Universiteit Utrecht

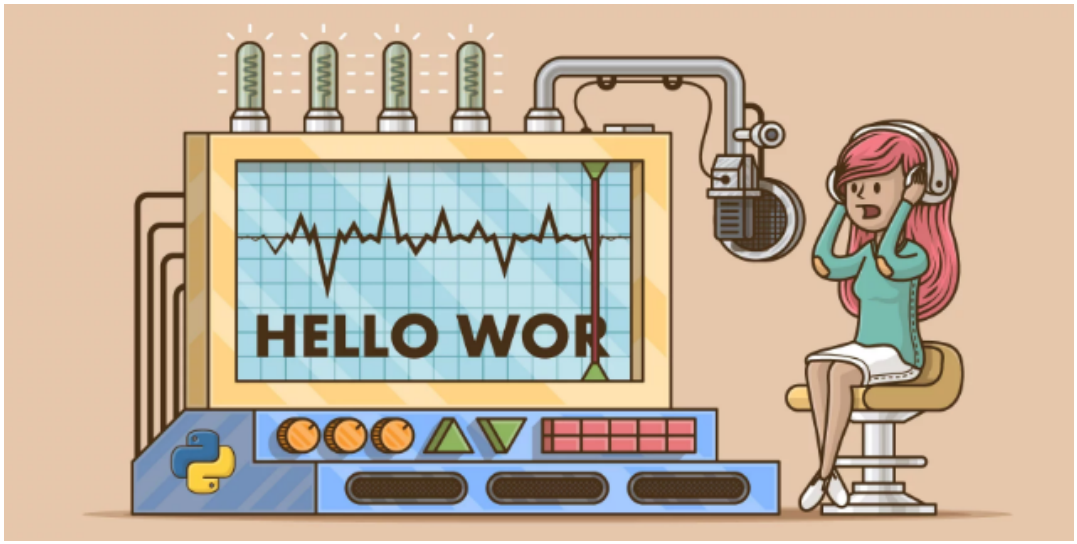
Wiskunde

Bachelorscriptie

Verborgen Markov modellen en hun toepassing bij spraakherkenning

Auteur
Jenthe Visscher
5637813

Begeleider
Dr. K. Dajani



11 januari 2019

Verborgen Markov modellen en hun toepassing bij spraakherkenning

Hoewel verborgen Markov modellen al in de jaren 50 en 60 geïntroduceerd en bestudeerd werden, is de populariteit ervan de laatste jaren erg gestegen. Een reden hiervoor is de grote toepasbaarheid van de modellen bij alledaagse toepassingen. Een alledaagse toepassing is spraakherkenning. De verborgen Markov modellen en de toepassing bij spraakherkenning zullen in deze scriptie beschreven worden.

Graag wil ik mijn begeleider dr. Karma Dajani bedanken voor haar hulp en ondersteuning bij het schrijven van mijn scriptie en in het bijzonder dat ze dit op afstand wilde doen.

Jenthe Visscher

Inhoudsopgave

Hoofdstuk 1

Introductie

Spraakherkenning in de digitale wereld is tegenwoordig een veelgebruikt fenomeen. Zo kan bij Google ook gezocht worden naar het gezochte, door in te spreken in plaats van te typen in de zoekbalk. De laatste jaren is er veel ontwikkeling geweest op het gebied van spraakherkenningssystemen. De systemen zijn steeds geavanceerder en beter geworden, maar de basis ervan is altijd het verborgen Markov model geweest. In deze scriptie gaan we deze verborgen Markov modellen uitgebreid bestuderen. Vervolgens maken we een uitstapje naar een toepassing ervan: spraakherkenning.

Als basis van de scriptie is *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition* gebruikt. We hebben dit document grondig bestudeerd en elementen er uit gekozen om te bespreken in de scriptie. Verschillende onderdelen uit dit document hebben we verder onderzocht en bewijzen voor gegeven.

In hoofdstuk 2 zullen we de Markovketen bespreken. Dit is het simpelste geval van een Markov model en ligt ten grondslag aan het verborgen Markov model. We geven de definitie en eigenschappen van de Markovketen aan de hand van een voorbeeld.

Daarna zullen we in hoofdstuk 3 het verborgen Markov model behandelen. Hiervoor beginnen we met een intuïtieve definitie en we geven een voorbeeld ter illustratie. Vervolgens bespreken we de elementen waaruit een verborgen Markov model bestaat en definiëren we deze elementen. In de rest van het verslag zullen we de elementen zoals gedefinieerd gebruiken. Ook introduceren we drie problemen die erg belangrijk zijn bij het gebruik van verborgen Markov modellen. De modellen zijn pas nuttig bij toepassingen als die problemen zijn opgelost. We laten zien wat de problemen zijn en hoe ze op een recursieve manier opgelost kunnen worden. Ook geven we een bewijs voor de manier van oplossen. Tot slot hebben we bij het voorbeeld uit het begin van het hoofdstuk voor ieder van de drie problemen een MATLAB-code geschreven die het oplost. Deze code hebben we bijgevoegd en zullen we bespreken in dit hoofdstuk.

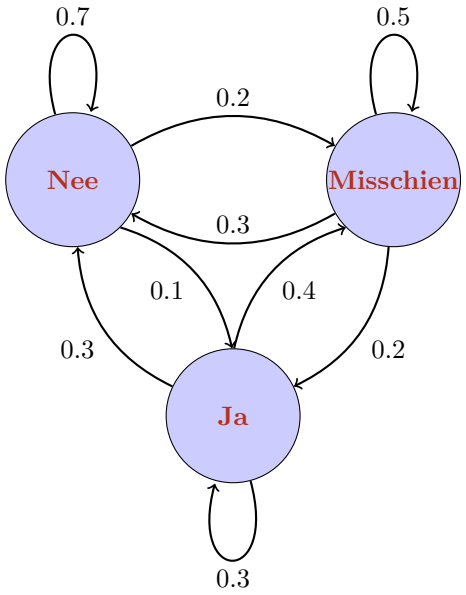
In hoofdstuk 4 wordt de toepassing van verborgen Markov modellen bij spraakherkenning behandeld. We laten zien hoe een (simpel) spraakherkenningssysteem werkt en wat de relatie hiervan is met een discreet verborgen Markov model. Verder leggen we uit hoe een spraaksignaal wordt omgezet naar een observatiereeks die gebruikt kan worden bij een discreet verborgen Markov model met behulp van Feature Analyse en Vector Kwantisering.

Hoofdstuk 2

Markov ketens

We beginnen met het simpelste geval van een Markov model: de Markovketen. De Markovketen is vernoemd naar de Russische wiskundige Andrej Markov. Het beschrijft stapsgewijs hoe een proces zich door verschillende toestanden beweegt. Dit gebeurt met behulp van de kans op een overgang naar een (andere) toestand. We leggen het principe uit aan de hand van een voorbeeld.

Stel iemand krijgt een reeks met vragen gesteld, waarop hij ja, nee of misschien kan antwoorden. Hij baseert zijn antwoorden echter niet op de waarheid, maar laat het afhangen van het antwoord dat hij op de vorige vraag heeft gegeven. Dit principe is in het onderstaande figuur schematisch weergegeven.



In het figuur is te zien dat hij met kans 0.3 een vraag met "Nee" beantwoordt als hij de vorige vraag met "Ja" beantwoord heeft. Dit is slechts één van de mogelijkheden. De andere overgangen werken op dezelfde manier.
Als het antwoord dat hij geeft alleen afhangt van het antwoord dat hij op de vorige vraag gegeven

heeft, noemen we dit proces een Markovketen.

Als we een formele definitie van de Markovketen willen geven, hebben we eerst de definitie van een stochastisch proces nodig.

Definitie (*Stochastisch proces*): Een stochastisch proces is een collectie $\{X_t, t \in T\}$ van stochastische variabelen. Hierbij is t de tijd, T de collectie van alle mogelijke tijdstippen en X_t de toestand van het proces op tijdstip t.

Definitie (*Discrete Markovketen*): Een discrete Markovketen is een stochastisch proces $\{X_t, t = 0, 1, 2, 3, ...\}$ met de mogelijke waarden van X_t uit een eindige of aftelbare verzameling M waarvoor geldt:

$$\mathbb{P}(X_{t+1} = j | X_t = i, X_{t-1} = i_{t-1}, ..., X_1 = i_1, X_0 = i_0) = \mathbb{P}(X_{t+1} = j | X_t = i) \tag{2.1}$$

voor alle $j, i, i_0, ..., i_{t-1} \in M$ en alle $t \geq 0$.

We bekijken alleen de Markovketens waarbij het niet uit maakt op welk tijdstip de transitie van toestand i naar toestand j plaatsvindt. Dit noemt men ook wel homogene Markovketens en we gebruiken hiervoor de volgende notatie:

$$\mathbb{P}(X_{t+1} = j | X_t = i) = p_{ij},$$

waarbij p_{ij} dus de kans is om van toestand i naar toestand j te gaan op ieder willekeurig tijdstip. Van alle mogelijke transities kunnen we een transitiematrix maken. Wanneer we in het voorbeeld uit het begin stellen: Nee=0, Misschien=1 en Ja=2 wordt de transitiematrix A van het voorbeeld:

$$A = \begin{pmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & p_{22} \end{pmatrix} = \begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.5 & 0.2 \\ 0.3 & 0.4 & 0.3 \end{pmatrix}. \tag{2.2}$$

Voordat we deze Markovketen kunnen gebruiken, moeten we een begintoestand $\pi_0 = (\pi_{0,1}, \pi_{0,2}, \pi_{0,3})^T$ definiëren waarbij $\pi_{0,i} = \mathbb{P}(X_0 = i)$ voor $i = 1, 2, 3$.

Uit de wet van totale waarschijnlijkheid volgt, dat:

$$\pi_{1,i} = \mathbb{P}(X_1 = i) = \sum_{j=1}^{3} \mathbb{P}(X_1 = i | X_0 = j)\mathbb{P}(X_0 = j) = p_{i1}\pi_{0,1} + p_{i2}\pi_{0,2} + p_{i3}\pi_{0,3}$$

In de laatste stap is goed te zien dat bovenstaande gelijk is aan de i^e component van de vector die volgt uit de matrixvermenigvuldiging van A met de begintoestand π_0. Al deze componenten vormen samen de vector π_1 die de kansverdeling van de toestand op tijdstip 1 weergeeft. Hiervoor geldt dus dat $\pi_1 = \pi_0 \cdot A$. We kunnen dit veralgemeniseren naar $\pi_n = \pi_{n-1} \cdot A$. Hiervoor geldt namelijk dezelfde wet van totale waarschijnlijkheid en hierbij kunnen we dezelfde berekening uitvoeren. Als we laatste vergelijking vaker toepassen zien we dat $\pi_n = \pi_0 \cdot A^n$. We kunnen dus met alleen een bekende begintoestand en de transitiematrix de toestand van een paar stappen verderop berekenen.

Hoofdstuk 3

Verborgen Markov modellen

Nu bekijken we een speciaal soort Markov model: het verborgen Markov model. Dit model bestaat uit twee verschillende processen. Het eerste proces is een Markovketen zoals we in het vorige hoofdstuk gezien hebben. Dit proces bepaalt de toestand van het gehele proces. Het verschil met de eerdere Markovketen is dat nu niet bekend is wat de transitiematrix van het proces is of zelfs in welke toestand het proces zich op tijdstip t bevindt. Wat wel waarneembaar is, is de uitkomst van het tweede proces. Dit proces vindt plaats in één van de toestanden van het model. De uitkomst van dit proces wordt ook wel een observatie genoemd. De observaties worden bepaald door een kansverdeling die per toestand verschilt. Deze kansverdeling hoeft niet per se een Markovketen te zijn. Hij hangt alleen af van de huidige toestand waarin het proces zich bevindt. Per stap wordt dus eerst, volgens een niet waarneembare proces, een toestand bepaald. Vervolgens wordt in die specifieke toestand een observatie bepaald, waarbij de observatie wel waarneembaar is. Om de definitie nog wat duidelijker te maken, geven we een simpel voorbeeld.

Anna en Bob spelen een spelletje. Bij dit spelletje pakt Anna meerdere keren achter elkaar stiekem een knikker uit één van drie verschillende vazen (genummerd 0,1 en 2) en deze knikker laat ze aan Bob zien. Bob ziet dus niet uit welke vaas Anna de knikker pakt. Hij ziet wel de kleur van de knikker die gepakt wordt. Er zijn knikkers in vijf verschillende kleuren (rood, blauw, geel, wit en zwart) en in iedere vaas zit een andere mix van knikkers. In vaas 0 is de verdeling bijvoorbeeld respectievelijk 2-3-2-1-4, in vaas 1 is de verdeling 5-1-1-1-3 en in vaas 2 is het 1-3-2-4-1. Anna kiest niet geheel willekeurig uit welke vaas ze knikkers gaat pakken, maar ze laat het afhangen van de vorige vaas en ze kiest de volgende vaas dan met behulp van een transitiematrix. Dit is bijvoorbeeld matrix 2.2 zoals bij ons eerdere voorbeeld. Welke vazen Anna voor de vorige vaas gekozen heeft, is ze weer vergeten dus deze hebben geen invloed. Verder begint ze sowieso bij vaas nummer 0. Uit voorgaande is af te leiden dat Anna de vazen kiest volgens een Markovproces. Dit is de Markovketen van het verborgen Markov model en de vazen zijn dus de verschillende toestanden. De kansverdeling in iedere aparte toestand is in dit geval dus de kans op een bepaalde kleur knikker. De waarneembare observatie is de kleur van de gepakte knikker.

3.1 Elementen

Nu we een beeld hebben van wat een verborgen Markov model is, kunnen we de verschillende elementen die nodig zijn voor zo'n model definiëren.

1. N, dit is het aantal toestanden van het model. De verschillende toestanden noteren we als $S = \{S_1, S_2, ..., S_N\}$. In het voorbeeld waren dit de verschillende vazen. De toestand op tijdstip t wordt door q_t aangegeven
2. M, dit is het aantal mogelijke verschillende observatie symbolen per toestand. De verschillende observatie symbolen worden genoteerd als $V = \{v_1, v_2, ..., v_M\}$. In het voorbeeld waren dit de verschillende kleuren knikkers. De observatie op tijdstip t wordt met O_t aangegeven.
3. $A = \{a_{ij}\}$, dit is de transitiematrix van de verschillende toestanden van het model. Er geldt dus $a_{ij} = \mathbb{P}(q_{t+1} = S_j | q_t = S_i)$ voor $1 \leq i, j \leq N$.
4. $B = \{b_j(k)\}$, dit is de kansverdelingsdistributie in toestand j. Er geldt $b_j(k) = \mathbb{P}(O_t = v_k | q_t = S_j)$ met $1 \leq j \leq N$ en $1 \leq k \leq M$.
5. $\pi = \{\pi_i\}$, dit is de begintoestand, waarbij geldt $\pi_i = \mathbb{P}(q_1 = S_i)$.

Dit is de benodigde parameterverzameling om een verborgen Markovmodel te construeren. Het is te zien dat N en M ook in A en B zijn opgenomen. De belangrijkste parameters zijn dus A,B en π. We gebruiken hiervoor ook wel de notatie $\lambda = (A, B, \pi)$.

Met behulp van bovenstaande elementen kunnen we een observatiereeks $O = (O_1, O_2, ..., O_T)$ genereren. Kies hiervoor een begintoestand $q_1 = S_i$ volgens π. In S_i kiezen we nu de observatie $O_1 = v_k$ volgens $b_i(k)$. Deze observatie wordt in de reeks geplaatst. Nu laten we een transitie plaatsvinden van q_1 naar q_2 volgens A. Hier kiezen we weer een nieuwe observatie volgens de kansverdeling van die toestand. Nu vindt er weer een transitie van toestand plaats en zo gaan we door tot we bij tijdstip T zijn.

3.2 De 3 basisproblemen van verborgen Markov modellen

Omdat een deel van een verborgen Markov model niet bekend is, zijn ze niet direct goed te gebruiken in de praktijk. Er zijn 3 problemen die opgelost moeten worden om het model nuttig te maken. De problemen zijn als volgt:

- **Probleem 1** Stel dat de observatiereeks $O = O_1O_2...O_T$ en de modelparameters $\lambda = (A, B, \pi)$ gegeven zijn. We zoeken nu een efficiënte manier om $\mathbb{P}(O|\lambda)$ te berekenen. $\mathbb{P}(O|\lambda)$ geeft weer hoe goed het model λ bij de observatiereeks past.
- **Probleem 2** Stel dat de observatiereeks $O = O_1O_2...O_T$ en het model λ gegeven zijn dan zoeken we een reeks van toestanden $Q = q_1q_2...q_T$ die het best past bij de gegeven observatiereeks. We proberen dus het verborgen deel van het model zichtbaar te maken. Omdat dat deel onbekend is, bestaat er geen correct antwoord, maar wel een die het beste past.
- **Probleem 3** Stel dat de observatiereeks $O = O_1O_2...O_T$ gegeven is dan zoeken we λ zodat $\mathbb{P}(O|\lambda)$ gemaximaliseerd wordt. We zoeken dus een model dat de observatiereeks het best beschrijft.

Wanneer we deze problemen oplossen, krijgen we een beter beeld van hoe het verborgen Markov model eruit ziet en kunnen we hem gebruiken bij praktische toepassingen en in het bijzonder bij spraakherkenning. Hoe de toepassing van verborgen Markov modellen op spraakherkenning precies in zijn werk gaat, komen we in hoofdstuk 4 op terug. In het huidige hoofdstuk leggen we uit hoe we bovenstaande problemen kunnen oplossen of benaderen. Belangrijke elementen bij het oplossen van de problemen zijn de voorwaartste en achterwaartse benadering.

3.3 Voorwaartse/Achterwaartse benadering

Bij de voorwaartste en achterwaartse benadering maken we gebruik van de eigenschap van een Markovketen dat de toestand q_{t+1} alleen van de vorige toestand afhangt dus voor alle s en t geldt dat q_t onafhankelijk is van q_s als $s \neq t-1, t, t+1$ Verder geldt dat de observaties onafhankelijk zijn van elkaar dus voor $s \neq t$ geldt dat O_s onafhankelijk is van O_t. De observaties zijn alleen afhankelijk van de laatste voorgaande toestand. Dit komt, omdat de kansverdeling van de observatie O_t wordt bepaald door q_t dus daar is O_t afhankelijk van, maar q_t wordt weer bepaald door q_{t-1} dus als q_t niet gegeven is, is O_t ook afhankelijk van q_{t-1}. Verder maken we bij de uitleg gebruik van de definitie van de conditionele kans, met name in de vorm

$$\mathbb{P}(X) = \frac{\mathbb{P}(X \bigcap Y)}{\mathbb{P}(Y|X)},$$

waarbij X en Y gebeurtenissen zijn. Tot slot maken we bij beide benaderingen gebruik van de parameters $\lambda = (A, B, \pi)$.

3.3.1 Voorwaartse benadering

Voor de voorwaartse benadering definiëren we de variabele

$$\alpha_t(i) = \mathbb{P}(O_1 O_2 ... O_t, q_t = S_i|\lambda).$$

We zoeken een manier om deze variabele voor alle t en i op een recursieve manier te berekenen. Dit doen we volgens de volgende stelling.

Stelling 3.3.1: Voor alle $1 \leq i \leq N$ geldt dat

$$\alpha_1(i) = \pi_i b_i(O_1) \tag{3.1}$$

en voor alle $1 \leq t \leq N$ en $1 \leq i, j \leq N$ geldt dat

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}). \tag{3.2}$$

Bewijs: Voor $t = 1$ geldt dat:

$$\alpha_1(i) = \mathbb{P}(O_1, q_1 = S_i|\lambda) = \mathbb{P}(O_1|q_1 = S_i, \lambda)\mathbb{P}(q_1 = S_i|\lambda) = b_i(O_1)\pi_i.$$

Nu gaan we de tweede vergelijking van de stelling bewijzen. Er geldt dat:

$$\begin{aligned}
\alpha_{t+1}(j) &= \mathbb{P}(O_1O_2...O_tO_{t+1}, q_{t+1} = S_j|\lambda) \\
&= \sum_{i=1}^{N} \mathbb{P}(O_1O_2...O_tO_{t+1}, q_{t+1} = S_j, q_t = S_i|\lambda) \\
&= \sum_{i=1}^{N} \mathbb{P}(O_{t+1}, q_{t+1} = S_j|O_1O_2...O_t, q_t = S_i, \lambda)\mathbb{P}(O_1O_2...O_t, q_t = S_i|\lambda) \\
&= \sum_{i=1}^{N} \mathbb{P}(O_{t+1}|q_{t+1} = S_j, q_t = S_i, \lambda)\mathbb{P}(q_{t+1} = S_j|q_t = S_i)\mathbb{P}(O_1O_2...O_t, q_t = S_i|\lambda) \\
&= \sum_{i=1}^{N} \alpha_t(i)a_{ij}b_j(O_{t+1}).
\end{aligned}$$

In bovenstaande geldt in de derde stap dat we de reeks $O_1...O_t$ uit
$\mathbb{P}(O_{t+1}, q_{t+1} = S_j|O_1O_2...O_t, q_t = S_i, \lambda)$ en in de vierde stap q_{t+1} uit
$\mathbb{P}(O_{t+1}|q_{t+1} = S_j, q_t = S_i, \lambda)$ kunnen wegstrepen wegens de eerdergenoemde onafhankelijkheid. Hiermee is de stelling bewezen.

Met behulp van de formule voor $\alpha_{t+1}(i)$ kunnen we nu op een recursieve manier de waarden van $\alpha_1(i)$ tot $\alpha_T(j)$ voor alle $1 \leq i, j \leq N$ berekenen. Hiervoor beginnen we voor iedere toestand i de waarde voor $\alpha_1(i)$ te berekenen. Met behulp van deze gegevens en de formule kunnen we nu voor alle toestanden j de waarde van $\alpha_2(j)$ berekenen. Zo kunnen we verder gaan tot we T bereikt hebben.

3.3.2 Achterwaartse benadering

Voor de achterwaartse benadering definiëren we de volgende variabele

$$\beta_t(i) = \mathbb{P}(O_{t+1}O_{t+2}...O_T|q_t = S_i, \lambda) \text{ voor } t = 1, ..., T-1.$$

en

$$\beta_T(i) = 1$$

We zoeken nu een manier om deze variabele voor alle t en i op een recursieve manier te berekenen. We doen dit met behulp van de volgende stelling.

Stelling 3.3.2: Stel dat voor $t = T$ en $1 \leq i \leq N$ per definitie geldt

$$\beta_T(i) = 1. \tag{3.3}$$

dan geldt voor $t = T-1, T-2, ..., 1$ en $1 \leq i, j \leq N$. dat

$$\beta_t(i) = \sum_{j=1}^{N} b_j(O_{t+1})\beta_{t+1}(j)a_{ij}. \tag{3.4}$$

Bewijs: We maken voor het bewijs gebruik van het feit dat $\sum_{j=1}^{N} a_{ij} = 1$. Er geldt nu dat:

$$\begin{aligned}
\beta_t(i) &= \mathbb{P}(O_{t+1}O_{t+2}...O_T|q_t = S_i, \lambda) \\
&= \sum_{j=1}^{N} \mathbb{P}(O_{t+1}O_{t+2}...O_T, q_{t+1} = S_j|q_t = S_i, \lambda) \\
&= \sum_{j=1}^{N} \mathbb{P}(O_{t+1}|O_{t+2}...O_T, q_{t+1} = S_j, q_t = S_i, \lambda)\mathbb{P}(O_{t+2}...O_T, q_{t+1} = S_j|q_t = S_i, \lambda) \\
&= \sum_{j=1}^{N} \mathbb{P}(O_{t+1}|O_{t+2}...O_T, q_{t+1} = S_j, q_t = S_i, \lambda)\mathbb{P}(O_{t+2}...O_T|q_{t+1} = S_j, q_t = S_i, \lambda)\mathbb{P}(q_{t+1} = S_j|q_t = S_i, \lambda) \\
&= \sum_{j=1}^{N} \mathbb{P}(O_{t+1}|q_{t+1} = S_j, \lambda)\mathbb{P}(O_{t+2}...O_T|q_{t+1} = S_j, \lambda)\mathbb{P}(q_{t+1} = S_j|q_t = S_i, \lambda) \\
&= \sum_{j=1}^{N} b_j(O_{t+1})\beta_{t+1}(j)a_{ij}.
\end{aligned}$$

Hiermee is de stelling bewezen. Met behulp van deze formule kunnen we ook op een recursieve manier alle waarden van $\beta_T(i)$ tot $\beta_1(j)$ berekenen voor alle waarden van $1 \leq j \leq N$.

3.4 Het oplossen van probleem 1

Zoals genoemd in sectie 3.2 is probleem 1: Stel dat de observatiereeks $O = O_1O_2...O_T$ en de modelparameters $\lambda = (A, B, \pi)$ gegeven zijn. We zoeken nu een efficiënte manier om $\mathbb{P}(O|\lambda)$ te berekenen.
Voor het oplossen van probleem 1 maken we gebruik van de de voorwaartse benadering. Er geldt namelijk dat:

$$\mathbb{P}(O|\lambda) = \sum_{i=1}^{N} \mathbb{P}(O_1O_2...O_T, q_T = S_i|\lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

Voor de oplossing dienen we dus volgens de voorwaartse benadering $\alpha_T(i)$ te berekenen voor alle $1 \leq i \leq N$.

3.4.1 Voorbeeld met MATLAB

Ter verdere illustratie van het oplossen van probleem 1 geven we een voorbeeld. Hiervoor hanteren we hetzelfde model als bij het voorbeeld in hoofdstuk 2. Het spelletje tussen Anna en Bob. Hierbij zijn namelijk de parameters van het model bekend. Als observatiereeks gebruiken we hiervoor de reeks blauw-rood-rood-zwart-wit-rood-geel. Omdat het om een recursie gaat die veel rekenwerk in beslag neemt, zullen we voor de berekeningen MATLAB gebruiken. De code die we hiervoor geschreven hebben, wordt hieronder beschreven.

We beginnen met het definiëren van het model in MATLAB. Dit zijn de twee kansverdelingen, de begintoestand en de observatiereeks. De code hiervoor ziet er als volgt uit:

Listing 3.1: Het model

```
A=[0.7, 0.2, 0.1;0.3, 0.5, 0.2; 0.3, 0.4, 0.3]
N=3
B=[1/6, 1/4, 1/6, 1/12, 1/3; 5/11, 1/11, 1/11, 1/11, 3/11;
    1/11, 3/11, 2/11, 4/11, 1/11];
M=5
pi=[1,0,0]
O=[2,1,1,5,4,1,3]
T=length(O)
```

Ook de kansverdeling per toestand is hierin als een matrix weergegeven. De rij staat hiervoor voor de toestand (de vaas)waarin het proces zich bevindt en in de verschillende kollomen staan de verschillende kleuren van de knikkers. Voor het programmeren hebben we de kleuren van de knikkers genummerd als rood=1, blauw=2, geel=3, wit=4 en zwart=5.

Nu gaan we over naar de initialisatie van de voorwaartse benadering. De code hiervoor is:

Listing 3.2: Initialisatie van $\alpha_t(i)$

```
a=zeros(1,N)
i=1
while i<N+1
    alpha=pi(i)*B(i,O(1));
    a(i)=alpha;
    i=i+1;
end
```

Er wordt dus een vector **a** gedefinieerd waarin $\alpha_1(i)$ voor alle verschillende waardes voor i staan. Deze waardes worden berekend met behulp van de formule (3.1) voor de initiële waardes van de recursie. In het geval van het voorbeeld geldt $(\alpha_1(1), \alpha_1(2), \alpha_1(3)) = (0.25, 0, 0)$. Nu we deze hebben berekend, kunnen we de recursie zelf gaan uitvoeren en vanuit $\alpha_T(i)$ de kans $\mathbb{P}(O|\lambda)$ berekenen. De code is:

Listing 3.3: Recursie van $\alpha_t(i)$

```
t=2
while (t<T+1)
    j=1
    while (j<N+1)
        x=0
        i=1
        while (i<N+1)
            Alpha=B(j,O(t))*a(i)*A(i,j)
            x=x+Alpha
            i=i+1
        end
        a(j)=x
        j=j+1
    end
    t=t+1
```

```
end
i=1
P=0
while  i<N+1
      P=P+a(i)
      i=i+1
end
```

In de code bekijken we per tijdstap t een vector met als componenten $\alpha_t(j)$ voor de verschillende waardes van j. Deze verschillende waardes van $\alpha_t(j)$ worden weer met behulp van een while-loop over de verschillende waardes van i berekend met behulp van de formule (3.2). Uiteindelijk volgt hieruit een vector die voor de verschillende i de $\alpha_T(i)$ voorstelt. In het geval van het voorbeeld is deze vector $1.0 \cdot 10^{-5}(0.1100, 0.0204, 0.0042)$. De elementen van deze vector worden nu bij elkaar opgeteld en dan verkrijgen we $\mathbb{P}(O|\lambda) = 1.3452 \cdot 10^{-6}$. Dit is inderdaad een erg kleine kans, maar voor een reeks van lengte 7 zoals we nu als voorbeeld hebben genomen, zijn erg veel mogelijkheden waardoor de kans op één speciale reeks erg klein is. Wanneer we een kortere observatiereeks, bijvoorbeeld rood-blauw-wit, is met behulp van de code te zien dat de kans op deze observatiereeks bij het gegeven model gelijk is aan 0,002 dus al een stuk groter.

3.5 Het oplossen van probleem 2

Zoals genoemd in sectie 3.2 luidt probleem 2: Stel dat de observatiereeks $O = O_1O_2...O_T$ en het model λ gegeven zijn dan zoeken we een reeks van toestanden $Q = q_1q_2...q_T$ die het best past bij de gegeven observatiereeks.

In tegenstelling tot probleem 1 is dit probleem niet exact op te lossen. Er bestaat namelijk niet één goede oplossing, maar alleen een die het beste past. Er zijn verschillende manieren waarop een optimale reeks van toestanden bij een observatiereeks gekozen kan worden. Namelijk per q_t apart de optimale bepalen of in groepjes, maar de meest gebruikte manier is de totale reeks $Q = (q_1 = S_{i_1}, ..., q_T = S_{i_T})$ nemen en dan $\mathbb{P}(Q|O, \lambda)$ over Q te maximaliseren. Hierbij geldt dat:

$$\max_Q \mathbb{P}(Q|O, \lambda) = \max_Q \frac{\mathbb{P}(Q, O|\lambda)}{\mathbb{P}(O|\lambda)} = \mathbb{P}(O|\lambda) \max_Q \mathbb{P}(Q, O|\lambda).$$

We kunnen dus ook $\mathbb{P}(Q, O|\lambda)$ maximaliseren om tot hetzelfde resultaat te komen. Een manier om dit te doen is volgens het Viterbi Algoritme. Hiervoor definiëren we:

$$\delta_t(i) = \max_{i_1, i_2, ..., i_{t-1}} \mathbb{P}(q_1 = S_{i_1}, q_2 = S_{i_2}, ..., q_t = S_i, O_1O_2...O_t|\lambda).$$

Deze functie geeft dus de hoogste kans weer van een reeks $q_1, ..., q_t$ op tijdstip t die eindigt in S_i. De bijbehorende reeks noemen we dan de optimale reeks van toestanden. We willen $\delta_t(i)$ op een recursieve manier kunnen oplossen. Dit doen we met behulp van de volgende stelling.

Stelling 3.5: Voor $1 \leq i \leq N$ geldt dat

$$\delta_1(i) = \pi_i b_i(O_1) \tag{3.5}$$

en voor alle $1 \leq t \leq T$ en $1 \leq i, j \leq N$ geldt dat

$$\delta_{t+1}(j) = \max_i \delta_t(i) a_{ij} \cdot b_j(O_{t+1}). \tag{3.6}$$

Bewijs: Voor $\delta_1(i)$ geldt dat

$$\delta_1(i) = \mathbb{P}(q_1 = S_i, O_1|\lambda) = \mathbb{P}(q_1 = S_i|\lambda)\mathbb{P}(O_1|q_1 = S_i, \lambda) = \pi_i b_i(O_1).$$

Dus hiermee is het eerste deel van de stelling bewezen. Voor het tweede deel geldt dat.

$$\begin{aligned}
\delta_{t+1}(j) &= \max_{i_1,i_2,...,i_{t-1},i} \mathbb{P}(q_1 = S_{i_1}, q_2 = S_{i_2}, ..., q_t = S_i, q_{t+1} = S_j, O_1O_2...O_{t+1}|\lambda) \\
&= \max_{i_1,i_2,...,i_{t-1},i} \mathbb{P}(q_1 = S_{i_1}, q_2 = S_{i_2}, ..., q_t = S_i, O_1O_2...O_t|q_{t+1} = S_j, O_{t+1}, \lambda)\mathbb{P}(O_{t+1}, q_{t+1} = S_j|\lambda) \\
&= \max_{i_1,i_2,...,i_{t-1},i} \mathbb{P}(q_1 = S_{i_1}, q_2 = S_{i_2}, ..., q_t = S_i, O_1O_2...O_t|\lambda)\mathbb{P}(O_{t+1}|q_{t+1} = S_j, \lambda)\mathbb{P}(q_{t+1} = S_j|\lambda) \\
&= \max_{i_1,i_2,...,i_{t-1},i} \mathbb{P}(q_1 = S_{i_1}, q_2 = S_{i_2}, ..., q_t = S_i, O_1O_2...O_t|\lambda)\mathbb{P}(q_{t+1} = S_j|\lambda)b_j(O_{t+1}) \\
&= \max_i \max_{i_1,i_2,...,i_{t-1}} \mathbb{P}(q_1 = S_{i_1}, q_2 = S_{i_2}, ..., q_t = S_i, O_1O_2...O_t|\lambda)\mathbb{P}(q_{t+1} = S_j|\lambda)b_j(O_{t+1})
\end{aligned}$$

Verder geldt dat

$$\begin{aligned}
a_{ij} &= \mathbb{P}(q_{t+1} = S_j|q_t = S_i, \lambda) \\
&= \frac{\mathbb{P}(q_t = S_i|q_{t+1} = S_j, \lambda)\mathbb{P}(q_{t+1} = S_j|\lambda)}{\mathbb{P}(q_t = S_i|\lambda)} \\
&= \frac{\mathbb{P}(q_t = S_i|\lambda)\mathbb{P}(q_{t+1} = S_j|\lambda)}{\mathbb{P}(q_t = S_i|\lambda)} \\
&= \mathbb{P}(q_{t+1} = S_j|\lambda).
\end{aligned}$$

Uit bovenstaande volgt nu dat

$$\delta_{t+1}(j) = \max_i \delta_t(i)a_{ij}b_j(O_{t+1}),$$

waarmee de stelling is bewezen.

Met behulp van de formule voor $\delta_{t+1}(i)$ kunnen we de kans van de optimale reeks van toestanden bepalen, maar we willen natuurlijk de reeks toestanden zelf weten. We moeten hiervoor het argument van het maximum bepalen. De optimale reeks noemen we $\psi_t(j)$ en hiervoor geldt:

$$\psi_t(j) = \operatorname*{argmax}_i \delta_{t-1}(i)a_{ij}b_j(O_{t+1}) = \operatorname*{argmax}_i \delta_{t-1}(i)a_{ij}.$$

We zien dat $b_j(O_{t+1})$ niet afhankelijk is van i, dus het maximum niet beïnvloedt. Vanwege deze onafhankelijkheid is het argument van het maximum onafhankelijk van $b_j(O_{t+1})$. We kunnen $b_j(O_{t+1})$ weg laten. Nu we de formule voor de optimale reeks op tijdstip t weten kunnen we ook de optimale reeks aan het einde van het proces bepalen. Dit doen we door eerst $\delta_T(i)$ te bepalen voor $1 \leq i \leq N$ volgens (3.6) en daarna te bepalen voor welke i $\delta_T(i)$ maximaal is. Deze i bepaalt de laatste toestand van de reeks en dan hebben we de optimale reeks q_T^* gevonden. We kunnen dit ook schrijven als

$$q_T^* = \operatorname*{argmax}_{1 \leq i \leq N} \delta_T(i)$$

Nu volgt de rest te achterhalen door terug te rekenen. Dit gebeurt op de volgende manier:

$$q_t^* = \psi_t(q_{t+1}^*) \text{ voor } t = T-1, T-2, ..., 1$$

We weten nu dus welke reeks van toestanden $\mathbb{P}(Q, O|\lambda)$ en dus ook $\mathbb{P}(Q|O, \lambda)$ maximaliseert en hebben hiermee de meest waarschijnlijke oplossing gevonden voor het tweede probleem.

3.5.1 Voorbeeld met MATLAB

Net als in sectie 3.4 gaan we met behulp van MATLAB weer het voorbeeld uit hoofdstuk 3 behandelen. Echter veranderen we nu de begintoestand in $\pi = (0.5, 0.3, 0.2)$. Het definiëren van het model gaat zoals in de vorige sectie. Vervolgens programmeren we de initialisatie van het probleem.

Listing 3.4: Initialisatie van $\delta_t(i)$

```
d=zeros(1,N)
i=1
while i<N+1
    delta=pi(i)*B(i,O(1));
    d(i)=delta;
    i=i+1;
end
```

Voor iedere i wordt dus $\delta_1(i)$ berekend met behulp van (3.5). Nu gaan we over naar de recursie van $\delta_t(i)$ en $\psi_t(i)$.

Listing 3.5: Recursie van $\delta_t(i)$ en $\psi_t(i)$

```
x=zeros(1,N)
psi=zeros(N,T)
t=2
j=1
while t<T+1
    j=1
    while j<N+1
        i=1
        while i<N+1
            delta=d(i)*A(i,j)*B(j,O(t))
            d(i)=delta
            i=i+1
        end
        x(j)=max(d)
        y=find(d==x(j))
        psi(j,t-1)=y
        j=j+1
    end
    t=t+1
end
```

Hierin worden de $\delta_t(i)$ en $\psi_t(i)$ berekend voor alle verschillende i, j en t met behulp van (3.6). Hierbij wordt de vector d met alle $\delta_t(i)$ voor de verschillende waarden van i telkens aangepast, waarbij in tijdstap $t + 1$ nog wel de waarden van d uit tijdstap t worden gebruikt. Van de waarden van $\psi_t(j)$ wordt een matrix gemaakt met op plaats (j, t) de i die het maximum $d_t(j)$ veroorzaakt. Tot slot moeten we nu nog bepalen voor welke i de $\delta_T(i)$ maximaal is en vanuit daar terugrekenen om een reeks te verkrijgen.

Listing 3.6: Bepalen van $\max_i \delta_T(i)$ en reeks q^*

```
P=max(x)
Q=find(x==P)
q=zeros(1,T)
q(T)=Q
t=T-1
while t>0
    q(t)=psi(q(t+1),t)
    t=t-1
end
```

Hierin geldt $P = \max_i \delta_t(i)$. Ook wordt eerst $q_T = Q$ berekend. Met behulp van een while-loop wordt daarna de reeks q^* terugberekend. De reeks van toestanden die hieruit volgt bij het gegeven voorbeeld is $q^* = (3, 3, 3, 3, 3, 3, 1)$. De werkelijke onbekende reeks van toestanden kan hier uiteraard van afwijken, maar de meest waarschijnlijke reeks is dus dat het proces in toestand 3 begint, daar een tijdje blijft en eindigt in toestand 1.

3.6 Het oplossen van probleem 3

In sectie 3.2 hebben we gezien dat probleem 3 is: Stel dat de observatiereeks $O = O_1O_2...O_T$ gegeven is dan zoeken we λ zodat $\mathbb{P}(O|\lambda)$ gemaximaliseerd wordt.

Net als bij probleem 2 is het niet mogelijk om de exacte oplossing te vinden. Het is namelijk niet mogelijk om $\mathbb{P}(O|\lambda)$ globaal te maximaliseren. Wel kunnen we het lokaal maximaliseren door een Lagrangiaan van de functie te gebruiken en hierop een iteratie toe te passen. Omdat het moeilijk is om $\mathbb{P}(O|\lambda)$ (partieel) te differentiëren en we daardoor geen maximum kunnen bepalen, passen we eerst de onderstaande stelling toe.

Stelling 3.6.1: Stel dat een observatiereeks O, een toestandreeks Q, een model λ en een alternatief model $\bar{\lambda}$ gegeven zijn zodanig dat

$$\sum_Q \mathbb{P}(Q|O,\lambda)\log(\mathbb{P}(Q,O|\bar{\lambda})) > \sum_Q \mathbb{P}(Q|O,\lambda)\log(\mathbb{P}(Q,O|\lambda)).$$

Dan geldt ook dat

$$\mathbb{P}(O|\bar{\lambda}) > \mathbb{P}(O|\lambda).$$

Bewijs: We maken gebruik van het feit dat $\sum_Q \mathbb{P}(Q|O,\lambda) = 1$. Hieruit volgt dat

$$\begin{aligned}
&\log \mathbb{P}(O|\bar{\lambda}) - \log \mathbb{P}(O|\lambda) \\
&= \sum_Q \mathbb{P}(Q|O,\lambda) \log \mathbb{P}(O|\bar{\lambda}) - \sum_Q \mathbb{P}(Q|O,\lambda) \log \mathbb{P}(O|\lambda) \\
&= \sum_Q \mathbb{P}(Q|O,\lambda) \log(\frac{\mathbb{P}(O,Q|\bar{\lambda})}{\mathbb{P}(Q|O,\bar{\lambda})} - \sum_Q \mathbb{P}(Q|O,\lambda) \log(\frac{\mathbb{P}(O,Q|\lambda)}{\mathbb{P}(Q|O,\lambda)} \\
&= \sum_Q \mathbb{P}(Q|O,\lambda) \log(\frac{\mathbb{P}(O,Q|\bar{\lambda})}{\mathbb{P}(Q|O,\bar{\lambda})} + \sum_Q \mathbb{P}(Q|O,\lambda) \log(\frac{\mathbb{P}(Q|O,\lambda)}{\mathbb{P}(O,Q|\lambda)} \\
&= \sum_Q \mathbb{P}(Q|O,\lambda)(\log \frac{\mathbb{P}(O,Q|\bar{\lambda})}{\mathbb{P}(Q|O,\bar{\lambda})} + \log \frac{\mathbb{P}(Q|O,\lambda)}{\mathbb{P}(O,Q|\lambda)}) \\
&= \sum_Q \mathbb{P}(Q|O,\lambda)(\log(\mathbb{P}(O,Q|\bar{\lambda}) + \log \frac{1}{\mathbb{P}(Q|O,\bar{\lambda})} + \log \mathbb{P}(Q|O,\lambda) + \log \frac{1}{\mathbb{P}(O,Q|\lambda)}) \\
&= \sum_Q \mathbb{P}(Q|O,\lambda)(\log \frac{\mathbb{P}(O,Q|\bar{\lambda})}{\mathbb{P}(O,Q|\lambda)} + \log \frac{\mathbb{P}(Q|O,\lambda)}{\mathbb{P}(Q|O,\bar{\lambda})}) \\
&= \sum_Q \mathbb{P}(Q|O,\lambda) \log \frac{\mathbb{P}(O,Q|\bar{\lambda})}{\mathbb{P}(O,Q|\lambda)} + \sum_Q \mathbb{P}(Q|O,\lambda) \log \frac{\mathbb{P}(Q|O,\lambda)}{\mathbb{P}(Q|O,\bar{\lambda})}
\end{aligned}$$

De rechtersommatie kunnen vermenigvuldigen met $\frac{\mathbb{P}(Q|O,\bar{\lambda})}{\mathbb{P}(Q|O,\bar{\lambda})}$, en we kunnen stellen $f(X) = X\log(X)$ met $X \geq 0$. We zien dan dat de rechtersommatie gelijk is aan:

$$\sum_Q f(\frac{\mathbb{P}(Q|O,\lambda)}{\mathbb{P}(Q|O,\bar{\lambda})})\mathbb{P}(Q|O,\bar{\lambda})$$

Voordat we verder gaan met het bewijs, geven we eerst de definitie van een convexe funtie.

Definitie:*(Convexe functie)* Een functie $f(x)$ is convex op een bepaald interval I als voor iedere twee willekeurige punten $x, y \in I$ geldt dat voor $t \in [0,1]$

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y).$$

Voor $f(x) = x\log(x)$ met $x \geq 0$ geldt dat de functie stijgend is voor alle waarden uit het bereik. Hieruit volgt dat log()

Nu gaan we gebruik maken van de ongelijkheid van Jensen. Deze ongelijkheid zegt dat als $f(x)$ een convexe functie is en we a_i met $i = 1, ..., n$ hebben zodat $a_1 + ... + a_n = 1$ en $a_i \geq 0$ dan geldt dat $\sum_{i=1}^N a_i f(t_i) \geq f(\sum_{i=1}^N a_i x_i)$. Aangezien we te maken hebben met een kansverdeling en $f(X) = X\log(X)$ een convexe functie is, geldt dat:

$$\sum_Q f(\frac{\mathbb{P}(Q|O,\lambda)}{\mathbb{P}(Q|O,\bar{\lambda})})\mathbb{P}(Q|O,\bar{\lambda}) \geq f(\sum_Q \frac{\mathbb{P}(Q|O,\lambda)}{\mathbb{P}(Q|O,\bar{\lambda})}\mathbb{P}(Q|O,\bar{\lambda})) = f(\sum_Q \mathbb{P}(Q|O,\lambda)) = f(1).$$

Hierbij geldt nu dat $f(1) = 1\log(1) = 0$. Dit betekent dat de rechtersommatie groter is dan of gelijk is aan 0. Hieruit volgt dat:

$$\sum_Q \mathbb{P}(Q|O,\lambda) \log \frac{\mathbb{P}(O,Q|\bar{\lambda})}{\mathbb{P}(O,Q|\lambda)} + \sum_Q \mathbb{P}(Q|O,\lambda) \log \frac{\mathbb{P}(Q|O,\lambda)}{\mathbb{P}(Q|O,\bar{\lambda})} \geq \sum_Q \mathbb{P}(Q|O,\lambda) \log \frac{\mathbb{P}(O,Q|\bar{\lambda})}{\mathbb{P}(O,Q|\lambda)}.$$

Waaruit we de conclusie kunnen trekken dat:

$$\log \mathbb{P}(O|\bar{\lambda}) - \log \mathbb{P}(O|\lambda) \geq \sum_Q \mathbb{P}(Q|O,\lambda) \log \mathbb{P}(O,Q|\bar{\lambda}) - \sum_Q \mathbb{P}(Q|O,\lambda) \log \mathbb{P}(O,Q|\lambda).$$

Dus als de rechterkant van de vergelijking groter dan 0 is, betekent dat dat de linkerkant van de vergelijking ook groter dan 0 is. Nu volgt dat

$$\log \mathbb{P}(O|\bar{\lambda}) \geq \log \mathbb{P}(O|\lambda)$$

En hieruit kunnen we de conclusie trekken dat dan ook geldt

$$\mathbb{P}(O|\bar{\lambda}) \geq \mathbb{P}(O|\lambda),$$

waarmee de stelling is bewezen.

We willen nu $\sum_Q \mathbb{P}(Q|O,\lambda) \log(\mathbb{P}(Q,O|\bar{\lambda}))$ met behulp van de Lagrangiaan maximaliseren over $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$. Hieruit komt geen absoluut maximum, maar alleen een lokaal maximum gebaseerd op een voorwaarde die we stellen. Omdat deze functie nog steeds niet gemakkelijk te differentiëren is, schrijven we deze eerst om naar een vorm waarbij dit wel het geval is. Bij deze berekening maken we gebruik van de volgende stelling die we eerst gaan bewijzen.

Stelling 3.6.2: Stel λ geeft een verborgen Markov model weer met $O_1...O_T$ de observatiereeks en $q_1...q_T$ de reeks van toestanden. Nu geldt dat

$$\mathbb{P}(O_1...O_T|q_1 = S_{i_1}, ..., q_T = S_{i_T}, \lambda) = \prod_{t=1}^{T} \mathbb{P}(O_t|q_t = S_{i_t}, \lambda)$$

voor alle $T \in \mathbb{N}$.

Bewijs: We geven een bewijs met behulp van inductie. Stel eerst $T = 1$ dan is het triviaal dat geldt

$$\mathbb{P}(O_1|q_1 = S_{i_1}) = \prod_{t=1}^{1} \mathbb{P}(O_t|q_t = S_{i_t}).$$

Stel nu als inductiehypothese dat voor $T = T'$ geldt

$$\mathbb{P}(O_1...O_{T'}|q_1 = S_{i_1}, ..., q_{T'} = S_{i_{T'}}, \lambda) = \prod_{t=1}^{T'} \mathbb{P}(O_t|q_t = S_{i_t}, \lambda)$$

Voor $T = T' + 1$ geldt nu dat

$$\begin{aligned}
&\mathbb{P}(O_1...O_{T'+1}|q_1 = S_{i_1}, ..., q_{T'+1} = S_{i_{T'+1}}, \lambda) \\
=&\mathbb{P}(O_1...O_{T'}|q_1 = S_{i_1}, ..., q_{T'} = S_{i_{T'}}, q_{T'+1} = S_{i_{T'+1}}, \lambda)\mathbb{P}(O_{T'+1}|q_1 = S_{i_1}, ..., q_{T'+1} = S_{i_{T'+1}}, \lambda)) \\
=&\mathbb{P}(O_1...O_{T'}|q_1 = S_{i_1}, ..., q_{T'} = S_{i_{T'}}, \lambda)\mathbb{P}(O_{T'+1}|q_{T'+1} = S_{i_{T'+1}}, \lambda)
\end{aligned}$$

De laatste stap volgt hierbij uit de onafhankelijkheidsregels van een verborgen Markov model. Als we nu de inductiehypothese toepassen, zien we dat bovenstaande gelijk is aan:

$$\begin{aligned}
&\mathbb{P}(O_{T'+1}|q_{T'+1} = S_{i_{T'+1}}, \lambda)) \prod_{t=1}^{T'} \mathbb{P}(O_t|q_t = S_{i_t}) \\
=& \prod_{t=1}^{T'+1} \mathbb{P}(O_t|q_t = S_{i_t}).
\end{aligned}$$

Uit de methode van volledige inductie volgt nu dan de stelling 3.6.2 geldt voor alle $T \in \mathbb{N}$.

Met behulp van deze stelling gaan we $\sum_Q \mathbb{P}(Q|O,\lambda)\log(\mathbb{P}(Q,O|\bar{\lambda}))$ omschrijven.

$$
\begin{aligned}
&\sum_Q \mathbb{P}(Q|O,\lambda)\log(\mathbb{P}(Q,O|\bar{\lambda})) \\
=&\sum_Q \mathbb{P}(Q|O,\lambda)\log(\mathbb{P}(q_1 = S_{i_1}...q_T = S_{i_T}, O_1...O_T|\bar{\lambda})) \\
=&\sum_Q \mathbb{P}(Q|O,\lambda)\log(\mathbb{P}(O_1...O_T|q_1 = S_{i_1}...q_T = S_{i_T},\bar{\lambda})\mathbb{P}(q_1 = S_{i_1}...q_T = S_{i_T}|\bar{\lambda}) \\
=&\sum_Q \mathbb{P}(Q|O,\lambda)\log(\prod_{t=1}^{T}\mathbb{P}(O_t|q_t = S_{i_t},\bar{\lambda})\mathbb{P}(q_1 = S_{i_1}...q_T = S_{i_T}|\bar{\lambda}) \\
=&\sum_Q \mathbb{P}(Q|O,\lambda)\log(\prod_{t=1}^{T}\mathbb{P}(O_t|q_t = S_{i_t},\bar{\lambda})\mathbb{P}(q_T|q_1 = S_{i_1}...q_T = S_{i_{T-1}},\bar{\lambda})\mathbb{P}(q_1 = S_{i_1}...q_T = S_{i_{T-1}}|\bar{\lambda}) \\
=&\sum_Q \mathbb{P}(Q|O,\lambda)\log(\prod_{t=1}^{T}\mathbb{P}(O_t|q_t = S_{i_t},\bar{\lambda})\mathbb{P}(q_T = S_{i_T}|q_{T-1} = S_{i_{T-1}},\bar{\lambda})\mathbb{P}(q_1 = S_{i_1}...q_T = S_{i_{T-1}}|\bar{\lambda})
\end{aligned}
$$

Als we de laatste twee stappen herhaaldelijk toepassen dan zien we dat het bovenstaande gelijk is aan

$$
\begin{aligned}
&\sum_Q \mathbb{P}(Q|O,\lambda)\log(\prod_{t=1}^{T}\mathbb{P}(O_t|q_t = S_{i_t},\bar{\lambda})\prod_{t=1}^{T}\mathbb{P}(q_{t+1} = S_(i_{t+1})|q_t = S_{i_t},\bar{\lambda})\mathbb{P}(q_1 = S_{i_1}|\bar{\lambda})) \\
=&\sum_Q \mathbb{P}(Q|O,\lambda)\sum_{t=1}^{T}[\log \bar{b}_{Si_t}(O_t) + \log \bar{a}_{S_{i_{t-1}}S_{i_t}}] + \log(\bar{\pi}_{S_{t_1}}) \\
=&\sum_Q \mathbb{P}(Q|O,\lambda)\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{M}\sum_{t=1}^{T}[\mathbf{1}\{S_{i_t} = S_i \wedge O_t = v_k\}\log \bar{b}_i(k) + \mathbf{1}\{S_{i_t} = S_i \wedge S_{i_{t+1}} = S_j\}\log \bar{a}_{ij} \\
&+ \mathbf{1}\{S_{i_1} = S_i\}\log(\bar{\pi}_i)].
\end{aligned}
$$

Om bovenstaande functie met behulp van de Lagrangiaan te kunnen maximaliseren, hebben we een willekeurige voorwaarde nodig waaraan de functie moet voldoen. Dit is gelijk de reden waarom we de functie slechts lokaal kunnen maximaliseren. Aangezien we te maken hebben met een kansverdeling, weten we dat $\sum_{j=1}^{N}\bar{a}_{ij} = 1$, $\sum_{k=1}^{M}\bar{b}_i(k) = 1$ en $\sum_{i=1}^{N}\bar{\pi}_i = 1$. We kunnnen deze eigenschappen gebruiken bij het opstellen van een voorwaarde. Wanneer we een voorwaarde kiezen waarvoor slechts dit hoeft te gelden, weten we dat dit voor alle parameters van een verborgen Markov model geldt en dat dus altijd aan de voorwaarde voldaan wordt. Om deze reden hebben we gekozen voor onderstaande voorwaarde bij de Lagrangiaan:

$$\sum_{i=1}^{N}\varepsilon_i(1 - \sum_{k=1}^{M}\bar{b}_i(k)) + \sum_{i=1}^{N}\delta_i(1 - \sum_{j=1}^{N}\bar{a}_{ij}) + \zeta(1 - \sum_{i=1}^{N}\bar{\pi}_i) = 0. \tag{3.7}$$

Met behulp van deze voorwaarde kunnen we nu een bijbehorende Lagrangiaan opstellen. Deze

Lagrangiaan is als volgt:

$$\mathfrak{L}(\bar{\lambda}, \delta, \varepsilon, \zeta) =$$

$$\sum_Q \mathbb{P}(Q|O,\lambda) \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{M}\sum_{t=1}^{T}[\mathbf{1}\{Si_t = S_i \wedge O_t = v_k\} \log \bar{b}_i(k) + \mathbf{1}\{Si_t = S_i \wedge S_{i_{t+1}} = S_j\} \log \bar{a}_{ij} +$$

$$\mathbf{1}\{Si_1 = S_i\} \log(\bar{\pi}_i)] + \sum_{i=1}^{N} \varepsilon_i (1 - \sum_{k=1}^{M} \bar{b}_i(k)) + \sum_{i=1}^{N} \delta_i (1 - \sum_{j=1}^{N} \bar{a}_{ij}) + \zeta(1 - \sum_{i=1}^{N} \bar{\pi}_i).$$

Om het lokale maximum te bepalen gaan we de Lagrangiaan maximaliseren over $\bar{\lambda}$. Dit gaan we doen door de partiële afgeleiden van de Lagrangiaan over $\bar{a}_{ij}, \bar{b}_i(k), \bar{\pi}_i, \delta_i, \varepsilon_i$ en ζ te bepalen en gelijk te stellen aan 0. Voor de partiële afgeleide over $\bar{a}_{ij}$ geldt:

$$\frac{\partial \mathfrak{L}}{\partial \bar{a}_{ij}} = \sum_Q \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} \mathbf{1}\{S_{i_t} = S_i \wedge S_{i_{t+1}} = S_j\} \frac{1}{\bar{a}_{ij}} - \delta_i = 0.$$

Hieruit volgt dus dat

$$\bar{a}_{ij} = \frac{\sum_Q \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} \mathbf{1}\{S_{i_t} = S_i \wedge S_{i_{t+1}} = S_j\}}{\delta_i}$$

Nu moeten we de waarde van δ_i nog berekenen. Voor deze partiële afgeleide geldt:

$$\frac{\partial \mathfrak{L}}{\partial \delta_i} = 1 - \sum_{j=1}^{N} \bar{a}_{ij} = 1 - \frac{1}{\delta_i} \sum_{j=1}^{N} \sum_Q \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} \mathbf{1}\{S_{i_t} = S_i \wedge S_{i_{t+1}} = S_j\} = 0.$$

Hieruit volgt nu dat

$$\delta_i = \sum_{j=1}^{N} \sum_Q \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} \mathbf{1}\{S_{i_t} = S_i \wedge S_{i_{t+1}} = S_j\}.$$

Nu kunnen we ook de waarde van $\bar{a}_{ij}$ berekenen. Deze is gelijk aan

$$\bar{a}_{ij} = \frac{\sum_Q \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} \mathbf{1}\{S_{i_t} = S_i \wedge S_{i_{t+1}} = S_j\}}{\sum_{j=1}^{N} \sum_Q \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} \mathbf{1}\{S_{i_t} = S_i \wedge S_{i_{t+1}} = S_j\}}.$$

Nu gaan we de waarde van $\bar{b}_i(k)$ berekenen door de partiële afgeleides naar $\bar{b}_i(k)$ en ε_i te bepalen en 0 te stellen. Voor de eerste geldt dat

$$\frac{\partial \mathfrak{L}}{\partial \bar{b}_i(k)} = \sum_Q \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} \mathbf{1}\{S_{i_t} = S_i \wedge O_t = v_k\} \frac{1}{\bar{b}_i(k)} - \varepsilon_i = 0.$$

Dus hieruit volgt dat

$$\bar{b}_i(k) = \frac{\sum_Q \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} \mathbf{1}\{S_{i_t} = S_i \wedge O_t = v_k\}}{\varepsilon_i}.$$

En voor de partiële afgeleide over ε_i geldt dat

$$\frac{\partial \mathfrak{L}}{\partial \varepsilon_i} = 1 - \sum_{k=1}^{M} \bar{b}_i(k) = 1 - \frac{1}{\varepsilon_i} \sum_{k=1}^{M} \sum_Q \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} 1\{S_{i_t} = S_i \wedge O_t = v_k\} = 0.$$

Dus er geldt dat

$$\varepsilon_i = \sum_{k=1}^{M} \sum_{Q} \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} \mathbf{1}\{S_{i_t} = S_i \wedge O_t = v_k\}.$$

Als we dit in de formule van $\bar{b}_i(k)$ invullen zien we dat

$$\bar{b}_i(k) = \frac{\sum_Q \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} \mathbf{1}\{S_{i_t} = S_i \wedge O_t = v_k\}}{\sum_{k=1}^{M} \sum_Q \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} \mathbf{1}\{S_{i_t} = S_i \wedge O_t = v_k\}}.$$

Tot slot geldt voor $\bar{\pi}_i$ dat de partiële afgeleide gelijk is aan

$$\frac{\partial \mathfrak{L}}{\partial \bar{\pi}_i} = \sum_{Q} \mathbb{P}(Q|O,\lambda) \mathbf{1}\{S_{i_1} = S_i\} \frac{1}{\bar{\pi}_i} + \zeta = 0$$

Hieruit volgt nu dat

$$\bar{\pi}_i = \frac{\sum_Q \mathbb{P}(Q|O,\lambda) \mathbf{1}\{S_{i_1} = S_i\}}{\zeta}.$$

Waarbij voor ζ geldt

$$\frac{\partial \mathfrak{L}}{\partial \zeta} = 1 - \sum_{i=1}^{N} \bar{\pi}_i = 1 - \frac{1}{\zeta} \sum_{i=1}^{N} \sum_{Q} \mathbb{P}(Q|O,\lambda) \mathbf{1}\{S_{t_1} = S_i\} = 0$$

Dus er geldt

$$\zeta = \sum_{i=1}^{N} \sum_{Q} \mathbb{P}(Q|O,\lambda) \mathbf{1}\{S_{i_1} = S_i\}.$$

Hieruit volgt dat

$$\bar{\pi}_i = \frac{\sum_Q \mathbb{P}(Q|O,\lambda) \mathbf{1}\{S_{i_1} = S_i\}}{\sum_{i=1}^{N} \sum_Q \mathbb{P}(Q|O,\lambda) \mathbf{1}\{S_{i_1} = S_i\}}$$

Om nu $\bar{a}_{ij}, \bar{b}_i(k)$ en $\bar{\pi}_i$ makkelijker te kunnen oplossen, willen we ze omschrijven naar een vorm met $\alpha_t(i)$ en $\beta_t(i)$, zodanig dat ze iteratief met de voorwaartse en achterwaartse benadering op te lossen zijn. We beginnen hiervoor met $\bar{a}_{ij}$ door $\sum_Q \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} 1\{q_t = S_i \wedge q_{t+1} = S_j\}$ om te schrijven. Hiervoor geldt:

$$\sum_{Q} \mathbb{P}(Q|O,\lambda) \sum_{t=1}^{T} \mathbf{1}\{S_{i_t} = S_i \wedge S_{i_{t+1}} = S_j\} \tag{3.8}$$

$$= \frac{1}{\mathbb{P}(O|\lambda)} \sum_{Q} \sum_{t=1}^{T} \mathbb{P}(Q,O|\lambda) \mathbf{1}\{S_{i_t} = S_i \wedge S_{i_{t+1}} = S_j\} \tag{3.9}$$

$$= \frac{1}{\mathbb{P}(O|\lambda)} \sum_{i_1,\ldots i_T} \sum_{t=1}^{T} \mathbb{P}(q_1 = S_{i_1} \ldots q_T = S_{i_T}, O_1 \ldots O_T|\lambda) \mathbf{1}\{Si_t = S_i \wedge S_{i_{t+1}} = S_j\} \tag{3.10}$$

$$= \frac{1}{\mathbb{P}(O|\lambda)} \sum_{i_1,\ldots i_T} \sum_{t=1}^{T} \mathbb{P}(O_1 \ldots O_T|q_1 = S_{i_1} \ldots q_T = S_{i_T}, \lambda) \mathbb{P}(q_1 = S_{i_1} \ldots q_T = S_{i_T}|\lambda) \mathbf{1}\{S_{i_t} = S_i \wedge S_{i_{t+1}} = S_j\} \tag{3.11}$$

$$=\frac{1}{\mathbb{P}(O|\lambda)}\sum_{t=1}^{T}\mathbb{P}(O_1...O_T|\lambda)\sum_{S_{i_1},...S_{i_T}}\mathbb{P}(q_1=S_{i_1}...q_T=S_{i_T}|\lambda)\mathbf{1}\{S_{i_t}=S_i\wedge S_{i_{t+1}}=S_j\} \tag{3.12}$$

$$=\frac{1}{\mathbb{P}(O|\lambda)}\sum_{t=1}^{T}\mathbb{P}(O_1...O_t|\lambda)\mathbb{P}(O_{t+1}|\lambda)\mathbb{P}(O_{t+2}...O_T|\lambda)\mathbb{P}(S_{i_t}=S_i,S_{i_{t+1}}=S_j|\lambda). \tag{3.13}$$

Voor de verschillende kansen in de laatste formule geldt nu dat:

$$\mathbb{P}(O_1...O_t|\lambda)=\frac{\mathbb{P}(O_1...O_t,S_{i_t}=S_i|\lambda)}{\mathbb{P}(S_{i_t}=S_i|O_1...O_t,\lambda)}=\frac{\alpha_t(i)}{\mathbb{P}(S_{i_t}=S_i)|\lambda)}, \tag{3.14}$$

$$\mathbb{P}(O_{t+1}|\lambda)=\frac{\mathbb{P}(O_{t+1},S_{i_{t+1}}=S_j,\lambda)}{\mathbb{P}(S_{i_{t+1}}=S_j|O_{t+1},\lambda)}=\frac{\mathbb{P}(O_{t+1},S_{i_{t+1}}=S_j|\lambda)}{\mathbb{P}(S_{i_{t+1}}=S_j|\lambda)}=b_j(O_{t+1}) \tag{3.15}$$

en

$$\mathbb{P}(O_{t+2}...O_T|\lambda)=\frac{\mathbb{P}(O_{t+2}...O_T,S_{i_{t+1}}=S_j|\lambda)}{\mathbb{P}(S_{i_{t+1}}=S_j|O_{t+2}...O_T,\lambda)}=\frac{\mathbb{P}(O_{t+2}...O_T,Si_{t+1}=S_j|\lambda)}{\mathbb{P}(S_{i_{t+1}}=S_j|\lambda)}=\beta_{t+1}(j). \tag{3.16}$$

Wanneer we dit invullen in de vergelijking zien we dat

$$\frac{1}{\mathbb{P}(O|\lambda)}\sum_{t=1}^{T}\mathbb{P}(O_1...O_t|\lambda)\mathbb{P}(O_{t+1}|\lambda)\mathbb{P}(O_{t+2}...O_T|\lambda)\mathbb{P}(S_{i_t}=S_i,S_{i_{t+1}}=S_j|\lambda) \tag{3.17}$$

$$=\frac{1}{\mathbb{P}(O|\lambda)}\sum_{t=1}^{T}\alpha_t(i)b_j(O_{t+1})\beta_{t+1}(j)\frac{\mathbb{P}(S_{i_t}=S_i,S_{i_{t+1}}=S_j|\lambda)}{\mathbb{P}(S_{i_t}=S_i)|\lambda)} \tag{3.18}$$

$$=\frac{1}{\mathbb{P}(O|\lambda)}\sum_{t=1}^{T}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j). \tag{3.19}$$

Voor het omschrijven van $\sum_{j=1}^{N}\sum_{Q}\mathbb{P}(Q|O,\lambda)\sum_{t=1}^{T}1\{q_t=S_i\wedge q_{t+1}=S_j\}$ kunnen we nu bijna hetzelfde principe volgen dus we schrijven dit niet helemaal uit. Er geldt:

$$\sum_{j=1}^{N}\sum_{Q}\mathbb{P}(Q|O,\lambda)\sum_{t=1}^{T}1\{S_{i_t}=S_i\wedge S_{i_{t+1}}=S_j\} \tag{3.20}$$

$$=\frac{1}{\mathbb{P}(O|\lambda)}\sum_{j=1}^{N}\sum_{t=1}^{T}\mathbb{P}(O_1...O_t|\lambda)\mathbb{P}(O_{t+1}...O_T|\lambda)\mathbb{P}(S_{i_t}=S_i|\lambda) \tag{3.21}$$

Dit volgt uit de berekening 3.8 tot en met (3.13), waarbij in stap (3.13) wordt gebruikt dat $\sum_{j=1}^{N}\mathbb{P}(S_{i_t}=S_i,S_{i_{t+1}}=S_j|\lambda)=\mathbb{P}(S_{i_t}=S_i|\lambda)$ om het als 3.21 te schrijven. Met behulp van (een variant van)(3.14) en (3.6) kunnen we nu (3.20) verder omschrijven tot:

$$\frac{1}{\mathbb{P}(O|\lambda)}\sum_{t=1}^{T}\alpha_t(i)\beta_t(i) \tag{3.22}$$

Als we nu (3.19) en (3.22) invullen in de vergelijking die we voor $\bar{a}_{ij}$ verkregen hebben, krijgen we

$$\bar{a}_{ij}=\frac{\frac{1}{\mathbb{P}(O|\lambda)}\sum_{t=1}^{T}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\frac{1}{\mathbb{P}(O|\lambda)}\sum_{t=1}^{T}\alpha_t(i)\beta_t(i)}=\frac{\sum_{t=1}^{T}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{t=1}^{T}\alpha_t(i)\beta_t(i)}. \tag{3.23}$$

Nu bekijken we de formule voor $\bar{b}_i(k)$. Hiervoor schrijven we eerst $\sum_Q \mathbb{P}(Q|O,\lambda)\sum_{t=1}^T \mathbf{1}\{S_{i_t} = S_i \wedge O_t = v_k\}$ om naar een vergelijking met $\alpha_t(i)$ en $\beta_t(j)$. Hierbij maken we gebruik van een paar van de berekingen die we hierboven hebben gebruikt, omdat de formules ongeveer hetzelfde zijn. Voor de formule geldt:

$$\begin{aligned}
&\sum_Q \mathbb{P}(Q|O,\lambda)\sum_{t=1}^T \mathbf{1}\{S_{i_t} = S_i \wedge O_t = v_k\} \\
=&\frac{1}{\mathbb{P}(O|\lambda)}\sum_{t=1}^T \mathbb{P}(O_1...O_T|\lambda) \sum_{S_{i_1}...,S_{i_T}T} \mathbf{1}\{S_{i_t} = S_i \wedge O_t = v_k\} \\
=&\frac{1}{\mathbb{P}(O|\lambda)}\sum_{t=1}^T \mathbb{P}(O_1...O_t|\lambda)\mathbb{P}(O_{t+1}...O_T|\lambda)\mathbb{P}(S_{i_t} = S_i|\lambda)\mathbf{1}\{O_t = v_k\} \\
=&\frac{1}{\mathbb{P}(O|\lambda)}\sum_{t=1}^T \frac{\alpha_t(i)}{\mathbb{P}(S_{i_t} = S_i|\lambda)}\beta_t(i)\mathbb{P}(S_{i_t} = S_i|\lambda)\mathbf{1}\{O_t = v_k\} \\
=&\frac{1}{\mathbb{P}(O|\lambda)}\sum_{t=1}^T \alpha_t(i)\beta_t(i)\mathbf{1}\{O_t = v_k\}.
\end{aligned}$$

Nu geldt voor $\bar{b}_i(k)$ dat

$$\bar{b}_i(k) = \frac{\frac{1}{\mathbb{P}(O|\lambda)}\sum_{t=1}^T \alpha_t(i)\beta_t(i)\mathbf{1}\{O_t = v_k\}}{\frac{1}{\mathbb{P}(O|\lambda)}\sum_{k=1}^M\sum_{t=1}^T \alpha_t(i)\beta_t(i)\mathbf{1}\{O_t = v_k\}} = \frac{\sum_{t=1}^T \alpha_t(i)\beta_t(i)\mathbf{1}\{O_t = v_k\}}{\sum_{t=1}^T \alpha_t(i)\beta_t(i)}. \tag{3.24}$$

Tot slot willen we $\bar{\pi}_i$ omschrijven naar een formule met $\alpha_t(i)$ en $\beta_t(i)$. Hiervoor schrijven we eerst $\sum_Q \mathbb{P}(Q|O,\lambda)\mathbf{1}\{q_1 = S_i\}$ om naar iets in die vorm. Voor deze formule geldt:

$$\begin{aligned}
&\sum_Q \mathbb{P}(Q|O,\lambda)\mathbf{1}\{S_{i_1} = S_i\} \\
=&\frac{1}{\mathbb{P}(O|\lambda)}\mathbb{P}(O_1|\lambda)\mathbb{P}(O_2...O_T|\lambda)\sum_Q \mathbb{P}(Q|\lambda)\mathbf{1}\{S_{i_1} = S_i\} \\
=&\frac{1}{\mathbb{P}(O|\lambda)}\frac{\alpha_1(i)}{\mathbb{P}(S_{i_1} = S_i)}\beta_1(i)\mathbb{P}(S_{i_1} = S_i) \\
=&\frac{1}{\mathbb{P}(O|\lambda)}\alpha_1(i)\beta_1(i)
\end{aligned}$$

Dus nu geldt:

$$\bar{\pi_i} = \frac{\frac{1}{\mathbb{P}(O|\lambda)}\alpha_1(i)\beta_1(i)}{\frac{1}{\mathbb{P}(O|\lambda)}\sum_{i=1}^N \alpha_1(i)\beta_1(i)} = \frac{\alpha_1(i)\beta_1(i)}{\sum_{i=1}^N \alpha_1(i)\beta_1(i)}. \tag{3.25}$$

Om nu daadwerkelijk het lokale maximum $\mathbb{P}(O|\lambda)$ te bepalen, moeten we het volgende doen. We beginnen met een willekeurig model $\lambda = (\{a_{ij}\}, \{b_i(k)\}, \{\pi_i\})$. Een keuze voor λ is bijvoorbeeld een model waarbij de transitie van de ene naar de andere toestand en ook de kansverdeling per toestand overal gelijke kansen hebben. Met behulp van deze λ en de formules (3.23), (3.24) en (3.25) kunnen we nu een $\bar{\lambda} = (\{\bar{a}_{ij}\}, \{\bar{b}_i(k)\}, \{\bar{\pi}_i\})$ berekenen zodat $\mathbb{P}(O|\bar{\lambda})$ het lokale maximum van $\mathbb{P}(O|\lambda)$ is. Dit is dus het model dat de observatiereeks het beste beschrijft.

3.6.1 Voorbeeld met MATLAB

Ook van de optimalisatie van de parameters van het verborgen Markov model hebben we een MATLAB-code geschreven. We maken hiervoor weer gebruik van de observatiereeks blauw-rood-rood-zwart-wit-rood-geel of in MATLAB $[2, 1, 1, 5, 4, 1, 3]$. Hiervoor definiëren we eerst een model waarmee we het algoritme beginnen.

Listing 3.7: Model voor het begin van het algoritme

```
A=[1/3,1/3,1/3;1/3,1/3,1/3;1/3,1/3,1/3];
B=[1/6, 1/4, 1/6, 1/12, 1/3; 5/11, 1/11, 1/11, 1/11, 3/11;
    1/11, 3/11, 2/11, 4/11, 1/11];
pi=[1/3,1/3,1/3];
```

Het maakt niet uit wat voor waarden we hier invullen. De uitkomst van het algoritme zal voor ieder model hetzelfde zijn. Voor de Markovketen en de begintoestand maken we gebruik van een gelijk verdeeld model. Voor de B gebruiken we dezelfde kansverdelingen als in de eerdere codes. Nu gaan we de voorwaarste en achterwaartse benadering programmeren. De voorwaartse benadering ziet er net zo uit als bij sectie 3.4.1 en de achterwaartse zoals hieronder besschreven.

Listing 3.8: Achterwaartse benadering

```
    b=zeros(T,N);
    i=1;
    while i<N+1
        b(T,i)=1;
        i=i+1;
    end
    t=T-1;
    while t>0
        i=1;
        while i<N+1
            x=0;
            j=1;
            while j<N+1
                Beta=A(i,j)*B(j,O(t+1))*b(t+1,j);
                x=x+Beta;
                j=j+1;
            end
            b(t,i)=x;
            i=i+1;
        end
        t=t-1;
    end
```

Hier wordt $\beta_t(i)$ voor iedere t,i en j berekend met behulp van formule (3.4) en in een matrix b gezet zodat we alle waarden later opnieuw kunnen gebruiken. De waarden van de voorwaartse benadering zijn in een matrix a gezet die ook dimensie (T, N) heeft.

Nu gaan we nieuwe matrices opstellen genaamd “Anieuw”, “Bnieuw” en “pinieuw” die de geoptimaliseerde matrices van het model gaan vormen. We beginnen hiervoor met drie nulmatrices

waarbij we de waardes later gaan invullen. Voor Anieuw worden de nieuwe waardes op onderstaande manier berekend.

Listing 3.9: Bepalen van $\bar{a}_{ij}$ voor alle i en j

```
i=1;
while i<N+1
    j=1;
    while j<N+1
        t=1;
        xna=0;
        xta=0;
        while t<T
            ana=a(t,i)*A(i,j)*B(j,O(t+1))*b(t+1,j);
            xna=xna+ana;
            ata=a(t,i)*b(t,i);
            xta=xta+ata;
            t=t+1 ;
        end
        Anieuw(i,j)=xna/xta;
        j=j+1;
    end
    i=i+1;
end
```

Dit is formule (3.23) gecodeerd voor alle i, j en t met behulp van while-loops. Als we nu $\bar{b}_i(k)$ gaan coderen met behulp van formule (3.24) dan ziet dat er als volgt uit:

Listing 3.10: Bepalen van $\bar{b}_i(k)$ voor alle i en k

```
i=1;
while i<N+1
    k=1;
    while k<M+1
        t=1;
        xnb=0;
        xtb=0;
        while t<T+1
            if O(t)==k
                anb=a(t,i)*b(t,i);
                xnb=xnb+anb;
            else
                xnb=xnb;
            end
            atb=a(t,i)*b(t,i);
            xtb=xtb+atb;
            t=t+1 ;
        end
        Bnieuw(i,k)=xnb/xtb;
        k=k+1;
```

```
    end
    i=i+1;
end
```

Tot slot moeten we dan nog $\bar{\pi}_i$ coderen. Dit doen we met behulp van de formule (3.25) en de code hiervoor staat hieronder:

Listing 3.11: Bepalen van $\bar{\pi}_i$ voor alle i

```
anp=zeros(1,N);
i=1;
xtp=0;
while i<N+1
    anp(i)=a(1,i)*b(1,i);
    xtp=xtp+anp(i);
    i=i+1;
end
i=1;
while i<N+1
    pinieuw(i)=anp(i)/xtp;
    i=i+1;
end
```

Hieruit komt het nieuwe model:

$$\bar{A} = \begin{pmatrix} 0.2646 & 0.4184 & 0.3170 \\ 0.3170 & 0.4032 & 0.2798 \\ 0.2582 & 0.5478 & 0.1940 \end{pmatrix}, \bar{B} = \begin{pmatrix} 0.3309 & 0.1920 & 0.1787 & 0.0730 & 0.2254 \\ 0.6766 & 0.0523 & 0.0731 & 0.0597 & 0.1383 \\ 0.1870 & 0.2170 & 0.2021 & 0.3302 & 0.0637 \end{pmatrix}$$

en $\bar{\pi} = (0.4074, 0.1481, 0.4444)$. Dit is dus het model dat het best bij de gegeven observatiereeks past volgens de bovenstaande oplossing voor het derde probleem.

3.7 Meerdere observatiereeksen

Voor sommige toepassingen van verborgen Markov modellen, zeker bij de toepassing in spraakherkenning, kan het handig zijn om meerdere observatiereeksen te gebruiken in plaats van één. Stel we hebben een verzameling van L observatiereeksen $\mathbf{O} = (O^{(1)}, O^{(2)}, ..., O^{(L)})$, waarbij $O^{(l)} = (O_1^{(l)}, O_2^{(l)}, ..., O_{T_l}^{(l)})$ de l-de observatiereeks is van lengte T_l. Met behulp van deze observatiereeksen willen we nu de parameters van het verborgen Markov model bepalen. We moeten dus λ bepalen zodat $\mathbb{P}(\mathbf{O}|\lambda)$ gemaximaliseerd wordt. Hiervoor gaan we er van uit de de verschillende observatiereeksen onafhankelijk zijn van elkaar. Dan geldt er namelijk dat:

$$\mathbb{P}(\mathbf{O}|\lambda) = \prod_{l=1}^{L} \mathbb{P}(O^{(l)}|\lambda).$$

Voor een model λ en een alternatief model $\bar{\lambda}$ geldt nu dat

$$\begin{aligned}&\log(\prod_{l=1}^{L}\mathbb{P}(O^{(l)}|\bar{\lambda})) - \log(\prod_{l=1}^{L}\mathbb{P}(O^{(l)}|\lambda))\\ =&\sum_{l=1}^{L}\log(\mathbb{P}(O^{(l)}|\bar{\lambda})) - \sum_{l=1}^{L}\log(\mathbb{P}(O^{(l)}|\lambda))\\ =&\sum_{l=1}^{L}[\log(\mathbb{P}(O^{(l)}|\bar{\lambda})) - \log(\mathbb{P}(O^{(l)}|\lambda))]\end{aligned}$$

Uit stelling 3.6 volgt nu dat

$$\begin{aligned}&\log(\prod_{l=1}^{L}\mathbb{P}(O^{(l)}|\bar{\lambda})) - \log(\prod_{l=1}^{L}\mathbb{P}(O^{(l)}|\lambda))\\ \geq&\sum_{l=1}^{L}\sum_{Q^{(l)}}\mathbb{P}(Q^{(l)}|O^{(l)},\lambda)\log(\mathbb{P}(O^{(l)}),Q^{(l)}|\bar{\lambda}) - \sum_{l=1}^{L}\sum_{Q^{(l)}}\mathbb{P}(Q^{(l)}|O^{(l)},\lambda)\log(\mathbb{P}(O^{(l)}),Q^{(l)}|\lambda)\end{aligned}$$

Hierin is $Q^{(l)} = (S_1^{(l)}, ..., S_{T_l}^{(l)})$ de bijbehorende reeks van toestanden bij observatiereeks l. In het eerste bewijs van sectie 3.6 is nu ook te zien dat we

$$\sum_{l=1}^{L}\sum_{Q^{(l)}}\mathbb{P}(Q^{(l)}|O^{(l)},\lambda)\log(\mathbb{P}(O^{(l)}),Q^{(l)}|\lambda)$$

kunnen maximaliseren over λ om op dezelfde λ uit te komen waarvoor $\mathbb{P}(\mathbf{O}|\lambda)$ gemaximaliseerd wordt over λ. De Lagrangiaan die we hiervoor gebruiken is:

$\mathfrak{L}(\bar{\lambda}, \delta, \varepsilon, \zeta) =$

$$\sum_{l=1}^{L}[\sum_{Q}^{(l)}\mathbb{P}(Q^{(l)}|O^{(l)},\lambda)\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{M}\sum_{t=1}^{T}[\mathbf{1}\{S_{i_t}^{(l)} = S_i \wedge O_t^{(l)} = v_k\}\log\bar{b}_i(k) + \mathbf{1}\{S_{i_t}^{(l)} = S_i \wedge S_{i_{t+1}}^{(l)} = S_j\}\log\bar{a}_{ij} +$$

$$\mathbf{1}\{S_{i_1}^{(l)} = S_i\}\log(\bar{\pi}_i)]] - [\sum_{i=1}^{N}\varepsilon_i(1 - \sum_{k=1}^{M}\bar{b}_i(k)) + \sum_{i=1}^{N}\delta_i(1 - \sum_{j=1}^{N}\bar{a}_{ij}) + \zeta(1 - \sum_{i=1}^{N}\bar{\pi}_i)].$$

Op ongeveer dezelfde manier als in hoofdstuk 3.6 kunnen we nu de waarden voor $\bar{a}_{ij}$, $\bar{b}_i(k)$ en $\bar{\pi}_i$ berekenen. Voor deze waarden geldt:

$$\bar{a}_{ij} = \frac{\sum_{l=1}^{L}\frac{1}{\mathbb{P}(O^{(l)}|\lambda)}\sum_{t=1}^{T_l-1}\alpha_t^l(i)a_{ij}b_j(O_{t+1}^{(l)})\beta_{t+1}^l(j)}{\sum_{l=1}^{L}\frac{1}{\mathbb{P}(O^{(l)}|\lambda)}\sum_{t=1}^{T_l-1}\alpha_t^l(i)\beta_t^l(i)} \tag{3.26}$$

,

$$\bar{b}_i(k) = \frac{\sum_{l=1}^{L}\frac{1}{\mathbb{P}(O^{(l)}|\lambda)}\sum_{t=1}^{T}\alpha_t^{(l)}(i)\beta_t^{(l)}(i)\mathbf{1}\{O_t^{(l)} = v_k\}}{\sum_{l=1}^{L}\frac{1}{\mathbb{P}(O^{(l)}|\lambda)}\sum_{t=1}^{T}\alpha_t^{(l)}(i)\beta_t^{(l)}(i)} \tag{3.27}$$

en

$$\bar{\pi}_i = \frac{\sum_{l=1}^{L}\frac{1}{\mathbb{P}(O^{(l)}|\lambda)}\alpha_1^{(l)}(i)\beta(l)_1(i)}{\sum_{l=1}^{L}\frac{1}{\mathbb{P}(O^{(l)}|\lambda)}\sum_{i=1}^{N}\alpha_1^{(l)}(i)\beta(l)_1(i)} \tag{3.28}$$

Met behulp van (3.26), (3.27) en (3.28) kunnen we dus bij meerdere observatiereeksen optimale model benaderen. De verdere berekening gaat precies hetzelfde als bij één enkele observatiereeks.

Hoofdstuk 4

De toepassing op spraakherkenning

Nu we verschillende eigenschappen van verborgen Markov modellen en de manier hoe ze op te lossen zijn, hebben besproken, gaan we verder met de toepassing op spraakherkenning ervan. Voor spraakherkenning worden woorden vaak opgedeeld in de verschillende spraakklanken. De Nederlandse taal heeft er hiervan 35. Dit zijn de letters van het alfabet, maar bijvoorbeeld ook de oe of de au. Het woord "woord"wordt bijvoorbeeld opgedeeld als w/oo/r/d. Het is uiteraard ook mogelijk om woorden op te delen in grotere delen zoals deelwoorden of het gewoon gehele woorden te laten. Wat dit betreft is er niet iets dat makkelijker is. Er geldt namelijk hoe simpeler de eenheid waarin woorden worden opgedeeld, hoe minder aparte eenheden er zijn, maar in het proces van het herkennen van een observatiereeks is dit ingewikkelder.

Er kunnen ook op verschillende manieren verborgen Markov modellen opgezet worden. De simpelste manier is volgens het geïsoleerde woord herkenningssysteem. Dan wordt er een woordenboek opgezet met W woorden en voor ieder woord wordt er een apart verborgen Markov model gedefinieerd. De parameters worden berekend met behulp van probleem 3. Er wordt hierbij gebruik gemaakt van één of meerdere observatiereeks(en). Zo'n observatiereeks volgt uit een Feature Analyse van ingesproken spraaksignaal van het desbetreffende woord. We komen later terug op hoe de Feature Analyse werkt.

Zo kan het woordenboek bijvoorbeeld bestaan uit de vier woorden"links", "rechts","stop" en"ga". Dit is dan een uiterst simpel spraakherkenningssysteem, want het kan alleen maar deze woorden herkennen. Deze vier woorden worden nu ieder door drie verschillende personen ingesproken. Per woord worden er dus drie observatiereeksen verkregen. De drie observatiereeksen van het woord "links"vormen nu de basis voor het verborgen Markov model behorende bij "links". Voor de andere drie woorden geldt hetzelfde. Met behulp van de berekeningen uit sectie 3.6 worden nu vier verschillende verborgen Markov modellen opgesteld. Per woord één.

Nu kunnen op verschillende manieren de toestanden in zo'n model worden beschreven. De eerste manier is dat het aantal toestanden in het model het aantal observatiereeksen dat gebruikt wordt om het model op te zetten is. Dit is dus het aantal keer dat het woord is uitgesproken door mogelijk verschillende personen, maar dit is niet noodzakelijk. De tweede manier is dat toestanden de manier weergeven waarop het woord bij de spraakherkenning is opgedeeld. Het aantal toestanden is in dit geval bijvoorbeeld het aantal spraakklanken van het woord.

In het voorbeeld van hierboven geldt dus dat bij de eerste manier dat ieder model 3 toestanden heeft. Mocht gekozen voor de tweede manier dan moeten we de woorden eerst opdelen. Links wordt opgedeeld als l/i/ng/k/s en zou dus 5 toestanden hebben, rechts wordt opgedeeld als r/e/g/t/s en heeft ook 5 toestanden, stop wordt s/t/o/p en heeft 4 toestanden en ga wordt g/a en heeft dus 2 toestanden. De wissel van toestand vindt dus plaats bij de overgang van spraakklank.

De verzameling V, bestaande uit de symbolen in de verschillende toestanden, moet alle opties uit de verschillende observatiereeksen dekken. Het bestaat in het geval van spraakherkenning dus uit alle verschillende symbolen die ontstaan uit de Feauture Analyse van de verschillende spraaksignalen. Dit kunnen bijvoorbeeld de verschillende spraakklanken zijn. De symbolen zijn te zien als de verschillende bouwstenen waaruit het woord is opgebouwd.

In ons voorbeeld kan deze verzameling dus als $V = (l, i, ng, k, s, r, e, g, t, o, p, a)$ gedefinieerd zijn. Bij ingewikkeldere spraakherkenningssystemen breidt dit zich uit tot alle verschillende spraakklanken. In dit systeem worden niet alle spraakklanken gebruikt dus ze zijn nu niet allemaal nodig.

Als voor alle woorden zo'n dergelijk model is opgezet, kan het gebruikt worden voor de spraakherkenning van een specifiek woord of zin. Hiervoor wordt het spraaksignaal van het betreffende woord eerst weer omgezet met behulp van dezelfde Feature Analyse in een bruikbare observatiereeks O. Nu wordt voor ieder model λ de kans $\mathbb{P}(O|\lambda)$ berekend. Het woord waarbij deze kans nu het grootst is, wordt gezien als het woord dat gezegd is.

Bij het bovengenoemde spraakherkenningssysteem kunnen dus de woorden "links", "rechts", "stop" en "ga" ingesproken worden en door het syteem herkend worden. Als bijvoorbeeld " links" ingesproken wordt, wordt hiervan weer een discrete observatiereeks O gemaakt en van deze observatiereeks wordt bij ieder model de kans berekend dat O past bij het model. De hoogste kans vindt waarschijnlijk plaats bij het model van het woord "links". Dit zal dan ook de uitkomst zijn van de herkenning. Mocht er een ander woord dan de vier waarvan een model is gemaakt, ingesproken worden en dan zal het systeem alsnog één van de vier woorden als uitkomst geven. Namelijk alsnog het woord waarbij $\mathbb{P}(O|\lambda)$ het grootst is.

Het is ook mogelijk om bijvoorbeeld voor iedere spraakklank apart een verborgen Markov model te definiëren. Hiermee zijn meer woorden te herkennen, omdat men nu niet gebonden is aan die W woorden waarvoor er een verborgen Markov model is opgesteld. Het principe van de spraakherkenning is wel lastiger, omdat er geen sprake meer is van een geïsoleerd woord systeem. Nu moeten de verschillende modellen aaneengeschakeld worden om een woord te kunnen herkennen.

Bij modellen die gebruik maken van kleinere woordonderdelen of waarin ook hele zinnen moeten worden omgezet, heeft het spraakherkenningssysteem vaak ook mechanismes ingebouwd om de woorden en zinnen correct te laten zijn. Dit houdt er rekening mee dat woorden die uit de bovengenoemde stappen volgen wel daadwerkelijk voorkomen in een woordenboek en dus bestaande woorden zijn. Ook wordt bij zinnen rekening gehouden met de grammatica, logische zinsopbouw en spelling. Ook wordt er soms rekening gehouden met de betekenis van woorden in een zin en of dit logisch is. Dit wordt gebruikt om de fouten bij de spraakherkenning kleiner te maken, maar we zullen hier niet verder op in gaan.

4.1 Feature Analyse en Vector Kwantisering

Aangezien het geluidssignaal van een spraakopname niet echt bruikbaar is bij een spraakherkenningssysteem, moet dit eerst omgezet worden naar een observatiereeks. Hiervoor wordt het geluidssignaal eerst met behulp van Feature Analyse omgezet in een reeks van observatievectoren. Daarna moet in ons geval deze reeks van observatievectoren nog omgezet worden naar een discrete observatiereeks, omdat we alleen hebben gekeken naar discrete verborgen Markov modellen.

4.1.1 Feature Analyse

Er komt een gedigitaliseerd geluidssignaal van de spraakopname binnen bij het spraakherkenningssysteem. Bij hoge tonen ontstaat veel ruis dus dan is het daadwerkelijke gezegde niet meer goed te herkennen. Daarom wordt bij hoge toonhoogtes de amplitude van het signaal verhoogd. Dit gebeurt pas van een bepaalde toonhoogte bijvoorbeeld vanaf 2122 Hz en hoe hoger de frequentie, hoe meer de amplitude verhoogd wordt. Dit kan bijvoorbeeld per verdubbeling van frequentie een verhoging van de amplitude met 6 dB zijn. Dit wordt ook wel het glad maken van het signaal genoemd.

Vervolgens wordt het verkregen signaal opgedeeld in verschillende analysevensters. De verschillende vensters overlappen deels met elkaar, zo is bijvoorbeeld de lengte van een analysevenster 10 ms en wordt aan het begin en het eind van het venster 2.5 ms overlapt door een ander venster. Deze overlapping zorgt voor extra zekerheid, omdat delen van het geluidssignaal twee keer voorkomen in een vector van de observatiereeks. Ieder deel van het geluidssignaal wordt dus ook vaker geanalyseerd door het verborgen Markov model, waardoor fouten kunnen worden opgevangen. We noemen de verschillende analysevensters A_t met $0 \leq t \leq \frac{T}{d}$, waarbij de T de lengte van het geluidssignaal is in seconden en d de lengte van het venster in seconden.

Van ieder zo'n venster wordt nu een continue observatievector gemaakt. Hiervoor worden eerst uit het venster een aantal (bijvoorbeeld 20) frequenties genomen, de verschillende toonhoogtes in een venster worden dus bekeken. Deze frequenties worden niet op een lineaire manier gekozen, maar met behulp van de mel-frequentieschaal. De mel-frequentie-schaal is een schaal van de toonhoogte gebaseerd op het menselijk gehoor. Zo klinkt een toon van 2000 mel twee keer zo hoog als een toon van 1000 mel, terwijl een toon van 2000 Hz niet twee keer zo hoog klinkt als een toon van 1000 Hz. Op deze 20 frequenties $(X_0, ...X_{19})$ wordt een Fast Fourier Transform (FFT) toegepast. De FFT is een snelle manier om een Discrete Fourier Transform (DFT) te berekenen. Bij DFT wordt een verzameling van waarden omgezet in een andere net zo grote verzameling van waarden $(Y_0, ..., Y_{19})$. De omzetting gebeurt met behulp van de volgende formule:

$$Y_k = \sum_{n=0}^{19} X_n e^{-\frac{2\pi i}{20} kn} = \sum_{n=0}^{19} X_n (\cos(\frac{2\pi kn}{20}) + i \sin(\frac{2\pi kn}{20})).$$

Er komt dus een reeks complexe getallen uit de DFT. Deze transformatie zet de variabelen die afhankelijk zijn van de tijd om in variabelen die afhankelijk zijn van de verschillende frequenties. Het geeft dus weer in welke mate een bepaalde frequentie voorkomt in het venster. Om dit voor iedere 10 ms met 20 waarden te doen, neemt erg veel tijd in beslag en daarom wordt de FFT uitgevoerd. Hierbij worden de frequenties $(X_0, ..., X_{19})$ opgesplitst in even variabelen dus $X' = X_{2m}$ en oneven variabelen dus $X'' = X_{2m+1}$ met $m \in \mathbb{Z}_{\geq 0}$. En stel dat de DFT van de

variabelen met even index gelijk is aan Y' en de DFT van de variabelen met oneven index gelijk is aan Y''. Volgens het algoritme van Cooley en Tukey geldt nu dat:

$$\begin{aligned} Y_k &= \sum_{n=0}^{9} X_{2n} e^{-\frac{2\pi i}{20} 2nk} + \sum_{n=0}^{9} X_{2n+1} e^{-\frac{2\pi i}{20}(2n+1)k} \\ &= \sum_{n=0}^{9} X'_n e^{-\frac{2\pi i}{10} nk} + e^{\frac{2\pi i}{20} k} \sum_{n=0}^{9} X''_n e^{-\frac{2\pi i}{10} nk} \\ &= \begin{cases} Y'_k + e^{\frac{2\pi i}{20} k} Y''_k & \text{als } k < 10 \\ Y'_{k-n} - e^{\frac{2\pi i}{20}(k-n)} Y''_{k-n} & \text{als } k \geq 10 \end{cases} \end{aligned}$$

Hierbij hoeft slecht 10 keer het deel met de e-macht berekend te worden dus dit scheelt bij veel vensters erg in het aantal berekeningen.

Nu de frequenties X_n met behulp van FFT zijn omgezet in waarden Y_n wordt op deze waarden een discrete cosinus transformatie (DCT) toegepast. Dit gebeurt, omdat de uitkomsten van de FFT van verschillende signalen nog erg op elkaar kunnen lijken en dit moet voorkomen worden. DCT is ook een soort DFT, maar dan alleen met de cosinus functie dus we krijgen geen complexe getallen als uitkomst. Er zijn verschillende DCT-functies.De verschillende soorten functies zijn genummerd. DCT II wordt bijvoorbeeld gegeven door:

$$C_k = \sum_{n=0}^{19} Y_n \cos(k(n + \frac{1}{2})\frac{\pi}{20}).$$

Hierin zijn Y_n de door FFT verkregen cofficienten en C_k de door DCT verkregen nieuwe coëficienten. Voor k geldt dus dat $0 \leq k \leq 19$. Deze 20 coëfficienten heten ook wel mel-frequentie cepstrale coëfficienten en vormen samen een deel van de feature analyse vector behorende bij analysevenster A_t. Deze vector noemen we $\mathbf{O_t}$. Het andere deel van de feature analyse vector bestaat uit eerste orde en tweede orde regressie coëfficienten die worden toegevoegd om de aanname van een verborgen Markov model dat verschillende observaties onafhankelijk zijn van elkaar te compenseren. De eerste orde coëfficient, ook wel $\mathbf{\Delta O_t}$ wordt verkregen door:

$$\mathbf{\Delta O_t} = \frac{\sum_{i=1}^{20} w_i(\mathbf{O_{t+i}} - \mathbf{O_{t-i}})}{2\sum_{i=1}^{20} w_i^2}.$$

Hierin zijn w_i regressie coëfficienten van de verschillende coëfficienten van $\mathbf{O_t}$.

De tweede orde coëfficient wordt nu gegeven door

$$\mathbf{\Delta^2 O_t} = \frac{\sum_{i=1}^{20} w_i(\mathbf{\Delta O_{t+i}} - \mathbf{\Delta O_{t-i}})}{2\sum_{i=1}^{20} w_i^2}.$$

Hierin zijn w_i regressie coëfficienten van de verschillende coëfficienten van $\mathbf{\Delta O_t}$. Hieruit volgt dan per venster A_t de observatievector $(\mathbf{O_t}, \mathbf{\Delta O_t}, \mathbf{\Delta^2 O_t})$, waarmee we verder gaan werken.

4.1.2 Vector Kwantisering

De observatievectoren die we met de Feature Analyse verkregen hebben uit een continu spraaksignaal zijn niet te gebruiken bij het discrete verborgen Markov model dat we besproken hebben.

Om dit model nu bruikbaar te maken, moeten deze vectoren omgezet worden naar een discrete observatiereeks. Dit gebeurt met behulp van Vector Kwantisering. Hierbij wordt een woordenboek gebruikt van grootte W met als inhoud de verzameling $(w_1, ..., w_W)$ van symbolen die in de uiteindelijke observatiereeks kunnen voorkomen. Het woordenboek koppelt de continue vectoren aan discrete symbolen. Eerst wordt er per woord waarvoor er een model bestaat een woordenboek gemaakt. Bij het opzetten van zo'n woordenboek worden dezelfde geluidssignalen gehanteerd die ook worden gebruikt bij het opzetten van het model van het betreffende woord. Per geluidssignaal wordt nu met behulp van Feature Analyse een reeks van observatievectoren gecreëerd. De observatievectoren van alle geluidssignalen worden nu opgesplitst in W disjuncte verzamelingen. Van iedere verzameling wordt nu het zwaartepunt gezocht. Dit zwaartepunt is een vector uit de verzameling die de totale verzameling van vectoren uit de verzameling het best representeert. Alle andere vectoren uit de verzameling worden aan dit zwaartepunt gekoppeld. De vector die bijvoorbeeld verzameling w_1 reprenseert, wordt nu gekoppeld aan het discrete symbool w_1, waardoor alle vectoren uit die verzameling aan het symbool w_1 gekoppeld worden.

Met behulp van dit woordenboek kan nu de door Feature Analyse verkregen reeks observatievectoren $\mathbf{O} = (\mathbf{O_1}, ..., \mathbf{O_K})$ worden omgezet naar een reeks $O' = (O'_1, ..., O'_K)$, waarbij geldt dat $O'_i \in \{w_1, ..., w_W\}$ voor $1 \leq i \leq K$. Dit gebeurt door bij iedere observatievector $\mathbf{O_t}$ de meest dichtbij liggende zwaartepuntvector te zoeken en de observatievector aan het bij dit zwaartepunt behorende discrete symbool te koppelen. Zo ontstaat er een observatiereeks die wel op te lossen is met behulp van verborgen Markov modellen en het principe zoals eerder in dit hoofdstuk beschreven.

Hoofdstuk 5

Conclusie

Nadat we een Markovketen gedefinieerd hebben, hebben we met behulp van deze definitie het verborgen Markov model geïntroduceerd. Dit is een model dat bestaat uit een geheime verborgen Markovketen en per toestand van de Markovketen een aparte kansverdeling waaruit een observatie komt die voor de waarnemer wel bekend is. Hierdoor ontstaat een observatiereeks O. De parameters van het verborgen Markov model worden weergegeven als $\lambda = (A, B, \pi)$, waarbij A de matrix behorende bij de Markovketen is, B de matrix behorende bij de kansverdelingen in de verschillende toestanden en π de begintoestand.

We hebben drie problemen gezien die opgelost moeten worden om het verborgen Markov model nuttig te maken voor toepassingen. Het eerste probleem gaf ons een observatiereeks O en een model λ en dan moest de kans berekend worden dat die observatiereeks voorkomt bij het gegeven model. In andere woorden $\mathbb{P}(O|\lambda)$ moest berekend worden. Bij het tweede probleem waren ook een O en λ gegeven, maar nu moest er een reeks van toestanden gevonden worden die het meest waarschijnlijk was bij de gegeven observatiereeks. We hebben gezien dat dit probleem niet exact op te lossen is, omdat er geen echt goede uitkomst is, maar dat we wel een goede benadering kunnen maken. Tot slot was bij het derde probleem alleen een observatiereeks O gegeven. We wilden een model verkrijgen dat O het best beschrijft dus in andere woorden: we zochten een model λ waardoor $\mathbb{P}(O|\lambda)$ gemaximaliseerd wordt.

We hebben besproken dat deze problemen gebruikt kunnen worden bij de toepassing van verborgen Markov modellen op spraakherkenning. Hiervoor wordt eerst per woord (of woordonderdeel) van het systeem een verborgen Markov model opgesteld. Dit gebeurt door ieder woord meerdere keren uit te spreken en dan met behulp van probleem 3 een model op te stellen. Het systeem bestaat dus uit verschillende verborgen Markov modellen: per woord één. Als het systeem opgesteld is, kan de spraakherkenning worden gedaan. Dit gebeurt door een observatiereeks te maken van een geluidssignaal van het woord. Van deze observatiereeks wordt in ieder verborgen Markov model de kans $\mathbb{P}(O|\lambda)$ bepaald. Het woord dat bij het model met de grootste kans hoort, is het woord dat uit de spraakherkenning komt.

We hebben kort gezien hoe een geluidssignaal van een spraakopname wordt omgezet in een discrete observatiereeks die we kunnen gebruiken bij de discrete verborgen Markov modellen die we besproken hebben. Het geluidssignaal wordt met behulp van Feature Analyse omgezet in een reeks van observatievectoren. Vervolgens wordt Vector Kwantisering gebruikt om de reeks van observatievectoren om te zetten in een discrete observatiereeks.

In de scriptie hebben we slechts de discrete vorm van verborgen Markov modellen besproken, maar er bestaat ook een continue vorm. Deze zorgt bij de toepassing op spraakherkenning voor een grotere nauwkeurigheid. Onder andere omdat de observatievectoren door Vector Kwantisering minder nauwkeurig worden. Hierdoor kunnen fouten onstaan. Bij het gebruik van een continu model hoeft dit namelijk niet gedaan te worden. De observatievectoren die volgen uit Feature Analyse kunnen direct gebruikt worden. Verder is het opstellen van een continu verborgen Markov model wel lastiger dan een discreet verborgen Markov model. Dit is de reden dat we er voor gekozen hebben deze niet te behandelen in de scriptie, maar dit is wel het model dat in werkelijkheid vaak gebruikt wordt bij spraakherkenning.

Bibliografie

[1] Lawrence R. Rabiner, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*
Proceedings of the IEEE, Vol.77, No. 2, Februari 1989

[2] Sheldon M. Ross, *Introduction to Probability Models*
Academic Press Elsevier, 11^e editie, 2014

[3] *The EM algorithm*
`http://pandamatak.com/people/anand/771/html/node25.html`, 10/11/2018

[4] Daniel Ramage, *Hidden Markov Models Fundamentals*
CS229 Section Notes, 1 December 2007
`http://cs229.stanford.edu/section/cs229-hmm.pdf`, 25/11/2018

[5] Mark Gales en Steve Young, *The Application of Hidden Markov Models in Speech Recognition*
Foundations and Trends in Signal Processing, Vol. 1, No. 3, 2007

[6] *Fast Fourier Transform*
`https://nl.wikipedia.org/wiki/Fast_Fourier_transform`, 27/12/2018

[7] *Discrete Fourier Transform*
`https://en.wikipedia.org/wiki/Discrete_Fourier_transform`, 27/12/2018

[8] *Discrete Cosinustransformatie*
`https://nl.wikipedia.org/wiki/Discrete_cosinustransformatie`, 27/12/2018