SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1880

Izvedba funkcija za skrivene Markovljeve modele

Mario Karuza

Zagreb, lipanj 2011.

SADRŽAJ

1. Uvod

Interakcija s računalom putem govora intrigira ljude još od početka računalne ere. Da bi takva interakcija bila moguća mora postojati dvosmjerna komunikacija čovjek - računalo. Sintetizacija glasa i *text-to-speech* sustavi već su došli do razine svakodnevnog korištenja, no na području prepoznavanja govora i dalje se provode istraživanja i aktivno se radi na poboljšanju takvih sustava koji su još, nažalost, daleko od svakodnevne upotrebe.

2. Skriveni Markovljevi modeli

Iako prvi put predstavljeni i proučavani krajem šezdesetih i početkom sedamdesetih godina prošlog stoljeća, statističke metode modeliranja procesa pomoću Markovljevih lanaca i skriveni Markovljevih modela postaju izrazito popularne. Razlog tome možemo naći u nekoliko činjenica. Kao prvo, ove metode imaju dobru matematičku pozadinu i tako mogu stvoriti teoretsku podlogu za daljnju nadogradnju. Drugo, ako se adekvatno primjene daju dobre rezutate i mogu se koristiti u raznim aplikacijama poput prepoznavanju govora, bioinformatici, itd.

2.1. Markovljevi lanci

Neka su vremenski trenuci pridruženi slikama procesa označeni indeksima $t = 1, 2, 3, ...$, te neka je slika procesa u određenom trenutku t označena sa q_t. Svako stanje u vremenskom trenutku q_t može se opisati skupom po volji odabranih slučajnih varijabli stanja x_i:

$$q_t = (x_1, x_2, ..., x_n) \tag{2.1}$$

Neka je konkretan uređen skup fiksiranih varijabli stanja koji opisuje točno određeno stanje procesa označen sa S_i:

$$S_i = (x_1 = x_{1i}, x_2 = x_{2i}, .., x_n = x_{ni}) \tag{2.2}$$

Promjenjivi proces može se predstaviti nizom Q slika procesa q_t u određenim vremenskim trenucima t, počevši od slike procesa q_1 u početnom trenutku t_1:

$$Q = \{q_0, q_1, .., q_t, ...\} = q_0 q_1 ... q_t ... \tag{2.3}$$

ili

$$S = S_0, S_1, .., S_t, ... = S_0 S_1 ... S_t ... \tag{2.4}$$

Pod pretpostavkom da se proces odvija pod određenim zakonitostima, proučavanje ponašanja procesa može se svest na proučavanje uvjetne vjerojatnosti da će proces u

trenutku $t+1$ biti određenom stanju S_j, ukoliko je dan slijed prethodnih stanja, od početnog trenutka t_0 do trenutka t

$$P = (q_{t+1} = S_j | q_t = S_k, q_{t-1} = S_l, ...) \tag{2.5}$$

Ovakav način razmišljanja krije u sebi dva problema. Trebalo bi definirati beskonačano veliki skup mogućih stanja procesa S_i, po jedno za svaki odabir slučajnih varijabli u svakom vremenskom trenutku. Nadalje svako stanje procesa q_t ima neograničen broj prethodnih stanja.

Prvi problem rješava se pretpostavkom o stacionarnom stanju, tj. pretpostavka da se proces podvrgava zakonima koji se ne mjenanju s vremenom. Time dobivamo konačan skup $S = \{S_1, S_2, .., S_N\}$ svih mogućih stanja u kojima se može nalaziti proces u bilo kojem vremenskom trenutku.

Drugi problem je rješiv tzv. *Markovljevom pretpostavkom* koja kaže da trenutno stanje procesa ovisi o konačnom broju prethodnih stanja. Ovakvi procesi zovi su Markovljevi procesi. Najjednostavnija vrsta procesa kod kojeg stanje procesa u trenutku $t+1$ ovisi samo o stanju procesa u trenutku t naziva se Markovljev lanac prvog reda. Prethodno definiratni izraz (2.5) tako prelazi u:

$$P(q_{t+1} = S_j | q_t = S_k) \tag{2.6}$$

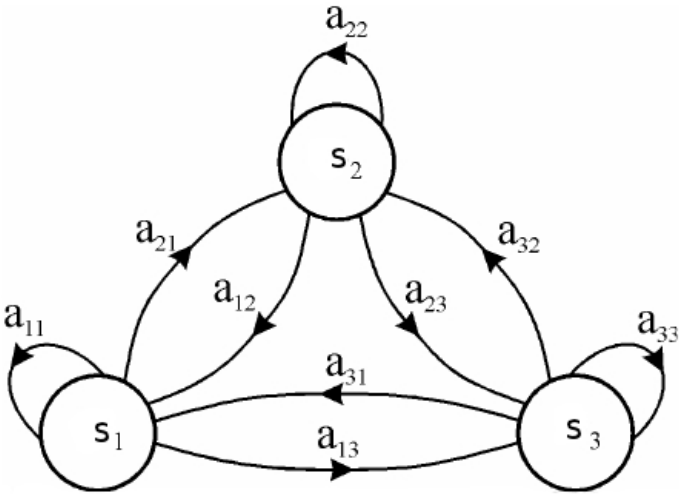
Nadalje, stacioniranost procesa nam govori da je vjerojatnost neovisna o vremenskom trenutku, što dovodi do zaključka da za promatrani proces postoji skup tranzicijskih vjerojatnosti prelaska procesa iz stanje u stanje a_{ij} oblika:

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_k), 1 \leq i, j \leq N, \forall t \tag{2.7}$$

Tranzicijske vjerojatnosti ne moraju nužno biti simetrične ($a_{ij} \neq a_{ji}$) i "lanac" se u određenom stanju može zadržati proizvoljno dugo($a_{ii} \neq 0$)

Također, treba definirati i skup inicijalnih vrijednosti π_i koje određuju vjerojatnosti početnih stanja:

$$\pi_i = P(q_0 = S_i), 1 \leq i \leq N \tag{2.8}$$



Slika 2.1: Markovljev lanac

2.2. Skriveni Markovljevi modeli

U prethodnom poglavlju promatrani su procesi kod kojih je moguće izravno odrediti u kojem se stanju proces nalazi. Često je problem složeniji i o stanju procesa možemo zaključiti samo posredno zaključujući preko dostupne nam varijable koju proces u tom stanju emitira. Općenito govoreći, proces koji je u nekom stanju S_i može sa određenom vjerojatnošću emitirati bilo koji element iz skupa simbola:

$$T = \{b_1, b_2, .., b_M\} \tag{2.9}$$

Značajku koji je proces emitirao u trentku t, u stanju q_t, označimo sa v_t. Sada, osim niza Q stanja kroz koje proces prolazi možemo promatrati i niz V opservacija u tim stanjima:

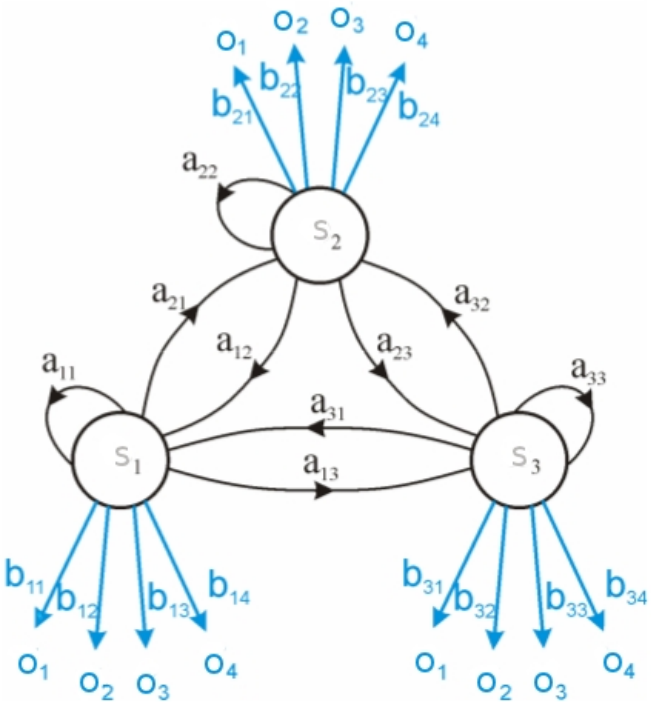
$$V = \{v_0, v_1, .., v_t, ..\} = v_0 v_1 ... v_t .. \tag{2.10}$$

ili

$$O = \{O_0, O_1, .., O_t, ..\} = O_0 O_1 ... O_t .. \tag{2.11}$$

Ovakvo poopćenja Markovljevog lanca kod kojeg su jedino vidljiva stanja O_t, dok su stanja S_i skrivena za okolinu nazivamo *skrivenim Markovljevim modelima*. Ponašanje modela ne razlikuje se mnogo od ponašanja lanca; u svakom koraku $(t \to t+1)$ sistem izvrši jedan prijelaz, ali u modelima se prilikom prijelaza emitira opservacijski simbol (vidljivo stanje).

Za svaki skriveni Markovljev model možemo definirati pet parametara koji ga jednoznačno određuju:



Slika 2.2: Skriveni Markovljev Model

1. N - broj stanja u kojima se proces može nalaziti

$$S = \{S_1, S_2, .., S_N\} \tag{2.12}$$

2. M - broj različitih opservacijskih simbola

$$T = \{b_1, b_2, .., b_M\} \tag{2.13}$$

3. A - matrica tranzicijskih vjerojatnosti prelaska proces iz stanja u stanje

$$A = [a_{ij}], a_{ij} = P(q_{t+1} = S_j | q_t = S_i), 1 \leq i, j \leq N \tag{2.14}$$

4. B - matrica vjerojatnosti emitiranja simbola

$$B = [b_j(k)], b_j(k) = P(O_t = b_k | q_t = S_i), 1 \leq j \leq N, 1 \leq k \leq M \tag{2.15}$$

5. Π - matrica inicijalnih tranzicijskih vrijednosti

$$\Pi = [\pi_i] \pi_i = P(q_0 = S_i), 1 \leq i \leq N \tag{2.16}$$

Budići da su prva dva parametra, N i M, implicitno definirana u ostalima, uređena troja $\lambda = (A, B, \Pi)$ smatra se potpunim opisam svakog skriven Markovljevog modela.

2.3. Osnovni problemi skrivenih Markovljevih modela

Skriveni Markovljevi modeli omogućuju jednostavno modeliranje procesa iz stvarnog svijeta, ali da bi ipak bili korisni u praski potrebno je riješiti tri osnovna problema:

1. Problem evaluacije - Dan je slijed opservacija $O^T = O_0O_1O_2...O_T$ i neka je dan model $\lambda = (A, B, \Pi)$. Potrebno je odrediti $P(O|\lambda)$, vjerojatnost da je dani model generirao dani slijed opservacija.

2. Problem dekodiranja - Dan je slijed opservacija $O^T = O_0O_1O_2...O_T$ i neka je dan model $\lambda = (A, B, \Pi)$. Potrebno je odrediti slijed stanja $S = q_1q_2...q_T$ koji bi s najvećom vjerojatnošću producirai dani niz opservacija.

3. Problem treniranja - Neka je dan skup opservacija $X = \{O_k\}$. Potrebno je odrediti skriveni Markovljev model $\lambda = (A, B, \Pi)$ koji maksimizira vjerojatnost emitiranja danog skupa opservacija.

2.3.1. Problem evaluacije

Kod evaluacijskog problema pitamo se koja je vjerojatnost da je zadani model generirao uočeni slijed opservacija. Ako postoje nekoliko modela koji su potencijalno generirali dani slijed, rješenje prvog problema nam govori koji je od njih najvjerojatniji.
Želimo izračunati vjerojatnost emitiranja slijeda opservacija $O^T = O_0O_1...O_T$ ukoliko je dan model $\lambda = (A, B, \Pi)$. Vjerojatnost da je model generirao tu sekvencu je:

$$P(V^T) = \sum_{r=1}^{r_{max}} P(O^T|s_r^T)P(s_r^T) \tag{2.17}$$

gdje r označava pojedinu sekvencu $S_r^T = \{s(1), s(2 =, ..., s(T)\}$ od T skrivenih stanja. U općem slučaju, ako postoji c skrivenih stanja računski dobivamo postojanje $r_{max} = c^T$ mogućih članova u sumi. Pri proračunu vjerojatnosti da je model generirao konkretan sljed, potrebno je uzeti u obzir svaku moguću kombinaciju skrivenih stanja, izračunati vjerojatnost da je ta kombinacija emitirala dati sljed i na kraju zbrojiti te vjerojatnosti. Vjerojatnost pojave pojedinog opservacijskog sljeda jednostavno je umnožak odgovarajućih tranzicijskih vjerojatnosti a_{ij} i vjerojatnost emitiranih simbola b_{jk} u svakom koraku.

Radimo sa Markovljevim procesima, pa drugi faktor u jednakosti (2.17) možemo zapisati kao:

$$P(s_r^T) = \Pi_{t=1}^T P(s(t)|s(t-1)) \tag{2.18}$$

što predstavlja umnožak svih elementa a_{ij} koji odgovaraju skrivenoj sekvenci stanja. Zbog pretpostavke da emitirane vjerojatnosti ovise samo od skrivenih stanja, u jednakosti (2.17), prvi faktor možemo zapisati i kao:

$$P(O^T|s_r^T) = \Pi_{t=1}^T P(O(t)|s(t)) \tag{2.19}$$

i predstavlja umnožak svih b_{jk} definiranih u odnosu na skriveni stanje i odgovorajauće vidljivo stanje.
Sada koristeći (2.18) i (2.19),jednakost možemo zapisati kao:

$$P(O^T) = \sum_{r=1}^{r_{max}} \Pi_{t=1}^T P(O(t)|s(t)) * P(s(t)|s(t-1)) \tag{2.20}$$

i ovu forumulu interpretirati na način da je vjerojatnost konkretnog sljeda T vidljivih stanja O^T jednaka sumi vjerojatnosti generiranja pojedinog prijelaza pomnoženih sa vjerojatnošću emitiranja vidljivog stanja u okviru promatranog slijeda, pri čemu se zbrajanje vrši po svim r_{max} mogućim sljedovima skrivenih stanja. Sve ove vjerojatnosti sadržane su u parametrima a_{ij} i b_{jk} i jednakost (2.20) se može direktno izračunati pri čemu dobivamo vremensku složenost $O(c^T * T)$.
Srećom, postoji efikasniji izračun, koji zahtjeva manji broj računskih operacija i koji koristi rekurziju.
Definirajmo prvo unaprijednu varijablu $\alpha_i(t)$:

$$\alpha_j(t) = \begin{cases} 0 & t = 0, j \neq \text{ početno stanje} \\ 1 & t = 0, j = \text{početno stanje} \\ \sum_i \alpha_i(t-1) a_{ij} b_{jk}^{O(t)} & t \neq 0 \end{cases} \tag{2.21}$$

Ova varijabla predstavlja vjerojatnost da, u trenutku t, model se nalazi stanju s_j i da je generirao prvih t elemenata slijeda O^T. Ovaj proračun je implementiran u tzv. *unaprijednom algoritmu* (Algoritam 1)

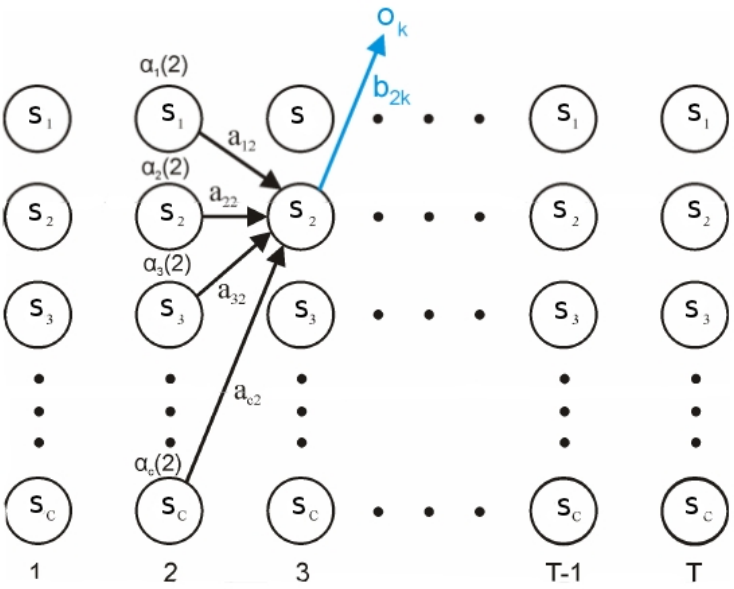
U liniji 5, α_0 označava vjerojatnost da slijed završi poznatim konačnim stanjem.

Pokušajmo ilustrirati na jednostavnom primjeru što pokušavamo postići unaprijednim algoritmom. Zamislimo da se model nalazi u stanju S_2 u trentuku $t = 3$ i da je u tom

Algoritam 1 – *Unaprijedni Algoritam*

1: **initialize** $t = 0, a_{ij}, b_{jk}, O^T, \alpha_j(0)$
2: **for** $t \leftarrow t + 1$ **do**
3: $\quad \alpha_i(t) = \sum_i^c \alpha_i(t-1) a_{ij} b_{jk}^{O(t)}$
4: **end for** $t = T$
5: **return** $p(O^T) \leftarrow \alpha_0(T)$
6: **end**

trenutku emitirao opservaciju O_k, potrebno je uočiti da u koraku $t = 2$ model bio u stanju S_j (i pri tome emitirao određenu opservaciju) koja iznosi $\alpha_j(2)$ za $j = 1, 2, ..., c$. Vjerojatnost $\alpha_2(3)$ dobije se kao suma svih umnožaka $\alpha_j(2) * a_{j2}$ pomnoženih sa vjerojatnošću b_{2k} emitiranja opservacije O_k. Broj operacija koje moramo izvršiti sad je



Slika 2.3: Ilustracija proračuna vjerojatnosti primjenom algoritmom unaprijed

$O(c^2 * T)$, što je mnogo manje nego u prethodnom slučaju.
Analogno, moguće je definirati unatražnu varijablu $\beta_i(t)$ kao vjerojatnost da se model nalazi u stanju S_i i da ćemo u narednim koracima generirati ostale elemente slijeda opservacija, od $t + 1$ do T, što matematički zapisujemo kao:

$$\beta_i(t) = \begin{cases} 0 & s_i(t) \neq s_0(t), t = T \\ 1 & s_i(t) = s_0(t), t = T \\ \sum_j \beta_j(t+1) a_{ij} b_{jk}^{O(t+1)} & t \neq 0 \end{cases} \tag{2.22}$$

Sada možemo definirati i *unazadni algoritam* koji u biti predstavlja unaprijedni algoritam invertiran u vremenu (Algoritam 2):

Algoritam 2 – *Unazadni Algoritam*

1: **initialize** $s(t), t = T, a_{ij}, b_{jk}, O^T$
2: **for** $t \leftarrow t - 1$ **do**
3: $\quad \beta_i(t) = \sum_{j=1}^{c} \beta_j(t+1) a_{ij} b_{jk}^{O(t+1)}$
4: **end for** $t = 1$
5: **return** $P(O^T) \leftarrow \beta_i(0)$
6: **end**

U linij 5 β_i označava vjerojatnost da slijed počinje poznatim stanjem.

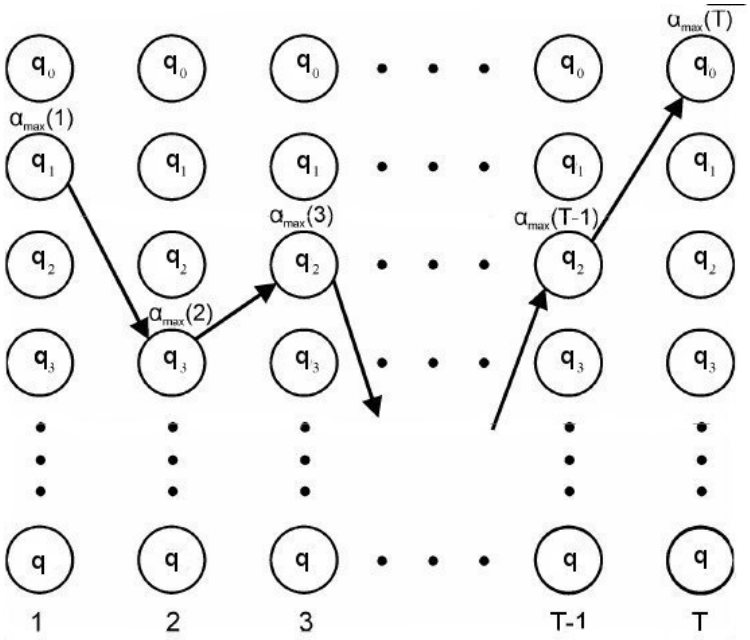
2.3.2. Problem dekodiranja

Rješavanjem problem dekodiranja trudimo se otkriti najvjerovatniji slijed skrivenih stanja koji su mogli producirati dati slijed opservacija. Koristeći jednotstavnu metodu, moglo bi razmišljati na način da se pronađe svaka moguća kombinacija stanja, tj. svaki mogući put kroz model i zatim, za svaku tu kombinaciju stanja, izračunati vjerojatnosti emitiranja datog slijeda opservacija. U tom slučaju, složenost bi opet bila $O(c^T * T)$, pa se primjenjuje jednostavniji algoritam poznat pod nazivom *Viterbijev algoritam* (Algoritam 3).

Algoritam 3 – *Viterbijev Algoritam*

1: **initialize** $Put = \{\}, t = 0$
2: **for** $t \leftarrow t + 1$ **do**
3: $\quad k \leftarrow 0$
4: $\quad$ **for** $k \leftarrow k + 1$ **do**
5: $\quad\quad \alpha_j(t) = b_{jk}^{O(t)} \sum_{i=1}^{c} \alpha_i(t-1) a_{ij}$
6: $\quad$ **end for** $k = c$
7: $\quad j = \underset{j}{\operatorname{argmax}} \alpha_j(t)$
8: $\quad$ dopuni Put sa s_j
9: **end for** $t = T$
10: **return** Put
11: **end**

U ovom slučaju razmišlja se na sličan način kao kod algoritma unaprijed, samo što



Slika 2.4: Ilustracija Viterbijevog algoritama

se u svakom vremenskom trenutku, umjesto sume računa umnožak$\alpha_j(t) * \alpha_{jt}$. Na taj način se u datom trenutku određuje najvjerovatnije stanje tj. stanje koje ima najveću vjerojatnost da bude posjećeno. Slijed tih stanja predstavlja najvjerovatniji put kroz model.

Složenost ovog algoritma je $O(c^2 * T)$. Neki algoritmi koriste logaritamsku vjerojatnosti ($log\alpha_i(t)$) i računaju ukupnu vjerojatnost kao sumu tih logaritama.

2.3.3. Problem treniranja

Osnovni zadatak treniranja skrivenih Markovljevih lanaca je određivanje parametara a_{ij} i b_{jk} koristeći skup uzoraka. Algoritam *Baum - Welch* ili *unaprijedno - unatražni algoritam* predstavlja specijalan slučaj algoritma *očekivanje - maksimizacija*. Osnova je u iterativnom ažuriranju parametara modela u cilju aproksimacije, odnosno približavanja opservacijskom trening slijedu. Veličine $\alpha_j(t)$ i $\beta_i(t)$ su samo procjene njihove stvarne vrijednost pošto u jednakostima (2.21) i (2.22) nisu poznate konkretne vrijednosti a_{ij} i b_{jk}. Moguće je zato izračunati poboljšane vrijednosti. Prvo je potrebeno definirati novu varijablu $\gamma_{ij}(t)$ kao vjerojatnost prelaza imeđu $q_i(t-1)$ i $q_j(t)$ pod uvjetom da je model generirao čitav trening slijed O^T, što daje:

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1)a_{ij}b_{jk}\beta_j(t)}{P(O^T|\lambda)} \tag{2.23}$$

gdje je $P(O^T|\lambda)$ vjerojatnost da je model generirao slijed O^T, sa bilo kojim prelaskom u koraku $t-1 \rightarrow t$.

Sada je moguće izračunati poboljšanu procjenu a_{ij}. Očekivani broj prelaza iz stanja s_i u stanje s_j, promatrano na cijelokupnom intervalu vremena T, jednak je $\sum_t^T \gamma_{ij}(t)$, dok je očekivani broj prelaza na s_j, iz bilo kojeg drugog stanja $\sum_t^T \sum_i \gamma_{ij}(t)$. Procjena a_{ij}, označena kao $\hat{a_{ij}}$, računa se kao odnos između očekivanog broja prelaza iz s_i u stanje s_j i ukupnog očekivanog broja prelaza u to stanje.

$$\hat{a_{ij}} = \frac{\sum_t^T \gamma_{ij}(t)}{\sum_t^T \sum_i \gamma_{ij}(t)} \tag{2.24}$$

Na sličan način dobiva se i poboljšana vrijednost parametra b_{jk}, označene kao $\hat{b_{jk}}$, računanjem odnosa između očekivanog broja posjećivanja stanja s_j, iz kojeg se emitira opservacija O_k i očekivanog ukupnog broja posjećivanja stanja s_j, sa emitiranjem bilo koje opservacije:

$$\hat{b_{jk}} = \frac{\sum_{t,s(t)=s_k}^T \sum_l \gamma_{jl}(t)|O_k}{\sum_t^T \sum_l \gamma_{jl}(t)} \tag{2.25}$$

Ukratko, počinje se sa grubo izabranim, proizvoljnim procjenama veličina a_{ij} i b_{jk}. Zatim se računaju poboljšane vrijednosti pomoću jednakosti (2.24) i (2.25) i postupak se ponavlja sve dok se ne dostegne određeni kriterij konvergencije, npr. dovoljno dobra točnost u smislu male promjene vrijednosti procjenjenih veličina, u dvije uzastopne iteracije.

Algoritam 4 – *Baum-Welch Algoritam*

1: **initialize** $a_{ij}, b_{jk}, O^T, z = 0$, kriterij konvergencije$\epsilon$
2: **repeat**
3: $z = z + 1$
4: racunanje $\hat{a}(z)$ iz $a(z-1)$ i $b(z-1)$ pomocu jednazbe (2.24)
5: racunanje $\hat{b}(z)$ iz $a(z-1)$ i $b(z-1)$ pomocu jednazbe (2.25)
6: $a_{ij}(z) = \hat{a_{ij}}(z-1)$
7: $b_{jk}(z) = \hat{b_{jk}}(z-1)$
8: **until** $\max_{i,j,k}[a_{ij}(z) - a_{ij}(z-1), b_{jk}(z) - a_{jk}(z-1)] < \epsilon$
9: **return** $a_{ij} = a_{ij}(z)$, $b_{jk} = a_{jk}(z)$
10: **end**

3. Automatsko prepoznavanje govora

Skriveni Markovljevi modeli nalaze svoju primjenu u raznim područjima računalne znanosti. Jedno od tih područja, gdje su najčešće korišteni, je područje statističkog prepoznavanja uzoraka. Kao jedno od tih područja, koji je i motiv ovog rada, je automatsko prepoznavanje govora (*eng. Automatic speech recognition - ASR*).
Formalno izražen, osnovni zadatak raspoznavanja uzoraka je problem pronalaženja decizijske funkcije g koja je u mogućnosti pridružit ulazni vektor $\overrightarrow{x} \in X \subseteq \Re^n$, gdje je X vektor značajki, jednom od prije poznatom, uzorku $c \in C$, gdje C je skup svih uzoraka koji se nalaze u sustavu.
Funkcija g je izvedena iz skupa već klasificiranih ulaznih vektora kojih zovemo skup za treniranje.

U statističkom prepoznavanju uzoraka decizijska funkcija sad može biti formulirana kao:

$$\hat{c} = g(\overrightarrow{x}) = \underset{c}{\operatorname{argmax}}\{p(c|\overrightarrow{x})\} \tag{3.1}$$

Jednažba nam kao rezultat vraća onaj uzorak koji maksimizira vjerojatnost $p(c|\overrightarrow{x})$.
Korištenjem *Bayesovog pravila* jednažbu (3.1) sad možemo zapisati kao:

$$\hat{c} = g(\overrightarrow{x}) = \underset{c}{\operatorname{argmax}}\{\frac{p(\overrightarrow{x}|c) * p(c)}{p(\overrightarrow{x})}\} \tag{3.2}$$

Kako nazivnik zapravo i nema bitnu ulogu, on se nalazi u Bayesovoj formuli samo da bi dobili razultati u rasponu $[0, 1]$, možemo ga zanemariti pa konačni izgled decizijske funkcije izgleda:

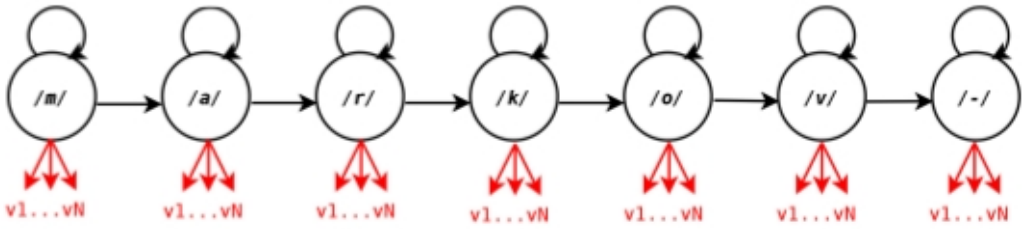
$$\hat{c} = g(\overrightarrow{x}) = \underset{c}{\operatorname{argmax}}\{p(\overrightarrow{x}|c) * p(c)\} \tag{3.3}$$

Ovu decizijsku funkciju možemo sad koristiti u stohatičkom automatskom prepoznavanu govora. U zadataku prepoznavanja izoliranih riječi, cilj je pridružiti akustičnom signalu pisanu riječ. Akustični signal prvo mora biti transformiran u u slijed akustičnih značajki $x_i^T = (x_1...x_T)$ u procesu nazvanom izlučivanje značajki. Vektor značajki

možemo zamislit kao odgovarajuće predočenje govornog signala podobno za korištenje u ovom problemu. Postoje mnoge metode koje se bave ovim problem, za naš zadatak koritit ćemo MFCC (*eng. Mel Frequency Cepstrum Coefficient*) metodu o kojoj će više biti riječi u poglavlju *3.1*.
Za raspoznavanje najvjerojatnije riječi $\overrightarrow{w}$ koju možemo dobiti iz vektora značajki, koritit ćemo preformulirano decizijsko pravilo (3.3):

$$\hat{w} = \underset{w}{\operatorname{argmax}}\{p(x_i^T|w) * p(w)\} \tag{3.4}$$

U izoliranom raspoznavanju riječi svaka riječ može se predočiti sa skrivenim Markovljevim modelima. U prepoznavanju govora određena riječ kao i njezine komponente nemogu biti direktno promatrane zbog toga što jedina informacija koju dobivamo je signal, tj. komponente su sakrivene i svaku komponentu možemo shvatiti kao stanje. Zapravo jedina vidljiva informacija koju možemo koristiti je vektor značajki, izlučen iz riječi, koji se promatrati i koji možemo vidjet kao emitiranje opservacijskih stanja iz određenom modela. Pogledajmo na primjeru. Riječ "markov" možemo predočiti kroz 7 stanja skrivenog Markovljevog lanca (slika 3.1)



Slika 3.1: Ilustracija riječi "markov" predočena Markovljevim modelom

Ovaj model je tipa *lijevo – desno* i često se koristi u prepoznavanju govora. Kroz njega je moguće kretanje, kroz stanja, samo unaprijed. Dodatno ovaj model ima završno stanje */-/* koje označava kraj riječi.
Skriveni Markovljev model definiran kao $\lambda = (A, B, \Pi)$, koristimo u decizijskom pravilu i sad jednažba (3.4) izgleda:

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmax}}\{p(x_i^T|\lambda) * p(\lambda)\} \tag{3.5}$$

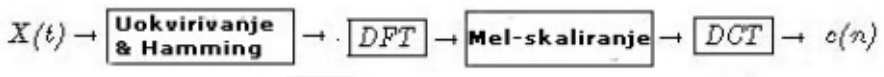
Kao što je vidljivo iz gornje jednažbe, problem pronalaženja riječi $\overrightarrow{w}$ je transformiran u problem pronalaženja najvjerojatnijeg modela $\hat{\lambda}$. Ovaj pristup donosi nam dva pitanja. Prvo, kako konstruirati model za različite riječi, i drugo, kako odrediti $p(\lambda)$ odnosno $p(x_i^T|\lambda)$.

Prvi problem riješavamo putem *Baum - Welch* algoritma. Prisjetimo se, ovaj algoritam može procjeniti sve tranzicijske vjerojatnosti kao i vjerojatnosti emitranja simbola koje su izračunate iz skupa za treniranje.
A-posteriori vjerojatnost $p(x_i^T|\lambda)$ za određeni model λ izvedena je uz pomoć *unaprijednog algoritma* ili *Viterbijevog algoritma*. Iako je prvi algoritam dobar za prepoznavanje izoliranih riječi, gdje se riječi sa modelom odnose jedan na jedan, u općenitom slučaju bolji je Viterbijev algoritam, koji radi prepoznavanje na temelju maksimalne vrijednosti optimalnog puta. Osim što je bolji kod prepoznavanja kontinuiranog govora, također je i puno brži.
Konačno *priori* vjerojatnost je dana sa tzv. modelom jezika. U raspoznavanju govora, model jezika može sadržavati informacije koje mogu dati vjerojatnost da određena riječ dolazi nakon prethodne riječi, ili vjerojatnost u okvirima određenog semantičkog konteksta. U raspoznavanju izoliranih riječi možemo izostaviti *priori* vjerojatnost u zadatku raspoznavanja.

3.1. MFCC

MFCC (*eng. Mel Frequency Cepstral Coefficients*) ili metoda Mel- frekvencijskih kepstralnih koeficijenata je metoda računanja parametara iz govornog signala. Ova metoda uzima u obzir karakterstike ljudskog slušnog sustava koje se upotrebljavaju u automatskim sustavima za prepoznavanje govora. Parametri su robusni i prilagodljivi promjenama govornika ili uvijetima snimanja. Zbog tih karakteristika MFCC metoda je odabrana kao metoda izlučivanja značajki.
Izračun MFF iz zvučnog signala sastoji se od sljedećih koraka:

1. Uokviravanje i množenje sa Hamming-ovim vremenskim otvorom
2. Furierova transformacija DFT
3. Mel-skaliranje poljem filtera
4. Inverzna kosinus transformacija DCT



Slika 3.2: Metoda MFCC

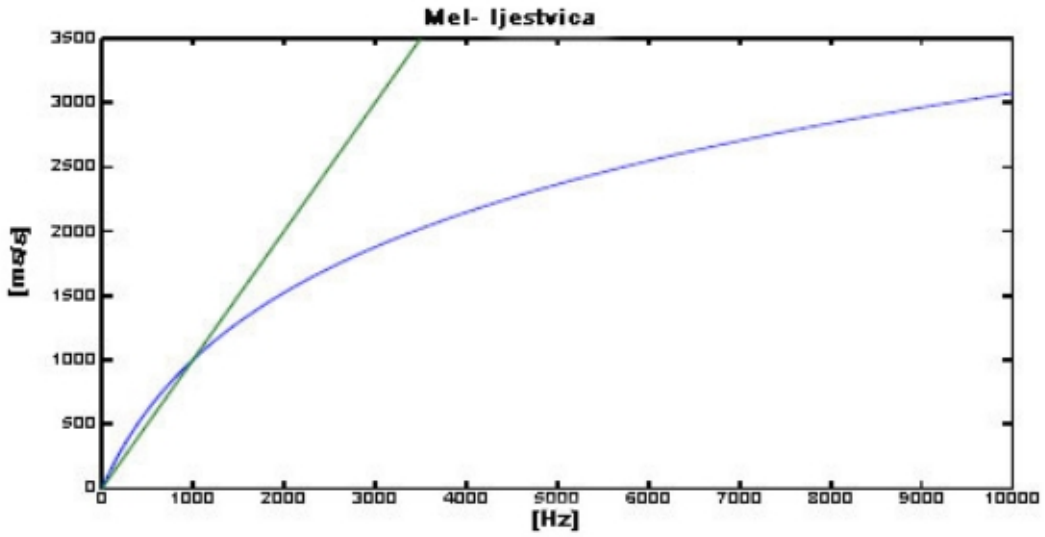
Na početku obrade, govor se dijeli u okvire koji imaju proizvoljan broj uzoraka signala. Kako bi izbjegli nagli prijelaz između okvira uvodi se preklapanje okvira, tj. naredni okvir pokrivaju dio informacija iz prethodnog okvira. Svi okviri se potom množe sa Hammingovim vremenskim otvorom i tako se izbjegavaju diskontinuiteti između rubova. Nakon toga se za svaki okvir Furierovom transformacijom DFT računa spektar signala. Spektar signala dan je formulom:

$$X(k) = \sum_{n=1}^{N} x(n) e^{-2j\Pi(k-1)(n-1)/N} \tag{3.6}$$

Optimalna duljina okvira je potencija broja 2.

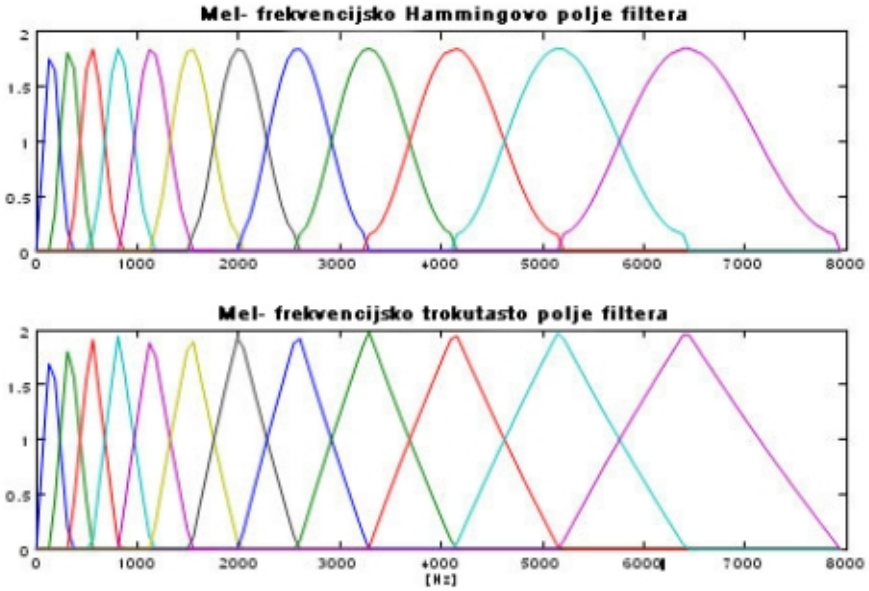
Nakon DFT transformacije dobiveni spektar se propušta kroz polje filtera. Filteri koji se koriste u ovoj metodi su konstruirani na osnovu ljudskog slušnog sustava. Dobra aproksimacija frekvencijske ljestvice u ljudskom slušnom sustavu predstavlja Mel-frekvencijska ljestvica.

$$m = 1127 * (1 * \frac{f}{700}) \tag{3.7}$$



Slika 3.3: Mel-frekvencijska ljestvica

Na osnovu Mel-frekvencijske ljestvice je konstruirano polje filetera (Slika 3.4). Polje je korišteno za propuštanje spektra dobijenog DFT transformacijom. Propuštanje signala je ostvarno množenjem vrijednosti filtera sa spektrom signala i zbrajanjem rezultata dobijenih množenjem. Konačni rezultat obrade je jedna vrijednost po fileru. Broj izlaznih vrijednosti ovog koraka je jednak broju filtera u polju.



Slika 3.4: Polje Mel-frekvencijskih filtera

Umjesto inverzne Furierove transformacije u posljednjem koraku obrade MFCC metodom, koristi se inverzna kosinus transformacija DCT koja rangira dobivene koeficijente prema važnosti.
Inverzna kosinus transformacija definirana je formulom:

$$y(k) = w(k) \sum_{n=1}^{N} *cos \frac{\Pi(2 * n - 1)(k - 1)}{2N}, k = 1, ..., N \tag{3.8}$$

gdje

$$w(k) = \sqrt{\frac{1}{N}}, k = 1 \tag{3.9}$$

$$w(k) = \sqrt{\frac{2}{N}}, 2 \leq k \leq N \tag{3.10}$$

0.-ti koeficijent dobivem ovom transformacijom se isključuje u daljnoj analizi. Iako nosi energiju, smatra se nepouzdanim. Ostali koeficijneti, kako je rečeno, rangirani su prema količini informacija koju nose, počevši od 1. koeficijenta.

4. Testiranje

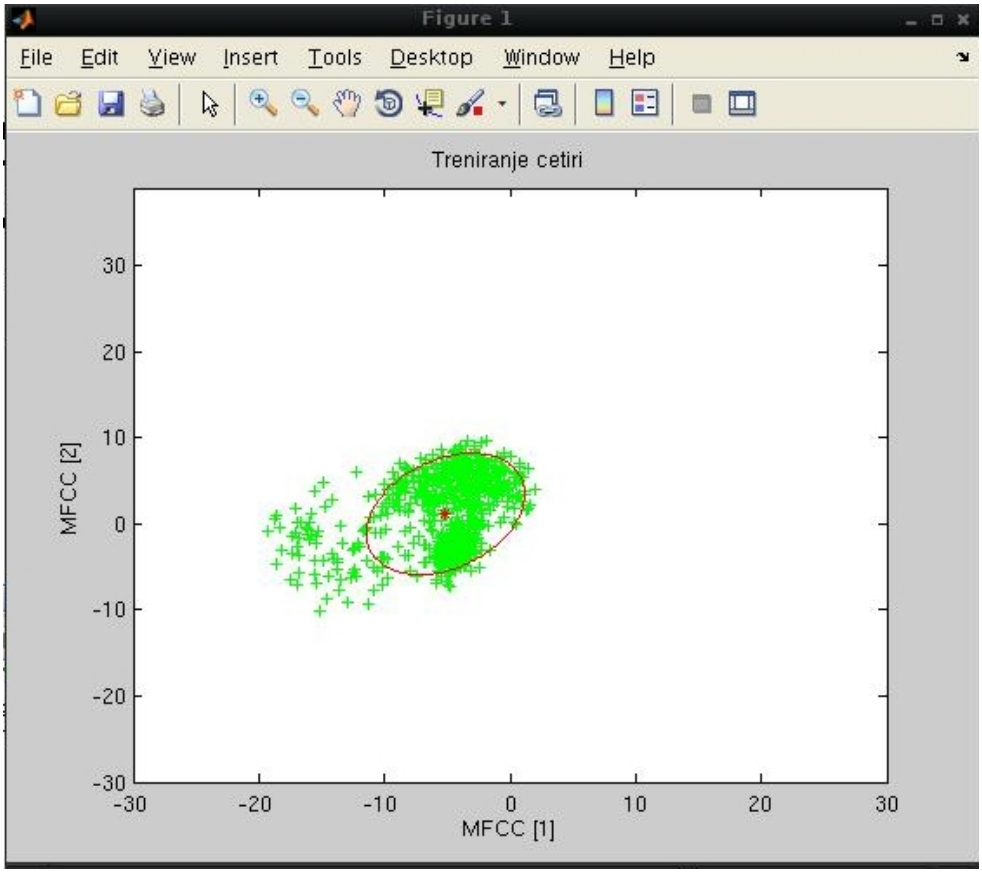
Testiranje je provedeno nad slijedećim uzorcima riječi: *jedan, dva, tri, četiri, pet, šest, sedam, osam, devet*. Ukupno je snimljeno 10 uzoraka za svaku skupinu, i samo testiranje se provodi *Leave-one-out* metodom. Ova metoda zasniva se na ideji da se $n - 1$ uzoraka koristi za treniranje i jedan za testiranje.

Za treniranje uzoraka korišten je *Baum - Welch algoritam*, dok je za samo testiranje korišten *unaprijdni algoritam*. Također korištena je i logaritamska metoda, pa kriterij odluke u ovom slučaju bit će najmanja logaritamska vrijednost koja se dobiva tesitranjem zadane riječi i svih unaprijed testiranih i generiranih modela.
Za kreiranje uzoraka korištena je frekvencija od $8000Hz$. Signal tijekom izlučivanja karaketeristika podjeljen je u prozore koji sadržavaju 80 uzoraka, to se podudara sa $10ms$, dok je za preklapanje uzoraka korišteno 20 uzoraka sa svake strane.

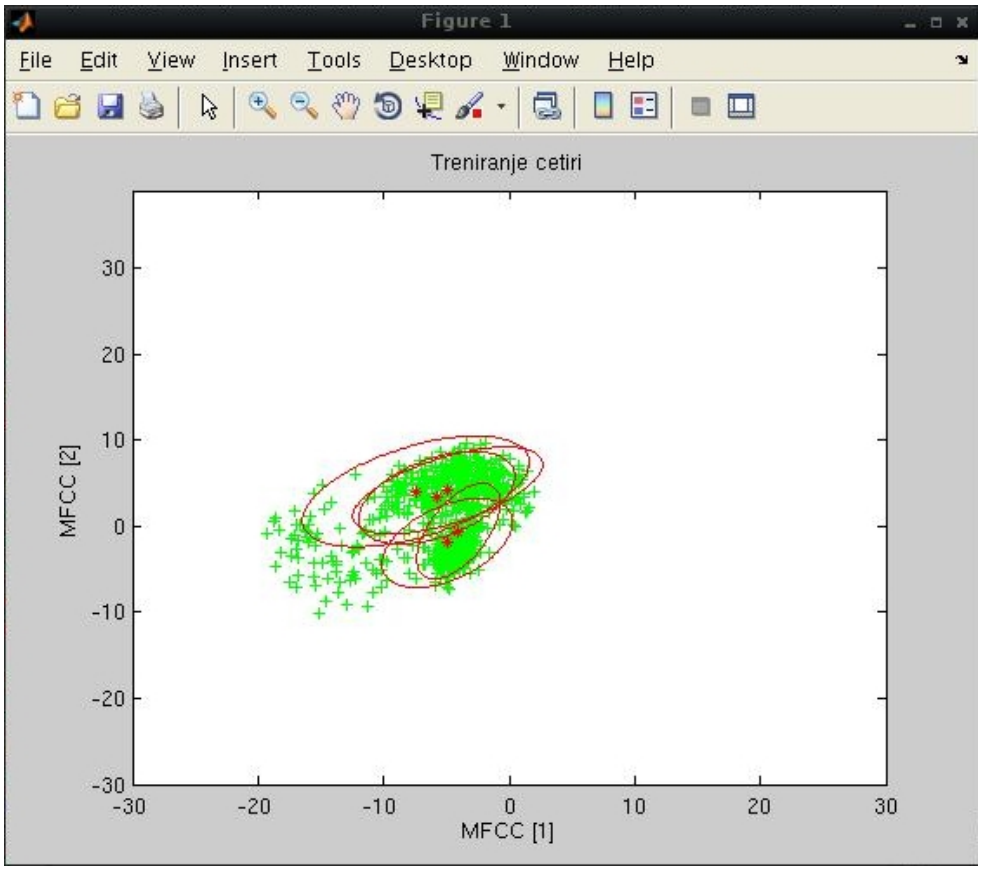
Za prvo testiranje koritit ćemo samo jedno stanje, tj. $N = 1$ što nam govori da će testirani skup biti definiran samo jednom Gaussovom distribucijom. Izgled konačnog treniranog modela za riječ *četiri* sa jednom distribucijom možemo vidjeti na slici *4.1*. Tablica *4.1* pokazuje nam logaritamske vrijednosti i odluke klasifikacije za ovaj slučaj.

Drugo testiranje provodili smo sa parametrom $N = 5$, izgled treniranog modela za riječ *četiri* sa pet distribucija možemo vidjeti na slici *4.2*. Tablica *4.2* pokazuje nam logaritamske vrijednosti i odluke klasifikacije za ovaj slučaj.

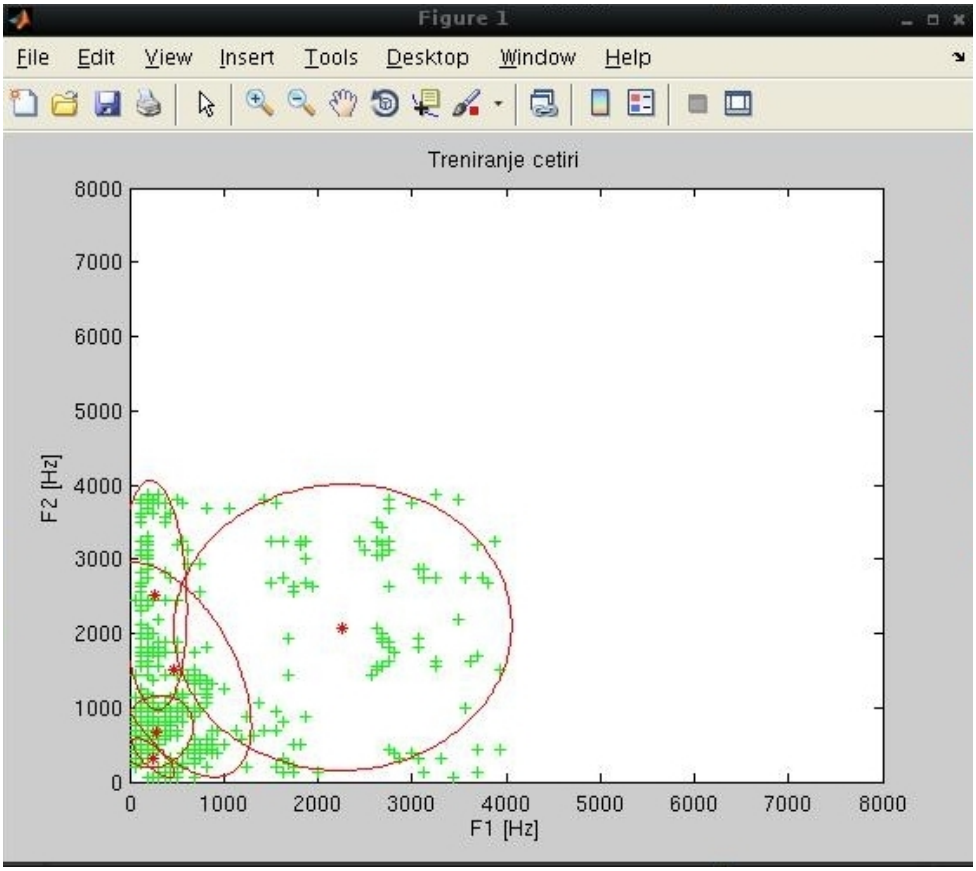
Za potrebe trećeg testiranja, metodu MFCC zamjenili smo jednostavnijom metod koja koja uzima značajke iz frekvencijske domene. Primjer crtanja prva dvije frekvencije za riječ *četiri* može se vidjeti na slici *4.3*, dok rezultate testiranja možemo vidjeti u tablici *4.3*.



Slika 4.1: Treniranje uz N=1, MFCC



Slika 4.2: Treniranje uz N=5, MFCC



Slika 4.3: Treniranje uz N=5, frekvencijska domena

$N = 1$	jedan	dva	tri	četiri	pet	šest	sedam	osam	devet
jedan	-3720.72	-5867.87	-5787.64	-5400.58	-4738.35	-6877.62	-4964.46	-6851.90	-4658.03
dva	-4731.69	-4389.99	-6304.26	-5858.47	-5535.50	-6737.03	-5775.08	-5762.78	-5689.59
tri	-4696.67	-7108.55	-4376.05	-4939.80	-4846.42	-6250.98	-5428.70	-7160.29	-5100.52
četiri	-4590.18	-6565.17	-5241.55	-4266.23	-4380.52	-5402.15	-5337.34	-6743.92	-5039.61
pet	-4818.50	-6670.75	-6024.91	-4882.91	-3878.68	-5674.59	-5310.81	-6975.95	-4856.36
šest	-4540.03	-6378.71	-5983.11	-4852.10	-4167.41	-4791.86	-5117.83	-6639.21	-4709.54
sedam	-3950.95	-5334.53	-5780.45	-5036.02	-4435.60	-5272.21	-4341.06	-5932.95	-4485.21
osam	-4426.94	-5183.53	-5641.15	-5858.38	-5176.14	-6047.48	-5120.75	-4891.78	-5238.74
devet	-4142.12	-6515.09	-5914.82	-5173.24	-4441.40	-5475.66	-4835.22	-6571.81	-4307.77
Odluka	jedan	dva	tri	četiri	pet	šest	sedam	osam	devet

Tablica 4.1: Testiranje sa N = 1, MFCC

$N = 5$	jedan	dva	tri	četiri	pet	šest	sedam	osam	devet
jedan	-3564.07	-7320.91	-5768.80	-6021.35	-5335.33	-7849.42	-5560.21	-7760.31	-4782.57
dva	-5242.45	-4116.12	-6892.49	-6210.00	-5742.06	-8255.33	-6151.16	-6651.68	-6076.12
tri	-5243.02	-7206.45	-4133.62	-5315.18	-5168.96	-6857.16	-5992.31	-8194.22	-5582.52
četiri	-4970.54	-6634.95	-5588.66	-4121.26	-4841.93	-5832.09	-5713.53	-7302.62	-5566.42
pet	-5217.17	-7449.33	-7343.82	-5315.64	-3829.36	-6068.13	-5610.58	-7601.82	-4979.50
šest	-5493.00	-6894.97	-6927.01	-6203.99	-4634.89	-4580.46	-6086.95	-7215.29	-5601.31
sedam	-4310.11	-6624.36	-6718.25	-5899.56	-4735.90	-5302.64	-3970.70	-6321.36	-4537.13
osam	-4857.92	-5945.11	-6419.47	-6478.45	-5573.31	-6713.86	-5522.75	-4565.42	-5552.02
devet	-4610.29	-7396.70	-5779.30	-5471.44	-4716.63	-5730.73	-5114.81	-7270.45	-4060.57
Odluka	jedan	dva	tri	četiri	pet	šest	sedam	osam	devet

Tablica 4.2: Testiranje sa N = 5, MFCC

Iz priloženih rezultate možemo zaključiti sljedeće. Korištenjem MFCC metode izlučivanja značajki dobivamo podudaranje 100%, tj. svi testni primjerci su dobro klasificirani. Sama razlika u korištenjem jedne ili pet gausovih distribucija (mješavina) je vidljiva u rezultatima dobivenim u logaritamskim vrijednostima. Dok kod $N = 1$ logaritamske vrijednosti će biti bliže željenom, dobrom rezultat, sa $N = 5$ dobivamo vrijednosti koje su više udaljenije željenom rezultatu. Iz ovih podataka zaključujemo možemo izvesti jednostavno pravilo: povećanjem stanja skrivenih markovljevih modela može se sigurnije i uspješnije detektirati željena riječ.

Također iz rezultata testiranja vidljivo je da je MFCC metoda puno uspješnija od jednostavnije metode izlučivanja značajki. Sa metodom koja se zasniva na frekvencijskoj domeni, tj. izlučivanje značajki iz frekvencijske domene samog signala, vidimo da smo za tri testna primjer krivo klasificirali riječ.

$N = 5$	jedan	dva	tri	četiri	pet	šest	sedam	osam	devet
jedan	-4976.49	-6001.13	-5992.16	-5906.27	-5448.62	-7275.42	-5715.44	-6121.42	-5709.73
dva	-5113.68	-5787.46	-6383.75	-6267.39	-5565.44	-7566.74	-5723.76	-6179.63	-5845.80
tri	-5112.04	-6108.23	-5949.24	-5948.17	-5631.38	-7209.65	-5805.50	-6193.77	-5940.29
četiri	-5002.21	-5970.76	-5917.78	-5642.74	-5382.04	-6549.35	-5745.97	-6063.01	-5725.72
pet	-5056.67	-6048.56	-6116.16	-5773.17	-5331.33	-6669.83	-5723.49	-6100.89	-5760.43
šest	-5068.40	-6080.48	-6130.85	-5785.06	-5423.32	-6561.63	-5769.33	-6114.32	-5760.01
sedam	-4992.48	-5986.54	-6076.23	-5741.77	-5367.19	-6593.89	-5626.00	-6064.30	-5706.94
osam	-5041.06	-5938.54	-6118.72	-5851.18	-5459.70	-6772.70	-5689.59	-6048.46	-5755.41
devet	-5030.13	-6010.66	-6176.67	-5826.18	-5365.69	-6699.08	-5669.74	-6086.14	-5750.31
Odluka	jedan	dva	**četiri**	četiri	pet	**četiri**	sedam	osam	**sedam**

Tablica 4.3: Testiranje sa N = 5, frekvencijska domena

5. Zaključak

Tema ovog završnog rada bila je pokazati praktično jednostavan sustav za prepoznavanje izoliranih riječi. Objašnjena je matematička podloga skrivenih Markovljevih modela kao i izlučivanje MFCC koeficijenata. U samom testiranju pažnja je bila usmjerena na pokazivanju ovisnosti broja stanja, tj. broja Gaussovih distribucija, sa točnošću dobivenih podataka. Vidjeli smo da korištenjem MFCC koeficijenata dobivamo sto postotnu točnost. Zadnje testiranje nam je pokazalo da je MFCC metoda daje točnije prepoznavanje za razliku od druge jednostavnije metode.
Iako naglasak nije bio na brzini izvođenja samog programa, možemo zaključiti da bi Viterbijev algoritam doprinio samoj brzini izvođenja programa.
Kao prijedlog nekih drugih načina, ideja sa kojima bi mogli ubrzati izvođenje programa možemo koristiti metode aktualne paralelizacije ili možda izvršavanja samih djelova programa na grafičkom procesoru (*eng. GPU*).

6. Dodatak - Matlab funkcije i skripte

main.m

```
clc
clear
close all

[audio_signals word_labels] = load_audio_from_folder('audio');
[test_signals word_test_lables] = load_audio_from_folder('test');

display(sprintf('Ucitano %d audio signala sljedecih rijeci:', length(audio_signals)))
display(unique(word_labels))

vocabulary = Vocabulary;

predicted_word_labels = vocabulary.train_test(audio_signals', word_labels', test_signals');
```

Vocabulary.m

```
classdef Vocabulary < handle
    properties
        words = {};
    end

    methods
        function add_word(self, word, audio_signals)
            new_word = Word(word);
            % ucitaj i treniraj
            new_word.train(extract_features(cell2mat(audio_signals)'));
            self.words.(char(word)) = new_word;
        end

        function prediction = recognize(self, audio_signal)
            max_likelihood = -Inf;
            likelihoods = [];

            for word = fieldnames(self.words)'
                likelihood = self.words.(char(word)).log_likelihood(extract_features(cell2mat(audio_s
                likelihoods = [likelihoods likelihood];

                display(sprintf('Testiranje sa %s [log %f ]',char(word),likelihood));

                if likelihood > max_likelihood
                    max_likelihood = likelihood;
```

```
                    prediction = word;
                end
            end
            display(sprintf('Prepoznata rijec kao %s!', char(prediction)))
            fclose(fid);
        end

        function predicted_word_labels = train_test(self, train_audio_signals, train_word_labels, tes
            [unique_word_labels, nothing , indices] = unique(train_word_labels);

            for i = 1:length(unique_word_labels)
                display(sprintf('Dodajem %s...', char(unique_word_labels(i))))
                self.add_word(unique_word_labels(i), train_audio_signals(indices == i))
            end

            predicted_word_labels = {};
            for test_audio_signal = test_audio_signals'
                predicted_word_labels(end + 1) = self.recognize(test_audio_signal);
            end
            predicted_word_labels = predicted_word_labels';
        end
    end

end
```

Word.m

```
classdef Word < handle
    properties
        N     =  1; % broj stanja
        A     = []; % NxN tranzicijska matrica
        prior = []; % Nx1 vektor pocetnih stanja
        mu    = []; % DxN vektor srednjih vrijednosti (D = broj spec. karakteristika )
        Sigma = []; % DxDxN konvarijaciska matrica
        name  = '';
    end

    methods
        function self = Word(name)
            self.name = char(name);
        end

        function log_likelihood = log_likelihood(self, observations)
            B = self.state_likelihood(observations);
            log_likelihood = forward(self, observations, B);
        end

        function [log_likelihood, alpha] = forward(self, observations, B)
            log_likelihood = 0;
            T = size(observations, 2);
            alpha = zeros(self.N, T);

            for t = 1:T
                if t == 1
                    % Inicijalizacija
```

```
            alpha(:, t) = B(:, t) .* self.prior;
        else
            % Indukcija
            alpha(:, t) = B(:, t) .* (self.A' * alpha(:, t - 1));
        end

        % Skaliranje
        alpha_sum      = sum(alpha(:, t));
        alpha(:, t)    = alpha(:, t) ./ alpha_sum;
        log_likelihood = log_likelihood + log(alpha_sum);
    end
end

function beta = backward(self, observations, B)
    T = size(observations, 2);
    beta = zeros(self.N, T);

    % Inicijalizacija
    beta(:, T) = ones(self.N, 1);

    for t = (T - 1):-1:1
        % Indukcija
        beta(:, t) = self.A * (beta(:, t + 1) .* B(:, t + 1));

        % Skalirannje
        beta(:, t) = beta(:, t) ./ sum(beta(:, t));
    end
end

% Evaluates the Gaussian pdfs for each state at the observations
% Returns a matrix containing B(s, t) = f(O_t | S_t = s)

function B = state_likelihood(self, observations)
    B = zeros(self.N, size(observations, 2));

    for s = 1:self.N
        B(s, :) = mvnpdf(observations', self.mu(:, s)', self.Sigma(:, :, s));
    end
end

function em_initialize(self, observations)
    % Random guessing
    self.prior = normalise(rand(self.N, 1));
    self.A     = mk_stochastic(rand(self.N));

    % All states start out with the empirical diagonal covariance
    self.Sigma = repmat(diag(diag(cov(observations'))), [1 1 self.N]);

    % Initialize each mean to a random data point
    indices = randperm(size(observations, 2));
    self.mu = observations(:, indices(1:self.N));
end

function train(self, observations)
```

```
    self.em_initialize(observations);

    for i = 1:9
        log_likelihood = self.em_step(observations);
        display(sprintf('Korak %02d: log_likelihood = %f', i, log_likelihood))
        self.plot_gaussians(observations);
    end
end

function log_likelihood = em_step(self, observations)
    B = self.state_likelihood(observations);
    D = size(observations, 1);
    T = size(observations, 2);

    [log_likelihood, alpha] = self.forward(observations, B);
    beta                    = self.backward(observations, B);

    xi_sum = zeros(self.N, self.N);
    gamma  = zeros(self.N, T);

    for t = 1:(T - 1)
        xi_sum      = xi_sum + normalise(self.A .* (alpha(:, t) * (beta(:, t + 1) .* B(:, t +
        gamma(:, t) = normalise(alpha(:, t) .* beta(:, t));
    end

    gamma(:, T) = normalise(alpha(:, T) .* beta(:, T));

    expected_prior = gamma(:, 1);
    expected_A     = mk_stochastic(xi_sum);

    expected_mu    = zeros(D, self.N);
    expected_Sigma = zeros(D, D, self.N);

    gamma_state_sum = sum(gamma, 2);

    gamma_state_sum = gamma_state_sum + (gamma_state_sum == 0);

    for s = 1:self.N
        gamma_observations = observations .* repmat(gamma(s, :), [D 1]);
        expected_mu(:, s)  = sum(gamma_observations, 2) / gamma_state_sum(s);

        expected_Sigma(:, :, s) = symmetrize(gamma_observations * observations' / gamma_state_
                                             expected_mu(:, s) * expected_mu(:, s)');
    end

    expected_Sigma = expected_Sigma + repmat(0.01 * eye(D, D), [1 1 self.N]);

    % M-step
    self.prior = expected_prior;
    self.A     = expected_A;
    self.mu    = expected_mu;
    self.Sigma = expected_Sigma;
end
```

```
        function plot_gaussians(self, observations)
            % crtamo samo prve dvije dimenzije (prikaz)
            type = 1;

            plot(observations(1, :), observations(2, :), 'g+')
            hold on
            plot(self.mu(1, :), self.mu(2, :), 'r*')

            for s = 1:size(self.Sigma, 3)
                error_ellipse(self.Sigma(1:2, 1:2, s), 'mu', self.mu(1:2, s), 'style', 'r-', 'conf',
            end
            if type == 2

                axis([0 8000 0 8000])
                hold off
                title(sprintf('Treniranje %s', self.name))
                xlabel('F1 [Hz]')
                ylabel('F2 [Hz]')
                drawnow
            elseif type == 1
                axis([-30 30 -30 39])
                hold off
                title(sprintf('Treniranje %s', self.name))
                xlabel('MFCC [1]')
                ylabel('MFCC [2]')
                drawnow
            end
            pause
        end
        %pause
    end
end
```

extract features.m

```
function framefrequencies = extract_features(sound)
    type =1;

    if type == 1
        framefrequencies = mfcc(sound)';

    elseif type == 2
        Fs = 8000;
        framesize = 80;
        overlap = 20;
        D = 6; % broj frekvencija u svakom vremenskom odsjecku

        frames = buffer(sound, framesize, overlap);
        w = hamming(framesize);
        [nothing2, N] = size(frames);
        framefrequencies = zeros(D, N);

        i = 1;
        NFFT = 2^nextpow2(framesize);
        for frame = frames
```

```
            x = frame .* w;
            X = fft(x, NFFT)/framesize;
            f = Fs/2*linspace(0, 1, NFFT/2 + 1);
            [ nothing , peaklocations] = findpeaks(abs(X(1:(NFFT/2 + 1))),'SORTSTR', 'descend');%

            if isempty(peaklocations)
                peaklocations = ones(D, 1);
            elseif length(peaklocations) < D
                peaklocations = padarray(peaklocations, D - length(peaklocations), 1, 'post');
            end

           framefrequencies(:, i) = f(peaklocations(1:D))';
            i = i + 1;
        end
    end
end
```

mfcc.m

```
function ceps = mfcc(signal)
bank=mel(24,256,8000,0,0.4,'m'); %24
bank=full(bank);
bank=bank/max(bank(:));

for k=1:12 % 12
  n=0:23;
  dctcoef(k,:)=cos((2*n+1)*k*pi/(2*24));
end

w = 1 + 6 * sin(pi * [1:12] ./ 12);
w = w/max(w);

xx=double(signal);
xx=filter([1 -0.9375],1,xx);

xx=enframe(xx,256,80);

for i=1:size(xx,1)
  y = xx(i,:);
  s = y' .* hamming(256);
  t = abs(fft(s));
  t = t.^2;
  c1=dctcoef * log(bank * t(1:129));
  c2 = c1.*w';
  m(i,:)=c2';
end

dtm = zeros(size(m));

for i=3:size(m,1)-2
  dtm(i,:) = -2*m(i-2,:) - m(i-1,:) + m(i+1,:) + 2*m(i+2,:);
end
dtm = dtm / 3;

ceps = [m dtm];
```

```
ceps = ceps(3:size(m,1)-2,:);
```

load audio from folder.m

```
function [audio_signals, word_labels] = load_audio_from_folder(audio_folder)
    audio_signals = {};
    word_labels = {};

    for word_folder = struct2cell(dir(audio_folder))
        for word_file = struct2cell(dir(sprintf('%s/%s/*.wav', audio_folder, char(word_folder(1)))))
            file_path = sprintf('%s/%s/%s', audio_folder, char(word_folder(1)), char(word_file(1)));

            audio_signals(end + 1) = {wavread(file_path)};
            word_labels(end + 1) = word_folder(1);
        end
    end
end
```

LITERATURA

1. Rabiner L. A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition
2. Kohlschein C. An introduction to Hidden Markov Models
3. Duda R., Hart P., Stork D. Pattern Classification
4. Chou W., Juang B. Pattern Recognition in Speech and Language Processing
5. Abdulla W., Kasabov N. The Concepts of Hidden Markov Model in Speech Recognition
6. Sigurdsson S., Petersen K. MFCC: An Evolution of Robustness of MP3 Encoded Music
7. Hasan R., et al. Speaker identification using MFCC

Izvedba funkcija za skrivene Markovljeve modele

Sažetak

Završni rad bavio se problematikom prepoznavanja uzoraka korištenjem MFCC koeficijenata i skrivenih Markovljevih modela. Svrha rada bila je prakična primjena jednostavnog programa koji bi bio u stanju detektirati izolirane riječi. Nakon implementiranja spomenutog alata, napravljeno je testiranje sa skupom uzoraka i pokazana je ovisnost skupa stanja kroz koje prolazi klasifikator sa točnošću dobivenih podataka.

Ključne riječi: Prepoznavanje uzoraka, skriveni Markovljevi modeli, MFCC

Function implementation for Hidden Markov Models

Abstract

The purpose of this paper was to create simple tools which can be used to identify isolated words. After realisation in Matlab, testings were made and there was shown correlation between markove model states and correctness of evaluated data.

Keywords: Pattern recognition, HMM, MFCC