LATVIJAS UNIVERSITĀTE

FIZIKAS UN MATEMĀTIKAS FAKULTĀTE

MATEMĀTIKAS NODAĻA

DISKRĒTIE SLĒPTIE MARKOVA MODEĻI UN TO PIELIETOJUMI

BAKALAURA DARBS

Autors: **Mārtiņš Eglītis**

Stud. apl. me08032

Darba vadītājs: doc. Dr.math. Jānis Valeinis

Rīga 2012

Anotācija

Šajā darbā tiks apskatīti mūsdienās lielu nozīmību un ievērību ieguvušie slēptie Markova modeļi (SMM), kurus plaši izmanto teksta un runas atpazīšanā, gēnu inženierijā, u.c. Neliels ieskats vēsturē iepazīstinās ar to izveides cēloņiem un sākotnējo pielietojumu, kā arī ar algoritmu aizsākumiem. Markova procesi un pseido piemēri izveidos pamatus slēpto Markova modeļu teorijai, tiks aprakstītas trīs SMM pamatproblēmas. Detalizētāka pamatproblēmu un to risināšanas algoritmu izpētei bija nepieciešams izveidot programmu. Visbeidzot tiks apskatīti arī praktiski piemēri, kas uzskatāmi parādīs SMM nozīmīgumu reālās dzīves situācijās.

Atslēgvārdi: Slēptie Markova modeļi (SMM), SMM *Forward*, *Backward*, Viterbi, Baum-Welch algoritmi, SMM pielietojuma piemēri.

Anotation

This work deals with important and noticable Hidden Markov Models (HMM) that are widely used in text and speech recognition, gene engeneering, e.t.c. A short insight into history will reveal the reasons for using HMM and the primary use as well as "cradle" of alghoritms underlying them. Markov process and pseudo examples will set the fundaments of the theory of Hidden Markov models; three basic problems will be formulated. More detailed research of the basic problems and alghorithms inspired me to write a script for solving them. Finally, real life examples are presented to focus the revelance of using HMM.

Keywords: Hidden Markov Models (HMM), HMM *Forward*, *Backward*, Viterbi, Baum-Welch algorithms, HMM applications.

Saturs

1. Ievads

Temats "Diskrētie slēptie Markova modeļi un to pielietojumi" tika izvēlēts, jo to dziļāka izpēte šķita ļoti interesanta gan teorētiskajā, gan arī praktiskajā ziņā. Uzņēmums, kurā brīvprātīgi strādāju, piedāvāja šādu tematu, jo tas tika uzskatīts par potenciālu ieguvumu darbībā un attīstībā.

Turpmāk apskatīsim un ar "SMM" sapratīsim diskrētos slēptos Markova modeļus. Šī darba mērķi un uzdevumi bija šādi:

- iepazīties ar teoriju, uz kuras balstās SMM;
- apskatīt algoritmus, kas veidoti uz šīs teorētiskās bāzes;
- rakstīt programmas kodus, kas, izmantojot algoritmus, risina pseido piemērus;
- apskatīt kā SMM tiek izmantoti reālām problēmām.

Darbā tiks apskatīta slēpto Markova modeļu teorija un pētītas trīs pamatproblēmas: novērtēšanas problēma, atšifrēšanas problēma un apmācības problēma. Katrai no šīm problēmām tiks izstrādāta datorprogramma, kas, izmantojot katras konkrētās problēmas risinājuma algoritmus, dos rezultātus pseido piemēriem. Darba nobeigumā SMM pielietojums tiks parādīts uz trīs konkrētiem piemēriem.

SMM tiek izmantoti, lai modelētu sistēmu, kas ir Markova process ar novērotajam slēptiem stāvokļiem. Tos var pielietot, piemēram, runas, rokrakstu, žestu atpazīšanā, signālu apstrādē, vajadzīgās informācijas iegūšanā no dokumentiem, mašīntulkošanā un gēnu noteikšanā.

Iesākumā tika vākti visdažādākie informācijas avoti: gan grāmatas, gan raksti, gan arī publikācijas (jāatzīmē, ka visi avoti ir ņemti no interneta). Apskatot iegūto, varēja secināt, ka to ir daudz, kas liecina par slēpto Markova modeļu aktualitāti un nozīmīgumu. Sīkāk apskatot un analizējot avotus, varēja ievērot, ka tie atšķiras gan ar apzīmējumiem, kas tika lietoti darba gaitā, gan arī pašas idejas izklāstu. Vecāka izdevuma gada grāmatās informācija bija gan neskaidri lasāma (izplūdusi druka, slikta skenēšanas kvalitāte, u.c.), gan arī smagnēja, uztverei bija nepieciešams daudz laika. Savukārt jaunākajos rakstos (internetā pieejamie raksti, semināru daļas, gandrīz visi pēc 2003. gada, lielākā daļa pēc 2008. gada), uz kuriem balstīts šis darbs, idejas nostādnes, apzīmējumu un piemēru sapratne bija daudz vieglāka un ātrāka. Slēpto Markova modeļu piemēri, kas arī atrodami interneta resursos, veicināja izpratni, kā arī gan salīdzinoši abstraktu, gan praktisku pielietojumu.

Paralēli informācijas avotu izpētei, tika rakstīts arī programmas kods, kas izpilda visus augstāk minētos algoritmus (*Forward*, *Backward*, Viterbi, Baum-Welch). Sākotnējais mērķis bija aprēķināt pseido piemēru rezultātus, bet vēlāk arī tika atrasts jauns mērķis:

padziļināt savas zināšanas programmēšanas valodā Python (versija 2.7), kuru plaši izmanto uzņēmums, kurā strādāju. Lai nu kā, bet programmēšana bija interesants process, kas sniedza gandarījumu par algoritmu precīzu darbību (piemēram, precīzām varbūtībām) un moderno ierīču izmantošanu slēptajos Markova modeļos. Pielikumā ir dots viss pseido piemēru aprēķināšanas kods, kas uzrakstīts programmēšanas valodā Python (versija 2.7).

Darba pirmajā nodaļā ir dotas vēsturiskas ziņas par SMM teoriju, kā arī ieskats Markova procesos. Otrajā nodaļā rakstīts par SMM, to trīs pamatproblēmām un atbilstošajiem algoritmiem. Trešajā nodaļā ir trīs reālu problēmu piemēri, kas parāda SMM pielietojumu. Ceturtajā nodaļā tiek izdarīti secinājumi par šī darba gaitu un iegūtajiem rezultātiem, bet piektajā nodaļā ir programmas kods.

1.1. Vēsture

Nodaļā 1.1 tiks izmantots [4] avots.

Šajā nodaļā tika ievāktas ziņas par autoru A.A. Markovu, slēpto Markova modeļu "pašiem pirmajiem soļiem", ar daudzajiem un SMM praktiskai pielietošanai nepieciešamajiem algoritmu aizsākumiem un to evolūciju, attīstoties zinātnei un tehnologijām.

Slēpto Markova modeļu vēsture sastāv no divām daļām. Pirmā daļa ir Markova procesu un Markova ķēžu vēsture, bet otrā - algoritmu vēsture, kas nepieciešami, lai attīstītu SMM līdz moderno pielietojamo zinātņu līmenim, izmantojot datoru vai citas elektroniskās ierīces.

Andrejs A.Markovs (dzimis 1856. gada 14. jūnijā, miris 1922. gada 20. jūlijā) bija izcils krievu matemātiķis, labi zināms ar darbiem par stohastiskajiem Markova procesiem. Viņa darbība vēlāk koncentrējās tieši ap Markova procesiem un Markova ķēdēm.

Jēdzieni "Markova procesi" un "Markova ķēdes" radās 1906. gadā, kad A.A.Markovs nāca klajā ar pirmajiem teorētiskajiem stohastisko procesu rezultātiem. Galīgu, saskaitāmu stāvokļu telpu vispārinājumu deva Kolmogorovs 1931. gadā. Markova ķēdes iesākumā tika saistītas ar Brauna daļiņu kustību, kā arī ar dažām fizikas problēmām. Matemātikā, galvenokārt varbūtību teorijā un matemātiskajā statistikā, Markova process var tikt uzskatīts par laikā mainīgu "parādību", kurai izpildās Markova īpašība (jeb nākotnes stāvokļa atkarība tikai no tagadnes stāvokļa).

Kopsolī ar spēcīgo datorzinātņu attīstību 1940. gados, pēc tādu slavenu zinātnieku kā John von Neuman, Turing un Konrad Zuse pētījumu rezultātiem, zinātnieki no visas pasaules aizsāka meklēt algoritmus, lai risinātu daudzas reālās dzīves problēmas.

EM (Expectation-Maximization) algoritms ir saistīts ar maksimālās ticamības (MT) algoritma vēsturi un tas radās 20. gadsimta sākumā. R.A.Fišers izmantoja, analizēja un rekomendēja pielietot MT algoritmu laikā no 1912. gada līdz 1922. gadam, kaut arī to iepriekš izmantoja Gauss, Laplass un citi matemātiķi. Izrādās, ka metode jau ir tikusi izmantota "speciālgadījumos", bet minētajā gadā tam jau bija izstrādāta vispārīgā teorija. EM algoritms tiek lietots statistikā, lai meklētu parametru maksimālos ticamības

novērtējumus. To bieži izmanto informācijas apkopošanai mašīnapmācībā un mašīnredzē.

Valodas apstrādē galvenais algoritms ir Baum-Welch (jeb *Forward-Backward*) algoritms. Faktiski, Baum-Welch algoritms ir atsevišķs gadījums vispārinātajam EM algoritmam. Tas tiek izmantots, lai uzlabotu slēptā Markova modeļa parametrus. 1970. gados L.E.Baum un L.R.Welch aprakstīja algoritmu savos darbos kā "maksimizācijas metodi, kas pielietojama Markova ķēžu varbūtību funkciju statistiskajā analīzē".

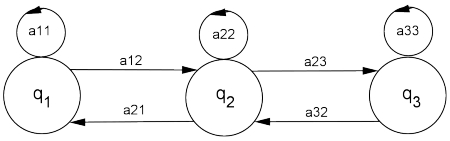
1967. gadā Andrew Viterbi ieviesa Viterbi algoritmu, kas atšifrē "īsto" signālu trokšņainos datos. Tas ir dinamiskās programmēšanas algoritms, lai atrastu visticamāko stāvokļu virkni, sauktu arī par Viterbi ceļu (vai vienkārši ceļu), kas iznākumā ar vislielāko varbūtību dod šos novērojumus. Pēdējo gadu laikā šim algoritmam ir atrasts pielietojums mobilajās tehnologijās un tīklos, iezvanpieejas modemos, satelītos, visuma komunikācijās un bezvadu lokālajos tīklos. Piemēram, dažos runa-teksts atpazīšanas ierīcēs akustiskais signāls tiek uzskatīts par novērojumu virkni, bet teksta virkne tiek uzskatīta par akustiskā signāla slēpto cēloni. Viterbi algoritms atrod visticamāko teksta virkni, kas veido akustisko signālu.

1.2. Markova process

Nodaļā 1.2 tiks izmantots [1] avots.

Pieņemsim par zināmu (vēlāk to definēsim), ka stohastiskajam procesam ir Markova īpašība, ja procesa nākotnes stāvokļu gadījuma varbūtību sadalījums ir atkarīgs vienīgi no tagadnes stāvokļiem. Process ar šādu īpašību tiek saukts par Markova procesu.

Bieži jēdziens "Markova ķēde" tiek lietots, lai apzīmētu Markova procesu ar diskrētu (galīgu vai saskaitāmu) stāvokļu kopu. Parasti Markova ķēde ir definēta diskrētai laika kopai, taču daži autori izmanto šādu pašu terminologiju, apzīmējot nepārtrauktas laika vērtības.



1. att. Markova ķēdes piemēra vizualizācija

Pieņemsim, ka stohastiska procesa stāvokļu kopa S ir ar garumu N un novērojumu kopa V ir ar garumu M

$$S = (s_1, ..., s_N)\,,$$

$$V = (v_1, ..., v_M)\,.$$

Pieņemsim Q kā fiksētu stāvokļu virkni ar garumu T un attiecīgos fiksētos novērojumus O arī ar garumu T

$$Q = q_1, ..., q_T,$$

$$O = o_1, ..., o_T.$$

Definīcija 1.

Markova īpašība ir spēkā un procesu sauc par Markova procesu, ja izpildās nosacījums

$$P\left(q_t \mid q_{t-1}, q_{t-2}, ..., q_1\right) = P\left(q_t \mid q_{t-1}\right), \, 2 \leqslant t \leqslant T.$$

Šo var interpretēt šādi: varbūtība

$$P\left(q_t \mid q_{t-1}, q_{t-2}, ..., q_1\right) = P\left(q_t \mid q_{t-1}\right), \, 2 \leqslant t \leqslant T$$

nav atkarīga no neviena pagātnes stāvokļa $q_{t-1}, q_{t-2}, ..., q_1$. Tātad, nākotnes stāvoklis q_{t+1} nav atkarīgs no pagātnes stāvokļiem $q_{t-1}, q_{t-2}, ..., q_1$, bet gan tikai no tagadnes stāvokļa q_t.

Definīcija 2.

Markova ķēdei izpildās šāda īpašība

$$P(q_1, q_2, ..., q_{t-1}, q_t) = P(q_1) * P(q_2 \mid q_1) * \cdots * P(q_t \mid q_{t-1}), \, 2 \leqslant t \leqslant T.$$

Piemērs 1.

Piemērā 1 tiks izmantots [6] avots.

Izmantosim tā dēvēto Urnu uzdevumu, bet šoreiz apskatīsim tikai pašas urnas. Ir dotas urnas s_1, s_2, s_3. Pieņemsim, ka s_1-"bronzas urna", s_2-"sudraba urna", s_3-"zelta urna", stāvokļu pārejas varbūtības ir uzdotas matricā

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 0.3 & 0.7 & 0 \\ 0.7 & 0.1 & 0.2 \\ 0 & 0.5 & 0.5 \end{pmatrix},$$

kur (a_{ij}), $1 \leqslant i, j \leqslant N$, ir varbūtība, ka process laika momentā t atrodas stāvoklī s_j, ja laikā $t-1$ process atradās stāvoklī s_i.

$$a_{ij} = P\left(q_t = s_j \mid q_{t-1} = s_i\right), \, 1 \leqslant i, j \leqslant N, \, 2 \leqslant t \leqslant T,$$

šeit q_t apzīmē pašreizējo stāvokli.

Ja koeficients (a_{ij}) ir nulle, tad pāreja no s_i uz s_j nav iespējama. Stāvokļu pārejas varbūtībām ir jāapmierina nevienādības

$$a_{ij} \geqslant 0,\ 1 \leqslant i, j \leqslant N,$$

$$\sum_{j=1}^{N} a_{ij} = 1,\ 1 \leqslant i \leqslant N.$$

Pieņemsim, ka stāvokļu sākuma varbūtības $\pi = (\pi_i)\,,\ 1 \leqslant i \leqslant N$, kur

$$\pi_i = P\left(q_1 = s_i\right),\ 1 \leqslant i \leqslant N,$$

ir

$$\pi = (0.7, 0.2, 0.1)\,.$$

Mēs vēlamies aprēķināt varbūtību fiksētai stāvokļu virknei: $q_1 =$ "zelta urna", $q_2 =$ "zelta urna", $q_3 =$ "sudraba urna", $q_4 =$ "bronzas urna". Markova ķēdes īpašība

$$\begin{aligned} P\left(q_1, q_2, q_3, q_4\right) &= P\left(q_1, q_2, q_3\right) P\left(q_4 \mid q_1, q_2, q_3\right) = \\ &= \ldots = P\left(q_1\right) P\left(q_2 \mid q_1\right) P\left(q_3 \mid q_2\right) P\left(q_4 \mid q_3\right) \end{aligned}$$

mums atļauj atrast šo varbūtību. Tad

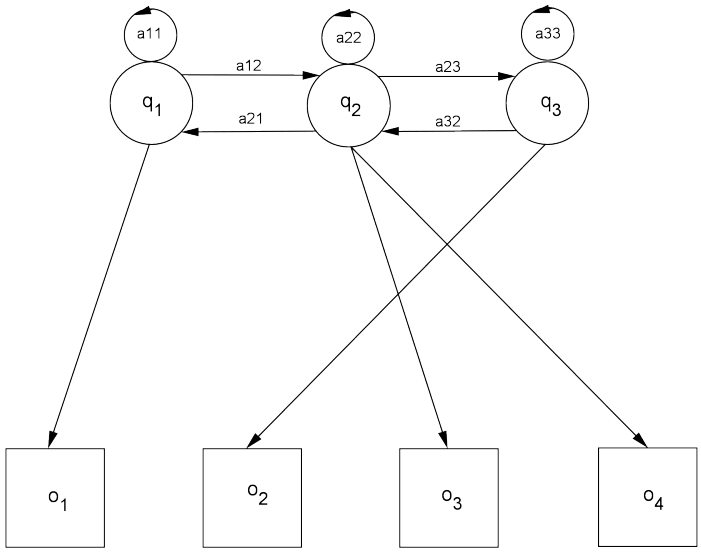
$$\begin{aligned} P\left('zelta\,urna', 'zelta\,urna', 'sudraba\,urna', 'bronzas\,urna'\right) &= \\ = 0.1 * 0.5 * 0.5 * 0.7 = 0.0175 \qquad\qquad & . \end{aligned}$$

2. Slēptais Markova modelis

Nodaļā 2 tiks izmantoti [1, 3, 4] avoti.

Par matemātisko modeli tiek uzskatīts sistēmas apraksts, izmantojot matemātiskos jēdzienus un valodu. Parastajā (atklātajā) Markova modelī stāvokļi novērotājam ir redzami atklātā veidā, tāpēc arī stāvokļu pārejas varbūtības ir vienīgie parametri.

Vizualizēsim atklāto Markova modeli ar 3 stāvokļiem un 4 novērojumiem, kur stāvokļi un to dotie novērojumi ir brīvi izvēlēti.



2. att. Atklātā Markova modeļa piemēra vizualizācija

Komentēsim 2. attēlu: stāvokļi s_1, s_2, s_3 dod attiecīgi novērojumus o_1, o_3, o_4, o_2. Novērotājam ir precīzi zināms, kurš stāvoklis ir devis konkrēto novērojumu. Faktiski, tas ir speciālgadījums slēptajam Markova modelim. Šeit varbūtība stāvoklī iegūt noteikto novērojumu ir vai nu 1 vai arī 0 un katru novērojumu dod tikai viens stāvoklis. Tomēr šādus gadījumus turpmāk neapskatīsim.

Slēptajā Markova modelī stāvokļi novērotājam nav tieši redzami, bet ir redzams novērojums, kas atkarīgs no stāvokļa. Katram stāvoklim ir varbūtību sadalījums pār iespējamajiem novērojumiem. Tādējādi novērojumu virkne, ko izveidojis SMM, sniedz mums informāciju par stāvokļu virkni. Jāievēro, ka vārds "slēptais" attiecas uz stāvokļu virkni, kuru izmanto šis modelis. Pat ja modeļa parametri (stāvokļu kopa, stāvokļu sākuma varbūtības, stāvokļu pārejas varbūtības, novērojumu varbūtības) ir precīzi zināmi, modelis joprojām ir "slēptais".

Slēptais Markova modelis sastāv no diviem stohastiskiem procesiem. Pirmais stohastiskais process ir Markova ķēde, kuru raksturo stāvokļi un stāvokļu pārejas varbūtības. Ķēdes stāvokļi nav redzami, tātad apslēpti. Otrais stohastiskais process katrā laika momentā veido novērojumu. Ir svarīgi atcerēties, ka, definējot slēpto Markova modeli, vārds "slēptais" attiecas uz Markova ķēdes stāvokļiem, bet ne uz modeļa parametriem.

Var uzskatīt, ka tas ir divkāršs stohastisks process ar šādiem diviem aspektiem:

- Pirmais stohastiskais process (jeb arī Markova ķēde) ir galīga stāvokļu kopa, kur katrs no tiem ir saistīts ar daudzdimensionālu varbūtību sadalījumu. Pārejas starp dažādiem stāvokļiem nosaka varbūtību kopa, kur šīs varbūtības sauc par stāvokļu pārejas varbūtībām.
- Otrajā stohastiskajā procesā katrs no stāvokļiem dod iznākumu jeb novērojumu. Tā kā mēs analizēsim tikai to, ko mēs iegūstam (neredzot, no kura stāvokļa), tad stāvokļi ir slēpti novērotājam.

Katrs SMM ir definēts ar stāvokļiem, stāvokļu pārejas varbūtībām, novērojumu varbūtībām un stāvokļu sākuma varbūtībām.

Definīcija 3.

Definēsim SMM:

1. Dota stāvokļu kopa $S = (s_1, ..., s_N)$ ar garumu N.
2. Dota katra stāvokļa novērojumu kopa $V = (v_1, ..., v_M)$ ar garumu M.
3. Dots stāvokļu pārejas varbūtību sadalījums $A = (a_{ij})$, kur a_{ij} ir varbūtība, ka process laika momentā t atrodas stāvoklī s_j ir dots, kad laikā $t-1$ process atrodas stāvoklī s_i. Ja koeficients a_{ij} ir nulle, tad pāreja no s_i uz s_j nav iespējama.

$$a_{ij} = P\left(q_t = s_j \mid q_{t-1} = s_i\right), \, 1 \leqslant i, j \leqslant N, \, 2 \leqslant t \leqslant T,$$

šeit q_t apzīmē pašreizējo stāvokli. Stāvokļu pārejas varbūtībām ir jāapmierina nevienādības

$$a_{ij} \geqslant 0, \, 1 \leqslant i, j \leqslant N$$

un

$$\sum_{j=1}^{N} a_{ij} = 1, \, 1 \leqslant i \leqslant N.$$

4. Dota katra stāvokļa novērojumu matrica $B = (b_i(k))$, kur $b_i(k)$ ir varbūtība k-tajam novērojumam atrasties stāvoklī i, neatkarīgs no laika t

$$B = (b_i(k)), \, b_i(k) = P\left(o_t = v_k \mid q_t = s_i\right), \, 1 \leqslant i \leqslant N, \, 1 \leqslant k \leqslant M, \, 1 \leqslant t \leqslant T,$$

kur v_k apzīmē k-to novērojumu, bet o_t apzīmē pašreizējo parametru vektoru. Ir jābūt apmierinātām šādām varbūtībām

$$b_i(k) \geqslant 0, \, 1 \leqslant i \leqslant N, \, 1 \leqslant k \leqslant M,$$

$$\sum_{k=1}^{M} b_i(k) = 1,\ 1 \leqslant i \leqslant N.$$

5. Dotas stāvokļu sākuma varbūtības $\pi = (\pi_i)$, kur π_i ir varbūtība, ka process laikā $t = 1$ atrodas stāvoklī s_i ar nosacījumu

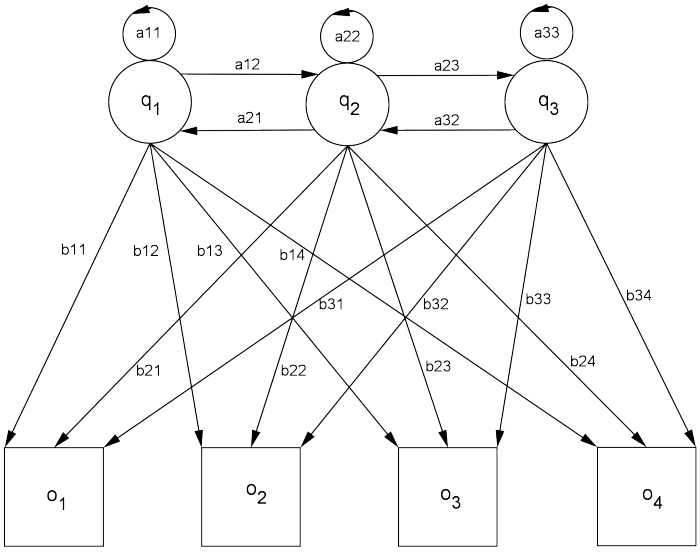
$$\pi_i = P\left(q_1 = s_i\right),\ 1 \leqslant i \leqslant N.$$

Definējot slēpto Markova modeli, ļoti svarīgi ir precizēt vai modelis būs diskrēts, nepārtraukts vai jaukta veida. Lietosim šādu pierakstu

$$\lambda = (A, B, \pi),$$

lai apzīmētu diskrēto SMM, tātad ar diskrētiem varbūtību sadalījumiem. Jāatgādina, ka nepārtrauktie slēptie Markova modeļi šeit netiks apskatīti.

Šoreiz vizualizēsim slēpto Markova modeli ar 3 stāvokļiem un 4 novērojumiem.



3. att. Slēptā Markova modeļa vizualizācija

Piemērs 2.

Piemērā 2 tiks izmantots [6] avots.

Tagad pieņemsim, ka katrā no urnām s_1, s_2, s_3 var izvēlēties lodītes v_1, v_2, v_3, v_4, kuru vinnesti ir attiecīgi "5 Ls", "10 Ls", "100 Ls" un "500 Ls". Atšķirībā no atklātā Markova modeļa, mums nav tieši zināms, kurš stāvoklis dos attiecīgo novērojumu. Šie stāvokļi mums ir doti ar noteiktām varbūtībām. Atcerēsimies stāvokļu pārejas varbūtības, kas uzdotas matricā

$$A = \begin{pmatrix} 0.3 & 0.6 & 0.1 \\ 0.7 & 0.1 & 0.2 \\ 0.5 & 0.4 & 0.1 \end{pmatrix}.$$

Uzdosim novērojumu varbūtības $B = (b_i(k))$, kur $1 \leqslant i \leqslant N$, $1 \leqslant k \leqslant M$. Tātad

$$\begin{array}{llll} b_1(o_1) = 0.1, & b_1(o_2) = 0.5, & b_1(o_3) = 0.2, & b_1(o_4) = 0.2 \\ b_2(o_1) = 0.6, & b_2(o_2) = 0.2, & b_2(o_3) = 0.1, & b_2(o_4) = 0.1 \\ b_3(o_1) = 0.3, & b_3(o_2) = 0.3, & b_3(o_3) = 0.3, & b_3(o_4) = 0.1 \end{array}.$$

Uzskatāmības labad, uzrakstīsim to matricā

$$B = \begin{pmatrix} 0.1 & 0.5 & 0.2 & 0.2 \\ 0.6 & 0.2 & 0.1 & 0.1 \\ 0.3 & 0.3 & 0.3 & 0.1 \end{pmatrix}.$$

Visbeidzot uzdosim stāvokļu sākuma varbūtības matricā

$$\pi = (0.7, 0.2, 0.1).$$

Saglabājot jau pirmajā piemērā pieņemto urnu secību: $q_1 =$ "zelta urna", $q_2 =$ "zelta urna", $q_3 =$ "sudraba urna", $q_4 =$ "bronzas urna", aprēķināsim varbūtību kāda ir izvilkt šādas lodītes no attiecīgajām urnām : $o_1 =$ "5 Ls", $o_2 =$ "100 Ls", $o_3 =$ "5 Ls" un $o_4 =$ "500 Ls" (jeb, ka no "zelta urnas" tiks izvilkta "5 Ls" lodīte, u.t.t.)

$$\begin{gathered} P(o_1, o_2, o_3, o_4 \mid q_1, q_2, q_3, q_4) = \\ = P(q_1)\, P(o_1 \mid q_1)\, P(q_2 \mid q_1)\, P(o_2 \mid q_2)\, P(q_3 \mid q_2)\, P(o_3 \mid q_3)\, P(q_4 \mid q_3)\, P(o_4 \mid q_4) = \\ = 0.1 * 0.3 * 0.1 * 0.3 * 0.4 * 0.6 * 0.7 * 0.2 = 0.00003024. \end{gathered}$$

Iepazīstināsim ar trīs fundamentālām SMM problēmām, kuru analīze un sīkāks apskats atrodams lielākajā daļā no ievāktās informācijas avotiem. Dosim tikai necilu aprakstu par katru no tām, jo turpmākās trīs apakšnodaļas nodarbosies ar to padziļināti.

Slēpto Markova modeļu trīs pamatproblēmas ir šādas:

- Novērtēšanas problēma: kāda ir varbūtība, ka fiksētie novērojumi $O = o_1, o_2, ..., o_T$ tiek iegūti ar modeli $P(O \mid \lambda)$, izmantojot doto slēpto Markova modeli λ ?
- Atšifrēšanas problēma: kāda ir visticamākā stāvokļu virkne dotajā SMM λ, kas deva stāvokļu virkni $O = o_1, o_2, ..., o_T$?
- Apmācības problēma: kā ir jāpiemeklē modeļa parametri (A, B, π), lai maksimizētu $P(O \mid \lambda)$, ja gan modelis λ, gan arī novērojumu virkne $O = o_1, o_2, ..., o_T$ ir dota ?

Novērtēšanas problēma var tikt lietota atsevišķu vārdu atpazīšanai. Atšifrēšanas problēma ir saistīta ar nepārtrauktu atpazīšanu un segmentēšanu. Apmācības problēma ir jāizmanto, ja ir vajadzība "uztrenēt" SMM secīgu atpazīšanas uzdevumu veikšanai.

2.1. Novērtēšanas problēma

Nodaļā 2.1 tiks izmantoti [3, 4] avoti.

Novērtēšanas problēmas uzdevums ir atrast varbūtību novērojumu virknei O. Šo varbūtību, faktiski, var meklēt gan ar *Forward*, gan ar *Backward* algoritmu, un varbūtības abos gadījumos būs vienādas. Abi algoritmi ir salīdzinoši vienkārši, tos izmantojot tiek samazināts arī izpildāmo darbību skaits, tā uzlabojot izpildes sarežģītību un laiku. Vizualizācija un pseido piemērs ļaus skaidrāk izprast šo pamatproblēmu.

Ja dots SMM $\lambda = (A, B, \pi)$ un novērojumu virkne $O = o_1, o_2, ..., o_T$, tad kā ir iespējams aprēķināt $P(O = o_1, o_2..., o_T \mid \lambda)$ jeb varbūtību fiksētai novērojumu virknei, kuru dod modelis λ ? Šo problēmu var uzskatīt kā mēgināājumu novērtēt cik labi modelis "paredz" doto novērojumu virkni, tādējādi atļaujot izvēlēties vislabāko modeli no modeļu kopas.

Varbūtība novērojumiem O pie fiksētas stāvokļu virknes Q ir

$$P(O \mid Q, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda) = b_{q_1}(o_1) b_{q_2}(o_2) \cdots b_{q_T}(o_T),$$

un varbūtība stāvokļu virknei ir

$$P(Q \mid \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T}.$$

Tātad doto novērojumu varbūtību, ko dod modelis λ, var aprēķināt šādi

$$P(O \mid \lambda) = \sum_{Q} P(O \mid Q, \lambda) P(Q \mid \lambda) = \sum_{q_1 \cdots q_T} \pi_{q_1} a_{q_1 q_2} b_{q_1}(o_1) \cdots a_{q_{T-1} q_T} b_{q_T}(o_T).$$

Šis rezultāts atļauj aprēķināt varbūtību novērojumu virknei O, bet tiešs aprēķins būtu ļoti neefektīvs (darbību skaits būtu $2TN^T$). Vērtīgāk būtu atpazīt daudzus liekos aprēķinus, kas rodas tieši rēķinot $P(O, \lambda)$, un tieši tāpēc aprēķinu sagrupēšana var atvieglot uzdevumu.

Saglabāsim informāciju režģī katrā laika solī, aprēķinot noglabāto informāciju katrā stāvoklī kā summu pa visiem stāvokļiem iepriekšējā laika solī. $\alpha_t(i)$ ir varbūtība fiksētai stāvokļu virknei $o_1, o_2, ..., o_t$, stāvoklim s_i un laikā t.

Definēsim "turpejošo" (turpmāk tomēr "labskanīguma" dēļ saglabāsim angļu "*Forward*") mainīgo

$$\alpha_t(i) = P(o_1, o_2, ..., o_T, q_t = s_i \mid \lambda), \; 1 \leqslant i \leqslant N, \; 1 \leqslant t \leqslant T.$$

Lai varētu izlauzties caur režģim, kas sastāv no $\alpha_t(i)$ vērtībām, jāiterē T reizes. Iegūstot beidzamo režga kolonnu un summējot tās elementus, mēs iegūstam novērojuma virknes varbūtību. Šis algoritms tiek saukts par *Forward* algoritmu.

Definīcija 4.
Definēsim *Forward* algoritmu:

1. Priekšnosacījumi
$$\alpha_1(j) = \pi_j b_j\left(o_1\right), \, 1 \leqslant j \leqslant N.$$

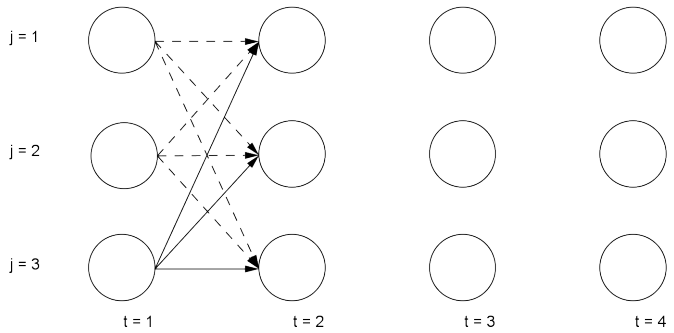
2. Indukcija
$$\alpha_{t+1}\left(j\right) = b_j\left(o_{t+1}\right) \sum_{i=1}^{N} \alpha_t\left(i\right) a_{ij}, \, 1 \leqslant t \leqslant T-1, \, 1 \leqslant j \leqslant N.$$

3. Iznākums
$$P\left(O \mid \lambda\right) = \sum_{j=1}^{N} \alpha_T\left(j\right).$$

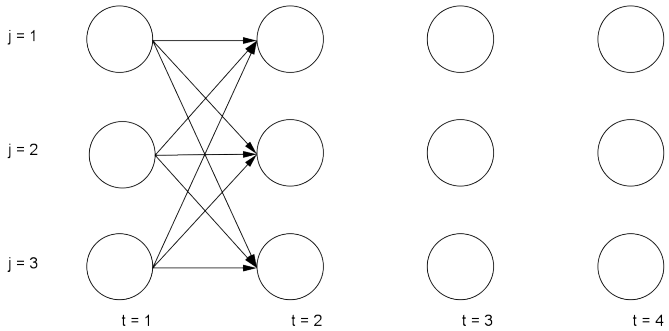
Indukcijas solis ir nozīmīgākā *Forward* algoritma daļa, tas ir attēlots 4. attēlā. Katram stāvoklim s_i mainīgajā $\alpha_i(t)$ tiek saglabāta varbūtība nokļūšanai šajā stāvoklī ar fiksēto novērojumu virkni līdz laikam t.

Tātad, apkopojot darbības un iegūstot $\alpha_t\left(i\right)$ vērtības, *Forward* algoritms samazina aprēķinu sarežģītību no $2TN^T$ uz N^2T.

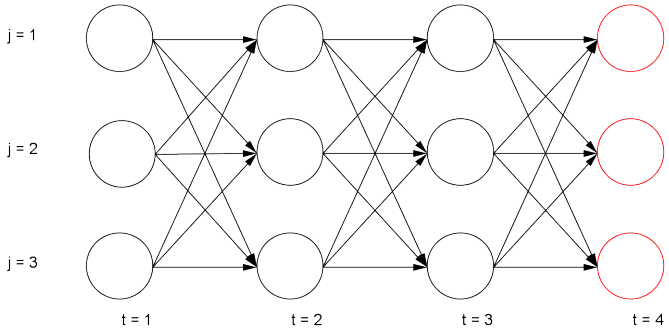
Vizualizēsim *Forward* algoritmu, izmantojot slēpto Markova modeli ar 3 stāvokļiem un 4 novērojumiem.



(a) *Forward* algoritma priekšnosacījumi



(b) *Forward* algoritma indukcija



(c) *Forward* algoritma iznākums

4. att. *Forward* algoritma vizualizācija

Var arī definēt "atpakaļnākošo" (turpmāk tekstā "*Backward*") algoritmu, kas ir ļoti līdzīgs *Forward* algoritmam, tikai ar *Backward* mainīgo

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T \mid q_t = s_i, \lambda), \; 1 \leqslant i \leqslant N, \; 1 \leqslant t \leqslant T-1,$$

kas ir fiksētas stāvokļu virknes varbūtība no $t+1$ līdz T, kur s_i ir pašreizējais stāvoklis. *Backward* mainīgo $\beta_t(i)$ var aprēķināt arī izmantojot rekursiju

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j)\, a_{ij} b_j(o_{t+1}), \; 1 \leqslant i \leqslant N, \; 1 \leqslant t \leqslant T-1,$$

kur $\beta_T(i) = 1$, $1 \leqslant i \leqslant N$.

Definīcija 5.

Definēsim *Backward* algoritmu:

1. Priekšnosacījumi
$$\beta_T(i) = 1,\ 1 \leqslant i \leqslant N.$$

2. Indukcija
$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j)\, a_{ij} b_j(o_{t+1}),\ 1 \leqslant i \leqslant N, 1 \leqslant t \leqslant T-1.$$

3. Iznākums
$$P(O \mid \lambda) = \sum_{j=1}^{N} \pi_j b_j(o_1)\, \beta_1(j).$$

Var redzēt, ka

$$P(O, q_t = s_i \mid \lambda) = \alpha_t(i)\, \beta_t(i),\ 1 \leqslant i \leqslant N, 1 \leqslant t \leqslant T,$$

tātad pastāv vairāki veidi kā aprēķināt $P(O \mid \lambda)$, izmantojot *Forward* vai *Backward* mainīgo, vai izmantojot tos abus

$$P(O \mid \lambda) = \sum_{i=1}^{N} P(O, q_t = s_i \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i)\, \beta_t(i),\ 1 \leqslant t \leqslant T.$$

Šis vienādojums var būt ļoti noderīgs, it īpaši izsakot formulas, kas nepieciešamas gradientu algoritmiem.

Piemērs 3.

Faktiski, novērtēšanas problēmai gan *Forward*, gan *Backward* algoritmiem ir jāsniedz vienādi rezultāti, tas ir, vienāda varbūtība, ka tiks izvēlēta notikumu virkne O.

Apskatīsim *Forward* algoritmu. Izmantosim jau iepriekšējos pseido piemēros lietoto stāvokļu sākuma varbūtību matricu π, stāvokļu pārejas matricu A un novērojumu varbūtību matricu B. Šoreiz mainīsim tikai novērojumu virkni, kas būs šāda

$$O = (o_1, o_2, o_3, o_4, o_5) = ('Ls100', 'Ls5', 'Ls500', 'Ls500', 'Ls100').$$

Izrakstīsim *Forward* algoritma locekļus $\alpha_t(i)$, kur $1 \leqslant t \leqslant T$ un $1 \leqslant i \leqslant N$.

$$\begin{array}{lllll} \alpha_1(o_1) = 0.14, & \alpha_1(o_2) = 0.0071, & \alpha_1(o_3) = 0.009288, & \alpha_1(o_4) = 0.0008655, & \alpha_1(o_5) = 0.0001521 \\ \alpha_2(o_1) = 0.02, & \alpha_2(o_2) = 0.0588, & \alpha_2(o_3) = 0.001266, & \alpha_2(o_4) = 0.0006223, & \alpha_2(o_5) = 0.0000634 \\ \alpha_3(o_1) = 0.03, & \alpha_3(o_2) = 0.0063, & \alpha_3(o_3) = 0.00131, & \alpha_3(o_4) = 0.0001313, & \alpha_3(o_5) = 0.0000672 \end{array}.$$

Varbūtība $P('Ls100',' Ls5',' Ls500',' Ls500',' Ls100')$, ka tiks izvilkti konkrētie novērojumi ir pēdējās kolonnas elementu summa

$$P('Ls100',' Ls5',' Ls500',' Ls500',' Ls100') =$$
$$0.0001521 + 0.0000634 + 0.0000672 = 0.0002827 \approx 0.0003 \quad .$$

2.2. Atšifrēšanas problēma

Nodaļā 2.2 tiks izmantoti [3, 4, 2] avoti.

Atšifrēšanas problēma atbild uz jautājumu: "ja ir zināma iznākumu virkne O, tad kāds ir visticamākais ceļš (stāvokļu virkne), kas man to dotu?". Viterbi algoritms, kas tika atzīts par vienu no labākajiem 20. gs. algoritmiem, tiek lietots, lai to atrisinātu. Vēlreiz jau pseido piemērs ļaus saskatīt ļoti būtisko atšķirību starp ar iepriekš aprakstītajiem algoritmiem iegūtajām varbūtībām.

Dota novērojumu virkne $O = o_1, o_2, ..., o_T$ un SMM $\lambda = (A, B, \pi)$. Kāda ir visticamākā stāvokļu virkne dotajā modelī, kas deva šo novērojumu virkni?

Vienā no veidiem var atrast ticamus stāvokļus q_t un izveidot no tiem virkni. Bet šī metode parasti nesniedz precīzus rezultātus (stāvokļu virkni), tāpēc jāizmanto cita metode- tā saucamais Viterbi algoritms. Izmantojot Viterbi algoritmu, tiek atrasta stāvokļu virkne ar maksimālo ticamību.

Ieviesīsim jaunu mainīgo, kas dos vislielāko varbūtību, fiksētai novērojumu virknei būt iegūtai no stāvokļu virknes (līdz laika momentam t un pie pašreizējā stāvokļa s_j)

$$\delta_t(j) = \max_{q_1, q_2, ..., q_{t-1}} P(q_1, q_2, ..., q_{t-1}, q_t = s_j, o_1, o_2, ..., o_{t-1} \mid \lambda).$$

Secinām, ka

$$\delta_{t+1}(j) = b_j(o_{t+1}) \max_{1 \leqslant i \leqslant N} \delta_t(i) a_{ij},\ 1 \leqslant j \leqslant N, 1 \leqslant t \leqslant T-1,$$

kur $\delta_1(j) = \pi_j b_j(o_1)$, $1 \leqslant j \leqslant N$. Tātad, lai aprēķinātu ticamāko stāvokļu virkni, tiek sākts ar $\delta_T(j)$, $1 \leqslant j \leqslant N$. Maksimuma meklēšanas operācijā mēs vienmēr atzīmējam "uzvarējušo stāvokli". Rezultāts ir stāvoklis $j*$, kur

$$j* = arg \max_{1 \leqslant j \leqslant N} \delta_T(j).$$

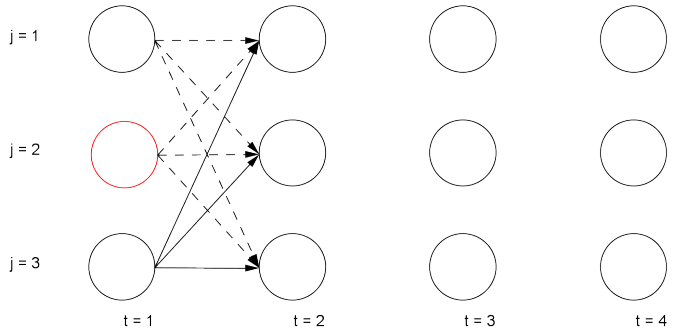
Mēs sākam no šī stāvokļa un "atmuguriski" atrodam stāvokļu virkni kā atzīmju virkni, ko uzrāda katrs "uzvarējušais stāvoklis". Tādējādi mēs iegūstam nepieciešamo stāvokļu kopu.

Jāatceras, ka atsevišķās varbūtības atšķiras no tām, ko atrod *Forward* un *Backward* algoritmi, jo tās, faktiski, ir varbūtības visticamākajam ceļam līdz stāvoklim laika momentā t, bet nevis novērojumu virknes varbūtība. Katram stāvoklim laika momentā $t = T$ būs sava varbūtība un savs labākais ceļš. Kopējo labāko ceļu var atrast izvēloties stāvokli,

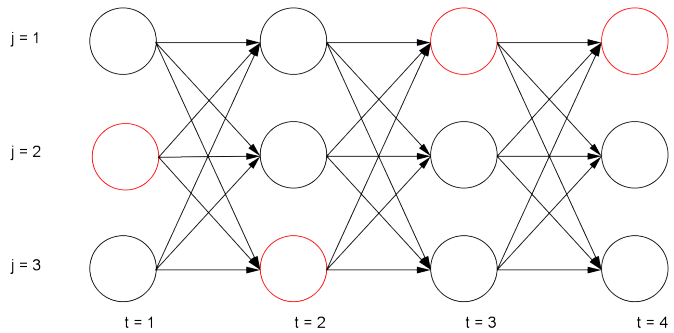
kuram ir vislielākā varbūtība, un tā labāko ceļu. Viterbi algoritmam, kas paredzēts lai atrastu labāko stāvokļu virkni, ir divas priekšrocības:

- izmantojot rekursiju, samazinās mašīnlaiks. Ieguvums ir analogs *Forward* un *Backward* algoritma ieguvumam salīdzinājumā pret "tiešo" aprēķinu, tātad no $2TN^T$ uz N^2T,
- algoritms nodrošina labu dot novērojumu interpretāciju.

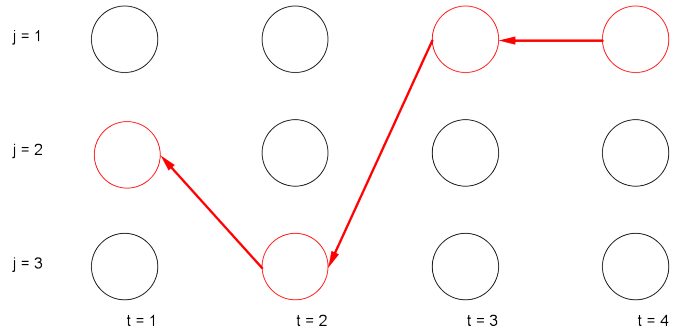
Šo algoritmu var vizualizēt kā meklēšanu grafā, kura virsotnes ir SMM stāvokļi s_j, $1 \leqslant j \leqslant N$, katrā laika momentā t, kur $1 \leqslant t \leqslant T$. Vizualizēsim Viterbi algoritmu, izmantojot slēpto Markova modeli ar 3 stāvokļiem un 4 novērojumiem.



(a) Viterbi algoritma priekšnosacījumi



(b) Viterbi algoritma indukcija



(c) Viterbi algoritma iznākums

5. att. Viterbi algoritma vizualizācija

Piemērs 4.

Piemērā 4 tiks izmantots [2] avots.

Apskatīsim Viterbi algoritmu. Nevienu no SMM modeļa $\lambda = (A, B, \pi)$ parametriem nemainīsim, saglabāsim arī iepriekšējā piemēra novērojumus

$$O = (o_1, o_2, o_3, o_4, o_5) = ('Ls100', 'Ls5', 'Ls500', 'Ls500', 'Ls100').$$

Izrakstīsim Viterbi algoritma locekļus $\delta_t(i)$, kur $1 \leqslant t \leqslant T$ un $1 \leqslant j \leqslant N$

$$\begin{array}{lllll}
\boldsymbol{\delta_1(o_1) = 0.14}, & \delta_1(o_2) = 0.0042, & \boldsymbol{\delta_1(o_3) = 0.007056}, & \delta_1(o_4) = 0.0004233, & \boldsymbol{\delta_1(o_5) = 0.0000592} \\
\delta_2(o_1) = 0.02, & \boldsymbol{\delta_2(o_2) = 0.0504}, & \delta_2(o_3) = 0.000504, & \boldsymbol{\delta_2(o_4) = 0.0004233}, & \delta_2(o_5) = 0.0000254 \\
\delta_3(o_1) = 0.03, & \delta_3(o_2) = 0.0042, & \delta_3(o_3) = 0.001008, & \delta_3(o_4) = 0.0000705, & \delta_3(o_5) = 0.0000254
\end{array}.$$

Tātad Viterbi algoritms deva atbildi, ka ar varbūtību $0.0000592 \approx 0.00006$ visticamākais ceļš (stāvokļu virkne) ir "Bronzas urna", "Sudraba urna","Bronzas urna", "Sudraba urna","Bronzas urna" (izcelts treknrakstā). Jāatceras, ka Viterbi algoritma rezultātā dotā varbūtība ir varbūtība, ka atrastais ceļš dos novērojumu virkni O, un šī varbūtība ir vislielākā pār visiem citiem ceļiem (nedrīkst sajaukt ar *Forward* un *Backward* algoritma iegūtajām varbūtībām).

2.3. Apmācības problēma

Nodaļā 2.3 tiks izmantoti [3, 4] avoti.

Beidzamās un sarežģītākās pamatproblēmas mērķis ir novērtēt un uzlabot slēpto Markova modeļu parametrus: stāvokļu sākuma varbūtības, stāvokļu pārejas varbūtības un novērojumu varbūtības. To var paveikt, izmantojot Baum-Welch algoritmu jeb tā saukto *Forward-Backward* algoritmu, kas ir ļoti jūtīgs pret parametru vērtībām un novērojumu skaitu un biežumu. Uzlabojumi SMM tiks demonstrēti, izmantojot pseido piemēru.

Kā var piemērot SMM parametrus, lai dotajam piemēram atrastu vislabāko modeli, kas atspoguļo doto novērojumu kopu (treniņa kopu) ? Atkarībā no piemēra, "daudzums", ko vajadzētu optimizēt apmācības procesā atšķiras. Tāpēc apmācībai ir dažādi optimizācijas kritēriji.

Literatūrā ir atrodami divi galvenie optimizācijas kritēriji:

- Maksimālās ticamības (MT) kritērijs.
- Maksimālās kopīgās informācijas (MKI) kritērijs.

Tagad aplūkosim sīkāk MT kritēriju un tā Baum-Welch algoritmu.

Dots slēptais Markova modelis λ_w, kurš pieder klasei w. Uzdevums ir maksimizēt doto novērojumu virknes (arī pieder klasei w) varbūtību atbilstoši modeļa λ_w parametriem. Matemātiski šī ticamība var tikt izteikta šādi

$$L_{tot} = P(O^w \mid \lambda_w).$$

Atmetot augšrakstu un apakšrakstu w (jo mēs vienā laika momentā apskatām tikai vienu klasi w), MT var pārrakstīt kā

$$L_{tot} = P\left(O \mid \lambda\right).$$

Modelis $\lambda = (A, B, \pi)$, kas maksimizē lielumu L_{tot} nevar tikt aprēķināts analītiski, jo nav zināma metode, kā to izdarīt. Izmantojot iteratīvas procedūras, piemēram, Baum-Welch algoritmu vai gradientu algoritmu, var lokāli to maksimizēt, izvēloties atbilstošus modeļa parametrus.

Baum-Welch algoritms jeb *Forward-Backward* algoritms ir iegūstams, izmantojot "parādīšanās skaitīšanas" argumentus vai lietojot matemātiskās analīzes zināšanas, lai maksimizētu lielumu

$$Q\left(\lambda, \overline{\lambda}\right) = \sum_{q} P\left(q \mid O, \lambda\right) ln\left(P\left(O, q, \overline{\lambda}\right)\right)$$

pēc $\overline{\lambda}$. Algoritma uzdevums ir atrast labākās stāvokļu pārejas varbūtības, novērojumu varbūtības un stāvokļu sākuma varbūtības. Papildus *Forward* un *Backward* mainīgajiem ir nepieciešams ieviest vēl divus mainīgos. Pirmais no tiem ir

$$\xi_t\left(i, j\right) = P\left(q_t = s_i, q_{t+1} = s_j \mid O, \lambda\right), \, 1 \leqslant i, j \leqslant N, \, 1 \leqslant t \leqslant T - 1,$$

kas var tikt pārrakstīts

$$\xi_t\left(i, j\right) = \frac{P\left(q_t = s_i, q_{t+1} = s_j, O \mid \lambda\right)}{P\left(O \mid \lambda\right)} \, 1 \leqslant i, j \leqslant N, \, 1 \leqslant t \leqslant T - 1.$$

Visbeidzot, izmantojot *Forward* un *Backward* mainīgos, pirmo mainīgo $\xi_t\left(i, j\right)$ var uzrakstīt šādi

$$\xi_t\left(i, j\right) = \frac{\alpha_t\left(i\right) a_{ij} \beta_{t+1}\left(j\right) b_j\left(o_{t+1}\right)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t\left(i\right) a_{ij} \beta_{t+1}\left(j\right) b_j\left(o_{t+1}\right)}, \, 1 \leqslant i, j \leqslant N, \, 1 \leqslant t \leqslant T - 1.$$

Savukārt otrais mainīgais ir

$$\gamma_t\left(i\right) = P\left(q_t = s_i \mid O, \lambda\right), \, 1 \leqslant i \leqslant N, \, 1 \leqslant t \leqslant T.$$

Izmantojot *Forward* un *Backward* mainīgos iepriekšējo mainīgo var pārrakstīt

$$\gamma_t\left(i\right) = \frac{\alpha_t\left(i\right) \beta_t\left(i\right)}{\sum_{i=1}^{N} \alpha_t\left(i\right) \beta_t\left(i\right)}, \, 1 \leqslant i \leqslant N, \, 1 \leqslant t \leqslant T - 1.$$

Tātad sakarība, kas saista $\gamma_t\left(i\right)$ un $\xi_t\left(i, j\right)$ ir

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j), \; 1 \leqslant i \leqslant N, \; 1 \leqslant t \leqslant T-1.$$

Tagad var aprakstīt Baum-Welch apmācības algoritmu, lai maksimizētu lielumu $P(O \mid \lambda)$. Pieņemsim, ka sākotnējais modelis ir $\lambda = (A, B, \pi)$ un aprēķināsim visus lielumus α un β, bet, kad tas būs izdarīts, arī lielumus ξ un γ. Nākamie vienādojumi mēdz tikt saukti par "pārnovērtēšanas formulām" un tiek lietoti, lai uzlabotu slēptā Markova modeļa parametrus

- Stāvokļu sākuma varbūtības

$$\overline{\pi}_i = \gamma_1(i), \; 1 \leqslant i \leqslant N.$$

- Stāvokļu pārejas varbūtības

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \; 1 \leqslant i,j \leqslant N.$$

- Novērojumu varbūtības

$$\overline{b}_j(k) = \frac{\sum_{\substack{t=1 \\ o_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}, \; 1 \leqslant j \leqslant N, \; 1 \leqslant k \leqslant M.$$

Šīs formulas var tikt izmainītas, lai tās atbilstu arī nepārtrauktam gadījumam.

Piemērs 5.

Apskatīsim Baum-Welch algoritmu. Zināms, ka Baum-Welch algoritms uzlabo parametrus, ko vēlāk var izmantot visos iepriekš apskatītajos algoritmos. SMM modeļa $\lambda = (A, B, \pi)$ sākotnējos parametrus nemainīsim, saglabāsim novērojumus

$$O = (o_1, o_2, o_3, o_4, o_5) = ('Ls100', 'Ls5', 'Ls500', 'Ls500', 'Ls100').$$

Izrakstīsim Baum-Welch algoritma (apskatīsim gadījumu, kad maksimālais iterāciju skaits = 2) parametrus, kur A, B, π ir attiecīgi stāvokļu pārejas varbūtības, novērojumu varbūtības un stāvokļu sākuma varbūtības, kā arī *Forward*, *Backward* un Viterbi algoritmu dotās varbūtības

- iterāciju skaits = 0 (sākuma stāvoklis)

$$A = \begin{pmatrix} 0.3 & 0.6 & 0.1 \\ 0.7 & 0.1 & 0.2 \\ 0.5 & 0.4 & 0.1 \end{pmatrix},$$

$$B = \begin{pmatrix} 0.1 & 0.5 & 0.2 & 0.2 \\ 0.6 & 0.2 & 0.1 & 0.1 \\ 0.3 & 0.3 & 0.3 & 0.1 \end{pmatrix},$$

$$\pi = (0.7, 0.2, 0.1)\,.$$

Forward un *Backward* algoritma varbūtība = 0.0002828.
Viterbi algoritma varbūtība = 0.0000592.

- iterāciju skaits = 1

$$A = \begin{pmatrix} 0.26 & 0.64 & 0.1 \\ 0.75 & 0.07 & 0.18 \\ 0.45 & 0.45 & 0.1 \end{pmatrix},$$

$$B = \begin{pmatrix} 0.04 & 0 & 0.39 & 0.58 \\ 0.57 & 0 & 0.03 & 0.4 \\ 0.2 & 0 & 0.32 & 0.48 \end{pmatrix},$$

$$\pi = (0.82, 0.05, 0.13)\,.$$

Forward un *Backward* algoritma varbūtība = 0.0090525.
Viterbi algoritma varbūtība = 0.0038.

- iterāciju skaits = 2

$$A = \begin{pmatrix} 0.17 & 0.75 & 0.08 \\ 0.81 & 0.03 & 0.16 \\ 0.42 & 0.48 & 0.1 \end{pmatrix},$$

$$B = \begin{pmatrix} 0.01 & 0 & 0.46 & 0.53 \\ 0.56 & 0 & 0 & 0.44 \\ 0.12 & 0 & 0.24 & 0.65 \end{pmatrix},$$

$$\pi = (0.91, 0, 0.09)\,.$$

Forward un *Backward* algoritma varbūtība = 0.0149031,
Viterbi algoritma varbūtība = 0.0092803.

Redzams, ka algoritmu aprēķinātās varbūtības uzlabojas, iterāciju skaitam pieaugot. Var atrast lokālo maksimuma punktu, palielinot iterāciju skaitu. Ja iterāciju skaits ir lielāks par 7, tad starpība starp divām secīgām *Forward*, *Backward* un Viterbi algoritmu iegūtajām varbūtībām ir mazāka par 10^{-10}. Iegūtais rezultāts var tikt uzskatīs par lokālo maksimumu: *Forward* un *Backward* algoritma varbūtība = 0.0324999, bet Viterbi algoritma varbūtība = 0.029904.

Jāatzīmē, ka ir arī iespējamas problēmas ar sākuma datiem (izejas parametriem). Šī problēma tiks aprakstīta vēlāk.

2.3.1. Koeficientu svēršana

Nodaļā 2.3.1 tiks izmantots [5] avots.

Lai redzētu, kāpēc ir svarīgi "nosvērt" parametrus, kad tiek pielietota pārnovērtēšanas procedūra. *Forward* mainīgais $\alpha_t(i)$ sastāv no liela skaita locekļu summas, kur katrs loceklis ir šāds

$$\prod_{s=1}^{N} P(s_i \mid s_{i+1}) \prod_{s=1}^{N} b_{s_i}(O).$$

Tā kā reizinājumu locekļi ir daudz mazāki par 1, tad var ievērot, ka $\alpha_t(i)$ tiecas uz 0, kad $t \to \infty$. Tas nozīmē, ka pēc pietiekami daudz laika jebkurš dators saskarsies ar precizitātes problēmu. Tāpēc ir vajadzīga svēršanas procedūra. Pamatprocedūra, kas tiek plaši pielietota, ir $\alpha_t(i)$ reizināšana ar svaru koeficientu, kas nav atkarīgs no stāvokļa s_i, kur $1 \leqslant t \leqslant T$. Svaru koeficienti c_t ir šādi

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)},\ 1 \leqslant t \leqslant T.$$

Tātad nosvērtie *Forward* mainīgie $\overline{\alpha}_t(i)$ ir

$$\overline{\alpha}_t(i) = c_t \alpha_t(i),\ 1 \leqslant i \leqslant N.$$

Līdzīga svēršanas procedūra var tikt pielietota arī *Backward* $\overline{\beta}_t(i)$ mainīgajiem

$$\overline{\beta}_t(j) = c_t \beta_t(j),\ 1 \leqslant j \leqslant N.$$

Var parādīt, ka aprēķinot stāvokļu pārejas matricas A mēs iegūstam tieši tādus pašus rezultātus, lietojot $\overline{\alpha}_t(i)$ un $\overline{\beta}_t(j)$ attiecīgi $\alpha_t(i)$ un $\beta_t(j)$ vietā. Vienīgā tiešām svarīgā izmaiņa ir varbūtības $P(O \mid \lambda)$ aprēķināšana, jo vairs nedrīkst saskaitīt visus $\overline{\alpha}_T(i)$ (tie jau ir nosvērti). Taču izrādās, ka ir joprojām var aprēķināt $logP(O \mid \lambda)$. Ja Viterbi algoritmā tiek izmantots logaritms, tad svēršanas procedūra nav nepieciešama. Tas nozīmē, ka var atrast varbūtību ar mazāk aprēķiniem un bez skaitliskām kļūdām.

2.3.2. Apmācības dati

Nodaļā 2.3.2 tiks izmantots [5] avots.

Acīmredzama problēma slēpto Markova modeļu apmācībai ir tā, ka novērojumu virkne ir galīga. Tas nozīmē, ka bieži mēdz būt nepietiekams novērojumu parādīšanās daudzums, lai dotu labus parametru novērtējumus. Loģisks veids, kā atrisināt šo problēmu, būtu jaunu datu ievākšana, bet tas bieži vien praktiskās situācijās ir neiespējami, un tādēļ ir nepieciešams atrast paņēmienu, kas darbojas ar pašreizējiem datiem.

Viens no risinājumiem ir vienkārši samazināt modeļa izmērus, tas ir, stāvokļu skaitu, iespējamo novērojumu skaitu katrā stāvoklī, u.c. Taču nereti, modelējot reālus modeļus, tas apraksta fizikālu procesu, tāpēc samazināšana nav veicama.

Otrs veids ir interpolēt vienu parametru novērtējumu kopu otrā parametru novērtējumu kopā (modelim, kam ir atbilstošs apmācības datu daudzums). Ideja ir izmantot apmācības datus, lai izveidotu divus modeļus: vienu, kas atbilst vēlamajam, bet otru, kas ir mazāks, bet ar pietiekamu apmācības datu daudzumu. Mazākais modelis tiek iegūts "sasienot" kopā vienu vai vairākas sākotnējā modeļa parametru kopas. Gala rezultāts tiek iegūts ar šo divu modeļu interpolāciju. Galvenais uzdevums ir saprast, kā novērtēt gan sākuma modeli, gan arī samazināto modeli.

2.3.3. Sākotnējo parametru novērtējumi

Nodaļā 2.3.3 tiks izmantots [5] avots.

Principā neeksistē viennozīmīgas atbildes, kā izvēlēties sākotnējos novērtējumus slēptā Markova modeļa parametriem. Liekas, ka stāvokļu sākuma varbūtības π un pārejas varbūtības (a_{ij}), kas uzdotas matricā A, nav atkarīgas no varbūtību sadalījuma. Taču novērojumu varbūtības $b_j(k)$, kas uzdotas matricā B, sākotnējās varbūtības ir nozīmīgas, it īpaši nepārtraukta varbūtību sadalījuma gadījumā. Ir virkne ieteikumu kā iegūt labus sākotnējos novērojumus, piemēram, novērojumu virknes manuāla segmentācija par stāvokļiem, viduvējot novērojumus šajos stāvokļos, novērojumu maksimālās ticamības segmentācija ar viduvēšanu.

Piemērs 6.

Apskatīsim gadījumu, kas parāda paragrāfā aprakstīto problēmu ar sākotnējo parametru novērtējumiem, kas īpaši ietekmē novērojumu matricu B, bet praktiski neietekmē stāvokļu pārejas matricu A un stāvokļu sākuma varbūtības π. Tam liela nozīme ir slēpto Markova modeļu apmācības problēmā.

Tātad sākotnējie parametri (netika mainīti) ir šādi

$$A = \begin{pmatrix} 0.3 & 0.6 & 0.1 \\ 0.7 & 0.1 & 0.2 \\ 0.5 & 0.4 & 0.1 \end{pmatrix},$$

$$B = \begin{pmatrix} 0.1 & 0.5 & 0.2 & 0.2 \\ 0.6 & 0.2 & 0.1 & 0.1 \\ 0.3 & 0.3 & 0.3 & 0.1 \end{pmatrix},$$

$$\pi = (0.7, 0.2, 0.1).$$

Izvēlēsimies šādus novērojumus

$$O = (o_1, o_2, o_3, o_4) = ('Ls5', 'Ls10', 'Ls100', 'Ls500').$$

Apskatot kaut vai tikai *Forward* algoritma (izpildot Baum-Welch algoritmu visiem modeļa parametriem, kur iterāciju skaits = 1) locekļus $\alpha_t(i)$, kur $1 \leqslant t \leqslant T$ un $1 \leqslant i \leqslant N$,

būs redzams "strupceļš"

$$\begin{array}{llll} \alpha_1(o_1) = 0.0513, & \alpha_1(o_2) = 0.143406, & \alpha_1(o_3) = 0.0231686, & \alpha_1(o_4) = 0 \\ \alpha_2(o_1) = 0.312, & \alpha_2(o_2) = 0.0088648, & \alpha_2(o_3) = 0.0220043, & \alpha_2(o_4) = 0 \\ \alpha_3(o_1) = 0.0364, & \alpha_3(o_2) = 0.0166041, & \alpha_3(o_3) = 0.0106923, & \alpha_3(o_4) = 0 \end{array} .$$

Uzlabojot visus parametrus, *Forward*, *Backward* un Viterbi algoritmu aprēķinātās varbūtības ir vienādas ar nulli. Arī turpmāka Baum-Welch algoritma darbība (ja iterāciju skaits > 1) ir liegta, jo dalīšana ar nulli nav iespējama.

Bet situācija var nebūt tik viennozīmīga un neizejama. Ja Baum-Welch algoritmam uzlabo tikai stāvokļu pārejas matricu A un stāvokļu sākuma varbūtības π, bet novērojumu matricu B fiksē (jeb matrica B netiek uzlabota), tad joprojām var iegūt lielākas visu algoritmu dotās varbūtības.

3. Slēpto Markova modeļu pielietojumi

Pēdējā pamatdaļas nodaļa "Slēpto Markova modeļu piemēri" apskata trīs reālus piemērus, kuriem atbilst un var izmantot SMM.

- Pirmais no tiem ir saistīts ar runas atpazīšanu, kas varētu būt saukts par sava veida SMM "klasiku", jo pētījumi šajā jomā aizsākās jau 20.gs. 80. gadu beigās.
- Otrais nodarbojas ar proteīnu modelēšanu, kas ir jau daudz jaunāks pētījumu objekts. Lai arī šķietami atšķirīgs, tam ar runas atpazīšanu ir daudz kopīgu iezīmju. Uzsvars tiek likts uz Viterbi algoritmu, kas atrod aminoskābju virkni, kas savukārt veido proteīnu.
- Trešais, vissavdabīgākais piemērs, ir par Hamptonese - sava veida pierakstu, ko izveidojis James Hampton (1909-1964). Šis pieraksts, atrasts viņa garāžā pēc autora nāves, tika analizēts. Iesākumā tika izteikta hipotēze par saderību ar angļu valodu, kas apstiprinājās. Pēcāk pielieto slēptos Markova modeļus tekstam angļu valodā, kā arī angļu valodas fonēmām.Visbeidzot Hamptonese tika salīdzināta ar citām valodām un tika secināts, ka tā neatbilst nevienai no tām, tā ir tikai pavirši uzrakstītas un šifrētas runātās skaņas jeb fonēmas.

3.1. Runas atpazīšana

Nodaļā 3.1 tiks izmantots [5] avots.

Nav apšaubāms, ka viena no vislietderīgākajiem slēptā Markova modeļa pielietojumiem ir runas atpazīšana. Šo piemēru 1989. gadā apskatīja L.Rabiner un tas ir saistīts ar izolētu vārdu atpazīšanu.

Pieņemsim, ka vārdi, kas ir jāatpazīst, pavisam ir skaitā V, un katrs runātais vārds parādās K reizes. Katra vārda parādīšanās veido novērojumu. Vārdu novērojumi parasti tiek attēloti ar spektra un/vai laika signālu veidā. Lai veiktu izolēto vārdu atpazīšanu ir jāveic divi uzdevumi:

- Vispirms ir vajadzīgs izveidot katram vārdam vārdnīcā savu SMM (jeb katram vārdam v ir nepieciešams novērtēt modeļa $\lambda_v = (A_v, B_v, \pi_v)$ parametrus (A_v, B_v, π_v), kas optimizē apmācības kopas vārda novērojumu vektoru ticamību.
- Katram nezināmajam vārdam jāveic novērojumu virknes analīze un jāaprēķina modeļa ticamība visiem iespējamajiem modeļiem. Modelis tad izdod atpazīto vārdu kā vārdu ar vislielāko ticamību.

3.2. Proteīnu modelēšana

Nodaļā 3.2 tiks izmantots [5] avots.

Proteīnu modelēšana nemaz nav tik atšķirīga no runas atpazīšanas gadījuma, kā tas pirmajā brīdī izliekas. Daudz vispārīgāka runas atpazīšana, kad tiek izmantota vārdu virkne vai fonēma, var tikt uzskatīta par iezīmju atpazīšanas uzdevumu. Tas izpildās arī proteīnu modelēšanā, kur uzdevums ir modelēt aminoskābju virkni, kas savukārt veido proteīnus. Faktiski, šeit vārdi atbilst 20 aminoskābēm - no tām sastāv ikviena proteīnu molekula. Šo piemēru 1994. gadā apskatīja A.Krogh.

Proteīna strukturālā intuīcija var tikt aplūkota šādos veidos:

- Pozīciju virkne, katra ar savu sadalījumu pār aminoskābēm.
- Varbūtība pārlekt pozīciju vai ievietot papildu aminoskābes starp divām secīgām pozīcijām.
- Pieļaujot varbūtību, ka ievietošanas vai izdzēšanas turpināšana ir daudz ticamāka par sākšanu.

A.Krogh veido slēptos Markova modeļus tā lai tie atbilstu augstāk minētajām īpašībām. SMM satur stāvokļu virkni ar garumu M, kas tiks sauktas par saderīgajiem stāvokļiem, atbilstoši pozīcijām proteīnā vai kolonnām vairākkārtīga sakārtojuma gadījumā. Katrs no M stāvokļiem var generēt burtu x no 20 burtu gara aminoskābju alfabēta atbilstoši sadalījumam $P(X = x \mid Z = m_k)$, kur $1 \leqslant k \leqslant M$, jeb arī, katrs radītais burts atbilsts konkrētai aminoskābei. Pieraksts $P(X = x \mid Z = m_k)$ nozīmē, ka katram no saderīgajiem stāvokļiem m_k, kur $1 \leqslant k \leqslant M$, ir atsevišķi sadalījumi.

Lai modelētu varbūtību pozīcijas pārlēkšanai, jāņem vērā, ka katram saderīgajam stāvoklim m_k ir savs dzēšanas stāvoklis d_k, kas gluži vienkārši ir beigu stāvoklis.

Lai modelētu varbūtību jaunas aminoskābes ievietošanai, pavisam ir $M + 1$ ievietošanas stāvoklis katrā pusē saderīgajiem stāvokļiem, kas generē burtus no aminoskābju alfabēta tieši tādā pašā veidā kā saderīgais stāvoklis, bet izmanto varbūtību sadalījumus $P(X = x \mid Z = i_k)$, kur $1 \leqslant k \leqslant M$. Vienkāršības labad, sākumā un beigās tika pievienoti fiktīvie stāvokļi, apzīmēti kā m_0 un m_{M+1}, kuri negenerē nevienu aminoskābi.

Jāievēro, ka modelis pieļauj dažas papildus aminoskābes, jo ievietotajiem stāvokļiem eksistē varbūtība pāriet pašiem sevī (jeb nākamajā solī palikt tajā pašā stāvoklī, kur atrodas šobrīd). Katram stāvoklim ir trīs iespējamās pārejas. Pārejas saderīgajos stāvokļos vai dzēšanas stāvokļos vienmēr ir vērstas prom no šiem stāvokļiem uz citiem stāvokļiem, bet pārejās ievietošanas stāvokļos tas neizpildās. Pārejas varbūtība no stāvokļa q uz stāvokli r ir $P(Z = z \mid Z = q)$, bet šeit apzīmēta $T(r \mid q)$, kas atbilst atpazīstamajam P_{rq} (iepriekšējās nodaļās a_{rq}).

Virkne no modeļa tiek generēta šādā veidā:

- Atrodoties fiktīvajā sākuma stāvoklī m_0, nejauši, bet saskaņā ar pārejas varbūtībām $T(m_1 \mid m_0)$,$T(d_1 \mid m_0)$ un $T(i_1 \mid m_0)$ attiecīgi tiek izvēlēta pāreja uz m_1,d_1 vai i_0 .
- Ja mēs atrodamies saderīgā stāvoklī vai ievietošanas stāvoklī, attiecībā pret aminoskābi tiek ģenerēts burts x. Piemēram, ja mēs atrodamies stāvoklī m_k, aminoskābe tiek ģenerēta atbilstoši varbūtību sadalījumam $P(X = x \mid Z = m_k)$, bet, ja mēs atrodamies uz dzēšanas stāvoklī, tad aminoskābe netiek ģenerēta.
- Nākošais solis tiek izvēlēts saskaņā ar iespējamajām pārejām pašreizējā stāvoklī. Procedūra tiek turpināta kamēr virkne sasniedz stāvokli m_{M+1}, kas ir fiktīvais beigu stāvoklis (aminoskābes netiek ģenerētas).

Iegūtā virkne $x_1, x_2, ..., x_L$ tagad ir burtu virkne, kas iegūta attiecīgi no dažādajām aminoskābēm, bet virkne ir iegūta sekojot stāvokļiem $q_0, q_1, ..., q_N, q_{N+1}$, kur $q_0 = m_0$ un $q_{N+1} = m_{M+1}$. Tā kā dzēšanas stāvokļi negenerē nevienu aminoskābi, tad var secināt, ka $N + 1$ (stāvokļu skaits ceļā) ir lielāks vai vienāds par L (virknes garums).

Ja q_i ir saderīgais stāvoklis vai ievietošanas stāvoklis, tad definēsim $l(i)$ kā indeksu virknē $x_1, x_2, ..., x_L$ aminoskābju procedūrai stāvoklī q_i. Varbūtība notikumam, ka ir iegūts ceļš $q_0, q_1 ... q_N, q_{N+1}$ un arīdzan ģenerēta virkne $x_1, x_2, ..., x_L$, ir šāda

$$P(x_1, x_2, ..., x_L, q_0, q_1, ..., q_N, q_{N+1} \mid modelis) = T(m_{N+1} \mid q_N) \prod_{i=1}^{N} T(q_i \mid q_{i-1}) P\left(x_{l(i)} \mid q_i\right),$$

kur $P\left(x_{l(i)} \mid q_i\right) = 1$, ja q_i ir dzēšanas stāvoklis. Varbūtība jebkādai aminoskābju virknei $x_1, x_2, ..., x_L$ var tikt atrasta summējot visus iespējamos ceļus, kas var radīt šo virkni

$$P(x_1, x_2, ..., x_L \mid modelis) = \sum_{ceļi\, q_0, ..., q_{N+1}} P(x_1, x_2, ..., x_L, q_0, q_1, ..., q_N, q_{N+1} \mid modelis).$$

Veids kā novērtēt modeļa parametrus ir šāds: dotai apmācības virkņu $s(1), ..., s(n)$ kopai var aprēķināt cik labi atbilst modelis, aprēķinot varbūtības, kas tos ģenerē. Tas ir locekļu, kas uzdoti summas veidā (kā tas uzdots iepriekšējā vienādojumā), reizinājums, kur katram $1 \leqslant j \leqslant n$ izpildās $x_1, x_2, ..., x_L = s(j)$. Rezultāts ir ticamības funkcija un maksimizējot to attiecībā pret modeļa parametriem mēs iegūstam labāko modeli, atsaucoties uz maksimālās ticamības metodi.

3.3. Hamptonese

3.3.1. Ievads

Nodaļā 3.3 tiks izmantots [7] avots.

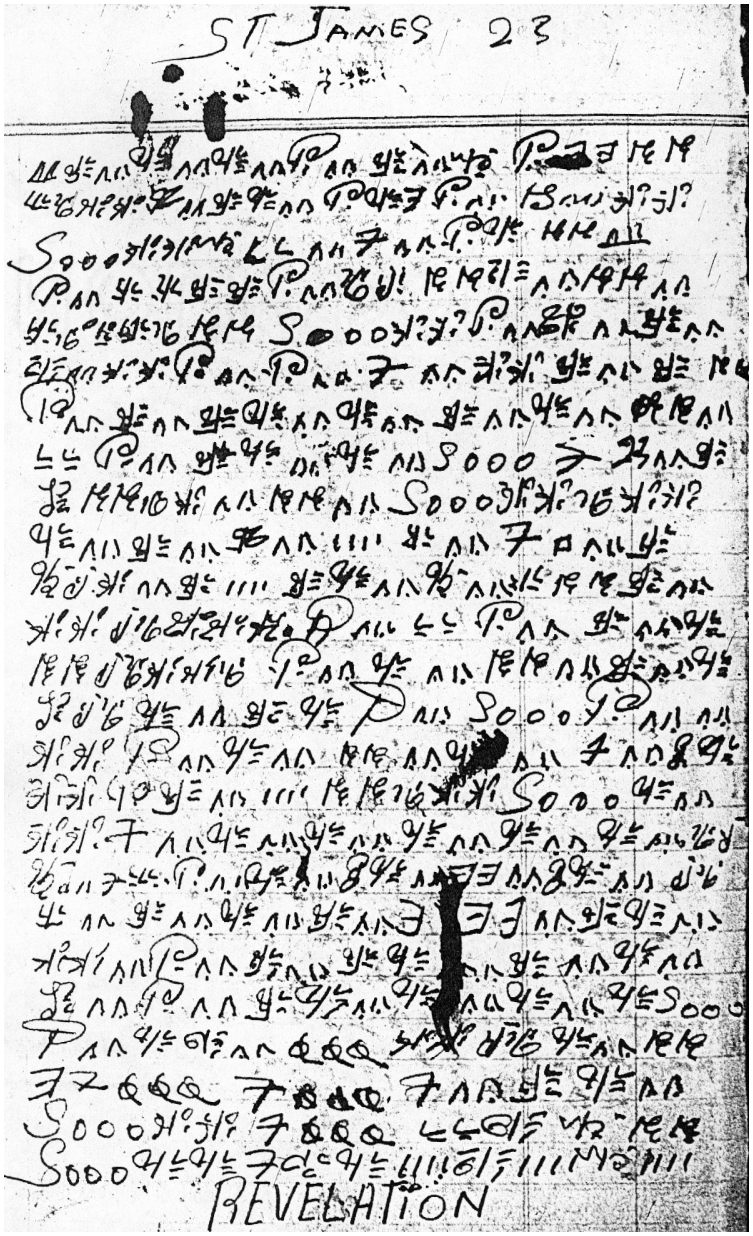
James Hampton (1909-1964) bija kluss un noslēgts cilvēks, kas atstājis pasaulei iespaidīgu vizuālās mākslas darbu kopumu, kā arī savādus pierakstus rokrakstā. J.Hamptona

raksts jeb Hamptonese savā ziņā ir interesants izpētes objekts. Piezīmju grāmatiņas, pēc autora nāves, tika atrastas viņa garažā. Tās satur izbārstītus angļu valodas vārdus, bet tajā dominē nezināmas izcelsmes pieraksts Hamptonese.

Kopumā 164 lapas var tikt iedalītas divās grupās, no kurām viena satur romiešu ciparus, Hamptonese zīmes, zīmējumus, kas visdrīzāk līdzinās uz kapenēm vai alu sienām atrodamajiem, un tekstiem angļu valodā, piemēram, "fear not". Otra kopa satur apmēram 100 lapas ar "tīru" Hamptonese, kas nesatur iepriekšējās kopas elementus, izņemot ļoti mazu daudzumu angļu valodas vārdus (piemēram, "Virgin Mary", "Revelation").

Katra lapa satur 25 vai 26 līnijas ar Hamptonese, simbolu daudzums rindā ir mainīgs. Nav ievērota lapu numerācija (tā ir haotiska), interpunkcija, teksts nav sakārtots paragrāfos, nav zināms, kā teksts ir lasāms (no augšas uz apakšu, no labās uz kreiso pusi).

Šīs 100 lapas arī kalpoja par pamatu analīzei. Salīdzinot Hamptonese ar kolekciju no rakstītajām valodām *http://www.omniglot.com/* netika atrasta neviena līdzīga.



6. att. Hamptonese fragments

3.3.2. Hamptonese entropija

Shannon entropija ir klasiskais informācijas vai nenoteiktības mērs. Bitu ziņā entropija tiek aprēķināta šādi

$$H(X) = -\sum_{x \in X} P(x) \log_2 P(x),$$

kur tiek pieņemts, ka $\log_2(0) = 0$. Ja angļu valoda sastāvētu no gadījuma izvēlēm, kas izdarītas no 27 simboliem (burtiem un atstarpes), tad angļu valodas nenoteiktība (atsevišķu simbolu ziņā) būtu:

$$H(X) = -\sum_{x \in X} \frac{1}{27} \log\left(\frac{1}{27}\right) = \log_2(27) \approx 4.75$$

jeb katrs burts satur aptuveni 4.75 bitus ar informāciju.

Rezultāti, kas apkopoti zemāk dotajā tabulā, parāda, ka Hamptonese ir saskaņojama ar angļu valodu. Taču tas nepierāda, ka Hamptonese ir atsevišķa valoda, bet dod pamatojumu turpināt analīzi.

1. tabula Entropiju salīdzinājums

	Entropija		
	1 simbols	2 simboli	3 simboli
Angļu valodas burti	4.11	3.54	2.68
Angļu valodas fonēmas	4.76	3.87	3.00
Hamptonese	4.41	3.63	3.08

3.3.3. SMM un angļu valoda

Cave un Neuwirth bija pirmie, kas pielietoja slēptos Markova modeļus tekstam angļu valodā. Viņi izvēlējās Brown Corpus (Brown University Standard Corpus of Present-Day American English). Tā ir galvenā teksta kolekcija, kas satur vairāk kā 1000000 vārdus. Viņi ignorēja visus ciparus, interpunkciju un speciālos simbolus, pārveidoja visus lielos burtus par mazajiem, atstājot tikai 27 simbolus - burtus un atstarpi.

Tad viņi pieņēma, ka eksistē Markova process ar diviem slēptiem stāvokļiem un novērojumiem, ko uzdod simboli no Brown Corpus. Rezultātā izveidojās stāvokļu pārejas matrica A, kas ir ar izmēru 2×2, un novērojumu varbūtību matrica B ar izmēru 2×27.

Atkārtojot Cave un Neuwirth eksperimentu, izmantojot 10000 novērojumus, pēc 200 iterācijām mēs iegūstam šādu tabulu

2. tabula SMM rezultāti angļu valodas tekstam

	Sākuma B vērtības		Beigu B vērtības	
Burts	Stāvoklis 0	Stāvoklis 1	Stāvoklis 0	Stāvoklis 1
a	0.0372642	0.0366080	0.0044447	0.1306242
b	0.0386792	0.0389249	0.0241154	0.0000000
c	0.0358491	0.0338276	0.0522168	0.0000000
d	0.0353774	0.0370714	0.0714247	0.0003260
e	0.0349057	0.0352178	0.0000000	0.2105809
f	0.0344340	0.0370714	0.0374685	0.0000000
g	0.0400943	0.0370714	0.0296958	0.0000000
h	0.0344340	0.0347544	0.0670510	0.0085455
i	0.0349057	0.0370714	0.0000000	0.1216511
j	0.0391509	0.0366080	0.0065769	0.0000000
k	0.0363208	0.0356812	0.0067762	0.0000000
l	0.0353774	0.0403151	0.0717349	0.0000135
m	0.0344340	0.0366080	0.0382657	0.0000000
n	0.0410377	0.0370714	0.1088182	0.0000000
o	0.0396226	0.0398517	0.0000000	0.1282757
p	0.0377358	0.0338276	0.0388589	0.0000047
q	0.0377358	0.0398517	0.0011958	0.0000000
r	0.0344340	0.0403151	0.1084196	0.0000000
s	0.0358491	0.0366080	0.1034371	0.0000000
t	0.0377358	0.0352178	0.1492508	0.0134756
u	0.0349057	0.0361446	0.0000000	0.0489816
v	0.0405660	0.0370714	0.0169406	0.0000000
w	0.0377358	0.0384615	0.0286993	0.0000000
x	0.0382075	0.0370714	0.0035874	0.0000000
y	0.0382075	0.0389249	0.0269053	0.0000003
z	0.0382075	0.0338276	0.0005979	0.0000000
atstarpe	0.0367925	0.0389249	0.0035184	0.3375209

Jāievēro, ka matrica B sākumā tika uzdota ar apmēram normāli sadalītām gadījuma vērtībām. Ir ļoti svarīgi, ka matrica nav uzdota ar precīzi normāli sadalītām gadījuma vērtībām, jo citādi Baum-Welch algoritms nedotu uzlabojumus.

Rezultāti 2. tabulā skaidri uzrāda, ka slēptais stāvoklis 1 atspoguļo "patskaņus", bet slēptais stāvoklis 0 atspoguļo "līdzskaņus". Kaut arī nav pārsteidzoši, ka patskaņi un līdzskaņi ir atdalīti, ir vērts uzsvērt, ka netika izdarīti nekādi a-priori pieņēmumi par slēpto stāvokļu dabu.

Tika aprēķināti arī slēptā Markova modeļa rezultāti angļu valodas fonēmām. Šim fonēmu SMM vispirms bija jāizveido fonētisks pārraksts no Brown Corpus, izmantojot izrunu vārdnīcu. Neievērojot akcentu zīmes, izrunu vārdnīca satur 39 simbolus. Ar diviem slēptiem stāvokļiem, izmantojot 50000 fonēmas un 200 iterācijas, mēs iegūstam rezultātus, kas apkopoti 3. tabulā.

3. tabula SMM rezultāti angļu valodas fonēmām

	Beigu B vērtības			Beigu B vērtības	
Fonēma	Stāvoklis 0	Stāvoklis 1	Fonēma	Stāvoklis 0	Stāvoklis 1
AH	0.0000000	0.2969759	EY	0.0000957	0.0460653
Z	0.0468621	0.0013497	F	0.0272705	0.0000000
AO	0.0000000	0.0353722	R	0.0772887	0.0000000
T	0.1115366	0.0164165	UW	0.0024273	0.0420961
W	0.0264391	0.0000000	N	0.1239146	0.0000000
AA	0.0000000	0.0443532	B	0.0270710	0.0000000
ER	0.0104758	0.0458586	G	0.0110797	0.0005440
K	0.0533438	0.0000000	M	0.0451959	0.0000000
D	0.0662517	0.0139417	EH	0.0000000	0.0763638
V	0.0367820	0.0000000	AE	0.0000000	0.0781199
IH	0.0000000	0.1464058	S	0.0804303	0.0101116
IY	0.0079505	0.0596027	L	0.0653828	0.0000000
OW	0.0000000	0.0273946	NG	0.0132029	0.0000000
SH	0.0161295	0.0000000	HH	0.0211180	0.0000000
AW	0.0001282	0.0131526	TH	0.0039602	0.0006984
AY	0.0001135	0.0282771	JH	0.0123050	0.0000000
P	0.0381176	0.0030524	CH	0.0094782	0.0000000
ZH	0.0007316	0.0000000	Y	0.0104094	0.0000000
DH	0.0545078	0.0000000	UH	0.0000000	0.0118911
OY	0.0000000	0.0019568			

Angļu valodas fonēmu SMM atkal var redzēt tīru atšķirību starp simboliem, kas iedalās šajos divos slēptajos stāvokļos. Interesanti, ka šajā fonēmu SMM viens no slēptajiem stāvokļiem "pārstāv" līdzskaņu skaņas, bet otrs- patskaņu skaņas, tātad analogi ar angļu valodas teksta SMM.

3.3.4. Hamptonese SMM

Aprēķinos atsevišķu Hamptonese simbolu skaits bija 42. Tika aprēķināti arī rezultāti $2, 3, 4, 5$ slēptajam stāvoklim. Rezultāti beigu matricas B gadījumam ar diviem slēptiem stāvokļiem ir uzdoti 4. tabulā.

4. tabula Hamptonese SMM rezultāti

	Beigu B vērtības			Beigu B vērtības	
Simbols	Stāvoklis 0	Stāvoklis 1	Simbols	Stāvoklis 0	Stāvoklis 1
2	0.0000000	0.0323915	14	0.0947011	0.0052195
ee	0.0059456	0.0091797	76	0.0504303	0.0132991
95	0.0013590	0.0108613	96	0.0071626	0.0220520
dc	0.0104561	0.0351127	EE	0.0141455	0.0141818
vv	0.4873287	0.0000000	M	0.0430382	0.0014101
F	0.0003230	0.0299307	N	0.0148755	0.0213175
g	0.0095264	0.0144809	g7	0.0005227	0.0024017
GG	0.0205943	0.0672325	Gi	0.0214469	0.0374094
Ki	0.0716056	0.0740697	d3	0.0000000	0.0312281
d4	0.0000000	0.0031228	Y3	0.0000000	0.2190866
Y4	0.0000000	0.0394331	qL3	0.0000000	0.0681507
qL4	0.0000241	0.0013892	4L	0.0068542	0.0112139
uL	0.0010389	0.0019307	J1	0.0055702	0.0069672
JJ	0.0089622	0.0288284	LL	0.0395888	0.0306004
nn	0.0108177	0.0120475	P	0.0000000	0.0461686
PL	0.0007135	0.0063528	P1	0.0000000	0.0184919
P2	0.0000000	0.0407190	o3	0.0230629	0.0000000
q3	0.0390770	0.0031463	S	0.0000000	0.0201452
3	0.0013213	0.0044007	T	0.0012893	0.0030790
A	0.0005299	0.0008652	A_	0.0044791	0.0075884
44	0.0027660	0.0042336	I	0.0004434	0.0002603

Pieņemsim, ka slieksnis, kas paredzēts, lai pievienotu simbolu stāvoklim 0 vai stāvoklim 1 ir 10 reizes. Tad rezultāti 4. tabulā uzrāda, ka pieci simboli $14, M, q3, vv, o3$ pieder stāvoklim 0, bet piecpadsmit citi simboli: $2, 96, F, g7, d3, d4, Y3, Y4, qL4, uL, P, P1, P2, S$ pieder stāvoklim 1. Taču joprojām atlikušie 22 simboli nevar tikt ierindoti nevienā no šiem stāvokļiem. Šis ir krass kontrasts starp angļu valodas burtu un fonēmu SMM rezultātiem, kur ,izmantojot tieši tādus pašus kritērijus, praktiski visi simboli ir pievienojami vienam vai otram stāvoklim.

Salīdzinot angļu valodas SMM rezultātus ar Hamptonese SMM rezultātiem, var secināt, ka Hamptonese rakstu zīmes neatspoguļo angļu valodas burtus vai fonēmas.

3.3.5. Secinājumi

Ir dažādi izskaidrojumi šiem Hamptonese SMM rezultāti. Piemēram, var gadīties, ka mums nav pietiekams daudzums Hamptonese datu. Ir iespējams iegūt angļu valodas SMM rezultātus, izmantojot tikai 10000 simbolus, kaut arī Hamptonese zīmju skaits ir vairāk kā 29000.

Ja datu daudzums ir pietiekams, tad datu kvalitāte var būt neapmierinoša, tas ir, Hamptonese dati ir ļoti trokšņaini. Tas var rasties sliktas pārrakstīšanas vai arī J.Hamptona pierakstu kļūdu dēļ.

Vēl viena no problēmām ir nepareiza Hamptonese datu interpretācija. Iespējams, ka rakstu zīmju kombinācijas ir jāsaprot kā simboli. Šajā gadījumā atsevišķām Hamptonese rakstu zīmēm ir maza vai praktiski nekāda semantiska nozīme, tātad slēptais Markova modelis nespēj tās klasificēt. Kā piemērs tam var kalpot rakstu zīme "Ki", kas tekstā parādās gan atsevišķi, gan arī kā "Ki Ki". Beidzamais noteikti varētu būt uzskatīts par atsevišķu rakstu zīmi.

Varbūt, ka Hamptonese ir šifrs. Galējā gadījumā, Hamptonese var būt šifrēts, izmantojot vienreizēju šifrēšanu, tātad to atšifrēt būtu praktiski neiespējami. Bet pētot J.Hamptona vēsturi, mēs varam pieņemt, ka viņš varējis būt izveidot ļoti vāju šifrēšanas sistēmu - salīdzinošo Hamptonese noteiktību var uzskatīt par pierādījumu tam. Slēptais Markova modelis sniedza atbildi, ka Hamptonese ir vienkārši aizvietošanas metodes šifrs angļu valodai. Pēc dziļākas Hamptonese kriptoanalīzes var secināt, ka, faktiski, Hamptonese ir slikts šifrs.

Protams, ka neskatoties uz līdzību valodai, Hamptonese ir tikai runāto skaņu pieraksts. Bet pat šajā gadījumā būtu interesanti turpināt šos pētījumus. 19. gadsimta beigās franču mistiķe Helene Smith uzskatīja, ka tad, kad viņa atrodoties transa stāvoklī, gari viņu aizved uz Marsu, kur viņa spēj sarunāties ar marsiešiem. Viņa marsiešu ziņojumus pierakstīja nezināmā rakstā, kas vēlāk izrādījās sakarīgs un ar reālas valodas iezīmēm. Šis pieraksts bija franču valodas vienkāršots variants.

4. Secinājumi

Darbā tika apskatīta diskrēto slēpto Markova modeļu teorija. Apskatot ar tematu saistīto literatūru var secināt, ka teorētiskā bāze tika iesākta tikai nesen, bet joprojām turpina strauji augt un tiek saistīta ar visdažādākajām zinātnes nozarēm: medicīnu, fiziku, informātiku, lingvistiku. Šī simbioze dod labumu abām pusēm: tiek izstrādāti jauni un uzlaboti vecie algoritmi un metodes, lai atrisinātu konkrēto problēmu, vienlaicīgi ļaujot izvērst teoriju (piemēram, Beijesa tīkli, neironu tīkli).

Lai risinātu trīs SMM pamatproblēmas, novērtēšanas, atšifrēšanas, apmācības, attiecīgi tika izmantoti algoritmi: *Forward* un *Backward*, Viterbi, Baum-Welch. Darba gaitā pierādījās, ka vislielākās problēmas sagādā tieši Baum-Welch algoritms, kam ir "jutīgs" pret izejas datiem un arī novērojumu virknes galīgumu (var rasties problēmas, ja ir nepietiekams daudzums novērojumu). Visi algoritmi tiek plaši izmantoti problēmu risināšanā.

Rakstot programmas kodus algoritmiem, tika apgūta jauna un ļoti progresīva programmēšanas valoda Python 2.7 (bezmaksas un atvērtā koda programma). Šī valoda noteikti noderēs nākotnē, galvenokārt, programmēšanas nozarē. Pašu programmas kodu rakstīšana nesagādāja problēmas, bet jāpiezīmē, ka tas bija laikietilpīgs process, ko vēl vairāk apgrūtināja iegūto rezultātu pārbaude un arī neprecizātes, kas mēdz būt sastopamas literatūrā. Pseido piemērs tika ar nodomu izvēlēts tāds, lai to varētu pielietot visiem algoritmiem (protams, ar nelielām izmaiņām), kas arī deva precīzus rezultātus. Kā tika parādīts apakšnodaļā 2.3, ir svarīgi, kādus sākotnējos parametrus izvēlēties, jo tas var novest pie aplamībām (dalīšanas ar nulli).

Faktiski, SMM tika lietoti reālajos piemēros, jo šie matemātiskie modeļi vislabāk atbilst dotajai problēmai, tiem ir izstrādāti algoritmi, kas ievērojami paātrina risināšanu vai uzlabo parametrus. Nav izslēgts, ka iespējams pastāv kāda cita, bet tikpat efektīva, alternatīva metode, bet tāda meklējot netika atrasta.

Nākotnes perspektīvas varētu būt saistītas ar slēpto Markova modeļu apskatīšanu ne tikai to diskrētajā, bet arī nepārtrauktā gadījumā.

Literatūra

[1] http://en.wikipedia.org/wiki/Hidden_Markov_model

[2] http://en.wikipedia.org/wiki/Viterbi_algorithm

[3] P. Blunsom. Hidden Markov Models, 2004.

[4] G. L. Kouemou. History and Theoretical Basics of Hidden Markov Models, 2010.

[5] M. Karlsson. Hidden Markov Models, 2008.

[6] M. Singh, J. Parker. Profile Hidden Markov Models. *Topics in Computational Molecular Biology*, 1999.

[7] M. Stamp, E. Le. Hamptonese and Hidden Markov Models, 2003.

5. Pielikums

```
from __future__ import division
from itertools import *
class HMM:
def __init__(self, obs, states, start_p, trans_p, emit_p):
self.obs = obs
self.states = states
self.start_p = start_p
self.trans_p = trans_p
self.emit_p = emit_p
def print_dptable(self, M):
print " ",
for i in range(len(M)): print "%7s" % ("%d" % i),
print
for j in M[0].keys():
print "%7s: " %j,
for t in range(len(M)):
print "%.7s" % ("%f" % M[t][j]),
print
def forward(self):
self.alpha = [{}]
self.prob_alpha = 0
for i in self.states:
self.alpha[0][i] = self.start_p[i] * self.emit_p[i][self.obs[0]]
for t in range(1,len(self.obs)):
self.alpha.append({})
for i in self.states:
self.alpha[t][i] = self.emit_p[i][self.obs[t]]*sum(self.alpha[t-1][j] * self.trans_p[j][i] for
j in self.states)
self.prob_alpha = sum(self.alpha[t][j] for j in self.states)
return (self.alpha, self.prob_alpha)
def backward(self):
self.beta = []
self.prob_beta = 0
for t in range(len(self.obs)):
self.beta.append({})
for i in self.states:
if t == len(self.obs) - 1:
self.beta[t][i] = 1
else:
```

```
self.beta[t][i] = 0
for t in reversed(range(len(self.obs))):
for i in self.states:
if t in range(len(self.obs)-1):
self.beta[t][i] = sum(self.beta[t+1][j] * self.trans_p[i][j] * self.emit_p[j][self.obs[t+1]]
for j in self.states)
self.prob_beta = sum(self.start_p[j] * self.emit_p[j][self.obs[0]] * self.beta[0][j] for j
in self.states)
return (self.beta, self.prob_beta)
def viterbi(self):
self.V = [{}]
self.prob_V = 0
self.path = {}
for i in states:
self.V[0][i] = self.start_p[i] * self.emit_p[i][self.obs[0]]
self.path[i] = [i]
for t in range(1,len(self.obs)):
self.V.append({})
newpath = {}
for i in states:
(self.prob_V, state) = max([(self.V[t-1][j] * self.trans_p[j][i] * self.emit_p[i][self.obs[t]],
j) for j in self.states])
self.V[t][i] = self.prob_V
newpath[i] = self.path[state] + [i]
self.path = newpath
(self.prob_V, state) = max([(self.V[len(self.obs) - 1][i], i) for i in self.states])
self.path = self.path[state]
return (self.V, self.path)
def baum_welch(self,iterations=3):
def reestimation():
self.xi = []
self.gamma = []
for t in range(len(self.obs)-1):
self.xi.append({})
for i in self.states:
self.xi[t][i] = {}
for j in self.states:
first_part = self.alpha[t][i] * self.trans_p[i][j] * self.beta[t+1][j] * self.emit_p[j][self.obs[t+1]]
second_part = sum(sum(self.alpha[t][i] * self.trans_p[i][j] * self.beta[t+1][j]\
* self.emit_p[j][self.obs[t+1]] for j in self.states) for i in self.states)
```

```
self.xi[t][i][j] = first_part / second_part
for t in range(len(self.obs)-1):
self.gamma.append({})
for i in states:
first_part = self.alpha[t][i] * self.beta[t][i]
second_part = sum(self.alpha[t][j] * self.beta[t][j] for j in states)
self.gamma[t][i] = first_part / second_part
start_p_updated = {}
for i in states:
start_p_updated[i] = round(self.gamma[0][i],2)
trans_p_updated = {}
for i in states:
trans_p_updated[i] = {}
for j in states:
first_part = sum(self.xi[t][i][j] for t in range(len(self.obs)-1))
second_part = sum(self.gamma[t][i] for t in range(len(self.obs)-1))
trans_p_updated[i][j] = round(first_part / second_part,2)
emit_p_updated = {}
for j in self.states:
emit_p_updated[j] = {}
for k in self.obs:
first_part = 0
second_part = 0
for t in range(len(self.obs)-1):
if k == self.obs[t]:
first_part += self.gamma[t][j]
second_part += self.gamma[t][j]
emit_p_updated[j][k] = round(first_part / second_part,2)
self.start_p = start_p_updated
self.trans_p = trans_p_updated
self.emit_p = emit_p_updated
for i in range(iterations):
HMM.forward()
HMM.backward()
reestimation()
def execute(self,*args):
for arg in args:
if arg == "forward":
print "Forward algoritms"
self.forward()
```

```
self.print_dptable(self.alpha)
print self.prob_alpha, observations
print("\n")
if arg == "backward":
print "Backward algoritms"
self.backward()
self.print_dptable(self.beta)
print self.prob_beta, observations
print("\n")
if arg == "viterbi":
print "Viterbi algoritms"
self.viterbi()
self.print_dptable(self.V)
print self.prob_V, self.path, observations
print("\n")
if arg == "baum_welch":
print "Baum-Welch algoritms"
self.forward()
self.backward()
self.baum_welch()
print "start_probability =",self.start_p
print "transition_probability =",self.trans_p
print "emission_probability =",self.emit_p
print("\n")
states = ("Bronza","Sudrabs","Zelts")
observations = ("Ls 10","Ls 500","Ls 10","Ls 5","Ls 100","Ls 500")
start_probability = {"Bronza":0.7,"Sudrabs":0.2,"Zelts":0.1}
transition_probability = {
"Bronza":{"Bronza":0.3,"Sudrabs":0.6,"Zelts":0.1},
"Sudrabs":{"Bronza":0.7,"Sudrabs":0.1,"Zelts":0.2},
"Zelts":{"Bronza":0.5,"Sudrabs":0.4,"Zelts":0.1}
}
emission_probability = {
"Bronza":{"Ls 5":0.1,"Ls 10":0.5,"Ls 100":0.2,"Ls 500":0.2},
"Sudrabs":{"Ls 5":0.6,"Ls 10":0.2,"Ls 100":0.1,"Ls 500":0.1},
"Zelts":{"Ls 5":0.3,"Ls 10":0.3,"Ls 100":0.3,"Ls 500":0.1}
}
HMM = HMM(observations, states, start_probability, transition_probability, emission_probab
HMM.execute("forward","baum_welch","forward")
```

Bakalaura darbs "Diskrētie slēptie Markova modeļi un to pielietojumi" izstrādāts LU Fizikas un matemātikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors:

_____ Mārtiņš Eglītis

(personiskais paraksts)

Rekomendēju darbu aizstāvēšanai

Vadītājs: doc. Dr.math. Jānis Valeinis

_____ _____

(personiskais paraksts) (datums)

Recenzents: lektors Jānis Smotrovs

Darbs iesniegts Matemātikas nodaļā_____

(datums)

Dekāna pilnvarotā persona: vecākā metodiķe Dzintra Holsta

_____

(personiskais paraksts)

Darbs aizstāvēts diplomdarba valsts pārbaudījuma komisijas sēdē

_____ prot. Nr. _____

Komisijas sekretāre: docente Margarita Buiķe _____

(personiskais paraksts)