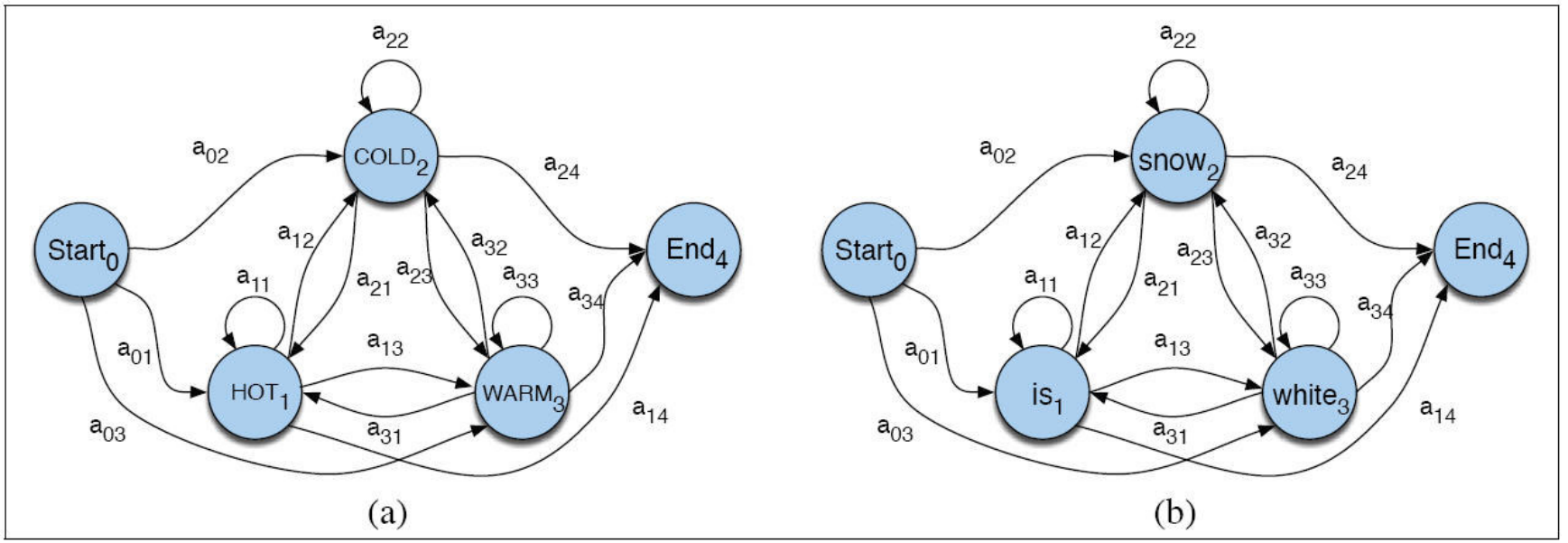
Speech and Language Processing

SLP Chapter 6
Hidden Markov Models:
the algorithms

Markov model

- Klassificatie probleem
  - Welke POS heeft een woord
  - Welke foneem behoort bij een stuk spraak
- Alternatieven, en kansen daarop
- Markov model als een vorm van een gewogen finite state transitie netwerk

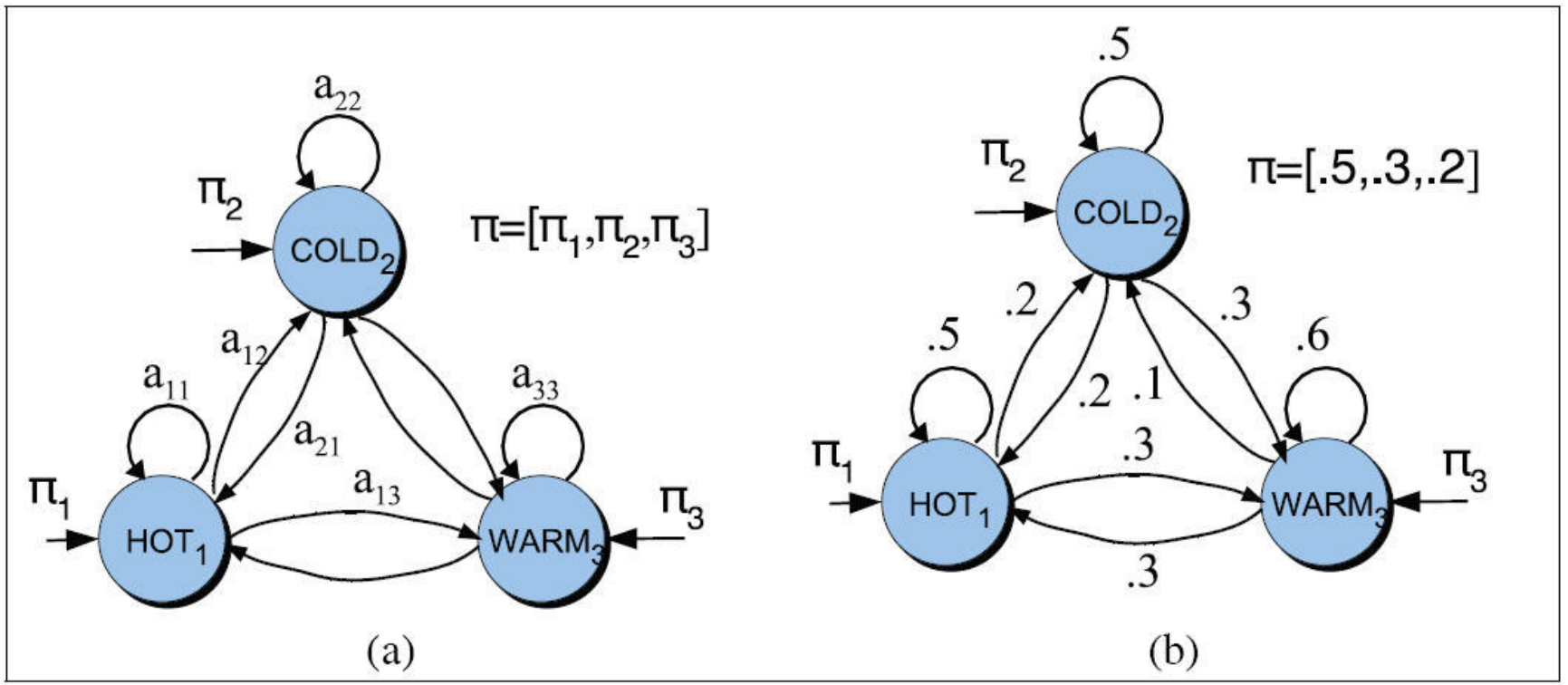
eenvoudige Markov keten



We hebben

- states q_i
- transities (met overgangskansen) a_{ij}
- start en eindstates (zonder output)

Het weer



Merk notatieverschil op: gebruik van π_i in plaats van a_{0i} voor $P(q_i|STAT)$

- Van eenvoudige Markov keten
- Naar Hidden Markov Model

HMM specificatie

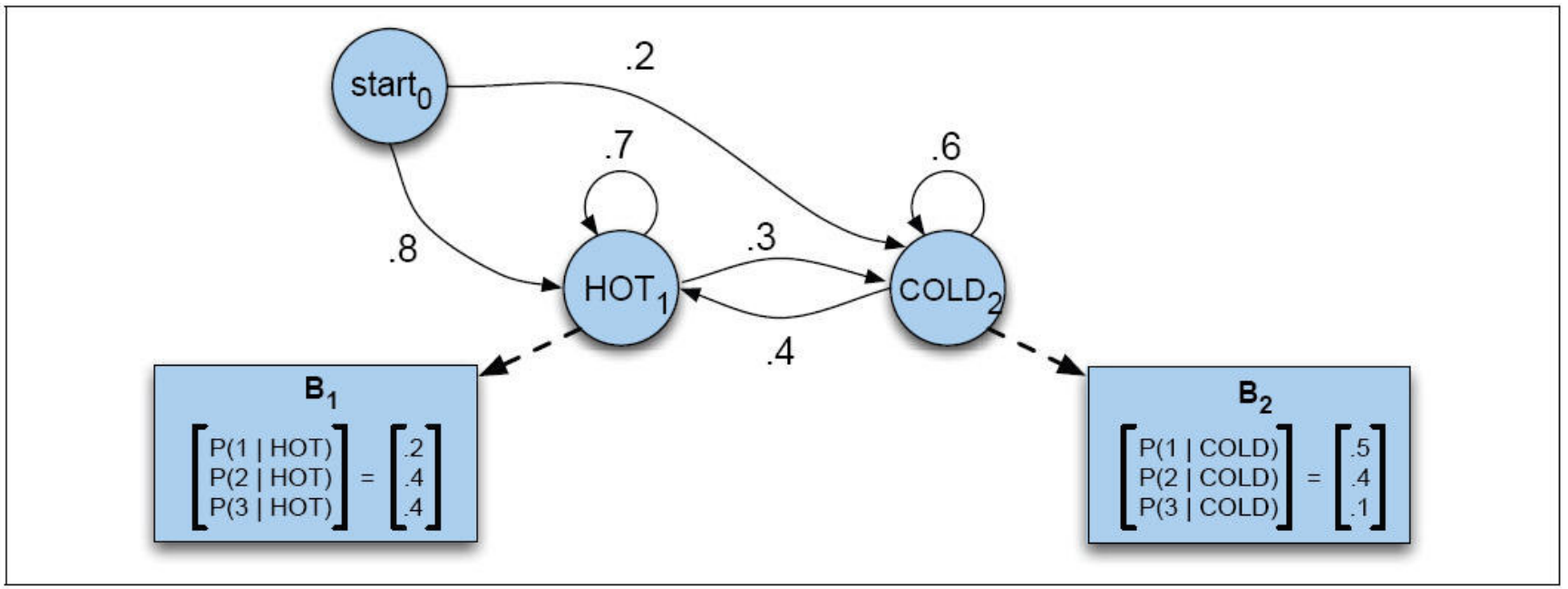
$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \ldots a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$
$QA = \{q_x, q_y ...\}$	a set $QA \subset Q$ of legal **accepting states**

Markov aanname

- De kans om in een state te zijn hangt alleen van de vorige state af (en niet van *alle* voorafgaande)
  $P(q_i|q_1...q_{i-1}) = P(q_i|q_{i-1})$

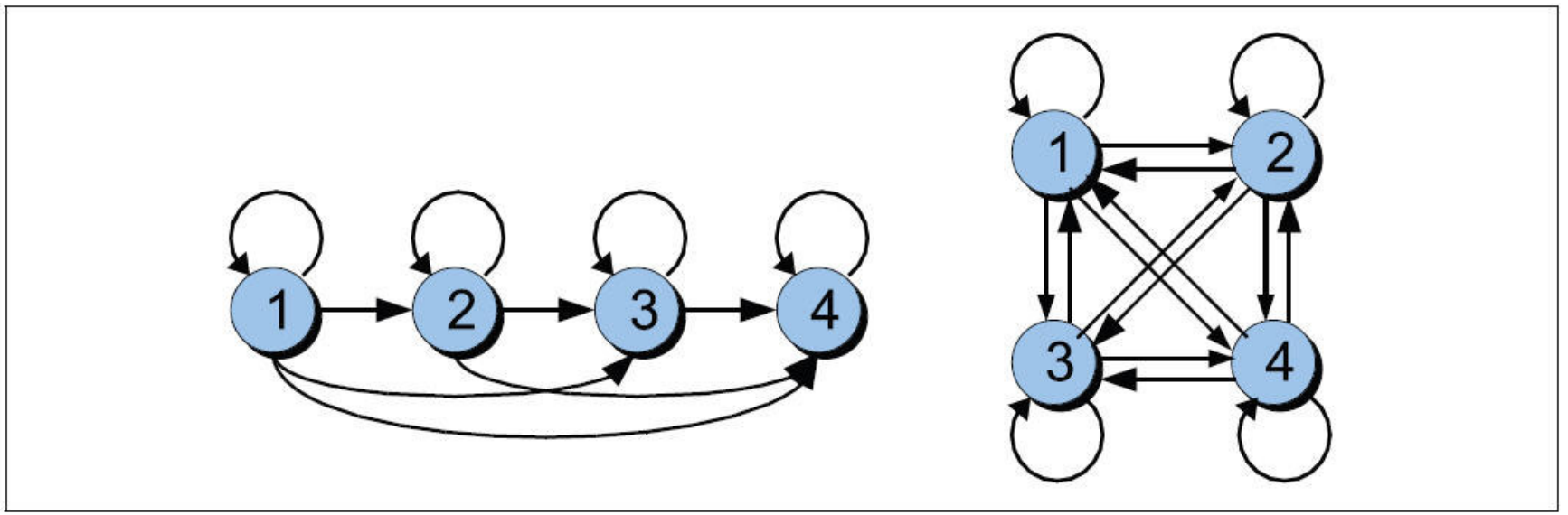
- De kans op een observatie hangt alleen van de huidige state af, en niet van alle alle observaties en voorgaande states
  $P(o_i|q_1 \ldots q_i, o_1\text{-}o_T) = P(o_i|q_i)$

Het ijsjes probleem



Als alle waarden van de HMM bekend zijn, dan kan de kans op een sequentie van warme en koude dagen worden afgeleid van de reeks gegeten ijsjes

Model architectuur



Links-Rechts model voor spraak
(kan niet terug in de tijd; hoe ziet A er dan uit?)

is bijzondere vorm van een

Ergodisch model: alle transities mogelijk

HMM: λ

HMM $\lambda = (A,B)$

ofwel, het Hidden Markov Model λ bestaat uit

A: transitiematrix
B: Output waarschijnlijkheden

De 3 HMM problemen

Problem 1 (Likelihood):	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Problem 2 (Decoding):	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Problem 3 (Learning):	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

POS toekenning al opgelost als Probleem 2 (met Viterbi algoritme)

Probleem 1: P(O|λ)

$$P(O|\lambda) = \Sigma P(O,Q)$$

de kans dat een HMM λ een observatiereeks O kan genereren, is gelijk aan de som van de kansen die langs alle mogelijke state reeksen (paden) bestaan.

$$P(O|\lambda) = \Sigma P(O,Q) = \Sigma P(O|Q)P(Q)$$

en dat voor HMM λ gelijk aan
(1) de kans op de state reeks (een pad) *
(2) de kans dat - gegeven dat pad – de observatiereeks wordt gegenereerd
(3) en dat gesommeerd over alle mogelijk state reeksen

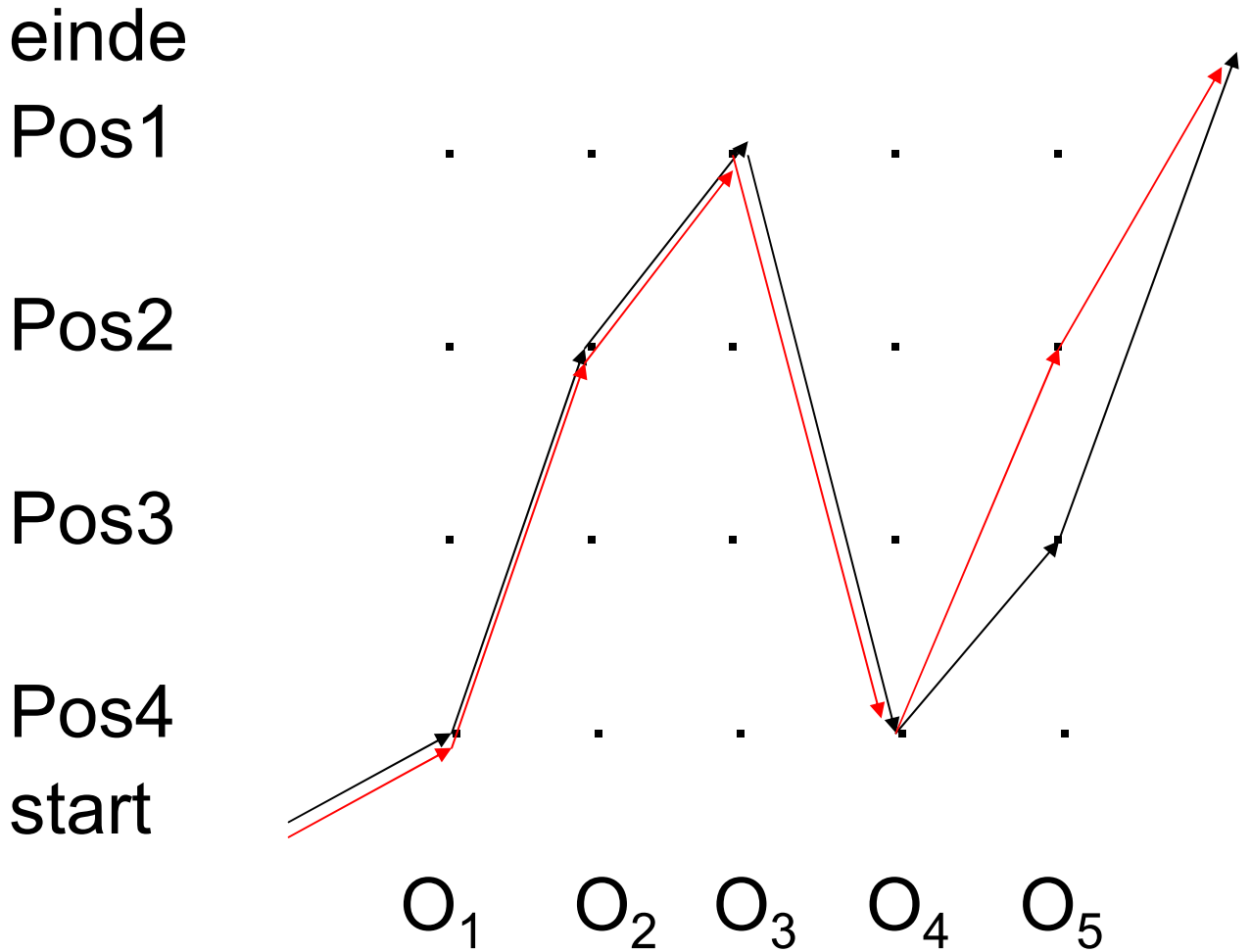
Probleem 1: P(O|λ)

- Kies een pad Q
- bereken P(O|Q) [dit hebben we al eens voor het ijsjes probleem gedaan]
- doe dat voor alle paden
- en sommeer alle verkregen P(O|Q)

- Maar dat is veel werk:

  N states, T observaties > N^T paden!

Grid van woorden en POS



Het rode pad verschilt alleen in voor O_5 van het zwarte pad.

Je doet de berekeningen over O_1 tot O_4 dubbel.

Forward algoritm

- Idee:
  maak gebruik van wat je al eerder hebt berekend

- definieer voorwaardse variabele op t

$$\alpha_t(j) = P(o_1, o_2 \ldots o_t, state_t = q_j | \lambda)$$

= de kans op alle observaties van o_1 tot en met o_t
en dat we op t in state q_j zijn

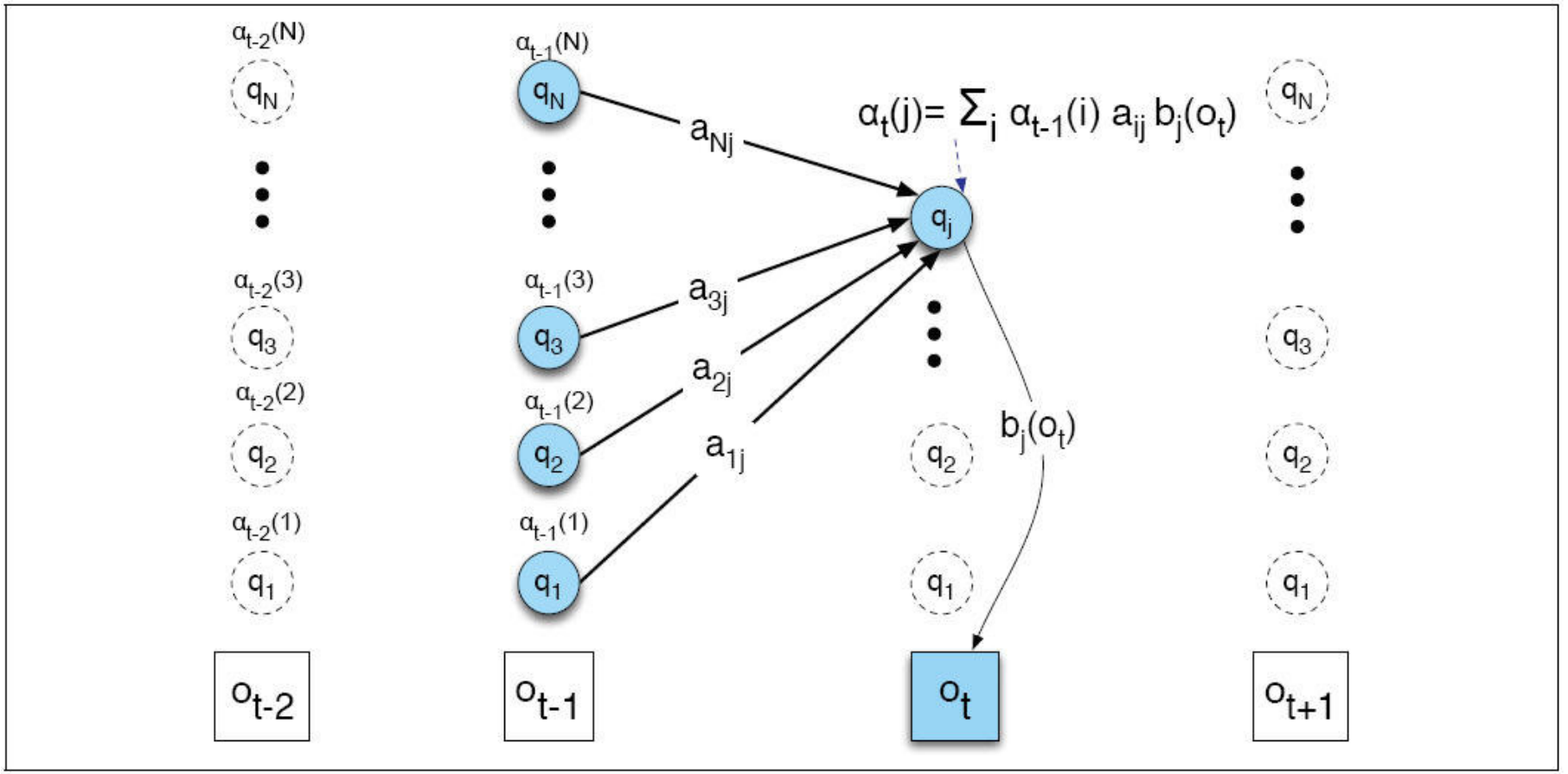
Forward algorithm

- $\alpha_t(j)$ kan in de voorwaardse variabele op t-1 worden uitgedrukt:

$$\alpha_t(j) = \Sigma\, \alpha_{t-1}(i)\, a_{ij}\, b_j(o_t)$$ (som over i=1 tot N)

$\alpha_{t-1}(i)$	the **previous forward path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

Recursie van $\alpha_t(j)$



Lijkt erg op Viterbi

Verschil

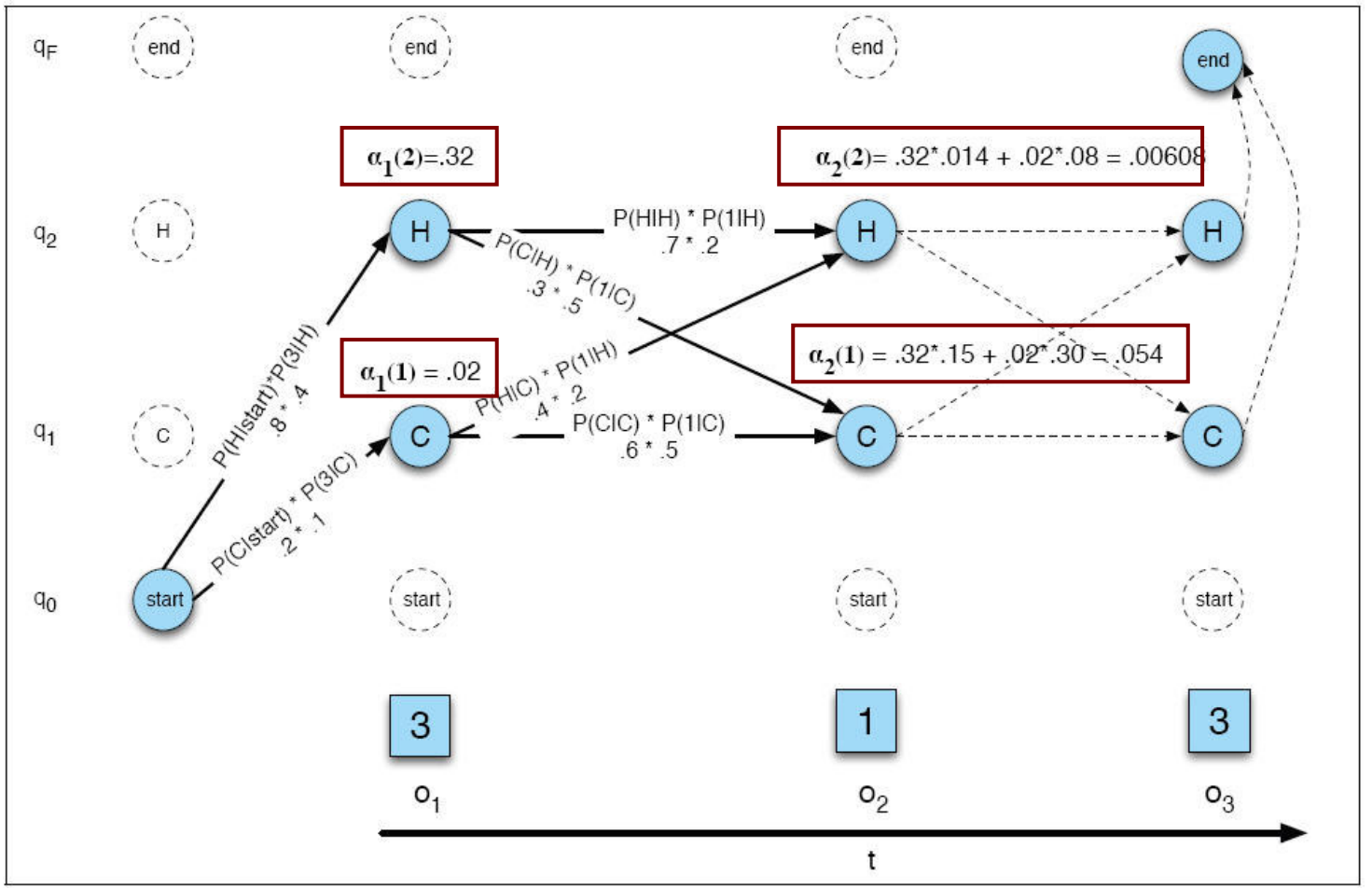
- *Viterbi* neemt alleen het maximum mee

$$viterbi_t(j) = \max \{viterbi_{t-1}(i)\, a_{ij}\, b_j(o_t)\}$$

- *Forward algoritme* sommeert alles (compleet)

$$\alpha_t(j) = \Sigma\, \alpha_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

3,1,3 ijsjes op hete of koude dagen



$P(O|\lambda)$

- Hoe begin je?
  - op t=1 kan dat in elke state i
  - $\alpha_1(i) = a_{01} * b_i(O_1)$
- Vervolgens recursie: $\alpha_t(i)$ afleiden uit alle $\alpha_{t-1}(j)$
- Hoe eindig je?
  - de laatste observatie (T)
  - je kunt eindigen in elke state i
  - voor elke state i heb je $\alpha_T(i)$
    - de kans op alle observaties, einde in state i en dan finish
- $P(O|\lambda)$ is dan $\sum \alpha_T(i)a_{iF}$ (sommeren over alle states)

alternatief: backward algorithm

- Forward algorithm werkt van voor naar achter
- Backward algorithm van achter naar voor

Backward algorithm

- Achterwaardse variabele op t

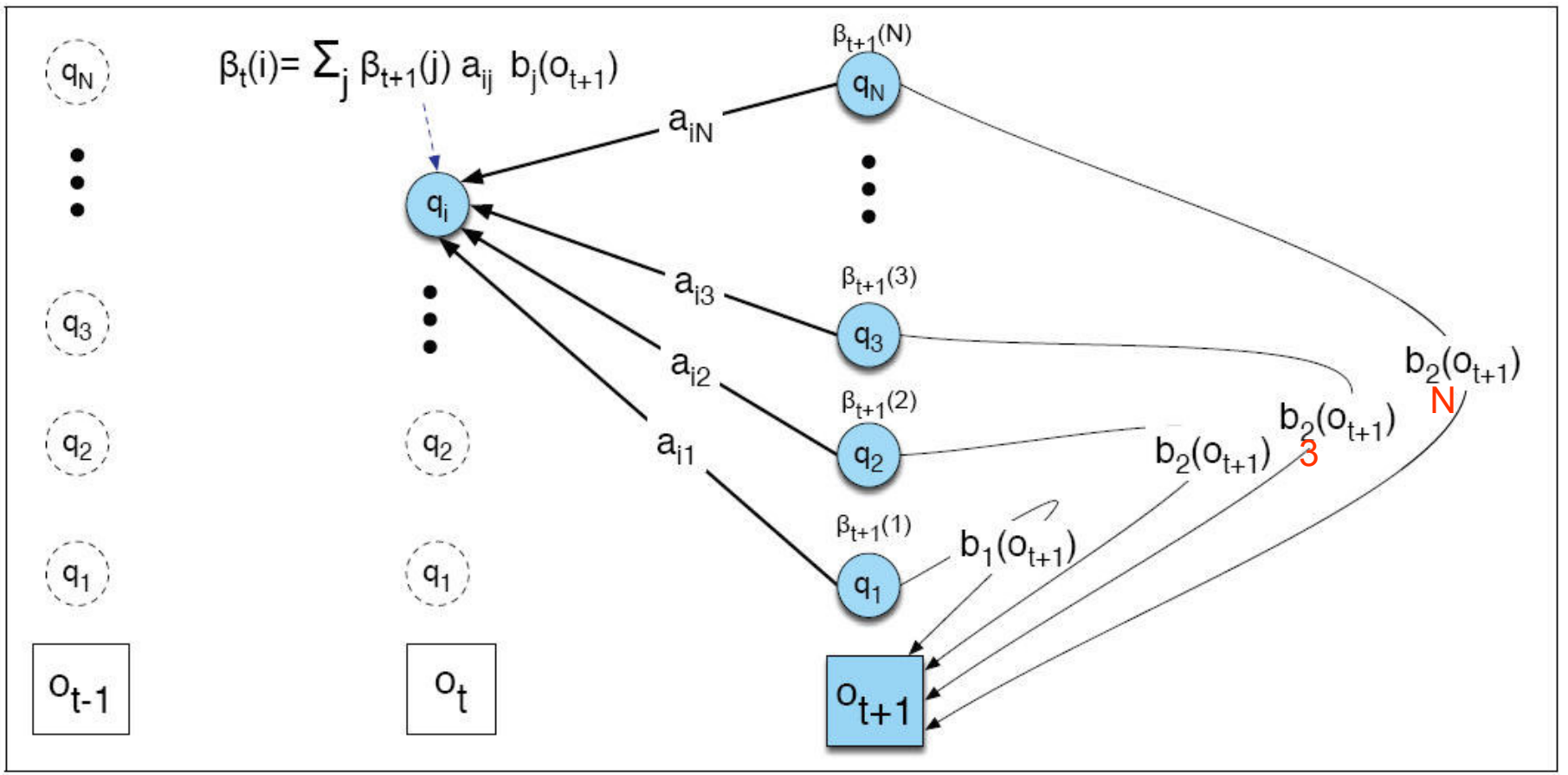
$$\beta_t(i) = P(o_{t+1}, o_{t+2} \ldots o_T \mid state_t = q_i , \lambda)$$

= de kans op alle observaties van o_{t+1} tot en met o_T
en dat we op t in state q_i zijn

- Deze kan in de achterwaardse variabele op t+1 worden uitgedrukt:

$$\beta_t(i) = \Sigma\, a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j)$$ (som over j=1 tot N)

Recursie van $\beta_t(i)$



P(O|λ) van achter naar voor

- Hoe begin je?
  - Op t=T kun je eindigen in elke state i, en vervolgens finish
  - $\beta_T(i) = a_{iF}$
- Vervolgens recursie: $\beta_t(i)$ afleiden uit alle $\beta_{t+1}(j)$
- Hoe eindig je de recursie?
  - met de eerste observatie (O_1)

Probleem 2: beste pad

- Al opgelost met Viterbi algoritme

- Door steeds te onthouden van welke state je komt, als je het maximum van $v_t(i)$ bepaalt

- Merk nogmaals op: Viterbi berekent uiteindelijk *alleen* de totale kans langs dit beste pad (en niet langs alle mogelijke paden)

Probleem 3: training van HMM

- Hoe komen we aan de transitiematrix A en de outputwaarschijnlijkheden B?

- Beschikbaar:
  - Modelarchitectuur in A (de positie van nullen)
  - Observaties die door het model gegenereerd zijn (training)

eenvoudig Markovmodel

- niets is verborgen (B=1)

  output bv: det, N, V, P, det, Adj, N,…

- alleen transitiematrix A te berekenen

- Voor a_{ij}

  - tel alle overgangen i naar j (det N = 1)
  - tel alle keren dat je i ziet (det = 2)
  - deel beide aantallen (a_{detN} = 0.5)

Hidden Markov Model

- Je kunt niet tellen, want de state is verborgen
- Er moet daarom met kansen worden gewerkt
  (je kunt niet meer eenvoudig tellen)

- De taak is om A en B te schatten, gegeven een observatiereeks

HMM algemene aanpak

- Kies een willekeurig model
  (dwz kies A en B willekeurig)
- Pas dit model toe op observatiereeks O
  (trainingsdata)
- Bereken dan een nieuwe schatting
  van A en B
  (deze is altijd beter!)
- Herhaal dit tot de schatting niet meer
  verandert

HMM training van A

- Nieuwe schatting van transitie $\hat{a}_{ij}$
  onder het huidige model

  = verwacht aantal transities van i naar j / verwacht aantal transities vanuit i

  = kans op transitie van i naar j / kans op transities vanuit i

  = kans op transitie van i naar j / kans om in state i te zijn = $\frac{P(q_t=i, q_{t+1}=j)}{P(q_t=i)}$

- Hoe bereken je deze kansen?
  -> doe dit voor een bepaalde t en sommeer dan over alle t

transitie van i naar j

- kans op overgang op t van i naar j:

  $P(q_t=i, q_{t+1}=j|O,\lambda)$

  $= P(q_t=i, q_{t+1}=j,O|\lambda) / P(O|\lambda)$

  – $P(O|\lambda)$ is oplossing probleem 1

  – $P(q_t=i, q_{t+1}=j,O|\lambda)$

    is de kans op een transitie van i naar j op t,en alle observaties,

    gegeven het huidige model

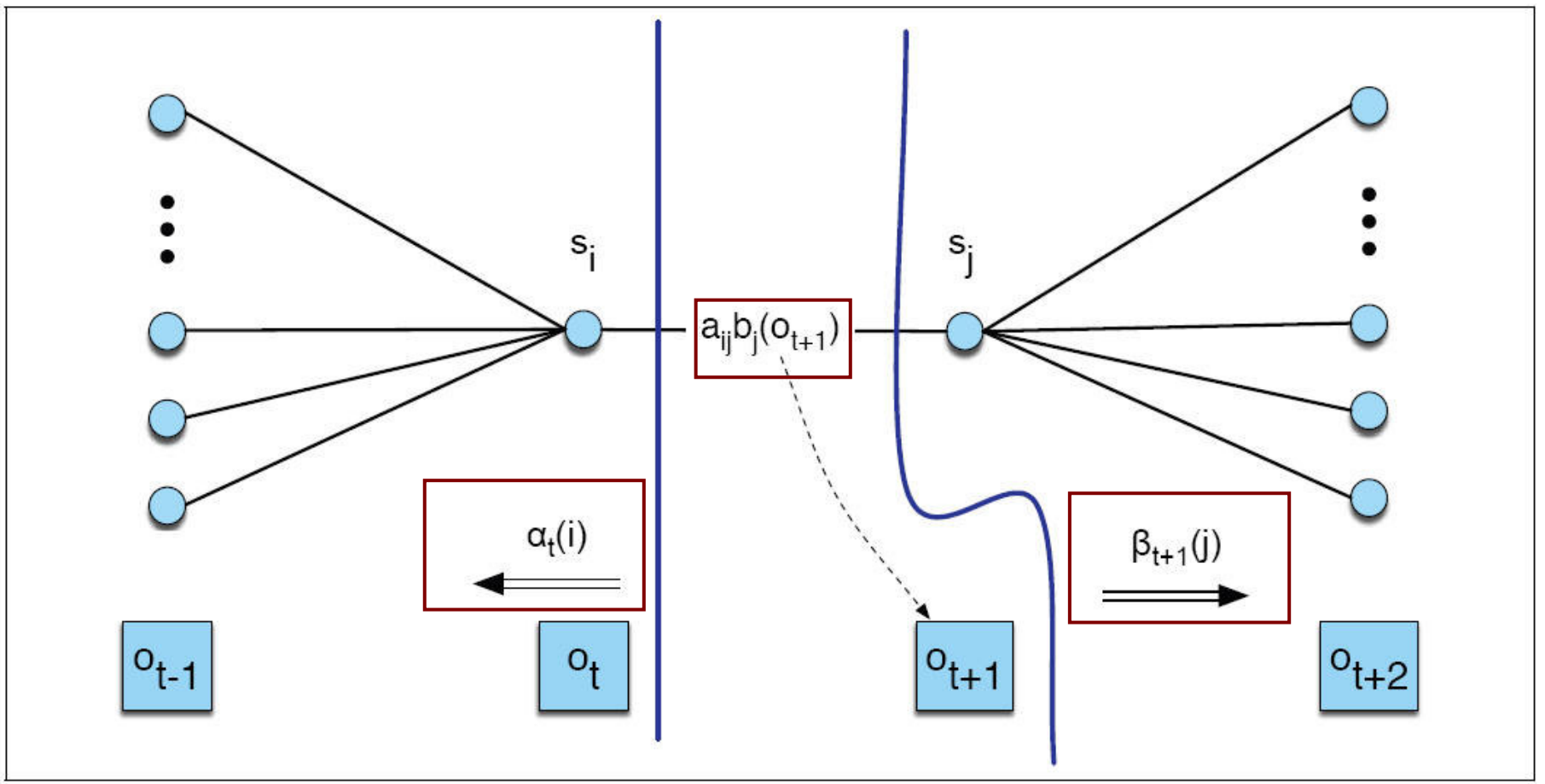
$P(q_t=i, q_{t+1}=j, O|\lambda)$

- kans daarop is op t

  $\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)$

  - eerst moet je in i zijn (en alle observaties t/m O_t gehad hebben): $\alpha_t(i)$
  - dan moet je een overgang van state i naar state j maken: a_{ij}
  - dan moet je de observatie O_{t+1} zien: $b_j(O_{t+1})$
  - en ten slotte moet je na state j in t+1, alle resterende observaties zien: $\beta_{t+1}(j)$

$P(q_t=i, q_{t+1}=j, O|\lambda)$



sommeren over t

- we berekenden voor t de kans op de overgang van state i naar state j (en de observatiereeks)
- dat sommeren we over alle 1 t/m T-1
- deel door $P(O|\lambda)$ (=normering)

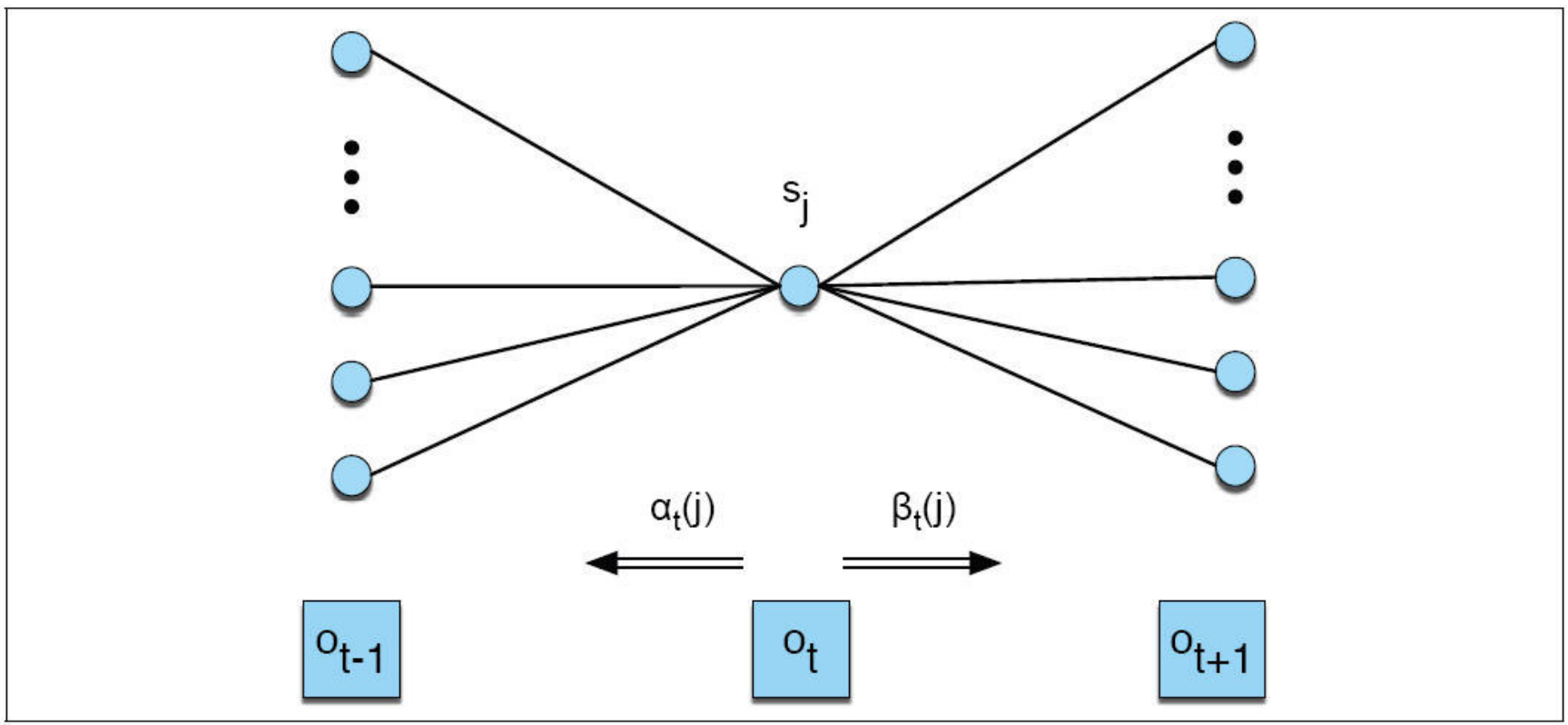
kans om in state i te zijn

op t:

- $P(q_t=i,O|\lambda) = \alpha_t(i)\ \beta_t(i) / P(O|\lambda)$
  - eerst moet je in i zijn (en alle observaties t/m O_t gezien hebben): $\alpha_t(i)$
  - dan moet je in i de observatie O_{t+1} zien en daarna alle resterende observaties : $\beta_t(i)$

en dan sommeren over 1 t/m T-1

$P(q_t=j,O|\lambda)$



Plaatje is j ipv i

samen

$$\hat{a}_{ij} = \frac{\sum \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum \alpha_t(i)\, \beta_t(i)}$$

- sommeren over 1 t/m T-1
- normeringen op $P(O|\lambda)$ vallen weg
- forward-backward (Baum-Welsh) algoritme

training van B

- Schatting van observatiewaarschijnlijkheid $b_j(v_k)$

  = $\frac{\text{verwacht aantal malen in j en dan } v_k \text{ observeren}}{\text{verwacht aantal malen in j}}$

  = $\frac{\text{kans om in state j te zijn en } v_k \text{ te zien}}{\text{kans om in state j te zijn}}$

kennen we al: $P(q_t=i,O|\lambda) = \alpha_t(i)\,\beta_t(i) / P(O|\lambda)$

Nieuwe schatting $\hat{b}_j(v_k)$

$\hat{b}_j(v_k)$ =

som over t van de kans om in state j te zijn EN v_k te observeren ($O_t = v_k$)

gedeeld door

som over t van de kans om in state j te zijn

Herhalen

- Het forward-backward algoritme geeft op basis van een eerste willekeurige schatting van A en B, elke keer daarna een betere schatting (op basis van O)