UNIVERZA V MARIBORU

FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO

Oddelek za matematiko in računalništvo

DIPLOMSKO DELO

Gregor Donaj

Maribor, 2011

UNIVERZA V MARIBORU

FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO

Oddelek za matematiko in računalništvo

Diplomsko delo

Algoritmi za reševanje treh osnovnih problemov prikritih markovskih modelov

Mentor:

izr. prof. dr. Dominik Benkovič

Kandidat:

Gregor Donaj

Maribor, 2011

ZAHVALA

Najbolj nerazumljiva stvar na svetu je, da je svet sploh razumljiv.
(Albert Einstein)

Zahvaljujem se svojemu mentorju izr. prof. dr. Dominiku Benkoviču za pomoč in vodenje pri izdelavi diplomske naloge.

Posebna zahvala velja še staršema, ki sta mi omogočila študij.

UNIVERZA V MARIBORU

FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO

IZJAVA

Podpisani Gregor Donaj, rojen 14. januarja 1986, študent Fakultete za naravoslovje in matematiko Univerze v Mariboru, študijskega programa Matematika, izjavljam, da je diplomsko delo z naslovom

Algoritmi za reševanje treh osnovnih
problemov prikritih markovskih modelov

pri mentorju izr. prof. dr. Dominiku Benkoviču avtorsko delo. V diplomskem delu so uporabljeni viri in literatura korektno navedeni; teksti niso uporabljeni brez navedbe avtorjev.

Maribor, 1. junij 2011

Gregor Donaj

Algoritmi za reševanje treh osnovnih problemov prikritih markovskih modelov

program diplomskega dela

V diplomski nalogi naj bodo predstavljeni prikriti markovski modeli in osnovni trije problemi, ki so z njimi povezani: problem ocenjevanja, problem učenja in problem dekodiranja. Podrobneje naj bodo opisani algoritmi, ki se uporabljajo za reševanje teh problemov.

Prestavljena naj bo tudi uporaba prikritih markovskih modelov na področju razpoznavanja govora in implementacija algoritmov na tem področju.

Osnovni viri:

1. J. R. Deller, jr., J. G. Proakis, J. H. Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan Publishing Company, 1993.
2. G. A. Fink, *Markov Models for Pattern Recognition, from Theory to Applications*, Springer, Berlin Heidelberg, 2008.
3. O. C. Ibe, *Markov Processes for Stochastic Modeling*, Elsevier Academic Press, 2009.

izr. prof. dr. Dominik Benkovič

DONAJ, G.: Algoritmi za reševanje treh osnovnih problemov prikritih markovskih modelov.
Diplomsko delo, Univerza v Mariboru, Fakulteta za naravoslovje in matematiko, Oddelek za matematiko in računalništvo, 2010.

IZVLEČEK

V diplomski nalogi se obravnavajo prikriti markovski modeli, ki se v praksi uporabljajo predvsem na področju razpoznavanja govora. Opisani so osnovni pojmi modelov in trije osnovni z njimi povezani problemi: problem ocenjevanja, problem dekodiranja in problem učenja. Opisane so metode za reševanje teh problemov za diskretne in zvezne prikrite markovske modele.

Na kratko je opisana tudi uporaba prikritih markovskih modelov v razpoznavanju govora na primeru enostavne aplikacije razpoznavanja izoliranih besed z majhnim slovarjem.

Ključne besede: prikriti markovski modeli, algoritem naprej, algoritem nazaj, Viterbijev algoritem, Baum-Welchev algoritem, Viterbijevo učenje, avtomatsko razpoznavanje govora.

Math. Subj. Class. (2010): 60J20, 62J05, 62H12.

DONAJ, G.: Solution algorithms for the three fundamental problems of hidden Markov models.
Graduation Thesis, University of Maribor, Faculty of Natural Scicences and Mathematics, Department of Mathematics and Computer Science, 2010.

ABSTRACT

This paper presents hidden Markov models, which are in practice mostly used in speech recognition. Basic terms of the models and their tree fundamental problems: the evaluation, the decoding and the training problem are described. So are methods for solving them in the cases of discrete and continuous Markov models.

A short description of the use of hidden Markov models in speech recognition is also presented on the example application of small vocabulary isolated word recognition.

Keywords: hidden Markov models, forward algorithm, backward algorithm, Viterbi algorithm, Baum-Welch algorithm, Viterbi learning, automatic speech recognition.

Math. Subj. Class. (2010): 60J20, 62J05, 62H12.

Kazalo

Uvod

Za matematično analizo pojavov iz narave uporabljamo različne modele, s katerimi poskušamo to obnašanje čim bolj natančno opisati. Za nekatere pojave predvidevamo, da lahko njihovo obnašanje opišemo z naborom različnih stanj, med katerimi pojav prehaja. Kadar je to prehajanje odvisno samo od trenutnega stanja, je smiselno uporabiti markovske verige.

Markovska veriga je naključni proces, ki ga prikazujemo z zaporedjem stanj. Verjetnost, da se na nekem mestu pojavi določeno stanje, je pri tem odvisna le od prejšnjega stanja in ne od stanj pred njim. Za nekatere praktične uporabe pa so markovske verige preveč enostaven model za uporabo. Namesto njih uporabljamo prikrite markovske modele. Te modele lahko opišemo kot dvojni naključni proces. Prvi proces je markovska veriga, ki je ne moremo neposredno opazovati. Drugi proces pa je realizacija naključnih spremenljivk. Pri tem vsako stanje v markovski verigi realizira vrednost naključne spremenljivke. Verjetnostna porazdelitvena funkcija je pri tem odvisna le od trenutnega stanja v markovski verigi. Opazujemo le naključni proces realizacije izhodov in ne same markovske verige, ki je skrita v ozadju. Od tega prihaja ime *prikriti* markovski modeli.

Markovske verige in prikriti markovski modeli spadajo v večjo skupino naključnih procesov, ki jih imenujemo markovski procesi. Ime so dobili ruskem matematiku Andreju Markovu, ki je leta 1907 vpeljal končne markovske verige. Eno od prvih področij, kjer so se uporabljali prikriti markovski modeli in po katerem so najbolj znani, je procesiranje govora. Danes se uporabljajo tudi v razpoznavanju pisave in gestike ter v bioinformatiki. Prikriti markovski modeli so predvsem zanimivi iz dveh vidikov. Imajo bogato matematično strukturo in izkazali so se kot zelo uspešni pri modeliranju nekaterih pojavov.

Markovski modeli se uporabljajo v razpoznavanju govora že od začetka sedemdesetih let dvajsetega stoletja. Uporaba prikritih markovskih modelov zahteva veliko število aproksimiranja različnih parametrov in izračunov verjetnosti, zato je pomembno, da pri tem uporabljamo hitre algoritme, ki nam omogočajo uporabo razpoznavalnikov govora v realnem času.

V prvem poglavju so predstavljeni osnovni pojmi s področja naključnih procesov, ki jih potrebujemo pri obravnavi prikritih markovskih modelov. Podali bomo tudi definicije markovske verige in prikritega markovskega modela ter nekaj osnovnih lastnosti. V naslednjih treh poglavjih so obravnavani trije osnovni problemi prikritih modelov Markova. Podani so opisi problemov, način reševanja, časovne zahtevnosti algoritmov in primeri za diskretne prikrite markovske modele. V petem poglavju so opisani zvezni prikriti markovski modeli in razlike med algoritmi za reševanje osnovnih problemov za diskretne in zvezne modele. V zadnjem poglavju diplomske naloge je opisana uporaba prikritih markovskih modelov v razpoznavanju govora. Opisane so strukture uporabljenih modelov in način razpoznavanja govora.

Poglavje 1

Naključni in markovski procesi

1.1 Naključni procesi

V tem poglavju bomo pogledali osnovne definicije in zakonitosti, ki jih bomo potrebovali v nadaljevanju. Najprej bomo vpeljali nekaj oznak. $(\Omega, \mathcal{A}, P)$ bo za nas verjetnostni prostor, kjer je $\mathcal{A}$ σ-algebra na prostoru dogodkov Ω, P pa je verjetnostna funkcija. Z $\mathcal{B}$ bomo označevali Borelovo σ-algebro na realnih številih. Začeli bomo z formalno definicijo naključne spremenljivke in njene porazdelitvene funkcije.

Definicija 1.1 *Naj bo $(\Omega, \mathcal{A}, P)$ verjetnostni prostor. Preslikava*

$$X : \Omega \to \mathbb{R} \tag{1.1}$$

je naključna spremenljivka, če je za vsak $B \in \mathcal{B}$ velja $X^{-1}(B) \in \mathcal{A}$.

Definicija 1.2 *Naj bo $(\Omega, \mathcal{A}, P)$ verjetnostni prostor in $X : G \to \mathbb{R}$ naključna spremenljivka. Porazdelitveno funkcijo naključne spremenljivke X definiramo s predpisom:*

$$\begin{aligned} &F_X : \mathbb{R} \to [0,1] \\ &F_X(x) = P(X \leq x) = P(X \in (-\infty, x]) \qquad \forall x \in \mathbb{R}. \end{aligned} \tag{1.2}$$

Za nas to pomeni, da je naključna spremenljivka funkcija, ki vsakemu dogodku priredi realno število. Velikokrat pa že same dogodke označujemo z realnimi števili. Tedaj lahko pogosto gledamo že na dogodek kot naključno spremenljivko. Naključne spremenljivke lahko ločimo na diskretne in zvezne. Pravimo, da je X *diskretna* naključna spremenljivka,

če obstajajo takšna realna števila x_i, da velja

$$\sum_i P(X = x_i) = 1. \tag{1.3}$$

Za diskretne naključne spremenljivke vpeljemo verjetnostno funkcijo s predpisom

$$p_X(x_i) = P(X = x_i). \tag{1.4}$$

Pravimo, da je X *zvezna* naključna spremenljivka, če lahko $P(X \in B)$ zapišemo v obliki

$$P(X \in B) = \int_B f(t)dt = \int_{-\infty}^{\infty} I_B(t)f(t)dt \tag{1.5}$$

za neko nenegativno integrabilno funkcijo f, ki jo imenujemo gostota verjetnosti. Tukaj smo uporabili funkcijo indikator dogodka $I_B(t)$, ki zavzame vrednost 1, ko je $t \in B$, sicer pa vrednost 0. Za gostoto verjetnosti mora veljati

$$\begin{aligned} &\int_{\mathbb{R}} f(t)dt = 1, \\ &F_X(x) = \int_{-\infty}^{x} f(t)dt. \end{aligned} \tag{1.6}$$

V nadaljevanju bomo govorili o naključnih procesih. To so družine naključnih spremenljivk. Te družine so sicer lahko končne, vendar običajno obravnavamo neskončne družine naključnih spremenljivk. Tako končne kot neskončne družine naključnih spremenljivk opisujemo s parametrom. Pogosto za parameter vzamemo čas. Glede na vrednosti, ki jih lahko čas zavzame, ločimo naključne procese na procese v diskretnem času in na procese v zveznem času. Za oba primera podajmo formalno definicijo.

Definicija 1.3 *Naključni proces v diskretnem času je družina naključnih spremenljivk X_n, kjer je n element neke podmnožice celih števil.*

Primeri takšnih družin so: $\{X_n | n = 1, 2, 3, ...\}$, $\{X_n | n = 0, 1, 2, ...\}$ in $\{X_n | n = 0; \pm 1, \pm 2, ...\}$. Naključne spremenljivke, ki sestavljajo naključni proces, so pri tem lahko zvezne ali pa diskretne.

Definicija 1.4 *Naključni proces v zveznem času je družina naključnih spremenljivk X_t, kjer je t element nekega intervala realnih števil.*

Primera takšne spremenljivke sta: $\{X_t|t \in \mathbb{R}, t \geq 0\}$ in $\{X_t|t \in \mathbb{R}, 0 \leq t \leq T\}$. Glede na do sedaj opisane lastnosti lahko razdelimo naključne procese v 4 skupine:

- diskretni naključni procesi v diskretnem času,
- diskretni naključni procesi v zveznem času,
- zvezni naključni procesi v diskretnem času,
- zvezni naključni procesi v zveznem času.

Poglejmo si primer Brownovega gibanja. Leta 1827 je Robert Brown opazoval gibanje majhnih delcev v tekočini. Pri tem je opazil, da se ti delci gibljejo po neenakomernih poteh. Označimo z (X_t, Y_t, Z_t) pozicijo nekega delca ob času t. Potem so X_t, Y_t in Z_t naključni procesi. Brownovo gibanje je primer zveznega naključnega procesa v zveznem času. V tem primeru lahko rečemo, da je naključni proces družina naključnih vektorjev. Primer Brownovega gibanja je na sliki 1.1. Zaradi preglednosti smo na sliki predstavili le gibanje v dveh dimenzijah.

Ker je naključni proces sestavljen iz naključnih spremenljivk, lahko za vsako od teh spremenljivk izračunamo pričakovano vrednost $E(X_t)$. Označimo s T množico vseh vrednosti, ki jih lahko zasede t. Definiramo lahko *povprečno funkcijo* (ang. mean function) procesa $m_X : T \to \mathbb{R}$ s predpisom $m_X(t) = E(X_t)$. Definiramo tudi korelacijo med dvema naključnima spremenljivkama v procesu s predpisom $R_X(t_1, t_2) = E(X_{t_1} X_{t_2})$. Če gledamo na ta predpis kot na funkcijo spremenljivk t_1 in t_2, potem jo imenujemo *korelacijska funkcija*.

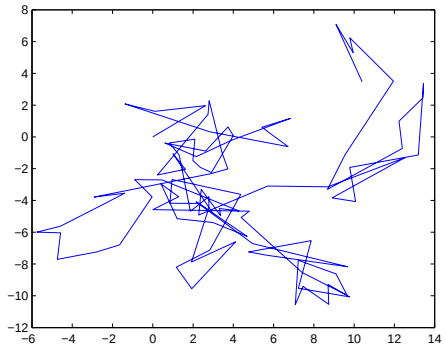
V nadaljevanju diplomske naloge se bomo omejili na markovske procese, za katere rečemo, da imajo omejen spomin.

Definicija 1.5 *Naključni proces $\{X_t|t \in T\}$ imenujemo markovski proces prvega reda, če je za vse $t_0 < t_1 < ... < t_n$ pogojna porazdelitvena funkcija spremenljivke X_{t_n} za dane vrednosti X_{t_0},...,$X_{t_{n-1}}$ odvisna le od $X_{t_{n-1}}$. Torej, če za vsak x_n velja:*

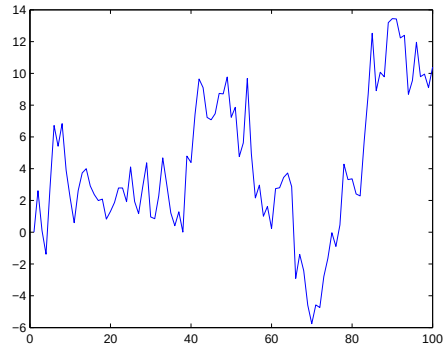
$$\begin{aligned} &P(X_{t_n} = x_n | X_{t_{n-1}} = x_{n-1}, X_{t_{n-2}} = x_{n-2}, ..., X_{t_0} = x_0) = \\ &P(X_{t_n} = x_n | X_{t_{n-1}} = x_{n-1}). \end{aligned} \tag{1.7}$$

To pomeni, da je porazdelitev naslednje naključne spremenljivke v naključnem procesu odvisna le od vrednosti, ki jo zaseda sedanja naključna spremenljivka, in ne od vrednosti prejšnjih naključnih spremenljivk. To lastnost imenujemo *markovska lastnost*.

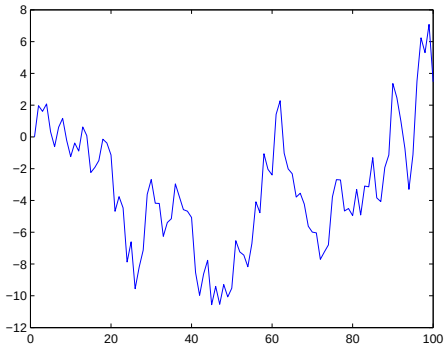
Podobno definiramo tudi markovske procese višjih redov. Markovski proces drugega reda je proces, kjer je naslednja naključna spremenljivka odvisna le od sedanje spremenljivke



(a)



(b)



(c)

Slika 1.1: Brownovo gibanje v dveh dimenzijah (a) prikaz gibanja v obeh dimenzijah (b) prikaz gibanja spremenljivke X po času (c) prikaz gibanja spremenljivke Y po času.

in zadnje spremenljivke, ne pa od spremenljivk pred njima. V splošnem je markovski proces i-tega reda proces, kjer za vsak $n > i$ velja

$$\begin{aligned} &P(X_{t_n}|X_{t_{n-1}} = x_{n-1}, X_{t_{n-2}} = x_{n-2}, ..., X_{t_0} = x_0) = \\ &P(X_{t_n}|X_{t_{n-1}} = x_{n-1}, ..., X_{t_{n-i}} = x_{n-i}). \end{aligned} \tag{1.8}$$

V večjem delu diplomske naloge bomo govorili o markovskih procesih prvega reda, zato bomo te procese imenovali preprosto markovske. Kadar bomo govorili o procesih višjega reda, bomo to posebej poudarili.

Tako kot vse naključne procese lahko tudi markovske procese ločimo glede na čas in glede na vrednosti naključnih spremenljivk. Markovske procese, ki jih sestavljajo diskretne naključne spremenljivke, imenujemo tudi markovske verige. Tako ločimo:

- markovske verige v diskretnem času,

- markovske verige v zveznem času,
- (zvezne) markovske procese v diskretnem času,
- (zvezne) markovske procese v zveznem času.

1.2 Markovske verige

Poglejmo si podrobneje markovske verige v diskretnem času, ki so osnova za razumevanje prikritih markovskih modelov. Povedali smo že, da gre za naključne procese v diskretnem času z diskretnimi naključnimi spremenljivkami. Množico vseh vrednosti, ki jih te diskretne spremenljivke lahko zasedejo imenujemo prostor stanj markovske verige, posamezne vrednosti pa stanja. Označujemo jih z naravnimi števili. Obravnavali bomo verige s končnim številom stanj, ki jih označujemo z naravnimi števili $1, 2, ..., N$.

V diplomski nalogi se bomo srečevali z markovskimi verigami v diskretnem času oblike $\{X_t | t = 0, 1, 2, ...\}$, ki jih bomo v nadaljevanju označevali z $(X_t)_{t\geq 0}$. Z indeksom t bomo nakazali splošni zapis markovske verige in kasneje prikritega markovskega modela v diskretnem času. Ponekod bomo za časovni indeks uporabljali tudi druge črke, na primer n in m.

1.2.1 Parametri markovske verige

Kadar zasede spremenljivka X_t v markovski verigi vrednost i, rečemo, da se markovska veriga nahaja v stanju i ob času t. Stanje, ki ga bo veriga zasedala v naslednjem trenutku $t + 1$ je odvisna sedaj od i. Definiramo pogojno verjetnost prehoda s predpisom

$$P(X_{t+1} = j | X_t = i). \tag{1.9}$$

V splošnem so verjetnosti prehoda iz enega v drugo stanje lahko odvisne tudi od časa t. Kadar so te verjetnosti neodvisne od časa rečemo, da je markovska veriga homogena. Za homogeno markovsko verigo vpeljemo oznako

$$p_{ij} = P(X_{t+1} = j | X_t = i). \tag{1.10}$$

Zaradi homogenosti je ta vrednost neodvisna od t. Hitro se prepričamo, da mora veljati

$$\sum_{j=1}^{N} p_{ij} = 1 \qquad \forall 1 \leq i \leq N. \tag{1.11}$$

Verjetnosti prehoda lahko zapišemo v obliki kvadratne matrike velikosti $N \times N$, kjer je N število stanj markovske verige. Pri tem zapišemo verjetnost p_{ij} v i-to vrstico in j-ti stolpec matrike. Matriko imenujemo prehodna matrika. Iz pogoja 1.11 sledi, da mora vsota elementov v vsaki vrstici matrike biti enaka 1. Takšno matriko imenujemo *stohastična*.

Verige, ki so oblike $(X_t)_{t\geq 0}$, imajo začetno naključno spremenljivko X_0, ki ni odvisna od drugih spremenljivk, zato moramo zanjo posebej definirati verjetnostno porazdelitev. Vpeljemo oznako

$$\pi_i = P(X_0 = i) \qquad \forall 1 \leq i \leq N. \tag{1.12}$$

Vrednosti π_i lahko združimo v vektor $\pi = (\pi_1, \pi_2, ..., \pi_N)^T$.

S pomočjo naštetih pojmov lahko podamo alternativno definicijo markovske verige, ki nam bo v nadaljevanju bolj priročna.

Definicija 1.6 *Markovska veriga v diskretnem času s končnim številom stanj je diskretni naključni proces oblike $(X_t)_{t\geq 0}$, kjer je t celo število in za katerega velja markovska lastnost. Parametri takšne markovske verige so prehodna matrika A in vektor π, ki vsebuje verjetnostno porazdelitev X_0.*

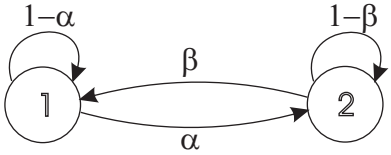
Zgled. Zamislimo si markovsko verigo, ki vsebuje dve stanji. Splošna oblika prehodne matrike je

$$A = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}, \tag{1.13}$$

vektorja začetnih verjetnosti pa

$$\pi = \begin{bmatrix} \pi_1 \\ \pi_2 = 1-\pi_1 \end{bmatrix}. \tag{1.14}$$

Grafično takšno markovsko verigo predstavimo kot množico stanj in usmerjenih povezav. Ta primer je predstavljen na sliki 1.2. Na sliki so zraven povezav zapisane verjetnosti prehoda po povezavi.



Slika 1.2: Splošni primer markovske verige z dvema stanjema

1.2.2 Večkoračne prehodne verjetnosti

Verjetnost p_{ij} iz enačbe 1.10 imenujemo tudi enokoračna prehodna verjetnost, ker nam pove kolikšna je verjetnost, da pridemo iz stanja i v stanje j v enem koraku. V homogeni markovski verigi definiramo m-koračno verjetnost

$$p_{ij}^{(m)} = P(X_{n+m} = j | X_n = i), \tag{1.15}$$

ki je tudi neodvisna od n in nam pove verjetnost, da preidemo iz stanja i v stanje j v m korakih.

Tukaj se seveda pojavi vprašanje, kako izračunamo te verjetnosti. Poglejmo si najprej primer za $m = 2$. Zaradi boljše preglednosti vstavimo za n vrednost 0. Izračunati hočemo

$$p_{ij}^{(2)} = P(X_2 = j | X_0 = i). \tag{1.16}$$

Zgornjo verjetnost lahko zapišemo kot popolno verjetnost po vseh možnih stanjih, v katerih se lahko nahaja veriga ob času 1.

$$\begin{aligned} p_{ij}^{(2)} &= \sum_k P(X_2 = j, X_1 = k | X_0 = i) \\ &= \sum_k P(X_2 = j | X_1 = k, X_0 = i) P(X_1 = k | X_0 = i) \\ &= \sum_k P(X_2 = j | X_1 = k) P(X_1 = k | X_0 = i) \\ &= \sum_k p_{ik} p_{kj} \end{aligned} \tag{1.17}$$

V zadnji vrstici zgornje enačbe vidimo, da smo iskano verjetnost zapisali kot skalarni produkt i-te vrstice in j-tega stolpca matrike verjetnosti prehoda, kar je element v i-ti vrstici in j-tem stolpcu kvadrata prehodne matrike A^2. Kasneje bomo to posplošili na poljuben m.

Naslednji izrek obravnava Chapman-Kolmogorove enačbe, ki dajejo posplošitev za zgoraj izpeljan rezultat.

Izrek 1.7 *Naj bosta r in m naravni števili, za kateri velja $r < m$. Potem za verjetnost prehoda v markovski verigi velja enačba*

$$p_{ij}^{(m)} = \sum_k p_{ik}^{(r)} p_{kj}^{(n-r)}. \tag{1.18}$$

Izrek pravi, da je verjetnost prehoda iz stanja i v stanje j v m korakih vsota po vseh stanjih

k produktov verjetnosti prehoda iz stanja i v stanje k v prvih r korakih in verjetnosti prehoda iz stanja k v končno stanje j v preostalih $m-r$ korakih. V zgornji izpeljavi smo dokazali primer za $m=2$ in $r=1$. Dokažimo sedaj še izrek.

Dokaz. Spomnimo se definicije verjetnosti prehoda v m korakih:

$$p_{ij}^{(m)} = P(X_m = j | X_0 = i). \tag{1.19}$$

Tudi ta izraz lahko zapišemo v obliki popolne verjetnosti glede na stanje X_r:

$$\begin{aligned} p_{ij}^{(m)} &= \sum_k P(X_n = j, X_r = k | X_0 = i) \\ &= \sum_k P(X_n = j | X_r = k, X_0 = i) P(X_r = k | X_0 = i) \\ &= \sum_k P(X_n = j | X_r = k) P(X_1 = k | X_0 = i) \\ &= \sum_k p_{ik}^{(r)} p_{kj}^{(n-r)}. \end{aligned} \tag{1.20}$$

□

Posledica 1.8 *Naj bo A prehodna matrika markovske verige. Verjetnost prehoda iz stanja i v stanje j v m korakih je enaka elementu v i-ti vrstici in j-tem stolpcu matrike A^m, oziroma*

$$p_{ij}^{(m)} = \{A^m\}_{ij}. \tag{1.21}$$

Dokaz. Posledico lahko dokažemo z indukcijo po številu korakov m. Za $m=1$ nimamo kaj dokazovati, saj je matrika A definirana tako, da je sestavljena iz verjetnosti prehoda v enem koraku.

Predpostavimo, da posledica velja za vse prehode dolžin $m-1$ korakov. Vstavimo v enačbo iz zgornjega izreka $r=1$. Tako dobimo

$$p_{ij}^{(m)} = \sum_k p_{ik} p_{kj}^{(m-1)}. \tag{1.22}$$

Elementi p_{ik} ustrezajo po definiciji i-ti vrstici matrike A, elementi $p_{kj}^{(m-1)}$ pa po indukcijski predpostavki j-temu stolpcu matrike A^{m-1}. Verjetnost $p_{ij}^{(m)}$ je torej enaka elementu v i-ti vrstici in j-tem stolpcu matrike $AA^{m-1} = A^m$. □

Zgled. Vzemimo markovsko verigo iz prejšnjega zgleda in vstavimo vrednosti $\alpha = 0.3$, $\beta = 0.6$ in $\pi_1 = 0.3$. Parametri verige so potem

$$A = \begin{bmatrix} 0.7 & 0.3 \\ 0.6 & 0.4 \end{bmatrix} \text{ in } \pi = \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix}. \tag{1.23}$$

Zanima nas verjetnost, da bo markovska veriga ob času $n = 2$ v prvem stanju. To verjetnost lahko izrazimo kot

$$\begin{aligned} P(X_2 = 1) &= P(X_2 = 1|X_0 = 1)P(X_0 = 1) + P(X_2 = 1|X_0 = 2)P(X_0 = 2) \\ &= p_{11}^{(2)} \cdot \pi_1 + p_{21}^{(2)} \cdot \pi_2. \end{aligned} \tag{1.24}$$

Verjetnosti $p_{11}^{(2)}$ in $p_{21}^{(2)}$ preberemo iz matrike

$$A^2 = \begin{bmatrix} 0.67 & 0.33 \\ 0.66 & 0.34 \end{bmatrix}. \tag{1.25}$$

Končno verjetnost dobimo

$$P(X_2 = 1) = 0.67 \cdot 0.3 + 0.66 \cdot 0.7 = 0.663. \tag{1.26}$$

1.2.3 Klasifikacija stanj markovske verige

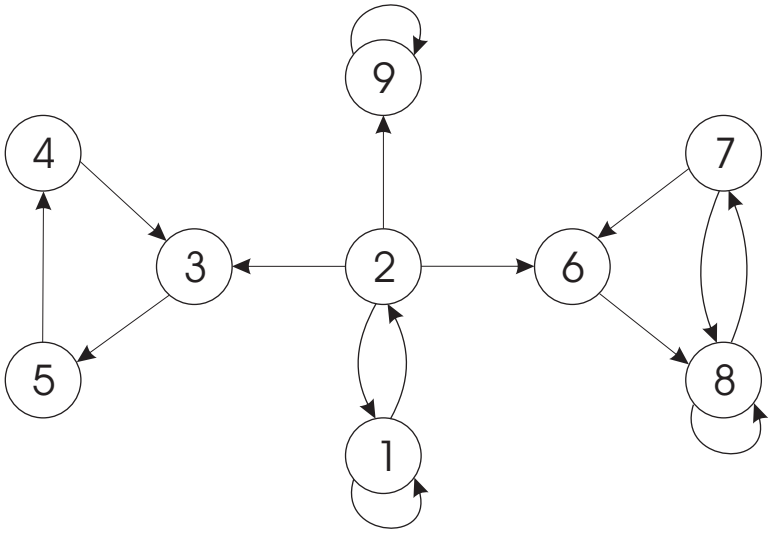
Stanja v markovski verigi lahko opisujemo glede na možnosti prehajanja med njimi. Enake izraze bomo pozneje uporabljali za opisovanje stanj v prikritem markovskem modelu.

Definicija 1.9 *Za stanja v markovski verigi lahko podamo naslednje lastnosti:*

- *Stanje j imenujemo dosegljivo iz stanja i, če je možno, da proces preide v nekem številu korakov iz stanja i v stanje j.*
- *Dve stanji sta povezani, če sta dosegljivi eno iz drugega.*
- *Stanje j imenujemo minljivo, če obstaja pozitivna verjetnost, da se proces nebo nikoli več vrnil v stanje j, potem ko ga enkrat zapusti.*
- *Stanje j imenujemo povrnjivo, če se bo proces z verjetnostjo 1 nekoč vrnil v to stanje, potem ko ga zapusti. Množica povrnljivih stanj tvori verigo, če vsako stanje v verigi komunicira z vsakim drugim stanjem v verigi.*
- *Povrnljivo stanje j imenujemo periodično, če obstaja naravno število $d > 1$, takšno, da je $p_{jj}^{(n)} = 0$ za vsak n različen od $d, 2d, 3d, ...$ d imenujemo perioda stanja j. Če je $d = 1$, imenujemo stanje aperiodično.*

- *Povrnljivo stanje j imenujemo pozitivno povrnljivo, če je pričakovan čas od zapustitve časa j do ponovne vrnitve v njega končen.*
- *Pozitivno povrnljivo aperiodično stanje imenujemo ergodično.*
- *Verigo iz ergodičnih stanj imenujemo ergodična veriga.*
- *Stanje j imenujemo absorbirajoče, če je* $p_{jj} = 1$.

Zgled. Za zgled si oglejmo markovsko verigo na sliki 1.3. Zaradi preglednosti na sliki nismo pisali verjetnosti prehoda. Vsaka povezava, ki je narisana predstavlja neničelno verjetnost prehoda med stanjema. Med stanji, kjer povezava ni narisana pa je verjetnost prehoda enaka 0. Stanji 1 in 2 sta primera minljivih stanj. Stanja 6, 7 in 8 so povrnljiva stanja, ki skupaj tvorijo verigo. Stanja 3, 4 in 5, ki prav tako tvorijo verigo, so periodična stanja s periodo 3. Stanje 9 je absorbirajoče, saj iz njega ne vodi nobena pot ven.



Slika 1.3: Primer markovske verige z različnimi vrstami stanj

1.2.4 Drugi markovski procesi

Markovske procese lahko zraven osnovne razdelitve glede na čas in verjetnostni prostor ločujemo tudi glede na to, ali lahko opazujemo stanja v katerih se nahaja veriga in ali imamo kakšen nadzor nad prehodi med stanji. Glede na te lastnosti poznamo zraven markovskih verig še markovske odločitvene procese, deloma vidne markovske odločitvene procese in prikrite markovske modele.

Drugi markovski procesi so še markovska naključna polja, markovski točkovni procesi, markovske čakalne vrste, markovske verige Monte Carlo, markovski obnovitveni procesi in markovski procesi prihodov.

1.3 Prikriti markovski modeli

Za markovske verige, ki smo je je do zdaj spoznali, lahko rečemo da spadajo med vidne markovske procese, saj lahko opazujemo stanja v realizaciji verige. V drugo skupino markovskih procesov pa spadajo prikriti markovski modeli (ang. hidden Markov model – HMM), v katerih ne moremo neposredno opazovati zaporedja stanj, ampak lahko le nanj sklepamo na podlagi realizacij drugega naključnega procesa.

1.3.1 Definicija prikritega markovskega modela

Definicija 1.10 *Diskretni prikriti markovski model je naključni proces v diskretnem času $(X_t, Y_t)_{t\geq 0}$, kjer je $(X_t)_{t\geq 0}$ markovska veriga in $(Y_t)_{t\geq 0}$ diskretni proces, v katerem za vsak n velja*

$$\begin{aligned} &P(Y_n = y_0 | X_0 = x_0, X_1 = x_1, ..., X_n = x_n, Y_0 = y_0, Y_1 = y_1, ..., Y_n = y_n) = \\ &P(Y_n = y_0 | X_n = x_n). \end{aligned} \tag{1.27}$$

Opišemo ga z naslednjimi parametri:

- *število stanj markovske verige N,*
- *prostor dogodkov $\Omega = \{o_1, ..., o_M\}$ naključnih spremenljivk Y_n, ki ga imenujemo tudi množica izhodnih simbolov,*
- *prehodno matriko stanj markovske verige A,*
- *množico verjetnostnih porazdelitev $\Phi = \{\phi_i | i = 1, ..., N\}$, kjer je $\phi_i(o_k)$ verjetnost $P(Y_n = o_k | X_n = i)$ in*
- *vektorjem začetnih verjetnosti stanj markovske verige π.*

Podobno definiramo tudi zvezni prikriti markovski model, v katerem je množica izhodnih signalov zvezna množica. Ponavadi je to kar interval realnih števil. Ker sta število stanj in množica izhodnih simbolov običajno dobro znana, obravnavamo predvsem ostale parametre modela, ki jih zapisujemo v obliki $\lambda = (A, \Phi, \pi)$.

Parametre HMM sestavljajo realna števila, ki so zapisana v obliki matrike, vektorja in množice verjetnostnih porazdelitev. Skupaj jih zapišemo z N^2+N+MN števili. Ker pa ta števila morajo zadoščati pogojem verjetnosti (razne vsote so enake 1), je HMM z N stanji in M izhodnimi simboli popolnoma opisan z $N(N-1)+(N-1)+N(M-1)$ parametri.

Ker je prostor dogodkov naključnih spremenljivk X_t množica stanj, bomo običajno uporabljali oznako s_t za stanje, ki ga zasede HMM ob času t. Zaporedje izhodnih simbolov bomo označevali z $O = o_1, o_2, ..., o_T$, zaporedje stanj pa z $S = s_1, s_2, ..., s_T$. Pogosto se bomo srečevali z verjetnostmi izhodnih simbolov in stanj. Ker bomo vedno uporabljali časovne indekse in enake simbole za obe zaporedji, lahko uporabljamo skrajšano oznako za verjetnost, da bo en od naključnih procesov iz HMM ob času n realiziral vrednost s_n oz. o_n. Verjetnosti bomo tudi pisali pogojne glede na parametre modela λ:

$$\begin{aligned} P(s_n|\lambda) &= P(X_n = s_n|\lambda), \\ P(o_n|\lambda) &= P(Y_n = o_n|\lambda). \end{aligned} \tag{1.28}$$

Zgled. Primeri oznak, ki jih bomo uporabljali so:

$$\begin{aligned} P(O|\lambda) &= P(Y_1 = o_1, ..., Y_T = o_T|\lambda), \\ P(o_n|\lambda) &= P(Y_n = o_n|\lambda), \\ P(S|\lambda) &= P(X_1 = s_1, ..., X_T = s_T|\lambda), \\ P(o_n|s_n, \lambda) &= P(Y_n = o_n|X_n = s_n, \lambda). \end{aligned} \tag{1.29}$$

Kadar obravnavamo prikrite markovske modele, opazujemo le naključni proces Y_t, ki ga tvori trenutno stanje v skriti verigi X_t. Tako lahko le z neko verjetnostjo rečemo, v katerem stanju je trenutno bil model.

Zgled. Zamislimo si igralca, ki meče kocko. Pri sebi ima tri nepoštene kocke. Vsakokrat vrže le eno kocko, potem pa se glede na to, katero kocko je pravkar vrgel odloči, katera kocka bo naslednja. Če ne gledamo koliko pik je bilo vrženih, ampak samo katera kocka je bila vržena, lahko opišemo obnašanje igralca z markovsko verigo. Naj bo prehodna matrika

$$A = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.4 & 0.6 \\ 0.3 & 0 & 0.7 \end{bmatrix} \tag{1.30}$$

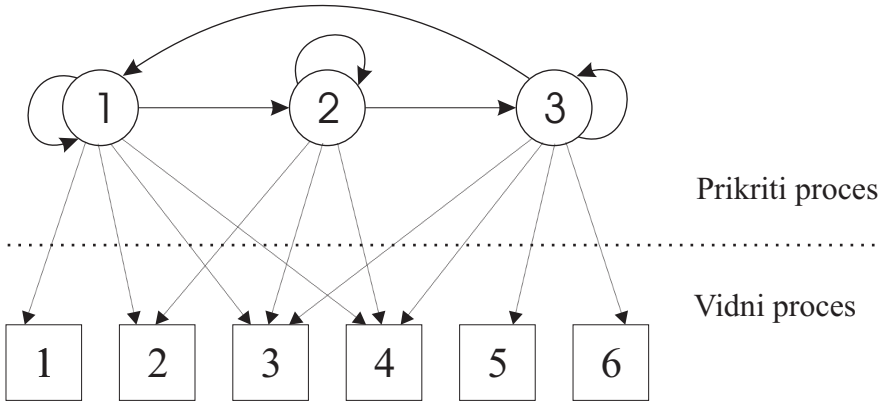
vektor začetnih verjetnosti pa

$$\pi = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{1.31}$$

Igralec bo vedno začel z prvo kocko in se bo po vsakem metu odločil ali bo spet vrgel isto kocko ali pa bo vzel naslednjo. Vsaki kocki priredimo verjetnostno porazdelitev:

$$\begin{aligned}
\phi_1 &\sim \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 0.5 & 0.2 & 0.1 & 0.2 & 0 & 0 \end{pmatrix}, \\
\phi_2 &\sim \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0.1 & 0.3 & 0.3 & 0 & 0 \end{pmatrix}, \\
\phi_3 &\sim \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 0.3 & 0.4 & 0.1 & 0.2 \end{pmatrix}.
\end{aligned} \tag{1.32}$$

Struktura takšnega HMM je prikazana na sliki 1.4. Opazujemo lahko le niz izhodnih simbolov. Če poznamo vse porazdelitvene funkcije izhodnih simbolov lahko sklepamo na zaporedje stanj. V tem zgledu lahko, ko vidimo izhodni simbol 1, 5 ali 6, z gotovostjo rečemo, v katerem stanju se v tistem trenutku nahaja model. Pri ostalih pa lahko povemo le verjetnosti, da se model nahaja v nekem stanju. Takšen model popolnoma opišemo z 23 parametri.

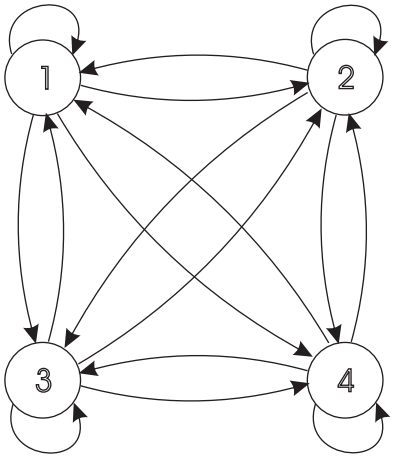


Slika 1.4: HMM s tremi stanji in šestimi izhodnimi simboli

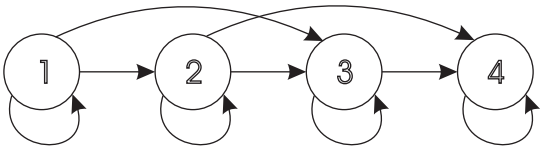
1.3.2 Tipi HMM

Podamo lahko tudi drugačno definicijo HMM, kjer model ne tvori izhodnih simbolov kadar se nahaja v nekem stanju, ampak ko prehaja med stanji. V tem primeru imamo toliko različnih verjetnostnih porazdelitev v množici Φ, koliko je povezav med stanji v modelu. Prvi tip HMM, ki smo ga prej definirali, imenujemo tudi *Moorov* HMM, pravkar opisanega pa *Mealyjev* HMM. Poimenovanje teh dveh tipov HMM prihaja iz poimenovanja Mooorovega in Mealyjevega stroja iz teorije avtomatov. Poljuben Mealyjev HMM se da pretvori v ekvivalenten Moorov HMM. Pri tem rečemo, da sta modela ekvivalentna, če se enako obnašata glede tvorjenja izhodnih simbolov, ne pa nujno glede prikritega procesa v ozadju. Zaradi te ekvivalence, bomo v nadaljevanju obravnavali le Moorov tip HMM.

Modele lahko ločimo tudi glede na možne prehode med njimi. Najbolj splošen primer je *ergodični* HMM, v katerem so vsa stanja ergodična. Primer takšnega HMM je na sliki 1.5. Drug značilen primer družine HMM so *levo-desni* ali *Bakisov* HMM. Uporabljamo jih za modeliranje govora. V takšnih modelih lahko stanja modela postavimo v vrsto tako, da bodo dovoljeni le prehodi iz leve v desno ali pa prehodi stanja samega vase. S takšnim vrstnim redom stanj bo v prehodni matriki za vsak $i > j$ veljalo $p_{ij} = 0$. Primer takšnega HMM je na sliki 1.6. V njem so vsa stanja tranzientna, zadnje pa je absorbirajoče. Definiramo lahko tudi strogo levo-desni HMM, v katerem velja tudi $p_{ii} = 0$.



Slika 1.5: Ergodični HMM



Slika 1.6: Levo-desni HMM

Poznamo tudi HMM s *tihimi stanji*. To so takšni modeli, ki vsebujejo običajna stanja, ki tvorijo izhodni simbol in pa tiha stanja, ki ga ne tvorijo. Običajno vpeljemo takšna stanja v modele za namenom, da zmanjšamo število povezav med stanji. Kadar se veliko stanj povezuje z veliko drugimi staji, lahko vstavimo tiho stanje, skozi katerega gredo prehodi in tako zmanjšamo število prehodov. Takšnih modelov v diplomski nalogi ne bomo obravnavali. Prav tako tudi ne veliko različnih možnih razširitev HMM na kompleksnejše modele, ki so možne za določena področja uporabe.

1.3.3 Osnovni problemi HMM

Pri obravnavi prikritih markovskih modelov se srečujemo z tremi osnovnimi problemi:

1. Problem ocenjevanja. Podan imamo model s parametri $\lambda = (A, \Phi, \pi)$ in zaporedje opazovanj $O = o_1, o_2, ..., o_T$ dolžine T, kjer je $o_i \in \Omega$ za vsak i. Izračunati želimo verjetnost, da je naš model s temi parametri tvoril takšno zaporedje izhodov:

$$P(O|\lambda). \tag{1.33}$$

2. Problem dekodiranja. Podan imamo model s parametri $\lambda = (A, \Phi, \pi)$ in zaporedje opazovanj $O = o_1, o_2, ..., o_T$ dolžine T, kjer je $o_i \in \Omega$ za vsak i. Določiti hočemo, katero je najbolj verjetno zaporedje stanj, skozi katera je model prehajal, da je tvoril to zaporedje opazovanj:

$$S' = \arg\max_S P(S|O, \lambda). \tag{1.34}$$

3. Problem učenja. Podana imamo le zaporedja opazovanj. Določiti hočemo parametre modela tako, da bomo maksimirali verjetnost $P(O|\lambda)$:

$$\lambda' = \arg\max_\lambda P(O|\lambda). \tag{1.35}$$

   Iščemo torej tak model, ki bo najbolj verjetno tvoril takšna zaporedja opazovanj. S tem določamo, kakšen model opisuje nek pojav, ki ga opazujemo.

Metode za reševanje teh problemov si bomo ogledali v naslednjih treh poglavjih.

Poglavje 2

Problem ocenjevanja

Kadar govorimo o problemu ocenjevanja, mislimo na izračun verjetnosti, da je določen model tvoril neko zaporedje opazovanj. Ocenjevanje uporabljamo, kadar imamo več modelov in hočemo ugotoviti, kateri izmed njih bi najverjetneje tvoril podano zaporedje opazovanj. Pri reševanju problema uporabljamo algoritem naprej ali pa algoritem nazaj.

V tem poglavju bo na primeru opisan problem ocenjevanja, ter njegovo reševanje s preprostim algoritmov, z algoritmoma naprej in algoritmom nazaj. Za algoritme bomo podali tudi njihove računske zahtevnosti.

2.1 Preprosti algoritem

Kadar obravnavamo (vidne) markovske verige, lahko govorimo le o verjetnosti, da je veriga prehajala skozi neko zaporedje stanj. To verjetnost lahko preprosto izračunamo. Naj bo $S = s_1, s_2, ..., s_T$ zaporedje stanj, skozi katera prehaja markovska veriga. Zgoraj omenjeno verjetnost izračunamo kot

$$P(S|A, \pi) = \pi_{s_1} p_{s_1 s_2} p_{s_2 s_3} \cdots p_{s_{T-1} s_T}, \tag{2.1}$$

kjer sta A in π parametra markovske verige, števila p_{ij} pa prehodne verjetnosti. Ta enačba sledi iz markovske lastnosti. Enako lahko zapišemo tudi verjetnost zaporedja prikritih stanj v HMM:

$$P(S|\lambda) = \pi_{s_1} \prod_{i=2}^{T} p_{s_{i-1} s_i}. \tag{2.2}$$

Za HMM s parametri λ lahko izračunamo verjetnost, da je določeno zaporedje stanj

$S = s_1, s_2, ..., s_T$ tvorilo neko zaporedje opazovanj $O = o_1, o_2, ..., o_T$. Ker so v HMM verjetnosti tvorjenja opazovanj odvisne le od trenutnega stanja in neodvisne med sabo, lahko zapišemo

$$\begin{aligned} P(O|S,\lambda) &= P(o_1|s_1,\lambda)P(o_2|s_2,\lambda)\dots P(o_T|s_T,\lambda) \\ &= \phi_{s_1}(o_1)\phi_{s_2}(o_2)\dots\phi_{s_T}(o_T) \\ &= \prod_{i=1}^{T}\phi_{s_i}(o_i), \end{aligned} \tag{2.3}$$

kjer so verjetnosti $\phi_i(o_k)$ definirane v definiciji 1.10.

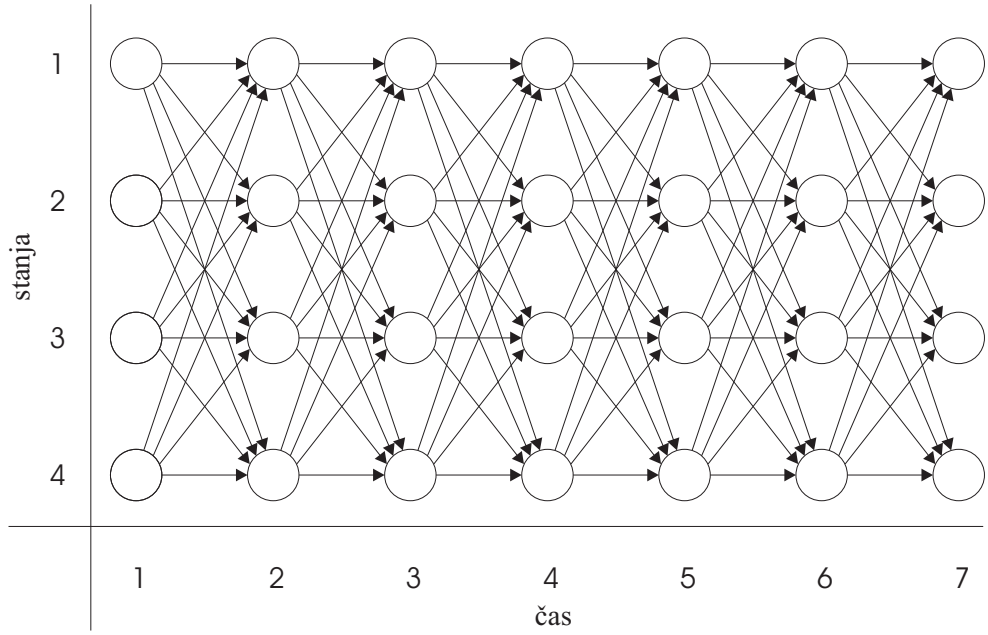
Ker pri praktični obravnavi HMM vidimo le zaporedje opazovanj, ki ga tvorijo skrita stanja, ne vemo, katero zaporedje stanj je tvorilo to zaporedje opazovanj. Za izračun verjetnosti moramo zato upoštevati vsa možna zaporedja stanj in verjetnosti teh zaporedij. Tako lahko zapišemo

$$\begin{aligned} P(O|\lambda) &= \sum_{S} P(O,S|\lambda) \\ &= \sum_{S} P(O|S,\lambda)P(S|\lambda) \\ &= \sum_{S}\left[\left(\prod_{i=1}^{T}\phi_{s_i}(o_i)\right)\pi_{s_1}\left(\prod_{i=2}^{T}p_{s_{i-1}s_i}\right)\right]. \end{aligned} \tag{2.4}$$

Kadar imamo model z N stanji, ki dovoljuje vse možne prehode med stanji in je tvoril T opazovanj, je možnih N^T različnih zaporedij stanj. Skupaj rabimo za ocenjevanje modela po enačbi 2.4 reda TN^T računskih operacij, natančneje $(2T-1)N^T$ množenj in $N^T - 1$ seštevanj. Že pri relativno majhnih vrednostih N in T postane ocenjevanje s tem preprostim algoritmom nepraktično, zato potrebujemo postopek kako izračunati verjetnost $P(O|\lambda)$ na hitrejši način.

2.2 Algoritem naprej

Preden si bomo pogledali algoritem za reševanje problema bomo predstavili shemo, ki predstavlja prehajanje markovske verige ali pa HMM med stanji in po času. Imenujemo jo graf poravnave, prikazana pa je na sliki 2.1. V njem imamo vozlišča, ki ustrezajo vsem stanjem ob vsakem možnem času, torej NT vozlišč. Razporejena so v vrstice in stolpce. Vsaka vrstica predstavlja eno stanje HMM ob vseh časih, vsak stolpec pa vsa stanja HMM ob nekem času. Povezave med vozlišči ustrezajo možnim prehodom med stanji med vsemi možnimi časi.



Slika 2.1: Graf poravnave za HMM s štirimi stanji in dolžino zaporedja stanj 7

Kadar ocenjujemo model s preprostim algoritmom iz prejšnjega razdelka, izvedemo veliko redundantnih računov. Rezultate teh računov nikjer ne shranjujemo. Ideja algoritma naprej je ta, da rezultate shranjujemo v obliki grafa poravnav. Vsakemu vozlišču v grafu priredimo verjetnost naprej.

Definicija 2.1 *Imejmo HMM s parametri λ in zaporedje opazovanj $O = o_1, o_2, ..., o_T$. Verjetnost naprej definiramo za vsako stanje i ob vsakem času t kot verjetnost, da je model tvoril delno zaporedje opazovanj $o_1, o_2, ..., o_t$ in se ob času t nahaja v stanju i:*

$$\alpha_t(i) = P(o_1, o_2, ..., o_t, s_t = i|\lambda) \qquad t = 1, ..., T, i = 1, ..., N. \tag{2.5}$$

S pomočjo verjetnostih naprej lahko sestavimo algoritem, s katerim ocenjujemo model. Ideja tega algoritma je, da verjetnost računamo postopoma po času. Vsa možna delna zaporedja stanj do nekega časa lahko razdelimo na N množic glede na to, v katerem stanju se končajo. Verjetnost delnega zaporedja opazovanj potem izračunamo z verjetnostmi delnih zaporedij, ki so za eno stanje krajše.

Algoritem je prikazan spodaj. Verjetnosti naprej računamo za vsa stanja ob nekem času. To storimo za vse čase v smeri od začetka proti koncu. Končno verjetnost dobimo kot vsoto verjetnosti za vsa stanja ob končnem času. Časovna zahtevnost tega algoritma je reda N^2T, natančneje zanj potrebujemo $N(N+1)(T-1)+N$ množenj in $N(N-1)(T-1)$ seštevanj.

Algoritem 1: Algoritem naprej

Vhod: O, λ
Izhod: $P(O|\lambda)$
Začni
  Za $i \in 1, ..., N$ **naredi**
    $\alpha_1(i) \longleftarrow \pi_i \phi_i(o_1)$
  Za $t \in 2, ..., T$ **naredi**
    Za $j \in 1, ..., N$ **naredi**
      $\alpha_t(j) \longleftarrow \left(\sum_{i=1}^{N} p_{ij} \alpha_{t-1}(i) \right) \phi_j(o_t)$
  $P(O|\lambda) \longleftarrow \sum_{i=1}^{N} \alpha_T(i)$

Dokaz. Za zapisan algoritem moramo dokazati, da nam pravilno izračuna verjetnosti naprej $\alpha_t(i)$ in končno verjetnost $P(O|\lambda)$. Pravilen izračun verjetnosti naprej dokažemo z indukcijo. Po definicijo verjetnosti naprej velja

$$\begin{aligned} \alpha_1(i) &= P(o_1, s_1 = i|\lambda) \\ &= P(o_1|s_1 = i\lambda)P(o_1|\lambda) \\ &= \phi_i(o_1)\pi_i, \end{aligned} \tag{2.6}$$

kar ustreza izračunu v algoritmu.

Predpostavimo, da algoritem pravilno izračuna verjetnosti naprej za vsa stanja i in za čas $t-1$, ki so po definiciji

$$\alpha_{t-1}(i) = P(o_1, o_2, ..., o_{t-1}, s_{t-1} = i|\lambda). \tag{2.7}$$

Verjetnost ob času t zapišemo kot

$$\begin{aligned} \alpha_t(j) &= P(o_1, o_2, ..., o_{t-1}, o_t, s_t = j|\lambda) \\ &= P(o_t|o_1, o_2, ..., o_{t-1}, s_t = j, \lambda)P(o_1, o_2, ..., o_{t-1}, s_t = j|\lambda) \\ &= P(o_t|s_t = j, \lambda)P(o_1, o_2, ..., o_{t-1}, s_t = j|\lambda) \\ &= P(o_t|s_t = j, \lambda) \sum_{i=1}^{N} (P(s_t = j \ s_{t-1} = i, \lambda)P(o_1, o_2, ..., o_{t-1}, s_{t-1} = i|\lambda)) \\ &= \phi_j(o_t) \sum_{i=1}^{N} p_{ij} \alpha_{t-1}(i), \end{aligned} \tag{2.8}$$

kar je enako predpisu v algoritmu. V izpeljavi smo uporabili le obrazce za popolno in

pogojno verjetnost. Izpeljava velja za poljubno stanje.

Hitro se prepričamo tudi, da je zaključek algoritma pravilen:

$$P(O|\lambda) = \sum_{i=1}^{N} P(O, s_T = i|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{2.9}$$

□

Zgled. Imejmo HMM z dvema stanjema in tremi izhodnimi simboli. Parametri modela naj bojo:

$$\begin{aligned} A &= \begin{bmatrix} 0.5 & 0.5 \\ 0.2 & 0.8 \end{bmatrix}, \\ \pi &= \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \\ \phi_1 &\sim \begin{pmatrix} 1 & 2 & 3 \\ 0.7 & 0.2 & 0.1 \end{pmatrix}, \\ \phi_2 &\sim \begin{pmatrix} 1 & 2 & 3 \\ 0.2 & 0.1 & 0.7 \end{pmatrix}. \end{aligned} \tag{2.10}$$

S pomočjo algoritma naprej hočemo izračunati, da je ta model tvoril zaporedje opazovanj

$$O = 1, 1, 2, 3, 3, 3, 2, 1, 3, 1. \tag{2.11}$$

Najprej izračunamo verjetnosti $\alpha_1(i)$ za obe stanji:

$$\begin{aligned} \alpha_1(1) &= \pi_1 \phi_1(o_1) = 0.5 \cdot 0.7 = 0.35, \\ \alpha_1(2) &= \pi_2 \phi_2(o_1) = 0.5 \cdot 0.2 = 0.1. \end{aligned} \tag{2.12}$$

Nato izračunamo preostale verjetnosti naprej za čase od 2 do T. Zaradi preglednosti zapišemo le vrednosti za čas $T = 10$:

$$\begin{aligned} \alpha_{10}(1) &= 1.2866 \cdot 10^{-5}, \\ \alpha_{10}(2) &= 1.2733 \cdot 10^{-5}. \end{aligned} \tag{2.13}$$

Izračunamo še končno verjetnost

$$P(O|\lambda) = \alpha_{10}(1) + \alpha_{10}(2) = 2.5619 \cdot 10^{-5}. \tag{2.14}$$

Verjetnost se zdi majhna, vendar ne pozabimo, da obstaja $3^{10} \approx 60.000$ možnih zaporedij

opazovanj dolžine 10. Za primerjavo lahko izračunamo verjetnost za zaporedje

$$O' = 3, 1, 2, 1, 3, 1, 2, 1, 3, 1. \tag{2.15}$$

Po istem postopku dobimo rezultat

$$P(O'|\lambda) = 6.9733 \cdot 10^{-6}. \tag{2.16}$$

V našem zgledu je $N = 2$ in $T = 10$. Za izračun končne verjetnosti bi z preprostim algoritmom iz prejšnjega razdelka potrebovali 20.479 računskih operacij, z algoritmom naprej pa smo jih potrebovali le 74, kar je skoraj 300 krat manj.

Prednost algoritma naprej v praktični uporabi je tudi ta, da lahko verjetnosti že računamo, čeprav še nismo videli celotnega zaporedja opazovanj. Ta lastnost je pomembna za delovanje algoritmov v realnem času. To pomeni, da moramo izračunati verjetnost tvorjenja opazovanj v času, primerljivem z časom tvorjenja opazovanj. Tukaj je pomembno dejstvo, da je časovna zahtevnost algoritma linearno odvisna od števila opazovanj.

2.3 Algoritem nazaj

V prejšnjem razdelku smo spoznali algoritem naprej, ki je ime dobil po tem, da postopoma računa verjetnosti tvorjenja izhodnih simbolov v smeri od prvega do zadnjega. Verjetnosti pa lahko računamo tudi v drugi smeri. Dualni algoritem k algoritmu naprej, ki tako deluje, se imenuje algoritem nazaj. Za njegovo razumevanje podobno kot prej priredimo vsakemu vozlišču v grafu poravnave verjetnost nazaj.

Definicija 2.2 *Imejmo HMM s parametri λ in zaporedje opazovanj $O = o_1, o_2, ..., o_T$. Verjetnost nazaj definiramo za vsako stanje i ob vsakem času t kot verjetnost, da je model tvoril delno zaporedje opazovanj $o_{t+1}, o_{t+2}, ..., o_T$, če se je ob času t nahajal v stanju i:*

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T|s_t = i, \lambda) \qquad t = 1, ..., T, i = 1, ..., N. \tag{2.17}$$

Definicija je podobna definiciji verjetnosti naprej, vendar v tem primeru gledamo za nek čas vsa opazovanja za nekim časom in brez trenutnega opazovanja, trenutno stanje pa je pogoj glede na katerega računamo verjetnost, medtem ko smo pri verjetnosti naprej vključili trenutno stanje kot dogodek.

Čeprav za ocenjevanje modela popolnoma zadošča že algoritem naprej, bomo pokazali tudi algoritem nazaj, ker ga bomo potrebovali še v kasnejših poglavjih. Algoritem je

prikazan spodaj. Deluje podobno kot algoritem naprej, vendar je prvi korak trivialen, v zadnjem koraku pa moramo uporabiti tudi verjetnosti naprej za čas 1. Ta algoritem ima tako kot algoritem naprej časovno zahtevnost TN^2, natančneje zanj rabimo $2(T-1)N^2+2N$ množenj in $(T-1)N(N-1)+N-1$ seštevanj.

Algoritem 2: Algoritem nazaj

Vhod: O, λ
Izhod: $P(O|\lambda)$
Začni
  Za $i \in 1, ..., N$ **naredi**
    $\beta_T(i) \longleftarrow 1$
  Za $t \in T-1, ..., 1$ **naredi**
    Za $i \in 1, ..., N$ **naredi**
      $\beta_t(i) \longleftarrow \sum_{j=1}^{N} p_{ij}\beta_{t+1}(j)\phi_j(o_{t+1})$
  $P(O|\lambda) \longleftarrow \sum_{i=1}^{N} \beta_1(i)\alpha_1(i) = \sum_{i=1}^{N} \beta_1(i)\pi_i\phi_i(o_1)$

Dokaz. Tudi za ta algoritem bomo dokazali njegovo pravilno delovanje. V prvem koraku algoritma določamo verjetnosti $\beta_T(i)$, glede na definicijo so to verjetnosti opazovanj od časa T naprej, vendar se zaporedje opazovanj ravno konča pri T, zato te verjetnosti postavimo na 1.

Osrednji korak dokazujemo podobno kot v prejšnjem algoritmu. Začnemo pri indukcijski osnovi T. Tukaj nimamo kaj dokazovati, saj so vse verjetnosti enake 1. V indukcijskem koraku predpostavimo, da je algoritem pravilen za čas $t+1$. Za verjetnost za čas t zapišemo:

$$\begin{aligned}
\beta_t(i) &= P(o_{t+1}, o_{t+2}, ..., o_T | s_t = i, \lambda) \\
&= \sum_{j=1}^{N} P(o_{t+1}, o_{t+2}, ..., o_T, s_{t+1} = j | s_t = i, \lambda) \\
&= \sum_{j=1}^{N} P(o_{t+1} | s_{t+1} = j) P(o_{t+2}, ..., o_T, s_{t+1} = j | s_t = i, \lambda) \\
&= \sum_{j=1}^{N} P(o_{t+1} | s_{t+1} = j) P(o_{t+2}, ..., o_T | s_{t+1} = j) P(s_{t+1} = j | s_t = i, \lambda) \\
&= \sum_{j=1}^{N} \phi_j(o_{t+1}) \beta_{t+1}(j) p_{ij}.
\end{aligned} \tag{2.18}$$

Dokazati še moramo zaključek algoritma:

$$\begin{aligned} P(O|\lambda) &= P(o_1, o_2, ..., o_T|\lambda) \\ &= \sum_{i=1}^{N} P(o_1, o_2, ..., o_T, s_1 = i|\lambda) \\ &= \sum_{i=1}^{N} P(o_1, s_1 = i|\lambda) P(o_2, ..., o_T|s_1 = i, \lambda) \\ &= \sum_{i=1}^{N} \alpha_1(i)\beta_1(i). \end{aligned} \tag{2.19}$$

Tudi v tem dokazu smo za vse izpeljave potrebovali le obrazca za pogojno in popolno verjetnost. □

Zgled. Rešimo z algoritmom nazaj enak primer HMM, kot smo ga v prejšnjem razdelku z algoritmom naprej. Parametri modela so

$$\begin{aligned} A &= \begin{bmatrix} 0.5 & 0.5 \\ 0.2 & 0.8 \end{bmatrix}, \\ \pi &= \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \\ \phi_1 &\sim \begin{pmatrix} 1 & 2 & 3 \\ 0.7 & 0.2 & 0.1 \end{pmatrix}, \\ \phi_2 &\sim \begin{pmatrix} 1 & 2 & 3 \\ 0.2 & 0.1 & 0.7 \end{pmatrix}. \end{aligned} \tag{2.20}$$

Najprej izračunamo verjetnost za zaporedje opazovanj

$$O = 1, 1, 2, 3, 3, 3, 2, 1, 3, 1. \tag{2.21}$$

Izračunamo vse verjetnosti nazaj. Zadnje verjetnosti so pri tem:

$$\begin{aligned} \beta_1(1) &= 6.1756 \cdot 10^{-5}, \\ \beta_1(2) &= 4.0038 \cdot 10^{-5}. \end{aligned} \tag{2.22}$$

Izračunamo še končno verjetnost

$$P(O|\lambda) = \beta_1(1)\pi_1\phi_1(o_1) + \beta_1(2)\pi_2\phi_2(o_1) = 2.5618 \cdot 10^{-5}. \tag{2.23}$$

Izračunajmo še verjetnost za zaporedje

$$O' = 3, 1, 2, 1, 3, 1, 2, 1, 3, 1. \tag{2.24}$$

Po istem postopku dobimo rezultat

$$P(O'|\lambda) = 6.9731 \cdot 10^{-6}. \tag{2.25}$$

Vidimo, da smo dobili enaka rezultata kot pa pri algoritmu naprej.

Poglavje 3

Problem dekodiranja

V prejšnjem poglavju smo si ogledali, kako izračunamo verjetnost, da je določen model tvoril neko zaporedje opazovanj. Ta izračun je lahko uporaben kadar imamo več modelov in hočemo ugotoviti, kateri najbolj ustreza opazovanjem. Ponekod nas lahko zanima tudi verjetnost najverjetnejšega zaporedja stanj, ponekod pa rabimo tudi informacijo o samem notranjem delovanju modela. V tem primeru hočemo ugotoviti, katero je najverjetnejše zaporedje stanj skozi katerega je model prehajal.

V tem poglavju bomo opisali sam problem dekodiranja in ga primerjali z ocenjevanjem. Opisali bomo izračun verjetnosti, da je najboljša pot tvorila neko zaporedje. Za reševanje problema dekodiranja se uporablja Viterbijev algoritem, ki se je razvil Andrew J. Viterbi in objavil v znanstvenem članku leta 1967. Prvotno se je uporabljal za dekodiranje konvolucijskih kod, potem pa se je začel uporabljati tudi na drugih področjih. Viterbijev algoritem spada med metode dinamičnega programiranja, s katerimi rešujemo kompleksne probleme z razbitjem na manjše probleme in združevanjem teh rešitev.

3.1 Opis problema

Imejmo HMM z parametri λ, ki je tvoril zaporedje opazovanj $O = o_1, o_2, ..., o_T$ dolžine T. Zanima nas, katero izmed vseh možnih zaporedij stanj je najverjetneje tvorilo to zaporedje opazovanj:

$$S' = \arg\max_{S} P(S|O, \lambda). \tag{3.1}$$

Z uporabo Bayesovega obrazca lahko verjetnost na desni strani zgornje enačbe preoblikujemo v

$$P(S|O,\lambda) = \frac{P(S,O|\lambda)}{P(O|\lambda)}. \tag{3.2}$$

Verjetnost v imenovalcu je nepomembna za maksimiranje zgornjega izraza, zato lahko zapišemo

$$\begin{aligned} S' &= \arg\max_S P(S,O|\lambda) \\ &= \arg\max_S P(O|S,\lambda)P(S|\lambda) \\ &= \arg\max_S \left[\left(\prod_{i=1}^{T}\phi_{s_i}(o_i)\right)\pi_{s_1}\left(\prod_{i=2}^{T}p_{s_{i-1}s_i}\right)\right]. \end{aligned} \tag{3.3}$$

Če hočemo problem dekodiranja rešiti neposredno po zgornji enačbi, moramo izračunati in primerjati verjetnosti vseh možnih zaporedij. Prav tako kot pri preprostem algoritmu ocenjevanja ima ta pristop računsko zahtevnost TN^T. Razlika v postopkih je le ta, da verjetnosti posameznih poti sedaj med sabo primerjamo, pri ocenjevanju pa smo jih seštevali, sicer pa za njiju rabimo enako število računskih operacij.

3.2 Posamezna optimalna stanja

Najprej si poglejmo postopek s katerim ugotovimo v katerem stanju se je model najverjetneje nahajal v nekem trenutku.

Trditev 3.1 *Naj bo podan HMM s parametri λ in zaporedje opazovanj $O = o_1, o_1, ..., o_T$. Naj bodo $\alpha_t(i)$ verjetnosti naprej in $\beta_t(i)$ verjetnosti nazaj. Potem za vsak t velja*

$$P(O, s_t = i|\lambda) = \alpha_t(i)\beta_t(i) \tag{3.4}$$

in

$$P(O|\lambda) = \sum_i \alpha_t(i)\beta_t(i) \tag{3.5}$$

Dokaz. Prvo enačbo trditve smo za $t = 1$ dokazali že pri algoritmu nazaj, podobno jo tudi dokažemo za poljuben t:

$$\begin{aligned} P(O, s_t = i|\lambda) &= P(o_1, ..., o_t, o_{t+1}, ..., o_T, s_t = i|\lambda) \\ &= P(o_1, ..., o_t, s_t = i|\lambda)P(o_{t+1}, ..., o_T, |o_1, ..., o_t, s_t = i, \lambda) \\ &= P(o_1, ..., o_t, s_t = i|\lambda)P(o_{t+1}, ..., o_T, |s_t = i, \lambda) \\ &= \alpha_t(i)\beta_t(i). \end{aligned} \tag{3.6}$$

Drugo enačbo dokažemo s pomočjo prve enačbe in popolne verjetnosti:

$$P(O|\lambda) = \sum_i P(O, s_t = i|\lambda) = \sum_i \alpha_t(i)\beta_t(i). \tag{3.7}$$

□

S pomočjo obrazca za pogojno verjetnost lahko iz zadnje enačbe dokažemo še naslednjo posledico.

Posledica 3.2 *Naj bo podan HMM s parametri λ, ki je tvoril zaporedje opazovanj $O = o_1, o_1, ..., o_T$. Za vsak t zanj velja*

$$\gamma_t(i) = P(s_t = i|O, \lambda) = \frac{P(O, s_t = i|\lambda)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)}. \tag{3.8}$$

Zgoraj smo vpeljali oznako $\gamma_t(i)$ za verjetnost, da se je model nahajal v stanju i ob času t. Posamezno najverjetnejše stanje dobimo kot

$$s_t^* = \arg\max_{1 \le i \le N} \gamma_t(i) \qquad \forall t \in 1, ..., T. \tag{3.9}$$

Iz posameznih optimalnih stanj lahko sicer tvorimo zaporedje $S^* = s_1^*, ..., s_T^*$, ki pa ne upošteva prehodnih verjetnosti med stanji in je zato skupno lahko manj verjetno kot pa katero drugo stanje. V skrajnem primeru lahko na ta način tvorimo zaporedje stanj, ki sploh ni možno, če je katera od prehodnih verjetnosti enaka 0. Kljub temu, da je ta postopek napačen za dekodiranje, bo nam rezultat iz zadnje posledice koristil v naslednjem poglavju. V nadaljevanju bomo spoznali postopek, ki nam da pravilno rešitev problema dekodiranja.

3.3 Viterbijev algoritem

Viterbijev algoritem deluje podobno kot algoritem naprej. Pri dekodiranju v grafu poravnave beležimo le verjetnosti, ki jih dobimo po najboljši poti do ustreznega vozlišča ob

ustreznem času.

Imejmo HMM s parametri λ, ki je tvoril zaporedje opazovanj $O = o_1, ..., o_T$. Najprej definiramo količino

$$\delta_t(i) = \max_{s_1,...,s_{t-1}} P(o_1, ..., o_t, s_1, ..., s_{t-1}, s_t = i|\lambda), \tag{3.10}$$

ki predstavlja za delno zaporedje opazovanj $o_1, ..., o_t$ verjetnost najboljše delne poti, ki se ob času t konča v vozlišču i. Definicija $\delta_t(i)$ nas spominja na definicijo verjetnosti naprej, le da verjetnosti posameznih delnih poti ne seštevamo ampak primerjamo in iščemo največjo. Vsakemu vozlišču v grafu poravnave, ki smo mu priredilo to verjetnosti določimo tudi kazalec

$$\psi_t(j) = \arg\max_i \delta_{t-1}(i)p_{ij}, \tag{3.11}$$

ki nam pove katero je predzadnje stanje v optimalnem delnem zaporedju stanj, ki se končna v stanju i ob času t.

S pomočjo teh dveh količin lahko zapišemo Viterbijev algoritem.

Algoritem 3: Viterbijev algoritem

Vhod: O, λ
Izhod: $P(S', O|\lambda), S'$
Začni
  Za $i \in 1, ..., N$ **naredi**
    $\delta_1(i) \longleftarrow \pi_i\phi_i(o_1)$
  Za $t \in 2, ..., T$ **naredi**
    Za $j \in 1, ..., N$ **naredi**
      $\delta_t(j) \longleftarrow \max_{1\leq i\leq N}\{\delta_{t-1}(i)p_{ij}\}\phi_j(o_t)$
      $\psi_t(j) \longleftarrow \arg\max_{1\leq i\leq N}\delta_{t-1}(i)p_{ij}$
  $P(S', O|\lambda) \longleftarrow \max_{1\leq i\leq N}\{\delta_T(i)\}$
  $s'_T \longleftarrow \arg\max_{1\leq i\leq N}\delta_T(i)$
  Za $t \in T-1, ..., 1$ **naredi**
    $s'_t \longleftarrow \psi_{t+1}(s'_{t+1})$

Dokaz. Pravilen izračun verjetnostih $\delta_t(i)$ dokazujemo z indukcijo. V prvem koraku algoritma določimo verjetnosti delnih zaporedij dolžine 1, ki se končajo v nekem stanju. Z vsako stanje je to samo eno zaporedje, za katerega velja $\delta_1(i) = \pi_i\phi_i(o_1)$.

Predpostavimo da velja algoritem za zaporedja dolžine $t-1$:

$$\delta_{t-1}(i) = \max_{s_1,...,s_{t-2}} P(o_1, ..., o_{t-1}, s_1, ..., s_{t-2}, s_{t-1} = i|\lambda). \tag{3.12}$$

Označimo z $\mathcal{S}_t$ množico vseh možnih zaporedij dolžine t. Preoblikujmo

$$\begin{aligned}\delta_t(j) &= \max_{s_1,...,s_{t-1}} P(o_1, ..., o_t, s_1, ..., s_{t-1}, s_t = j|\lambda) \\ &= \max\left\{P(o_1, ..., o_t, s_1, ..., s_{t-1}, s_t = j|\lambda)|s_1, ..., s_{t-1} \in \mathcal{S}_{t-1}\right\}.\end{aligned} \tag{3.13}$$

Iščemo maksimum zgornje množice. Najdemo ga tudi, če zgornjo množico razdelimo na podmnožice, poiščemo maksimume posameznih podmnožic in potem skupni maksimum. Zgornjo množico smo zapisali kot množico verjetnosti glede na zaporedja stanj $s_1, ..., s_{t-1}$. Razdelimo jih lahko glede na stanje s_{t-1}:

$$\begin{aligned}&\{P(o_1, ..., o_t, s_1, ..., s_{t-1}, s_t = j|\lambda)|s_1, ..., s_{t-1} \in \mathcal{S}_{t-1}\} \\ =&\left\{\left\{P(o_1, ..., o_t, s_1, ..., s_{t-1} = i, s_t = j|\lambda)|s_1, ..., s_{t-1} \in \mathcal{S}_{t-1}\right\}|i = 1...N\right\}\end{aligned} \tag{3.14}$$

Maksimum posamezne podmnožice je

$$\begin{aligned}&\max\{\, P(o_1, ..., o_t, s_1, ..., s_{t-1} = i, s_t = j|\lambda)|s_1, ..., s_{t-1} \in \mathcal{S}_{t-1}\,\} \\ =&\max\{\, P(o_1, ..., o_{t-1}, s_1, ..., s_{t-2}, s_{t-1} = i|\lambda) \\ &\qquad P(s_t = j|s_{t-1} = i, \lambda) \\ &\qquad P(o_t|s_t = j|\lambda)|s_1, ..., s_{t-2} \in \mathcal{S}_{t-2}\,\} \\ =&\max\{\, P(o_1, ..., o_{t-1}, s_1, ..., s_{t-2}, s_{t-1} = i|\lambda)|s_1, ..., s_{t-2} \in \mathcal{S}_{t-2}\,\}\, p_{ij}\phi_j(o_t).\end{aligned} \tag{3.15}$$

Verjetnosti p_{ij} in $\phi_j(o_t)$ smo pisali zunaj, ker sta neodvisni od zaporedja stanj $s_1, ..., s_{t-2}$, maksimum množice v zadnji vrstici pa je enak $\delta_{t-1}(i)$. Skupni maksimum potem iščemo

$$\max\{\delta_{t-1}(i)p_{ij}\phi_j(o_t)|i = 1, ...N\} = \max\{\delta_{t-1}(i)p_{ij}|i = 1, ...N\}\,\phi_j(o_t), \tag{3.16}$$

kjer smo spet $\phi_j(o_t)$ zapisali zunaj. Ta zadnji izraz pa je enak predpisu v algoritmu. Z $\psi_t(j)$ si zapomnimo pri katerem i ima zgornja množica maksimum.

Verjetnost najboljšega zaporedja izberemo kot največjo vrednost med vsemi najboljšimi zaporedji dolžine T, ki se končajo v različnih stanjih:

$$P(S', O|\lambda) = \max_{1\le i\le N}\{\delta_T(i)\}. \tag{3.17}$$

Ko smo določili verjetnost najboljšega zaporedja z izbiro največjega člena $\delta_T(i)$, smo s tem

členom določili v katerem stanju se konča optimalno zaporedje:

$$s'_T = \arg\max_{1\leq i\leq N} \delta_T(i). \tag{3.18}$$

Sedaj ko imamo končno stanje zaporedja in imamo za vsako najboljše delno zaporedje (med njimi tudi končno) definiran predzadnji člen zaporedja, lahko po vrsti od zadaj naprej zgradimo optimalno zaporedje tako, da glede na vsak trenutni člen določimo prejšnjega:

$$s'_t = \psi_{t+1}(s'_{t+1}). \tag{3.19}$$

□

Algoritem nam v svoji osnovni obliki da verjetnost $P(S', O|\lambda)$. Verjetnost $P(S|O, \lambda)$ dobimo potem s pomočjo enačbe 3.2.

Zgled. Vzemimo spet HMM iz prejšnjega poglavja. Njegovi parametri so:

$$\begin{aligned} A &= \begin{bmatrix} 0.5 & 0.5 \\ 0.2 & 0.8 \end{bmatrix}, \\ \pi &= \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \\ \phi_1 &\sim \begin{pmatrix} 1 & 2 & 3 \\ 0.7 & 0.2 & 0.1 \end{pmatrix}, \\ \phi_2 &\sim \begin{pmatrix} 1 & 2 & 3 \\ 0.2 & 0.1 & 0.7 \end{pmatrix}. \end{aligned} \tag{3.20}$$

Z Viterbijevim algoritmom določimo najboljša zaporedja stanj in pripadajoče verjetnosti za zaporedji opazovanj

$$O_1 = 1, 1, 2, 3, 3, 3, 2, 1, 3, 1 \tag{3.21}$$

in

$$O_2 = 3, 1, 2, 1, 3, 1, 2, 1, 3, 1. \tag{3.22}$$

Z Viterbijevim algoritmom izračunamo za prvo zaporedje:

$$\begin{aligned} &S'_1 = 1, 1, 1, 2, 2, 2, 2, 2, 2, 2 \quad \text{in} \\ &P(S'_1, O_1|\lambda) = 1.5420 \cdot 10^{-6}, \end{aligned} \tag{3.23}$$

za drugo zaporedje pa

$$\begin{aligned} &S_2' = 2,1,1,1,2,1,1,1,2,2 \quad \text{in} \\ &P(S_2', O_2|\lambda) = 1.6471 \cdot 10^{-7}. \end{aligned} \tag{3.24}$$

S pomočjo verjetnosti $P(O_1|\lambda)$ in $P(O_2|\lambda)$, ki smo ju izračunali v prejšnjem poglavju izračunamo še

$$\begin{aligned} &P(S_1'|O_1, \lambda) = 0.0602 \quad \text{in} \\ &P(S_2'|O_2, \lambda) = 0.0236. \end{aligned} \tag{3.25}$$

Poglavje 4

Problem učenja

V prejšnjih dveh poglavjih smo spoznali, kako računamo verjetnosti v nekem znanem modelu. V tem poglavju pa bomo spoznali, kako sploh določimo model oziroma kako določimo njegove parametre. Ocenjevanje lahko primerjamo z računanjem verjetnosti, da je neka vrednost naključne spremenljivke bila realizira od normalne porazdelitve. Problem učenja pa je primerljiv z določanjem parametrov porazdelitve. Kadar imamo opravka z normalno ali pa kakšno drugo dobro znano porazdelitvijo določimo parametre statistično s pomočjo parametrov vzorčne množice. Učenje HMM pa ni tako enostavno.

Učenje HMM je najtežji izmed osnovnih treh problemov. Preprosti zgled, ki smo ga videli v prejšnjih dveh poglavjih ima 2 stanji in 3 izhodne simbole. V tem modelu moramo oceniti 7 parametrov. Metoda, ki jo bomo spoznali za učenje modelov je Baum-Welchev algoritem. To je iterativen algoritem, ki vrednosti parametrov spreminja postopoma k bolj ugodni vrednosti. Kot vse metode, ki temeljijo na tem principu, tudi naš algoritem lahko najde le lokalni ekstrem optimizacijske funkcije. Druga metoda, ki jo bomo spoznali je Viterbijevo učenje, ki se razlikuje v optimizacijski funkciji učenja. Ker obe metodi poiščeta lokalni maksimum, imajo vedno začetni približki parametrov velik vpliv na končni rezultat algoritma.

V tem poglavju bomo naprej natančneje opredelili problem učenja. Nato bomo opisali oba algoritma za učenje HMM in na koncu bomo opisali kako v njih uporabljamo informacije iz več vzorčnih zaporedij opazovanj. Pravilnosti samega algoritma ne bomo dokazovali, ker bi bil dokaz preveč zamuden, njegovo delovanje pa temelji na bolj splošnem EM algoritmu. Bralec lahko podrobnosti o tem algoritmu najde v [8]. Današnji Baum-Welchev algoritem pa temelji na vrsti člankov, ki jih je z drugimi objavil L.E. Baum [1, 2, 3, 4, 5].

4.1 Opis problema

Opazujemo nek pojav, ki ga vidimo kot naključni proces in za katerega predpostavimo, da lahko njegovo obnašanje opišemo z HMM. Z znanjem o pojavu predpostavimo tudi število stanj v modelu in možne povezave med stanji. Z opazovanjem naključnega procesa poznamo tudi možne izhodne simbole.

Sedaj formuliramo problem učenja. Podano imamo zaporedje opazovanj $O = o_1, o_2, ..., o_T$, ki ga je tvoril HMM z znanim številom stanj N, znano množico izhodnih simbolov Ω in neznanimi parametri $\lambda = (A, \pi, \Phi)$. Podana zaporedja opazovanj (lahko jih je tudi več) v tem primeru imenujemo učni podatki. Določiti hočemo parametre modela tako, da maksimiramo optimizacijsko funkcijo

$$P(O|\lambda). \tag{4.1}$$

Parametri modela so latentni, kar pomeni da ne moremo neposredno opazovati njihovega vpliva na podatke. Problem je tako kompleksen, da ne poznamo nobene metode, ki bi nam omogočala analitično poiskati maksimum zgornje verjetnosti. Poznamo le iterativno metodo, ki poišče približek lokalnemu maksimumu.

4.2 Baum-Wechev algoritem

4.2.1 EM algoritem

EM algoritem (ang.: expectation-maximization (EM) algorithm) je iterativna metoda za ocenjevanje latentnih parametrov v statističnih modelih. Pri tej metodi izmenično izvajamo dva koraka. Prvi je E-korak (expectation) v katerem izračunamo pričakovano vrednost verjetnosti, da je model tvoril podane podatke, v našem primeru jih imenujemo opazovane vrednosti, glede na podane približke parametrov modela.

Drugi korak je M-korak (maximization) v katerem izračunamo nove vrednosti parametrov, s katerimi maksimiramo pričakovano vrednost iz prvega koraka.

Baum-Welchev algoritem je poseben primer EM algoritma za HMM. V tem algoritmu zamenjamo vse parametre v modelu z njihovimi pogojnimi pričakovanimi vrednostmi glede na trenutne vrednosti parametrov in učne podatke.

4.2.2 Pričakovane vrednosti

Za izračun pričakovanih vrednosti bomo potrebovali v prejšnjem poglavju opisano a posteriori verjetnost $\gamma_t(i) = P(s_t = i|O, \lambda)$. Dodatno bomo potrebovali tudi a posteriori verjetnosti prehoda med dvema stanjema i in j ob času t:

$$\begin{aligned}\gamma_t(i,j) &= P(s_t = i, s_{t+1} = j|O, \lambda) \\ &= \frac{P(s_t = i, s_{t+1} = j, O|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i)p_{ij}\phi_j(o_{t+1})\beta_{t+1}(j)}{P(O|\lambda)}.\end{aligned} \tag{4.2}$$

To verjetnost lahko določimo za vsak čas, razen za čas T, kjer ni več prehoda v naslednje stanje. Vse pričakovane verjetnosti, ki jih bomo opisovali v tem razdelku so izračunane iz a posteriori verjetnost glede na podano zaporedje opazovanj O.

S pomočjo zgoraj omenjenih dveh količin lahko opišemo pričakovano vrednost verjetnosti prehoda med stanji kot razmerje med pričakovano vrednostjo prehoda med tema dvema stanjema in pričakovano vrednostjo nahajanja v prvem stanju ob vseh časih razen ob času T:

$$\hat{p_{ij}} = \frac{\sum_{t=1}^{T-1} P(s_t = i, s_{t+1} = j|O, \lambda)}{\sum_{t=1}^{T-1} P(s_t = i|O, \lambda)} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}. \tag{4.3}$$

S pomočjo verjetnosti nahajanja v stanju ob času 1 določimo pričakovane vrednosti v vektorju začetnih verjetnosti:

$$\hat{\pi_i} = P(s_1 = i|o, \lambda) = \gamma_1(i). \tag{4.4}$$

Podobno dobimo tudi verjetnosti izhodnih simbolov. Pričakovano vrednost tvorjenja poljubnega izhodnega simbola $u \in \Omega$ v stanju i izračunamo:

$$\hat{\phi_i}(u) = \frac{\sum_{t=1}^{T} P(s_t = i, u = o_t|O, \lambda)}{\sum_{t=1}^{T} P(s_t = i|O, \lambda)} = \frac{\sum_{t=1}^{T} \gamma_t(i)\delta(u, o_t)}{\sum_{t=1}^{T} \gamma_t(i)}, \tag{4.5}$$

kjer smo uporabili funkcijo Kroneckerjev delta δ, za katero velja:

$$\delta(i,j) = \begin{cases} 1; & i = j \\ 0; & i \neq j \end{cases}. \tag{4.6}$$

Tako upoštevamo v števcu le tiste člene zaporedja, kjer se pojavi izhodi simbol u.

4.2.3 Celotni algoritem

Algoritem 4: Baum-Welchev algoritem

Vhod: O
Izhod: $\hat{\lambda}$
Začni
  Izberemo začetne vrednosti za A, π, Φ.
  $\hat{\lambda} \longleftarrow (A, \pi, \Phi)$
  Ponavljaj
    $\lambda \longleftarrow \hat{\lambda}$
    Izračunaj nove $\hat{A}$, $\hat{\pi}$ in $\hat{\Phi}$ glede na λ
    $\hat{\lambda} \longleftarrow (\hat{A}, \hat{\Pi}, \hat{\Phi})$
  dokler *Razlika med* $P(O|\hat{\lambda})$ *in* $P(O|\lambda)$ *je dovolj majhna*

Sama struktura algoritma je dokaj preprosta. Najprej izberemo začetne vrednosti za parametre modela, nato izvajamo iterativni korak. S tem prenehamo, ko vidimo, da se vpliv spremembe parametrov ne pozna več bistveno na verjetnosti, da je model tvoril zaporedje opazovanj. V iteraciji algoritma predstavlja λ stare vrednosti parametrov, $\hat{\lambda}$ pa nove vrednosti. $\hat{A}$, $\hat{\pi}$ in $\hat{\Phi}$ računamo s pomočjo enačb 4.3, 4.4 in 4.5.

4.3 Viterbijevo učenje

Viterbijevo učenje se od Baum-Welchevega algoritma razlikuje v tem, da za optimizacijsko funkcijo uporablja verjetnost, ki smo jo spoznali v poglavju o dekodiranju $P'(O|\lambda) = P(S', O|\lambda)$, kjer je S' najboljše zaporedje stanj, kot smo ga določili v prejšnjem poglavju. Medtem ko smo v Baum-Welchevem algoritmu za izračun parametrov uporabljali verjetnosti, da se model nahaja v nekem stanju glede na vse možne poti, bomo tukaj upoštevali le najbolj verjetno pot. Ker upoštevamo samo eno pot, bodo verjetnosti zasedale le vrednosti 0 ali pa 1.

Za opis delovanja Viterbijevega učenja definiramo verjetnost, da ima najboljše zaporedje stanj na mestu t stanje i:

$$\chi_t(i) = P'(s_t = i|O, \lambda) = \begin{cases} 1; & s'_t = i \\ 0; & \text{sicer} \end{cases}, \tag{4.7}$$

kjer je $S' = s'_1, ..., s'_T$ najboljše zaporedje stanj. Ta predpis nam pove le ali je neko stanje ob nekem času v najboljšem zaporedju stanj, ki ga določimo s pomočjo Viterbijevega algoritma.

Nove približke za prehodne verjetnosti izračunamo podobno kot pri Baum-Welchevem algoritmu. Razlika je v tem, da uporabimo verjetnosti χ, namesto γ.

$$\hat{p}_{ij} = \frac{\sum_{t=1}^{T-1} P(s_t = i, s_{t+1} = j|S', O, \lambda)}{\sum_{t=1}^{T-1} P(s_t = i|S', O, \lambda)} = \frac{\sum_{t=1}^{T-1} \chi_t(i)\chi_{t+1}(j)}{\sum_{t=1}^{T-1} \chi_t(i)}. \tag{4.8}$$

Viterbijevo učenje nam ne daje koristnih novih približkov za vektor začetnih verjetnosti. Dobimo le verjetnost 1 za prvo stanje v najboljšem zaporedju in 0 za vsa ostala stanja. Tudi predpis za verjetnosti izhodnih simbolov dobimo s pomočjo verjetnosti $\chi_t(i)$:

$$\hat{\phi}_i(u) = \frac{\sum_{t=1}^{T} P(s_t = i, u = o_t|S', O, \lambda)}{\sum_{t=1}^{T} P(s_t = i|S', O, \lambda)} = \frac{\sum_{t=1}^{T} \chi_t(i)\delta(u, o_t)}{\sum_{t=1}^{T} \chi_t(i)}. \tag{4.9}$$

Algoritem 5: Viterbijevo učenje

Vhod: O
Izhod: $\hat{\lambda}$
Začni
  Izberemo začetne vrednosti za A, π, Φ.
  $\hat{\lambda} \longleftarrow (A, \pi, \Phi)$
  Ponavljaj
    $\lambda \longleftarrow \hat{\lambda}$
    Z Viterbijevim algoritmom določi S' glede na O in λ
    Izračunaj nove $\hat{A}$, $\hat{\pi}$ in $\hat{\Phi}$ glede na λ in S'
    $\hat{\lambda} \longleftarrow (\hat{A}, \hat{\pi}, \hat{\Phi})$
  dokler *Razlika med* $P'(O|\hat{\lambda})$ *in* $P'(O|\lambda)$ *je dovolj majhna*

4.4 Več zaporedij opazovanj

Kot pri vsakem ocenjevanju parametrov statističnih modelov, je vedno bolje imeti več podatkov kot pa manj. Kadar ocenjujemo parametre v ergodičnem HMM, lahko imamo eno zelo dolgo zaporedje opazovanj, ki nam bo zagotovilo, da smo vsako stanje v modelu dovolj pogosto obiskali, da lahko dobimo uporabne ocene parametrov. Drugače je pri levo-desnih model. Zaradi njihove značilnost, da se v neko stanje ne vrnemo, potem ko ga enkrat zapu-

stimo, je težko dobiti dovolj podatkov za vsako stanje. V tem primeru lahko uporabljamo več zaporedij opazovanj.

Naj bo

$$\mathcal{O} = \{O^{(1)}, O^{(2)}, ..., O^{(K)}\} \tag{4.10}$$

množica K zaporedij opazovanj, ki so lahko različnih dolžin. Posamezno zaporedje v tej množici je oblike $O^{(k)} = O_1^{(k)}, O_2^{(k)}, ..., O_{T_k}^{(k)}$. Predpostavimo, da so zaporedja stanj med seboj neodvisna. Sedaj je naš cilj, da maksimiramo optimizacijsko funkcijo

$$\begin{aligned} P(\mathcal{O}|\lambda) &= \prod_{k=1}^{K} P(O^{(k)}|\lambda) \\ &= \prod_{k=1}^{K} P_k, \end{aligned} \tag{4.11}$$

kjer smo z P_k označili verjetnost posameznega zaporedja opazovanj.

Enačbe za ocenjevanje parametrov temeljijo na pričakovanih vrednostih, zato jih lahko enostavno priredimo tako, da upoštevajo več različno verjetnih zaporedij in seštevajo njihov vpliv. Tako na primer enačbo za ocenjevanje prehodnih verjetnosti

$$\hat{p}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \tag{4.12}$$

ki upošteva samo eno zaporedje preuredimo v enačbo

$$\hat{p}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \gamma_t^{(k)}(i,j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \gamma_t^{(k)}(i)}, \tag{4.13}$$

kjer vrednosti $\gamma_t^{(k)}(i)$ in $\gamma_t^{(k)}(i,j)$ izračunamo za vsako zaporedje ločeno. Tako v števcu kot v imenovalcu smo seštevali vplive posameznih zaporedij in jih utežili z obratno vrednostjo njunih verjetnosti.

Na enak način tudi preoblikujemo enačbe za ocenjevanje verjetnosti izhodnih simbolov. Verjetnost začetnega vektorja v levo-desnih modelih ne ocenjujemo, saj v njih običajno fiksno določimo začetno stanje, tako da velja $\pi_1 = 1$ in $\pi_i = 0$ za vsak $i \neq 1$.

Poglavje 5

Zvezni prikriti markovski modeli

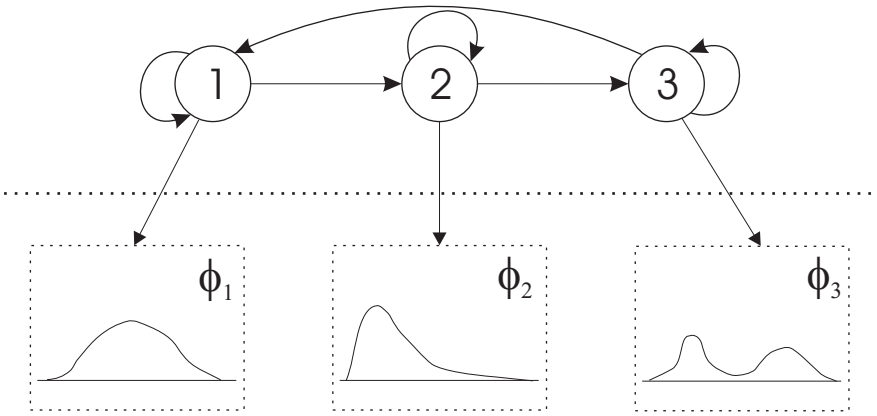
V prejšnjih treh poglavjih smo obravnavali algoritme le za primer diskretnih HMM. V tem poglavju bomo spoznali, kakšne so razlike z zveznimi modeli in kako moramo to upoštevati pri opisanih algoritmih.

Algoritma za ocenjevanje in dekodiranje se razlikuje samo v tem, da uporabljamo gostoto verjetnosti namesto verjetnostne funkcije. Drugačne zgodba je pri učenju modela, saj moramo ocenjevati parametre porazdelitev in ne posameznih verjetnosti. Učenje je odvisno od vrste porazdelitve, ki se uporablja. Najpogosteje se uporabljajo Gaussove porazdelitve. Opisali bomo modele z Gaussovimi porazdelitvami in enačbe za ocenjevanje parametrov teh porazdelitev.

5.1 Zvezne porazdelitve izhodnih simbolov

V prvem poglavju smo podali definicijo diskretnega HMM, kjer je množica izhodnih simbolov bila oblike $\Omega = \{o_1, ..., o_M\}$. Sedaj bomo pogledali zvezne HMM, kjer je ta množica oblike $\Omega = \mathbb{R}^M$ in kjer je množica $\Phi = \{\phi_i | i = 1, ..., N\}$ sestavljena iz zveznih porazdelitev. Primer takšnega HMM za $N = 3$ in $M = 1$ je prikazan na sliki 5.1.

V drugem poglavju smo obravnavali ocenjevanje modelov. Verjetnost, da je določeno zaporedje stanj tvorilo zaporedje opazovanj, smo zapisali kot produkt verjetnosti, da je posamezno stanje tvorilo istoležno opazovanje. Enako lahko storimo tudi z zveznimi modeli



Slika 5.1: Primer zveznega HMM

in dobimo enačbo

$$\begin{aligned} p(O|S,\lambda) &= p(o_1|s_1,\lambda)p(o_2|s_2,\lambda)...p(o_T|s_T,\lambda) \\ &= \phi_{s_1}(o_1)\phi_{s_2}(o_2)\ldots\phi_{s_T}(o_T) \\ &= \prod_{i=1}^{T}\phi_{s_i}(o_i). \end{aligned} \tag{5.1}$$

Razlika je le v tem, da smo tukaj uporabljali M-dimenzionalne gostote verjetnosti ϕ_{s_i} in smo na koncu dobili MT-dimenzionalno gostoto $p(O|S,\lambda)$. Ves postopek izpeljave algoritmov za ocenjevanje lahko od tu naprej izvedemo enako kot pri diskretnih modelih. Enako velja tudi za Viterbijev algoritem za dekodiranje.

5.2 Gaussovi modeli

Običajna (enodimenzionalna) Gaussova ali normalna porazdelitev ima obliko

$$\begin{aligned} p(x) &= \mathcal{N}(x;\mu,\sigma) \\ &= \frac{1}{\sqrt{2\pi}\sigma}e^{-(\frac{x-\mu}{\sigma})^2}, \end{aligned} \tag{5.2}$$

kjer je μ srednja vrednost in σ standardni odklon. Velikokrat pa za modeliranje naravnih pojavov uporabljamo večdimenzionalne Gaussove porazdelitve, ki jih opišemo z enačbo

$$\begin{aligned} p(x) &= \mathcal{N}_M(x;\mu,C) \\ &= \frac{1}{(2\pi)^{\frac{M}{2}}\sqrt{\det C}}e^{-\frac{1}{2}(x-\mu)^T C(x-\mu)}, \end{aligned} \tag{5.3}$$

kjer je μ M-dimenzionalen vektor srednje vrednosti, C pa pripadajoča kovariančna matrika. Včasih uporabljamo celo mešane porazdelitve, ki jih dobimo kot uteženo vsoto enostavnih porazdelitev:

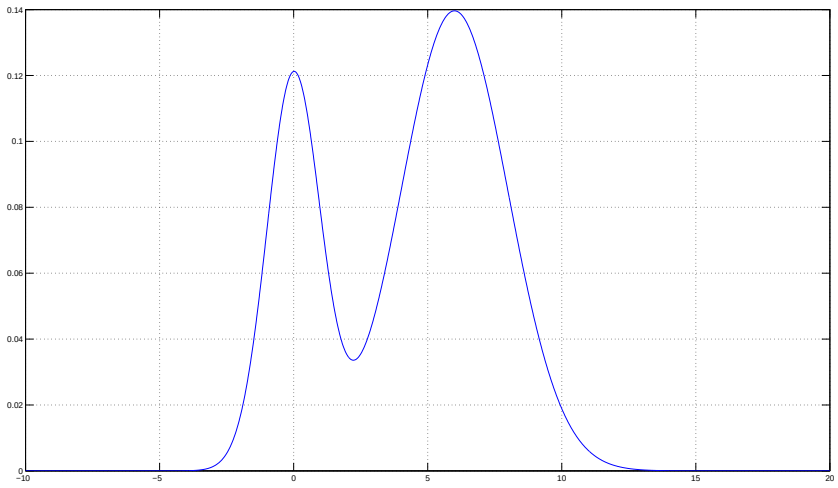
$$p(x) = \sum_{k=1}^{K} c_k \, \mathcal{N}_M(x, \mu_k, C_k). \tag{5.4}$$

Tukaj so c_k pozitivni utežni koeficienti, za katere velja, da je njihova vsota enaka 1. μ_k so M-dimenzionalni vektorji srednjih vrednosti posameznih komponent, C_k so pripadajoče kovariančne matrike. V teh primerih je x M-dimenzionalen naključni vektor. Gaussovih modelov ne bomo natančneje opisovali, ker ne nanašajo neposredno na našo osrednjo temo.

Zgled. Sestavimo mešano enodimenzionalno Gaussovo porazdelitev iz dveh komponent. V splošnem je verjetnostna funkcija takšne porazdelitve oblike

$$p(x) = c\,\mathcal{N}(x; \mu_1, \sigma_1) + (1-c)\mathcal{N}(x; \mu_2, \sigma_2) \qquad c \in (0,1), \mu_1, \mu_2, \sigma_1, \sigma_2 \in \mathbb{R}. \tag{5.5}$$

Če vstavimo vrednosti $c = 0.3$, $\mu_1 = 0$, $\sigma_1 = 1$, $\mu_2 = 6$, $\sigma_2 = 2$, dobimo verjetnostno funkcijo prikazano na sliki 5.2.



Slika 5.2: Primer mešane Gaussove porazdelitve

5.3 Učenje zveznih modelov

Uporaba zveznih porazdelitev nima vpliva na ocenjevanje novih vrednosti prehodnih verjetnosti in verjetnosti začetnega vektorja, zato uporabljamo iste enačbe, kot smo jih opisali

v prejšnjem poglavju. Dodali pa bomo enačbe za ocenjevanje parametrov porazdelitev.

Mešane večdimenzionalne Gaussove porazdelitve natančno opišemo, če za vsako komponento k podamo utež c_k, srednjo vrednost μ_k in kovariančno matriko C_k. V HMM jih podamo za vsako stanje j, zato uporabljamo oznake c_{jk}, μ_{jk} in C_{jk}.

Najprej definiramo pomožno količino

$$\xi_t(j,k) = P(s_t = j, M_t = k|O,\lambda) = \frac{\sum_{i=1}^{N} \alpha_{t-1}(i) p_{ij} c_{jk} \phi_{jk}(o_t) \beta_t(j)}{P(O|\lambda)}, \tag{5.6}$$

ki pove verjetnost, da je pri podanem zaporedju vektorjev opazovanj O model v stanju j izbral k-to komponento mešane porazdelitve in tvoril opazovanje o_t. Tukaj smo uporabili oznako ϕ_{jk}, ki predstavlja k-to komponento porazdelitve ϕ_j.

Enačbe s katerimi jih ocenjujemo v Baum-Welchevem algoritmu so:

$$\hat{c}_{jk} = \frac{\sum_{t=1}^{T} \xi_t(j,k)}{\sum_{t=1}^{T} \gamma_t(j)}, \tag{5.7}$$

$$\hat{\mu}_{jk} = \frac{\sum_{t=1}^{T} \xi_t(j,k) o_t}{\sum_{t=1}^{T} \xi_t(j,k)} \quad \text{in} \tag{5.8}$$

$$\hat{C}_{jk} = \frac{\sum_{t=1}^{T} \xi_t(j,k) o_t o_t^T}{\sum_{t=1}^{T} \xi_t(j,k)} - \hat{\mu}_{jk} \hat{\mu}_{jk}^T. \tag{5.9}$$

Poglavje 6

Uporaba prikritih markovskih modelov v razpoznavanju govora

V tem poglavju bomo spoznali eno od praktičnih uporab prikritih markovskih modelov. Najprej si bomo na enostavnem primeru ogledali osnovni princip statističnega razpoznavanja vzorcev. Nato bomo opisali predstavitev govornega signala v računalniku z njegovimi značilkami. Na koncu poglavja pa bomo pogledali razpoznavanje izoliran besed.

6.1 Razpoznavanje vzorcev

Razpoznavanje vzorcev delimo na dva pristopa. Prvi je nenadzorovano razpoznavanje, kjer poskušamo množico statističnih enot – vzorcev razdeliti na neke skupine glede na njihove značilnosti. Druga metoda je nadzorovano razpoznavanje, kjer imamo te skupine – imenujemo jih razrede – že vnaprej določene. Princip nadzorovanega razpoznavanje se da hitro prikazati na primeru.

Zamislimo si, da imamo neko statistično populacijo, ki jo lahko razdelimo na dva razreda c_1 in c_2 glede na neko njihovo lastnost. Na primer imamo neko vrsto živali, te pa lahko delimo na samce in samice. Iz populacije vzemimo reprezentativen vzorec in vsaki enoti v njemu določimo razred. Označimo relativni frekvenci razredov z f_1 in f_2. Vsakemu vzorcu izmerimo neko statistično lastnost, ki jo imenujemo značilka. Predpostavimo, da ima značilka znotraj vsakega razreda Gaussovo porazdelitev. Označimo srednje vrednosti in standardne odklone vzorcev v razredih z μ_1, μ_2, σ_1 in σ_2. S tem smo zaključili prvo fazo razpoznavanja vzorcev – učenje modela.

Druga faza je razpoznavanje. Vzemimo iz populacije nek primerek in mu poskušajmo določiti razred s pomočjo zgoraj naučenega modela. Izmerimo mu njegovo značilno x_0. Za

oba razreda zapišimo verjetnosti, da ima nek vzorec iz njiju vrednost značilke x_0:

$$\begin{aligned} p(x_0|c_1) &= \mathcal{N}(x_0; \mu_1, \sigma_1), \\ p(x_0|c_2) &= \mathcal{N}(x_0; \mu_2, \sigma_2). \end{aligned} \tag{6.1}$$

Iz teh vrednosti lahko s pomočjo Bayesovega obrazca izrazimo verjetnosti, da vzorec z vrednostjo značilke x_0 pripada nekemu razredu:

$$\begin{aligned} P(c_1|x_0) &= \frac{p(x_0|c_1)P(c_1)}{p(x_0)}, \\ P(c_2|x_0) &= \frac{p(x_0|c_2)P(c_2)}{p(x_0)}. \end{aligned} \tag{6.2}$$

Tukaj za verjetnosti $P(c_1)$ in $P(c_2)$ vstavimo prej izračunane relativne frekvence razredov f_1 in f_2. Naloga razpoznavanja na tem mestu je sprejeti odločitev kateremu razredu naj naš vzorec pripada. Najenostavnejši klasifikator je Bayesov klasifikator minimalne napake, ki pove, da naj x_0 pripišemo tistemu razredu, ki je v zadnji enačbi dobil večjo verjetnost. Na tak način minimiziramo pričakovano število napačno razpoznanih vzorcev. Drugi klasifikatorjev ne bomo omenjali.

Razpoznavanje govora poteka po podobnem principu, le da na mestu enostavnih Gaussovih porazdelitev uporabljamo HMM.

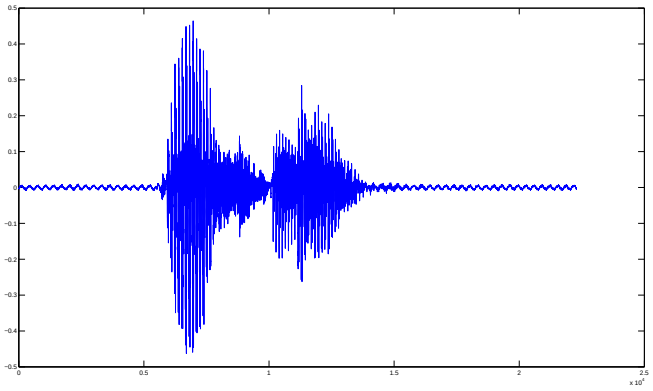
Kot smo že prej omenili, so značilke lastnosti vzorcev, s katerimi jih opišemo in ki nam pomagajo pri razpoznavanju. Značilke morajo biti izbrane tako, da po eni strani z njimi lahko dobro razločujemo med različnimi fonemi, ki jih izgovori neka oseba, po drugi strani pa se njihove vrednosti ne smejo preveč spreminjati, kadar nek fonem izgovorijo različne osebe.

V procesiranju govora jih z različnimi postopki izločamo iz govornega signala (ta je v računalniku predstavljeno kot zaporedje števil) v enakomernih časovnih presledkih. Tipične značilke, ki se uporabljajo so: kratkočasovna povprečna energija in amplituda, povprečno število prečkanja nivoja nič, kepstralni koeficienti in iz njih izpeljane dinamične značilke. Podrobnosti o teh značilkah lahko bralec najde v [7, 12, 13] ali kateri drugi knjigi s področja digitalnega procesiranja signalov ali razpoznavanja govora.

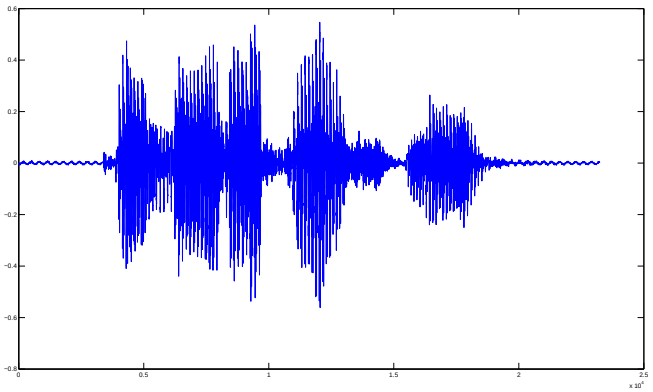
Značilke izločamo v časovnih presledkih, v katerih se lastnosti govora, ki so pomembne za razpoznavanja izgovorjenega ne spreminjajo. Tipičen čas je na primer 10 milisekund. V vsakem presledku izločimo množico značilk v obliki vektorja.

6.2 Razpoznavanje izoliranih besed z majhnim slovarjem

Opisali bomo le eno najmanj zahtevnih metod razpoznavanja govora. Pojem izolirane besede pomeni, da razpoznavamo besede, ki jih obdaja tišina in jim zato enostavno določimo začetek in konec. Tak primer je na sliki 6.1a. Drug način je razpoznavanje tekočega govora, kjer mej med besedami ne moremo enostavno določiti. Primer takšne izgovorjave je na sliki 6.1b. Pojem majhnega slovarja pomeni, da razpoznavalnik pozna le majhno (do 100) število besed, med kateri odloča katera je najbolj podobna izgovorjeni besedi, ki jo hočemo prepoznati.



(a)



(b)

Slika 6.1: Predstavitev govornega signala (a) za izolirano besedo *desno* (b) za tekoči besedi *kamera desno*.

Primer take metode je govorno krmiljena telefonska centrala. Tako imamo lahko primer centrale na podjetju, kjer povemo ime in priimek zaposlenega, ki ga hočemo poklicati,

sistem pa nas samodejno poveže naprej. Drug primer pa je lahko sistem, kjer povemo telefonsko številko, ki jo hočemo doseči. Tak sistem bi lahko razpoznaval med besedami *da*, *ne*, *popravek* in številkami (*nič, ena, ..., devet*).

6.2.1 Struktura modelov

Ideja uporabe HMM v razpoznavanju govora je ta, da vsaki besedi priredimo en model. HMM za besede v majhnem slovarju lahko gradimo na dva načina. Prvi je, da zgradimo najprej t.i. trifonske modele, ki ustrezajo vsakemu fonemu v besedi. Nato sestavimo model celotne besede tako, da zaporedoma združimo trifonske modele. Drug način, ki je uporabe predvsem pri zelo majhnem slovarju, je ta, da vsaki besedi priredimo popolnoma neodvisen model. Predstavili bomo slednjo metodo.

V prejšnjih poglavjih smo govorili o zaporedju opazovanj. Opazovanja v primeru razpoznavanja govora so zgoraj omenjeni vektorji značilk. Ti imajo običajno več deset dimenzij. Njihove porazdelitve pa modeliramo z večdimenzionalnimi Gaussovimi porazdelitvami.

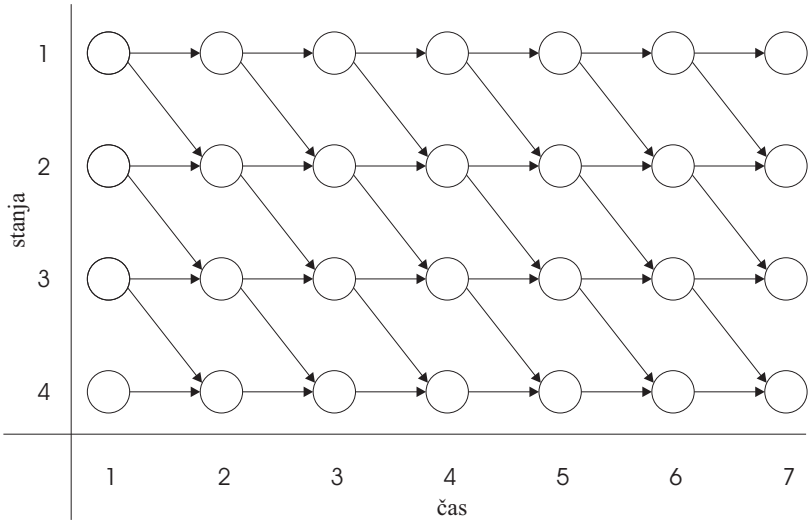
Modeli za razpoznavanje govora imajo levo-desno strukturo. Modele za besede oblikujemo s 15 do 30 stanji, ki dovoljujejo prehode iz stanja vase ali pa v naslednjo stanje. Takšen model je na sliki 6.2. Posamezna stanja predstavljajo značilnosti govora v posameznih trenutkih med izgovorjavo besed. Graf poravnave takšnega modela pa je na sliki 6.3.

Slika 6.2: Struktura HMM za modeliranje celih besed

6.2.2 Učenje

Pred učenjem modelov moramo zbrati učno množico. V njej so primeri izgovorjav besed, ki jih hočemo imeti v našem slovarju. Pri tem na splošno velja, da je bolje imeti večjo množico. Iz vseh enot v učni množici izločimo značilke. Besede so pri tem shranjene v takšni obliki, da je pred in za besedo le kratek šum ali pa tišina.

Drugi korak je priprava modelov. Stanja predstavljajo akustične značilnosti besed ob različnih delih besede. Prvo in zadnje stanje predstavljata pri tem šum, ki nastopa tik pred začetkom oz. tik po koncu besede. Ker se beseda vedno začne s tem šumom pred njo, moramo vektorju začetne verjetnosti določimo fiksno vrednost $\pi = [1, 0, ..., 0]^T$. Pred



Slika 6.3: Graf poravnave za levo-desni HMM

samim učenjem določimo tudi začetne vrednosti verjetnosti prehoda. To lahko storimo z različnimi vrednosti. S poskušanjem ugotovimo, katere se najbolje obnesejo.

Pri porazdelitvah izhodnih simbolov so pomembni dobri začetni približki. Te dobimo lahko z različnimi metodami, ki jih zaradi obsežnosti ne bomo podrobneje opisovali. Bralec jih lahko najde v [13].

Naslednji korak je, da izvedemo dejansko učenje z Baum-Welchevim algoritmom za več zaporedij opazovanj in zvezne porazdelitve, tako kot smo ga opisali v prejšnjih dveh poglavjih.

6.2.3 Razpoznavanje

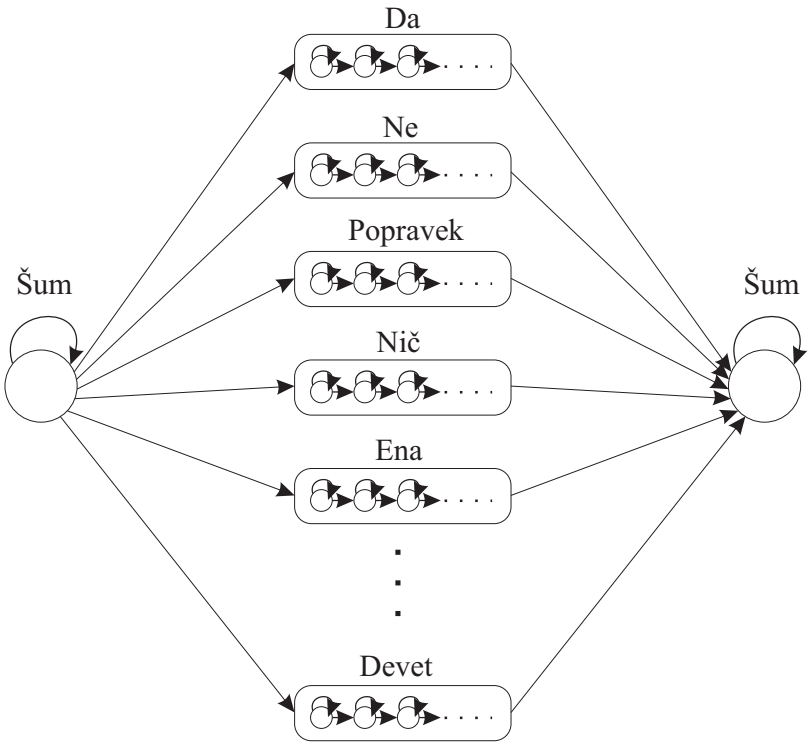
Za razpoznavanje uporabljamo algoritem naprej ali pa Viterbijev algoritem. Posamezni besedi w v slovarju lahko priredimo tudi a priori verjetnosti, da se pojavijo $P(w_i)$. Te verjetnosti lahko določimo statistično (kot v primeru na začetku poglavja), z nekimi pravili ali pa jih enostavno postavimo na enake vrednosti.

Imejmo slovar besed $\{w_1, w_1, ..., w_W\}$ s pripadajočimi modeli $\{\lambda_1, \lambda_2, ..., \lambda_W\}$. Kot rezultat razpoznavanja izberemo besedo, ki maksimira skupno verjetnost:

$$\hat{w} = \arg\max_{w_i} P(w_i)P(O|\lambda_i). \tag{6.3}$$

Algoritem za ocenjevanje lahko še nekoliko dodatno pospešimo s tem, da upoštevamo, kateri prehodi med stanji so levo-desnem modelu sploh dovoljeni.

Modele vseh besed lahko združimo tudi v skupni model, ki ga imenujemo slovnica. Primer takšnega modela, za razpoznavanje izoliranih besed z majhnim slovarjem je prikazan na sliki 6.4. V njem imamo dve stanji, ki predstavljata šum na začetku in na koncu besede.



Slika 6.4: Slovnica za razpoznavanje izoliranih besed

Dodajamo lahko tudi jezikovne modele, v katerih modeliramo verjetnosti besede glede na prejšnje besede:

$$P(w_i|w_1, w_2, ..., w_{i-1}). \tag{6.4}$$

Ta model imenujemo jezikovni model, ki ga običajno poenostavimo v t.i. m-gramski model:

$$P(w_i|w_{i-m+1}, w_{i-m+2}, ..., w_{i-1}), \tag{6.5}$$

kjer je verjetnost besede odvisna od $m-1$ prejšnjih besed. Ta model torej ustreza markovski verigi reda $m-1$. Samo učenje in uporaba teh modelov pa ni več s področja te diplomske naloge.

Literatura

[1] L. Baum, An inequality and associated maximization technique in statistical estimation for probalistic functions of Markov processes, Inequalities 3 (1972) str. 1-8.

[2] L. Baum, J. Eagon, An inequality with applications to statistical estimation for probalistic functions of Markov processes and to a model for ecology, American Mathematical Society Bulletin 73 (1967), str. 360–363.

[3] L. Baum, T. Petrie, Statistical inference for probabilistic functions of finite state Markov chains. Annals of Mathematicas Science 37 (1966) str. 1554–1563.

[4] L. Baum, T. Petrie, G. Soules, N. Weiss, A maximization technique occuring in the statistical analysis of probablistic functions of Markov chains, The Annals of Mathematical Statistics 41(1) (1970) str 164–171.

[5] L. Baum, G. Sell, Growth functions for transformations in manifolds, Pacific journal of Mathematics 27 (1968) str. 211-227.

[6] A. Hudoklin-Božič, *Stohastični procesi*, Moderna organizacija, Kranj, 1999.

[7] J. R. Deller, jr., J. G. Proakis, J. H. Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan Publishing Company, 1993.

[8] A.P. Dempster, N.M.Laird, D.B. Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, Journal of the Royal Statistical Society. Series B 39 (1977) str. 1–38.

[9] J. A. Gubner, *Probability and Random Processes for Electrical and Computer Engineers*, Cambridge University Press, 2006.

[10] G. A. Fink, *Markov Models for Pattern Recognition*, Springer, 2007.

[11] O. C. Ibe, *Markov Processes for Stochastic Modeling*, Elsevier Academic Press, 2009.

[12] Z. Kačič, *Komunikacija človek-stroj*, Fakulteta za elektrotehniko, računalništvo in informatiko, Maribor, 1995.

[13] L. Rabiner, B-H. Jaung, *Fundamentals of Speech Recognition*, PTR Prentice Hall, Englewood Cliffs, New Jersey, 1993.