Markov Modelle

Markov Modelle

Modelle für zeitlich voneinander abhängende Zustände und sich daraus ergebende Sequenzen:

$$\boldsymbol{\omega}^T = \{\omega(1), \omega(2), \ldots, \omega(T)\}$$

Markov Modelle erster Ordnung:

Zustand zum Zeitpunkt $t+1$ hängt vom Zustand zum Zeitpunkt t ab. Abhängigkeit wird beschrieben durch Übergangswahrscheinlichkeiten

$$P\left(\omega_j(t+1) \mid \omega_i(t)\right) = a_{ij}$$

Wie groß ist $P\left(\boldsymbol{\omega}^T \mid \boldsymbol{\theta}\right)$?

Markov Modelle

Antwort:

$P\left(\boldsymbol{\omega}^T \mid \boldsymbol{\theta}\right)$ ergibt sich als Produkt der beteiligten Übergangswahrscheinlichkeiten a_{ij} und der Startwahrscheinlichkeiten $P\left(\omega\left(1\right) = \omega_i\right)$

Bemerkung: Die Darstellung wird etwas einfacher, wenn wir einen festgelegten Startzustand $\omega\left(0\right)$ definieren, mit: $P\left(\omega\left(0\right) = \omega_0\right) = 1$

Markov Modell erster Ordnung:

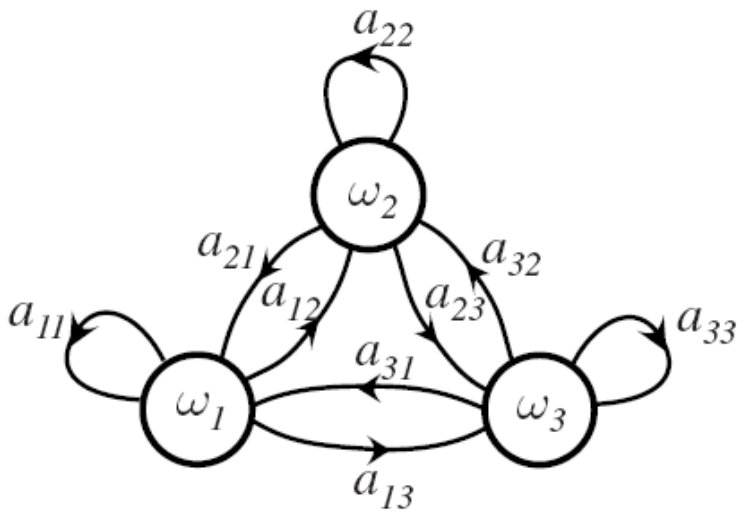


FIGURE 3.8. The discrete states, ω_i, in a basic Markov model are represented by nodes, and the transition probabilities, a_{ij}, are represented by links. In a first-order discrete-time Markov model, at any step t the full system is in a particular state $\omega(t)$. The state at step $t+1$ is a random function that depends solely on the state at step t and the transition probabilities. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Hidden Markov Models

Hidden Markov Model:

Die Zustände sind nicht beobachtbar, sondern nur daraus mit bestimmter Wahrscheinlichkeit $P\left(v_k\left(t\right)|\omega_j\left(t\right)\right)=b_{jk}$ folgende visible states:

$$\mathbf{V}^T=\left\{v\left(1\right),v\left(2\right),\ldots,v\left(T\right)\right\}$$

Hidden Markov Model:

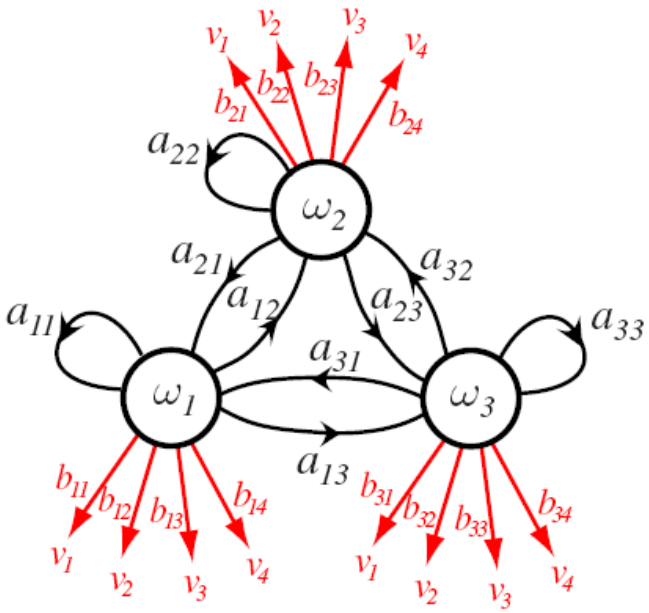


FIGURE 3.9. Three hidden units in an HMM and the transitions between them are shown in black while the visible states and the emission probabilities of visible states are shown in red. This model shows all transitions as being possible; in other HMMs, some such candidate transitions are not allowed. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Hidden Markov Models

Hidden Markov Models werden z.B. zur Spracherkennung eingesetzt.

Drei interessante Probleme:

1. Bewertungsproblem: Bestimme $P\left(\mathbf{V}^T \mid \boldsymbol{\theta}\right)$, gegeben: $\mathbf{V}^T, a_{ij}, b_{jk}$
2. Dekodierungsproblem: Bestimme wahrscheinlichste Folge $\boldsymbol{\omega}^T$, gegeben: $\mathbf{V}^T$
3. Lernproblem: Bestimme a_{ij}, b_{jk}, gegeben Menge von Trainingsdaten $\left\{\mathbf{V}^T{}_n\right\}$

HMM: Bewertung

1. Bewertungsproblem:

$$P\left(\mathbf{V}^T \mid \boldsymbol{\theta}\right) = \sum_{r=1}^{r_{\max}} P\left(\mathbf{V}^T \mid \boldsymbol{\omega}_r^T\right) P\left(\boldsymbol{\omega}_r^T\right)$$

Summe über alle möglichen Folgen, die Beobachtung hervorgerufen haben könnten.

Anzahl der möglichen Zustände: c

Anzahl der möglichen Folgen: $r_{\max} = c^T$

HMM: Bewertung

Es gilt: $$P\left(\boldsymbol{\omega}_r^T\right) = \prod_{t=1}^{T} P\left(\omega(t) \mid \omega(t-1)\right)$$

(Produkt der verschiedenen a_{ij})

und: $$P\left(\mathbf{V}^T \mid \boldsymbol{\omega}_r^T\right) = \prod_{t=1}^{T} P\left(v(t) \mid \omega(t)\right)$$

(Produkt der verschiedenen b_{jk})

Insgesamt erhalten wir also:

$$P\left(\mathbf{V}^T \mid \boldsymbol{\theta}\right) = \sum_{r=1}^{r_{\max}} \prod_{t=1}^{T} P\left(v(t) \mid \omega(t)\right) P\left(\omega(t) \mid \omega(t-1)\right)$$

Dabei ist normalerweise

$\omega(T) = \omega_0 ; a_{00} = 1$ (absorbierender Zustand)

HMM: Bewertung

Berechnungsaufwand:

$$P\left(\mathbf{V}^T \mid \boldsymbol{\theta}\right) = \sum_{r=1}^{r_{\max}} \prod_{t=1}^{T} P\left(v\left(t\right) \mid \omega\left(t\right)\right) P\left(\omega\left(t\right) \mid \omega\left(t-1\right)\right)$$

$$\boxed{O\left(c^T T\right)}$$

Beispiel: c=10, T=20, 10^{21} Berechnungen!

HMM: Bewertung

Viel besser: Forward Algorithmus

Wir definieren und berechnen rekursiv:

$$\alpha_j(t) = \begin{cases} 0 & t = 0,\ j \neq \text{Anfangszustand} \\ 1 & t = 0,\ j = \text{Anfangszustand} \\ \left[\sum_i \alpha_i(t-1)\, a_{ij}\right] b_{jk}\left(v(t) = v_k\right) & \text{sonst} \end{cases}$$

Aufwand: $\boxed{O\left(c^2 T\right)}$

Beispiel: c=10, T=20, 2000 Berechnungen!

HMM: Bewertung

Forward Algorithmus

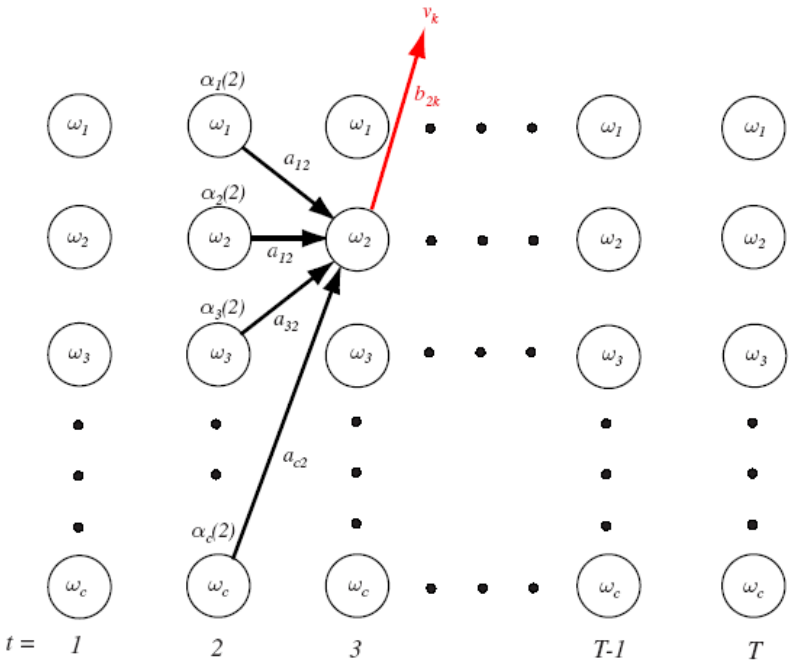
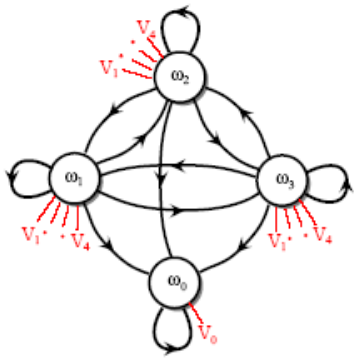


FIGURE 3.10. The computation of probabilities by the Forward algorithm can be visualized by means of a trellis—a sort of "unfolding" of the HMM through time. Suppose we seek the probability that the HMM was in state ω_2 at $t = 3$ and generated the observed visible symbol up through that step (including the observed visible symbol v_k). The probability the HMM was in state $\omega_j(t = 2)$ and generated the observed sequence through $t = 2$ is $\alpha_j(2)$ for $j = 1, 2, \ldots, c$. To find $\alpha_2(3)$ we must sum these and multiply the probability that state ω_2 emitted the observed symbol v_k. Formally, for this particular illustration we have $\alpha_2(3) = b_{2k} \sum_{j=1}^{c} \alpha_j(2) a_{j2}$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

HMM: Bewertung

Forward Algorithmus Beispiel:

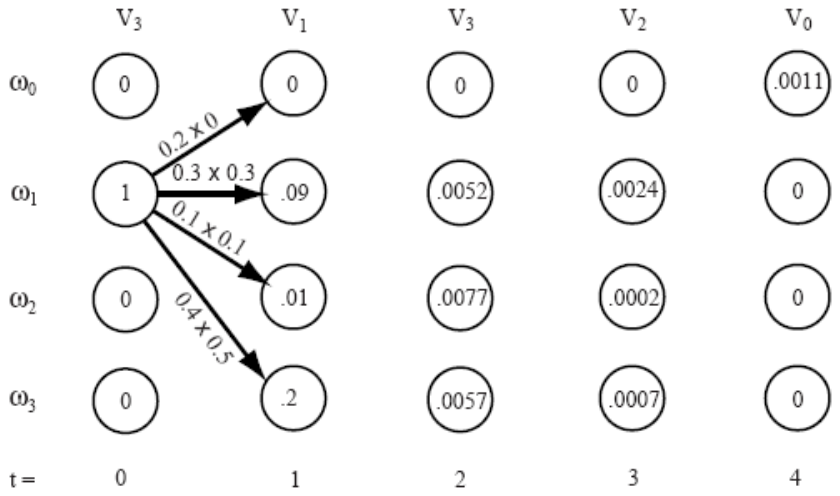
(Start mit ω_1)



$$a_{ij} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.2 & 0.3 & 0.1 & 0.4 \\ 0.2 & 0.5 & 0.2 & 0.1 \\ 0.8 & 0.1 & 0.0 & 0.1 \end{pmatrix} \text{ and}$$

$$b_{jk} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0.4 & 0.1 & 0.2 \\ 0 & 0.1 & 0.1 & 0.7 & 0.1 \\ 0 & 0.5 & 0.2 & 0.1 & 0.2 \end{pmatrix}.$$

$$\mathbf{V}^5 = \{v_3, v_1, v_3, v_2, v_0\}$$



HMM: Bewertung

Mustererkennung mit HMM:

Für jede Kategorie ein HMM, berechne alle Wahrscheinlichkeiten, dann Bayes:

$$P\left(\boldsymbol{\theta} \mid \mathbf{V}^T\right) = \frac{P\left(\mathbf{V}^T \mid \boldsymbol{\theta}\right) P\left(\boldsymbol{\theta}\right)}{P\left(\mathbf{V}^T\right)}$$

HMM: Bewertung

Bei Spracherkennung vereinfachtes Modell:

left-to-right HMM

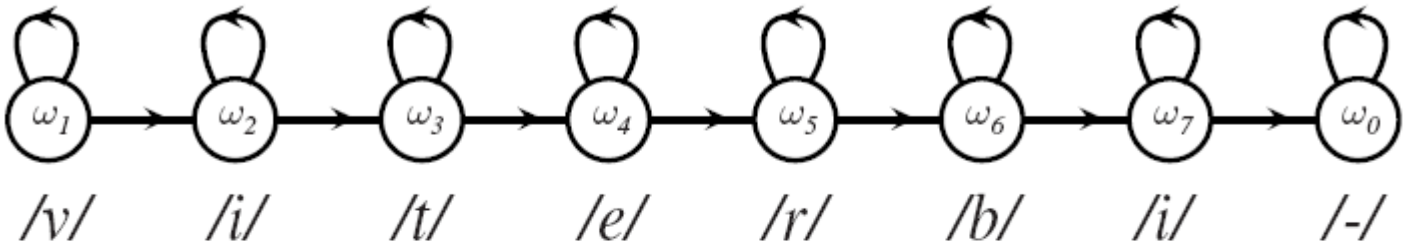


FIGURE 3.11. A left-to-right HMM commonly used in speech recognition. For instance, such a model could describe the utterance "viterbi," where ω_1 represents the phoneme /v/, ω_2 represents /i/,. . . , and ω_0 a final silent state. Such a left-to-right model is more restrictive than the general HMM in Fig. 3.9 because it precludes transitions "back" in time. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

2. Dekodierungsproblem

Dekodierung mit Forward-Algorithmus:

Gegeben: $\mathbf{V}^T$

Wir können wieder berechnen:

$$\alpha_j(t) = \left[\sum_i \alpha_i(t-1)\, a_{ij}\right] b_{jk}\left(v(t) = v_k\right)$$

und dann das ω_j auswählen, das zum maximalen $\alpha_j(t)$ gehört. Wir erhalten eine Sequenz von Zuständen $\boldsymbol{\omega}^T$, die allerdings nicht unbedingt erlaubt sein muss.

Forward Algorithmus

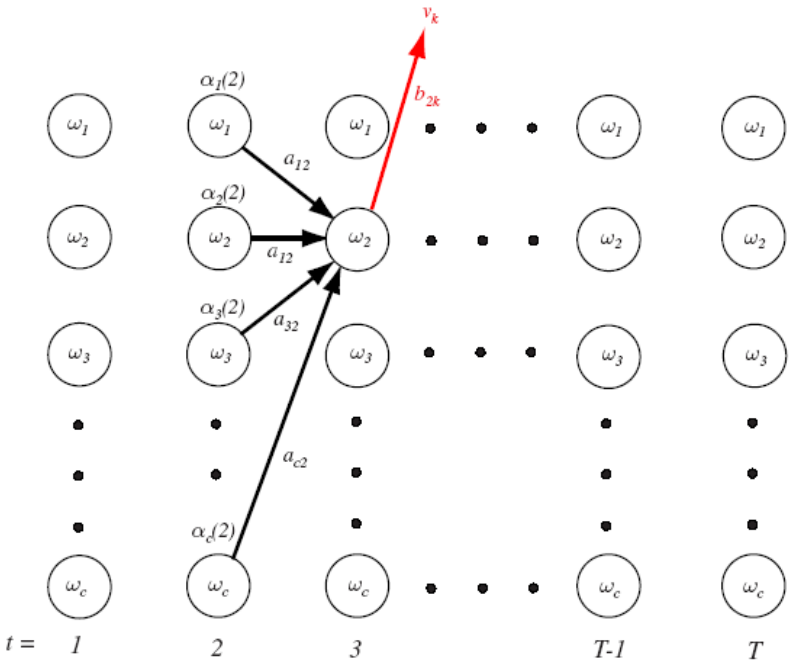


FIGURE 3.10. The computation of probabilities by the Forward algorithm can be visualized by means of a trellis—a sort of "unfolding" of the HMM through time. Suppose we seek the probability that the HMM was in state ω_2 at $t = 3$ and generated the observed visible symbol up through that step (including the observed visible symbol v_k). The probability the HMM was in state $\omega_j(t = 2)$ and generated the observed sequence through $t = 2$ is $\alpha_j(2)$ for $j = 1, 2, \ldots, c$. To find $\alpha_2(3)$ we must sum these and multiply the probability that state ω_2 emitted the observed symbol v_k. Formally, for this particular illustration we have $\alpha_2(3) = b_{2k} \sum_{j=1}^{c} \alpha_j(2) a_{j2}$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

HMM - Dekodierung

Dekodierung mit Forward-Algorithmus

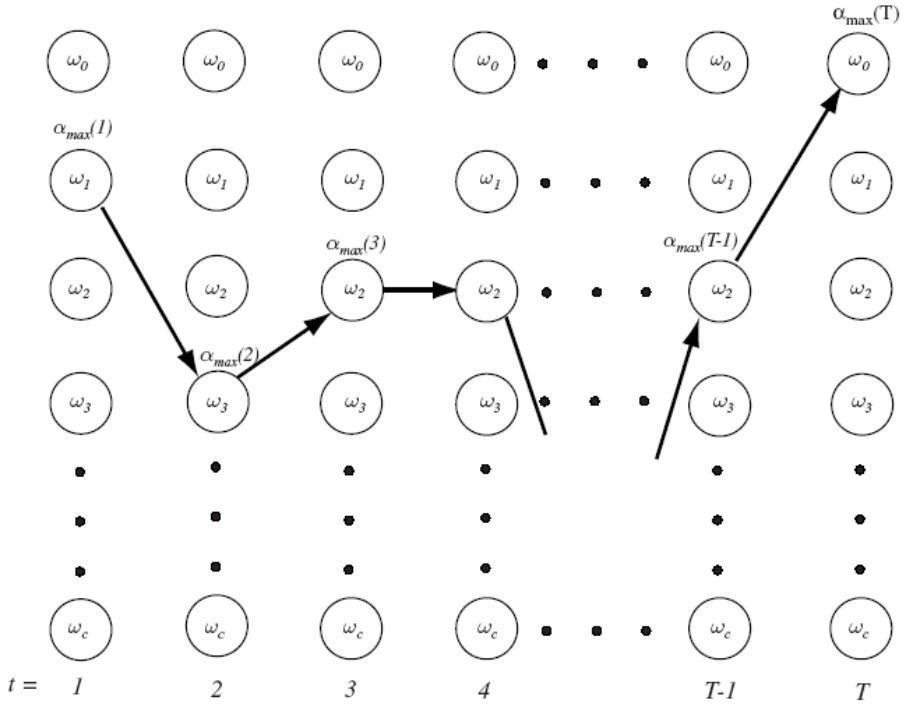


FIGURE 3.12. The decoding algorithm finds at each time step t the state that has the highest probability of having come from the previous step and generated the observed visible state v_k. The full path is the sequence of such states. Because this is a local optimization (dependent only upon the single previous time step, not the full sequence), the algorithm does not guarantee that the path is indeed allowable. For instance, it might be possible that the maximum at $t = 5$ is ω_1 and at $t = 6$ is ω_2, and thus these would appear in the path. This can even occur if $a_{12} = P(\omega_2(t+1)|\omega_1(t)) = 0$, precluding that transition. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Hidden Markov Models

Backward-Algorithmus:

Analog zu den $\alpha_j(t)$ beim Forward-Algorithmus definieren wir $\beta_i(t)$ für eine Sequenz $\mathbf{V}^T$ als Wahrscheinlichkeit, dass sich das Modell, wenn es sich im Zustand $\omega_i(t)$ befindet, den Rest der gegebenen sichtbaren Sequenz produziert:

$$\beta_i(t) = \begin{cases} 1 & t = T+1; i = c+1 \\ 0 & t = T+1; i \neq c+1 \\ a_{i,c+1} & t = T \\ \sum_j \beta_j(t+1) a_{ij} b_{jk} \left(v(t+1) = v_k\right) & \text{sonst} \end{cases}$$

Dabei ist $a_{i,c+1}$ die Wahrscheinlichkeit für Zustand ω_i, der letzte Zustand der Sequenz vor dem eindeutig definierten Endzustand ω_{c+1} zu sein.

Viterbi-Dekoder

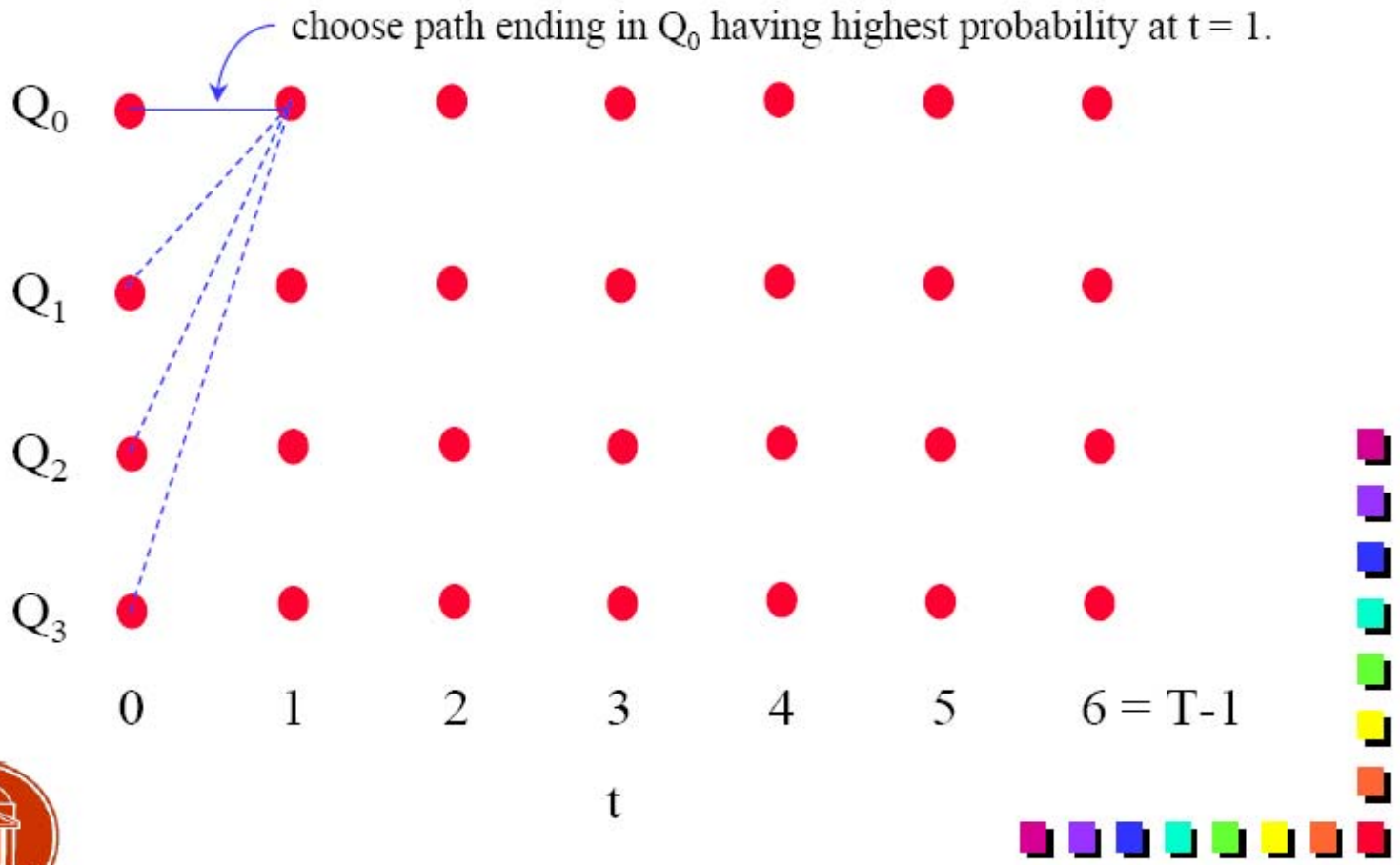
Wir berechnen rekursiv die Wahrscheinlichkeit $\phi_j(t)$ für den wahrscheinlichsten Pfad, der mit $v(t) = v_k$ endet und den vorherigen wahrscheinlichsten Pfad einschließt:

$$\phi_j(t) = \begin{cases} 0 & t = 0,\ j \neq 0 \\ 1 & t = 0,\ j = 0 \\ b_{jk}\left(v(t) = v_k\right) \max\limits_i \left[\phi_i(t-1)\, a_{ij}\right] & \text{sonst} \end{cases}$$

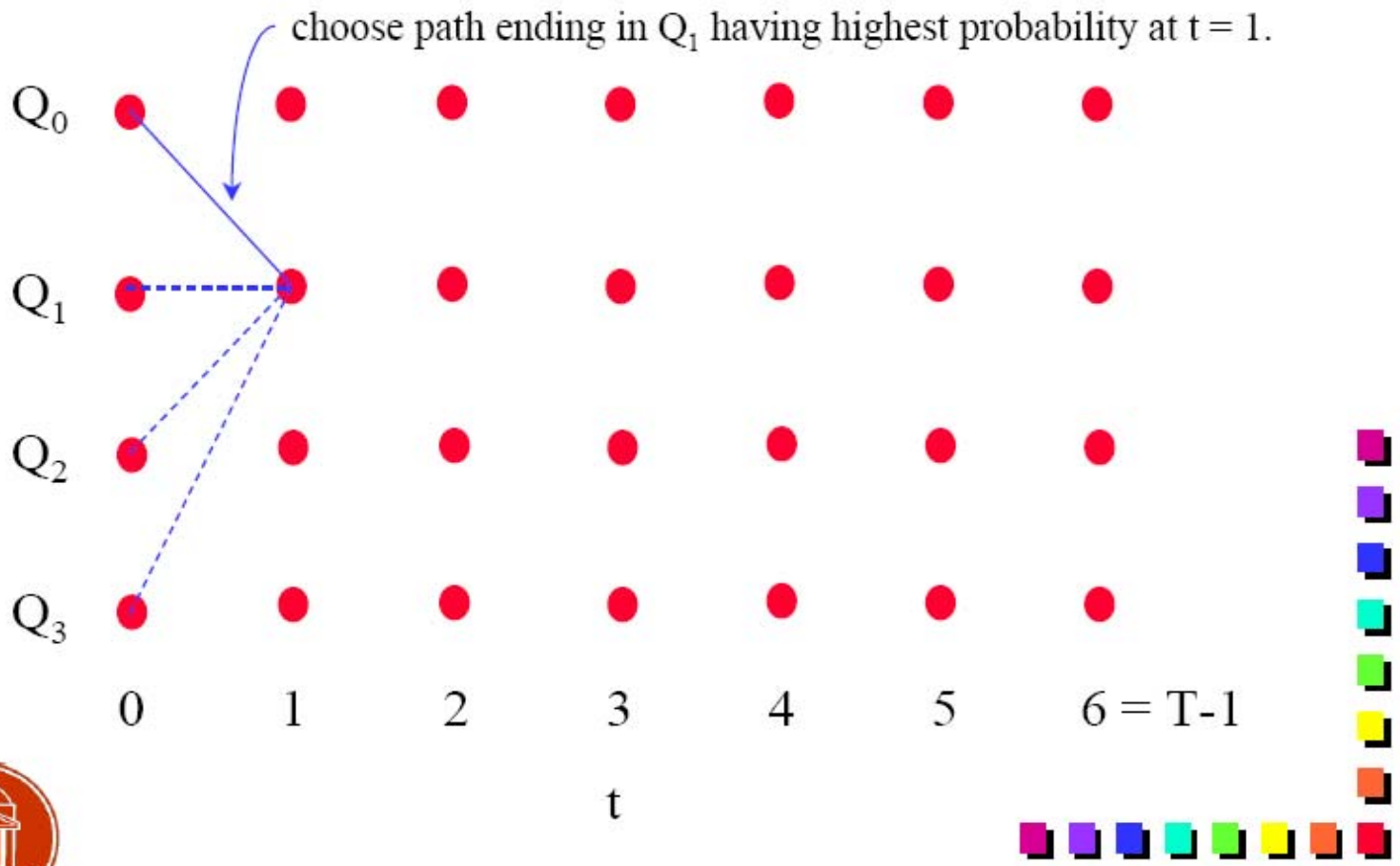
HMM – Viterbi-Dekoder

Achtung: in den folgenden Folien wird eine etwas andere Nomenklatur verwendet.

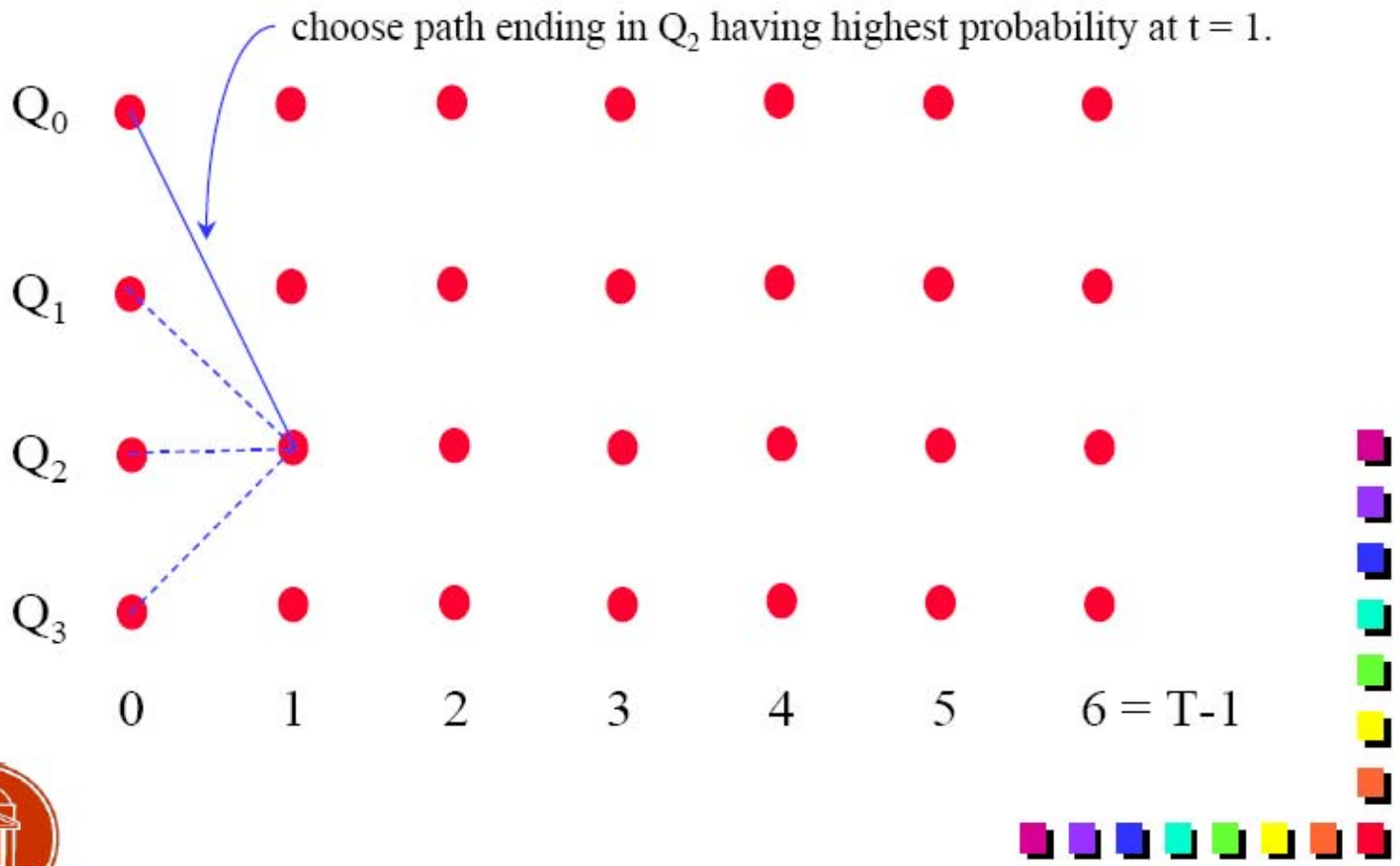
Viterbi Algorithm (cont.)



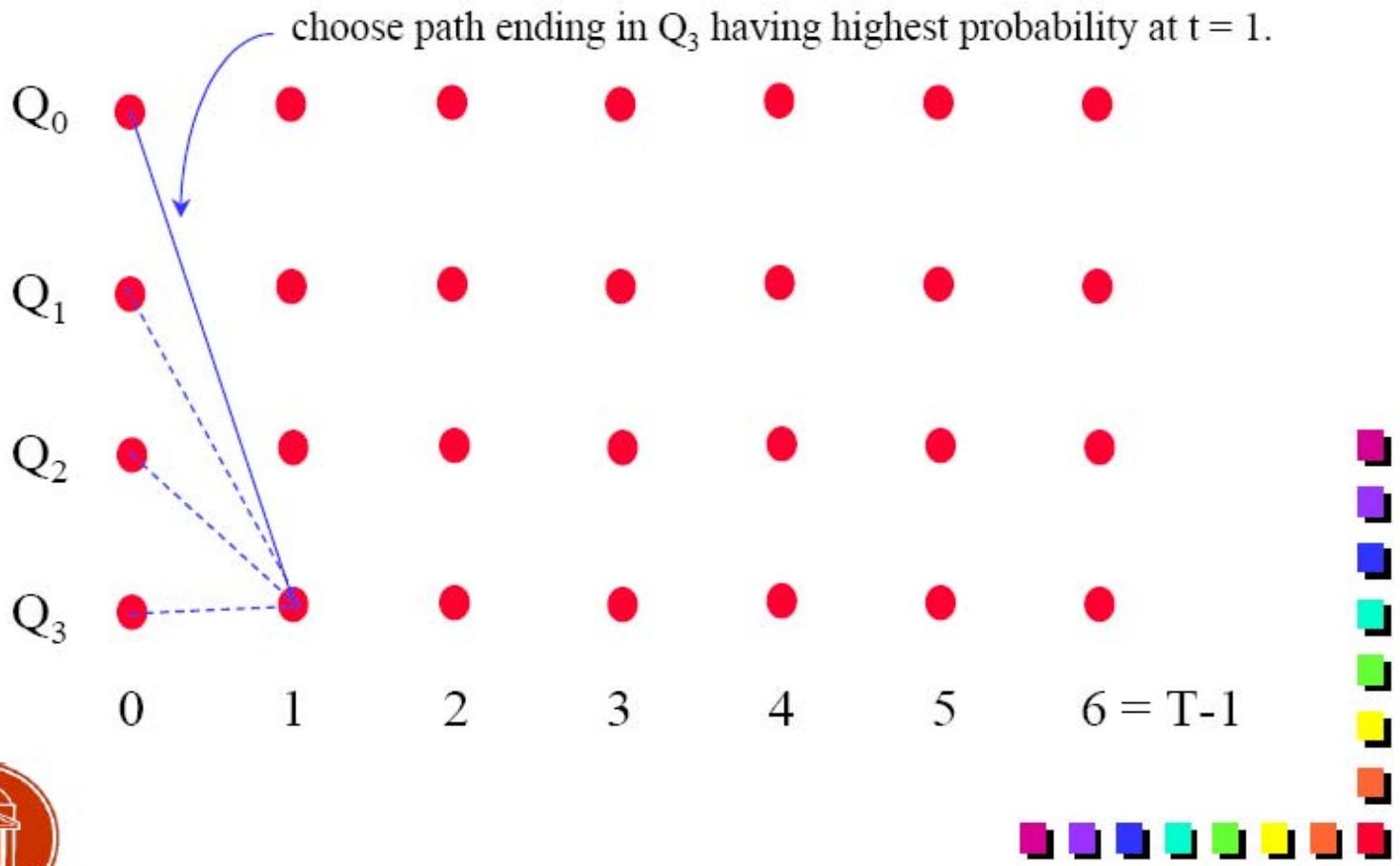
Viterbi Algorithm (cont.)



Viterbi Algorithm (cont.)

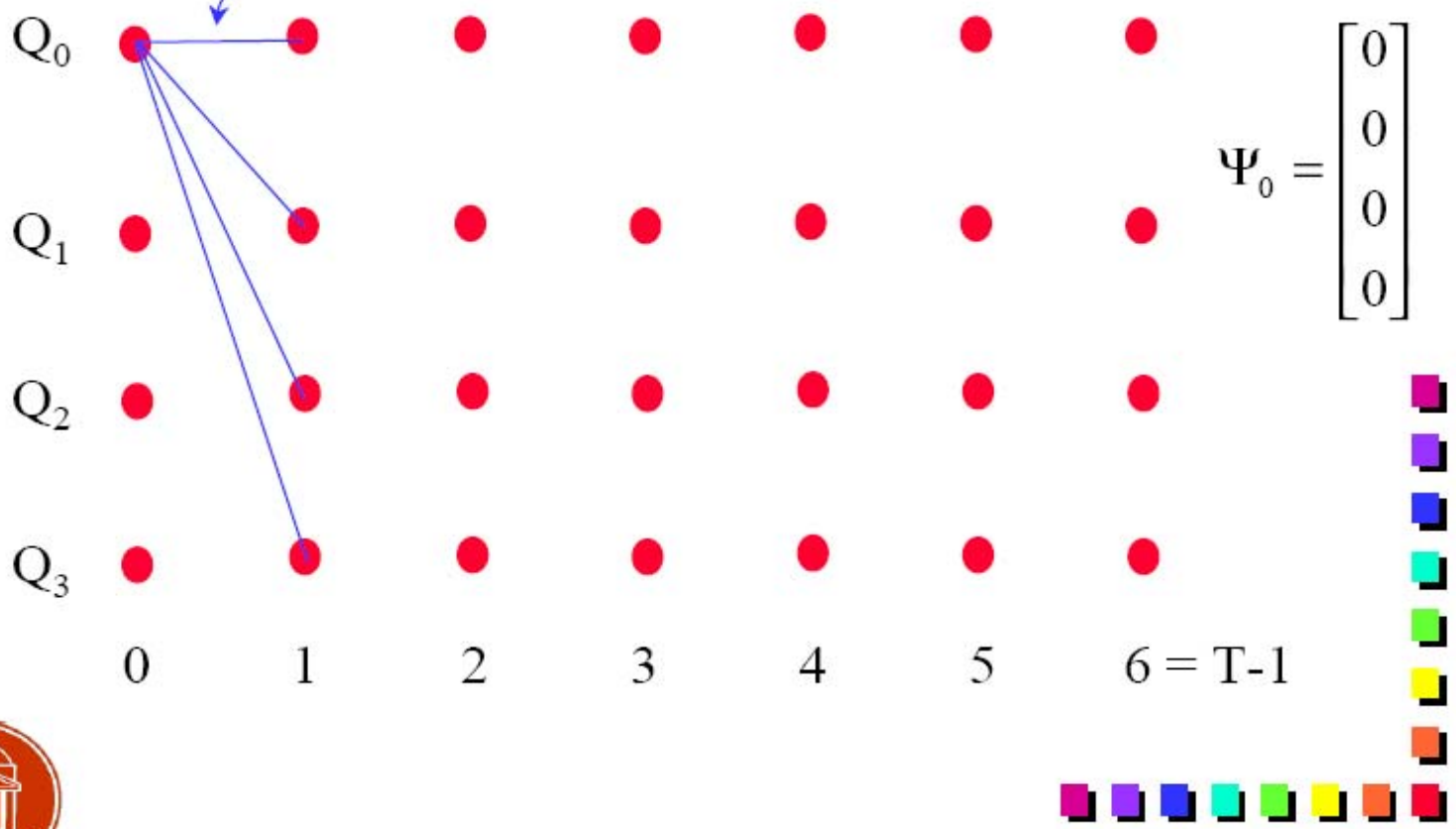


Viterbi Algorithm (cont.)

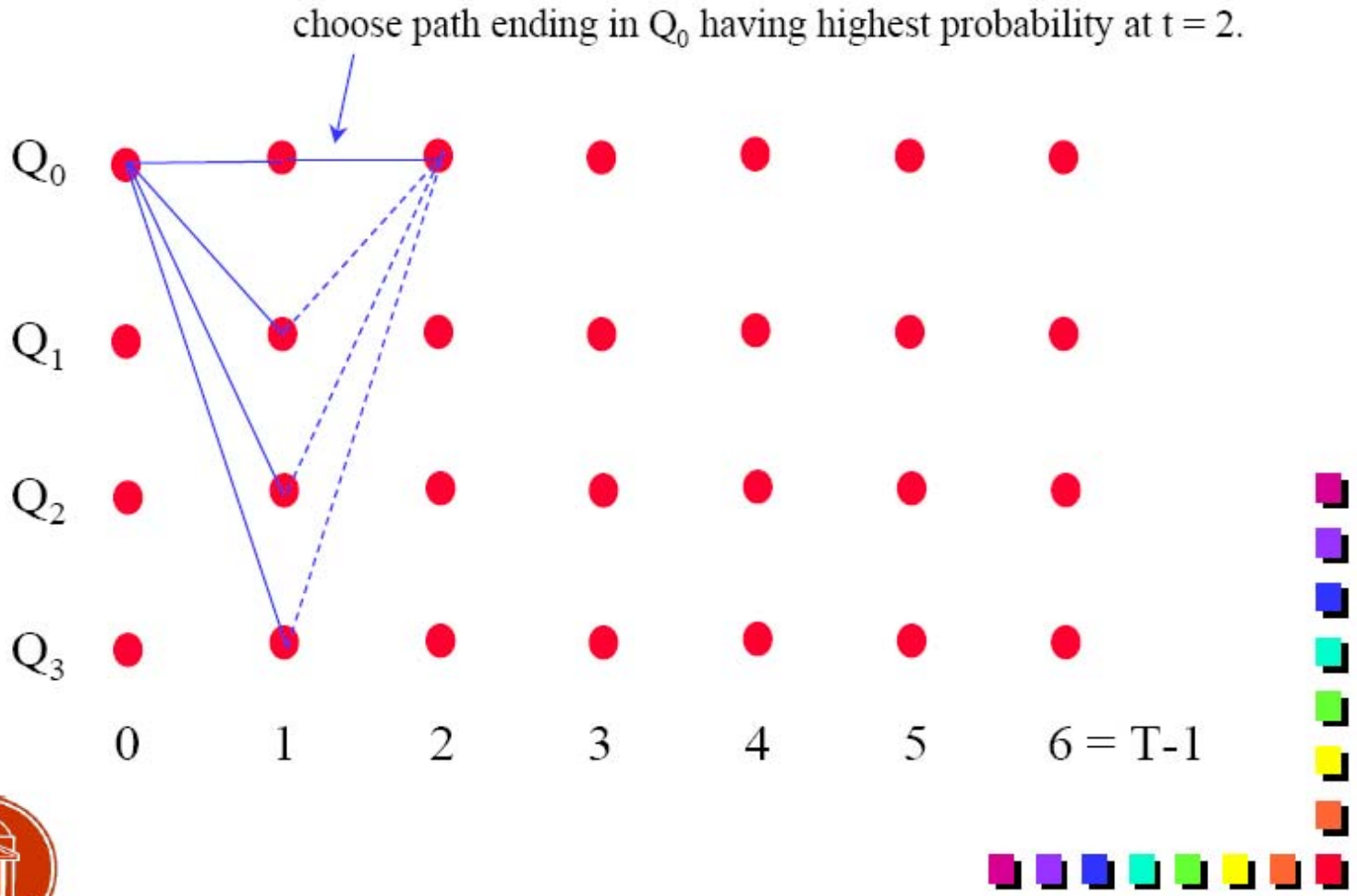


Viterbi Algorithm (cont.)

Save each of the N = 4 maximum probabilities in the vector δ_t
Save the state at t = 0 in each retained path in the vector Ψ_t

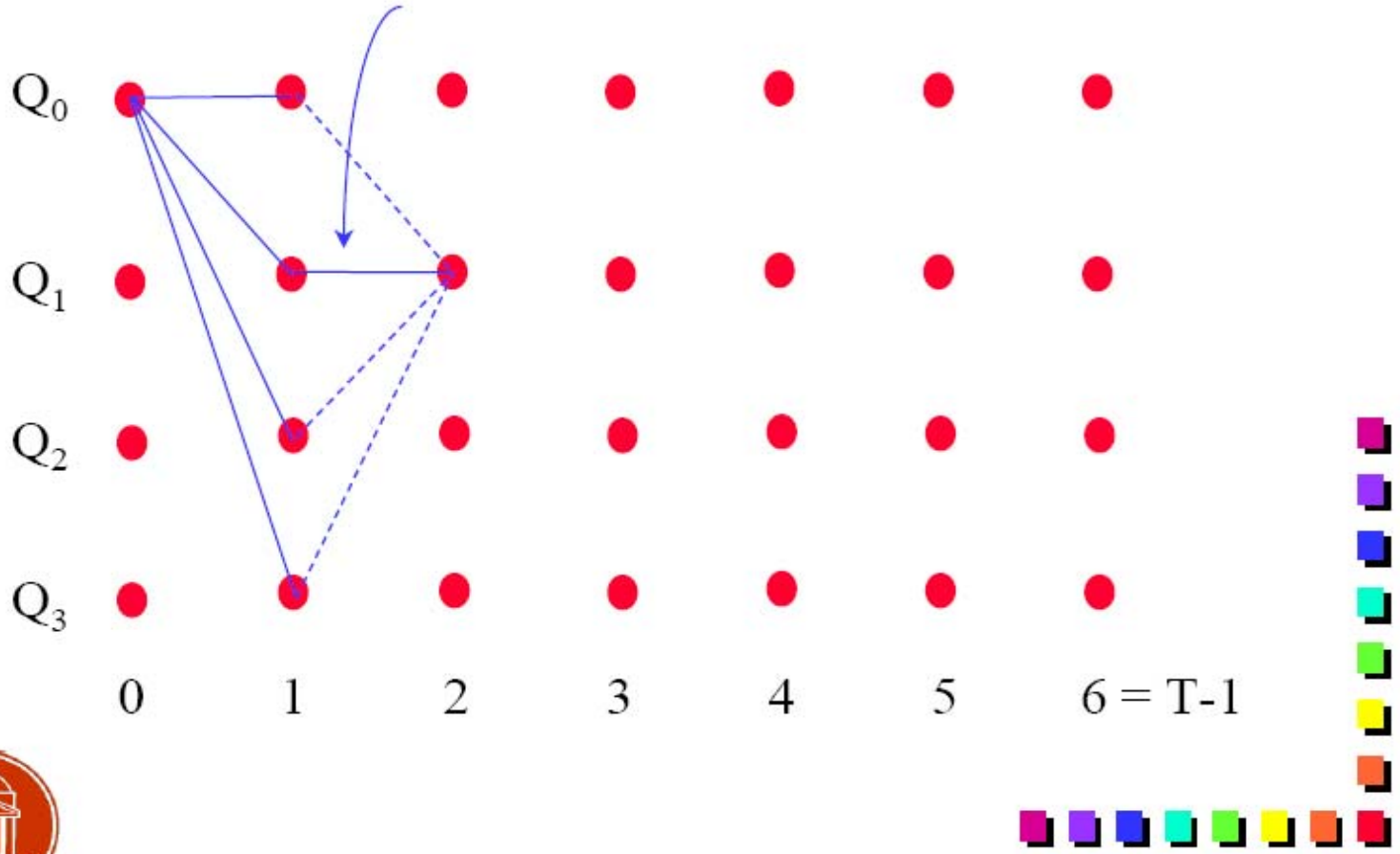


Viterbi Algorithm (cont.)



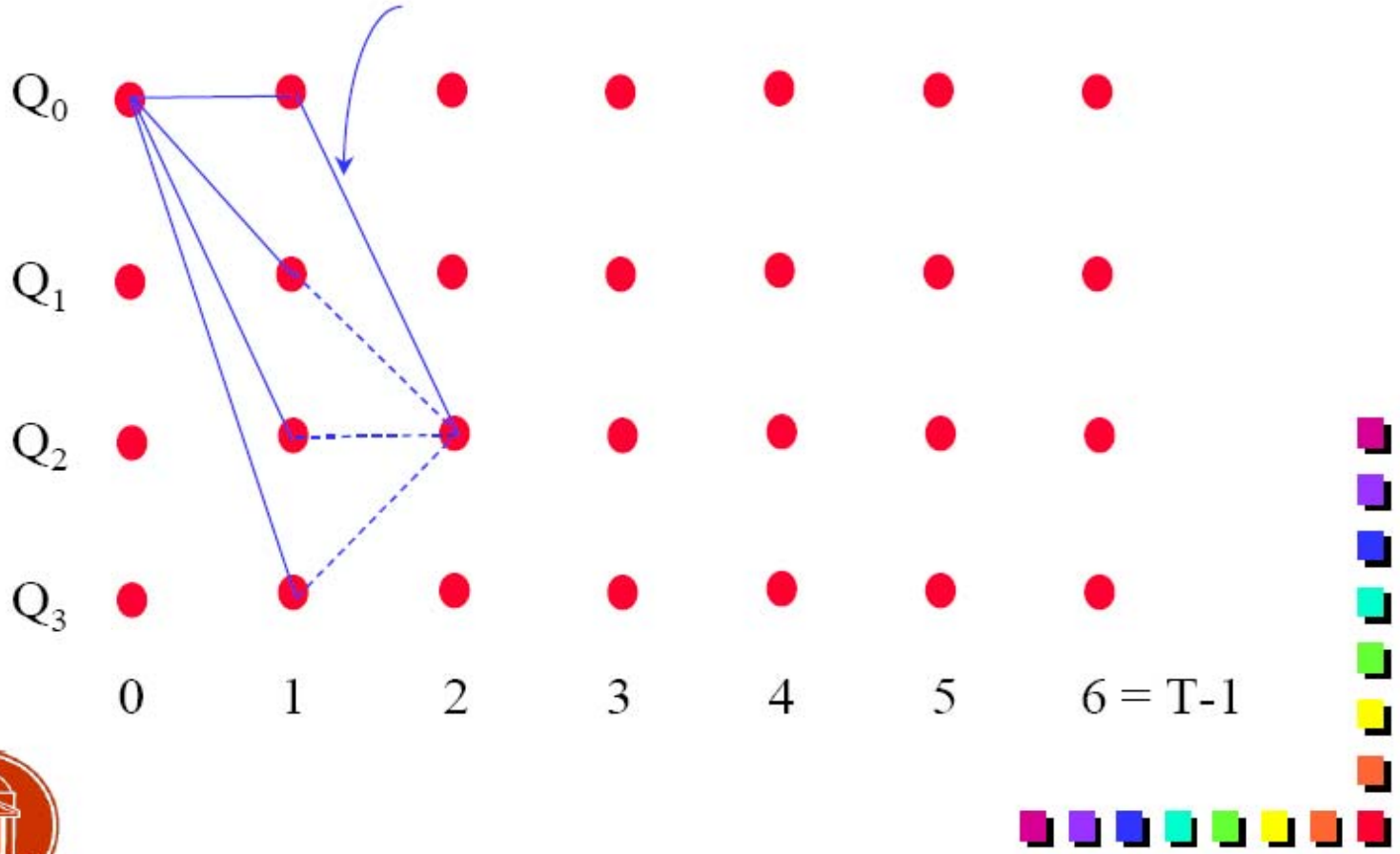
Viterbi Algorithm (cont.)

choose path ending in Q_1 having highest probability at t = 2.



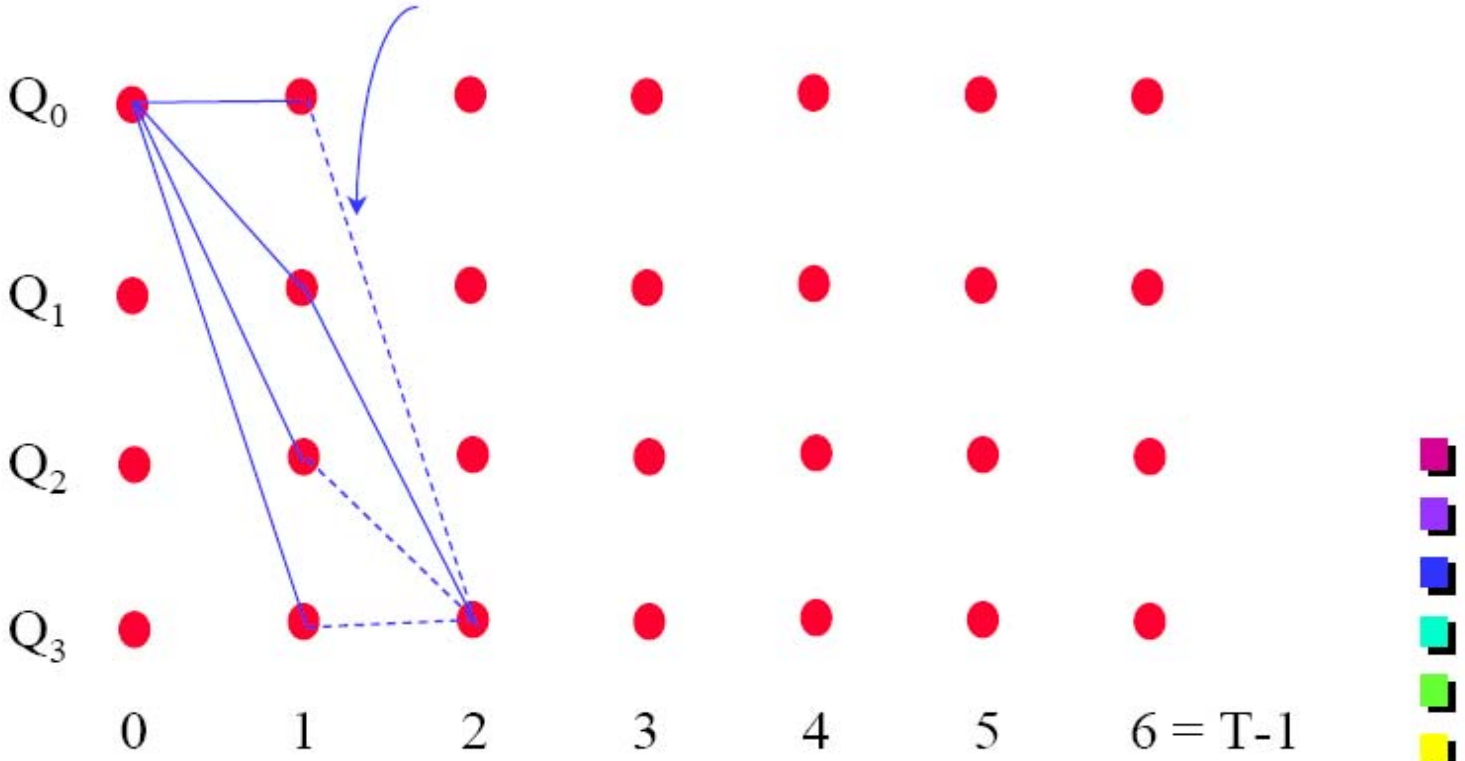
Viterbi Algorithm (cont.)

choose path ending in Q_2 having highest probability at t = 2.



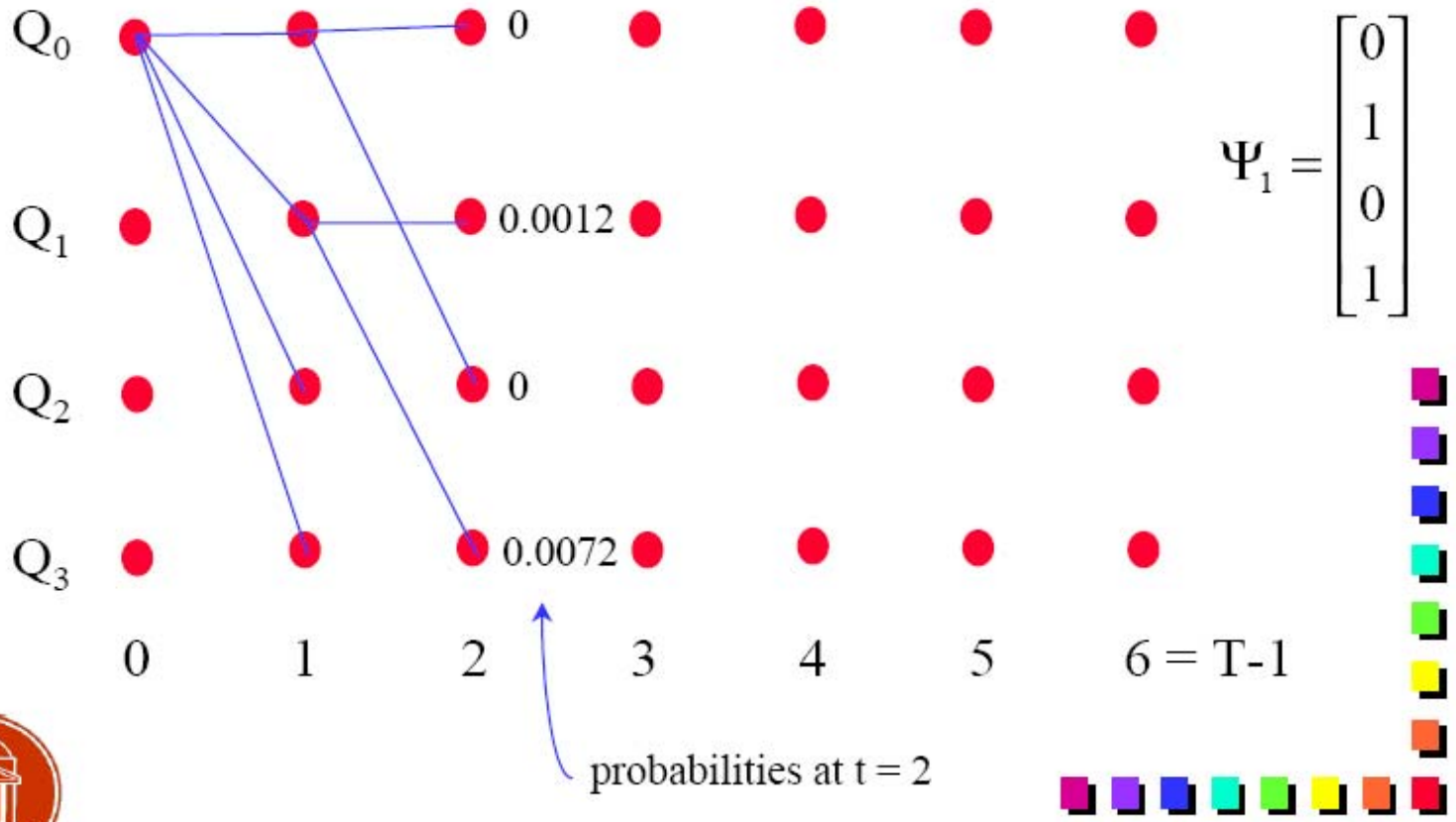
Viterbi Algorithm (cont.)

choose path ending in Q_3 having highest probability at t = 2.



Viterbi Algorithm (cont.)

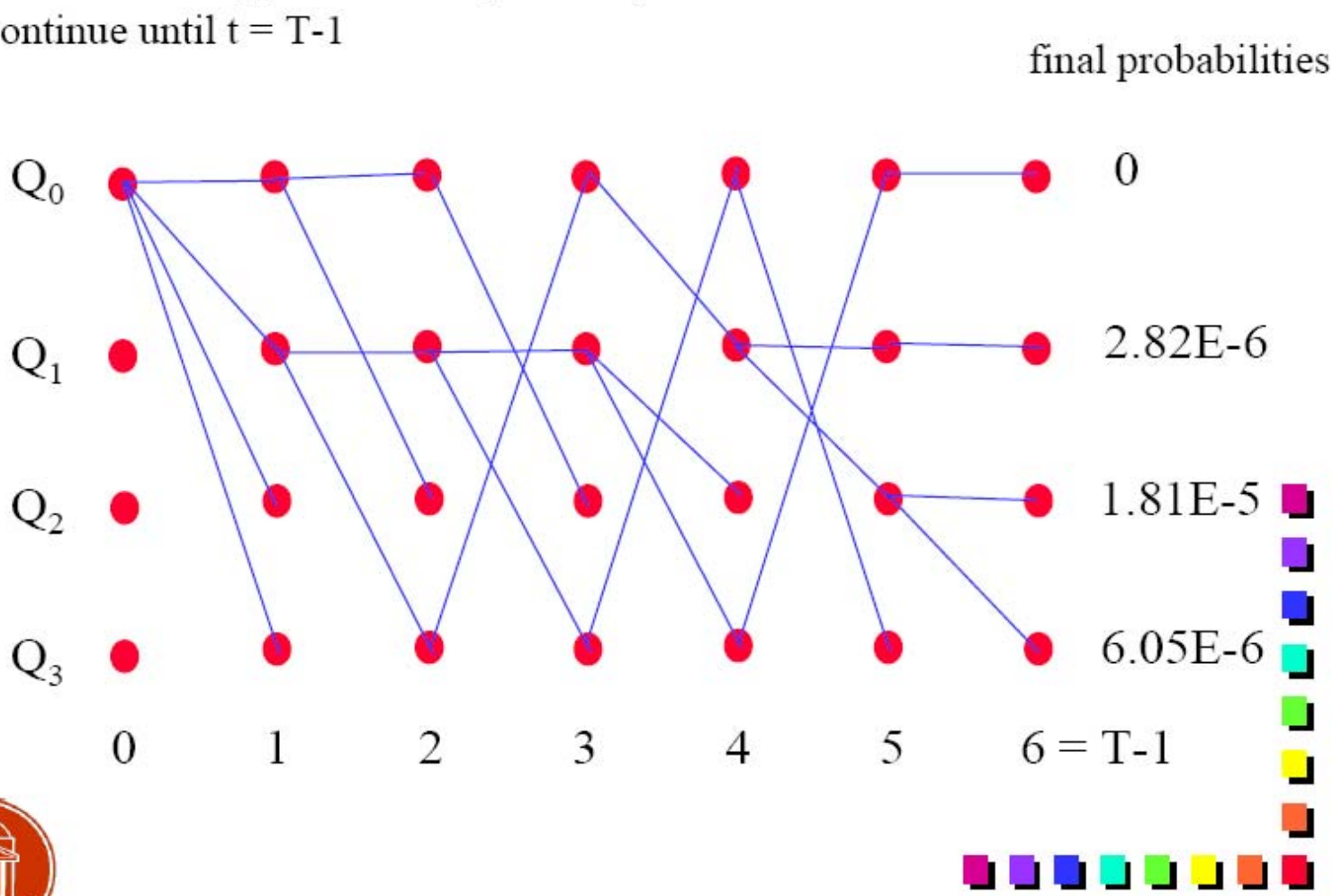
Save each of the N = 4 maximum probabilities in the vector δ_2
Save the state at t = 1 in each retained path in the vector Ψ_1



$$\Psi_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

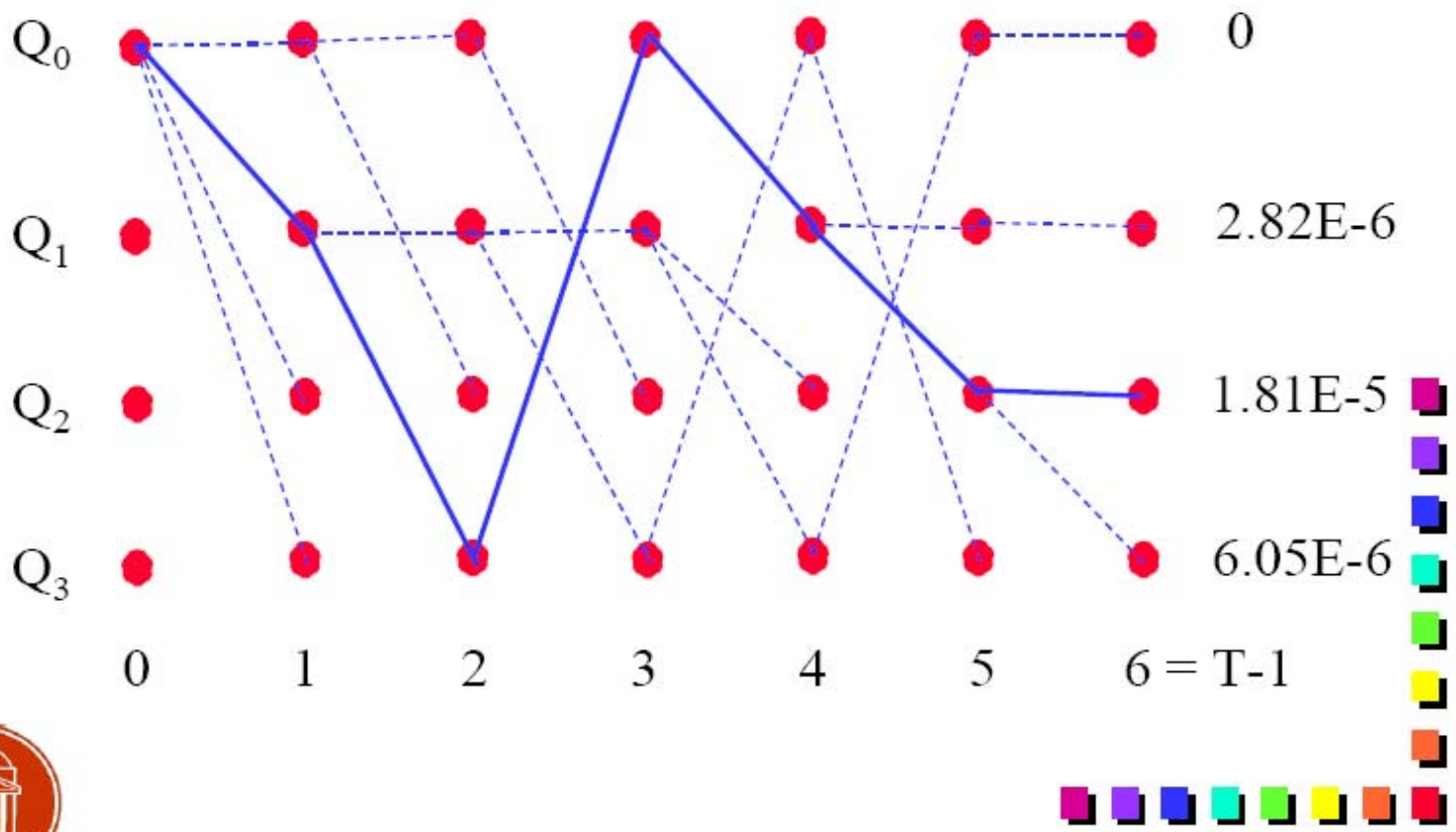
Viterbi Algorithm (cont.)

continue until t = T-1



Viterbi Algorithm (cont.)

- maximum final probability defines best path
- must backtrack through the Ψ_t to find it



HMM – Auswertung – Dekodierung

Aus numerischen Gründen rechnet man beim Viterbi-Dekoder statt mit den Wahrscheinlichkeiten mit den Logarithmen davon

- Warum?
- Wie?
- Können wir bei Forward- und Backward-Algorithmus auch mit Logarithmen rechnen?

HMM – Viterbi-Dekoder

Forward-Rekursion:

$$\alpha_j(t) = \begin{cases} 0 & t = 0,\ j \neq 0 \\ 1 & t = 0,\ j = 0 \\ \left[\sum_i \alpha_i(t-1)\, a_{ij}\right] b_{jk} & \text{sonst};\ v(t) = v_k \end{cases}$$

Logarithmieren und exponenzieren:

$$\alpha'_j(t) = \begin{cases} -\infty & t = 0,\ j \neq 0 \\ 0 & t = 0,\ j = 0 \\ \log\left[\sum_i a_{ij} e^{\alpha'_i(t-1) - \bar{\alpha}'(t-1)}\right] + \log b_{jk} + \bar{\alpha}'(t-1) & \text{sonst};\ v(t) = v_k \end{cases}$$

HMM – Viterbi-Dekoder

Beispiel: Dishonest Casino

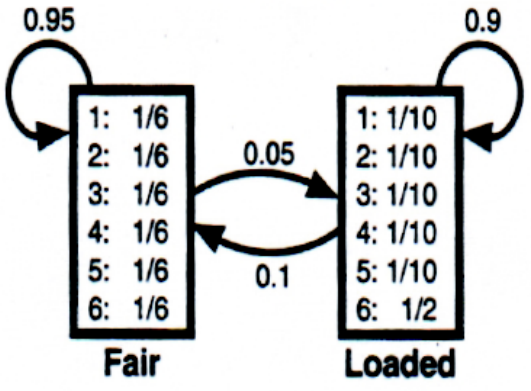
Fairer Würfel: $P(n) = \frac{1}{6}$

Gezinkter Würfel: $P(6) = 0.5; P(n; n \neq 6) = 0.1$

Wahrscheinlichkeiten für Wechsel:

Fair -> gezinkt: 0.05

Gezinkt -> fair: 0.1



Aus: Durbin,Eddy,Krough,Mitchison: Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, 1998., nach Skript zu Shamir: Algorithms for Molecular Biology Tel-Aviv 2001

HMM – Viterbi-Dekoder

Ergebnis mit Viterbi-Dekoder:

300 Würfe , L=Loaded (gezinkt) F=Fair

```
Rolls   315116246446644245311321631164152133625144543631656626566666
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLL

Rolls   651166453132651245636664631636663162326455236266666625151631
Die     LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLFFFFFFFFF
Viterbi LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF

Rolls   222555441666566563564324364131513465146353411126414626253356
Die     FFFFFFFFLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls   366163666466232534413661661163252562462255265252266435353336
Die     LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
*
Rolls   233121625364414432335163243633665562466662632666612355245242
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLFFFFFFFFFF
```

Beispiel Spracherkennung

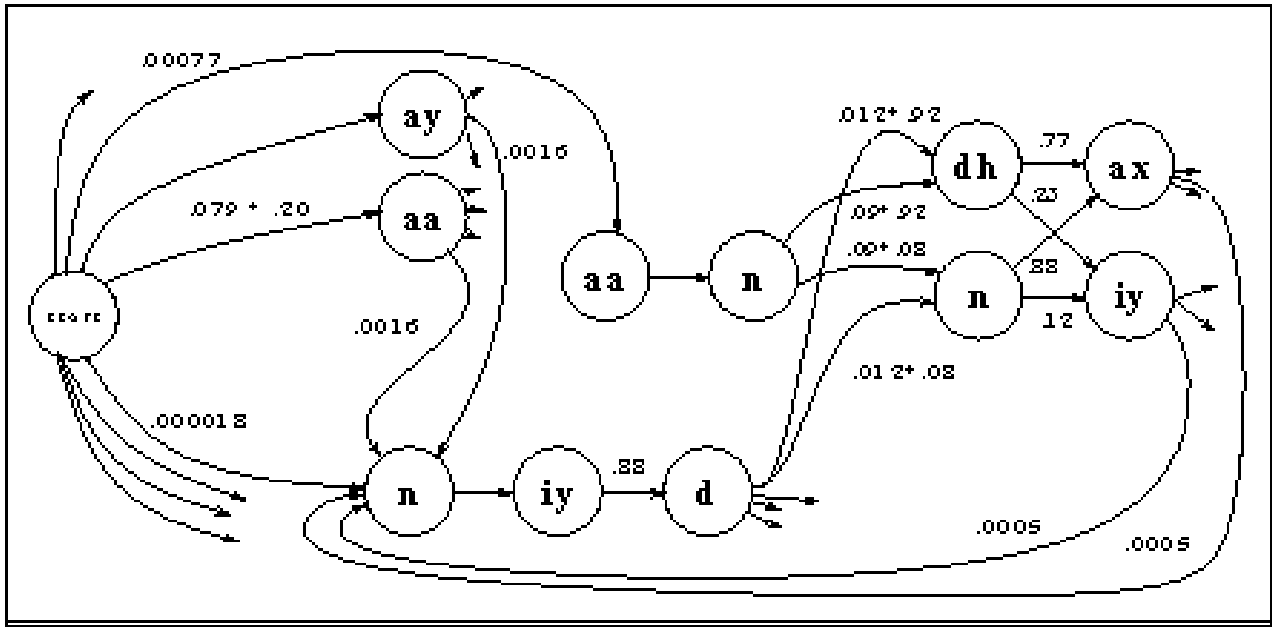


Figure 7.8 Single automaton made from the words *I*, *need*, *on*, and *the*. The arcs between words have probabilities computed from Figure 7.7. For lack of space the figure only shows a few of the between-word arcs.

Aus Michael Metear: *Natural Language Processing, University of Ulster, 2004, http://www.infj.ulst.ac.uk/nlp/powerpoint/speechrecog.ppt*

Beispiel Spracherkennung; Ergebnisse

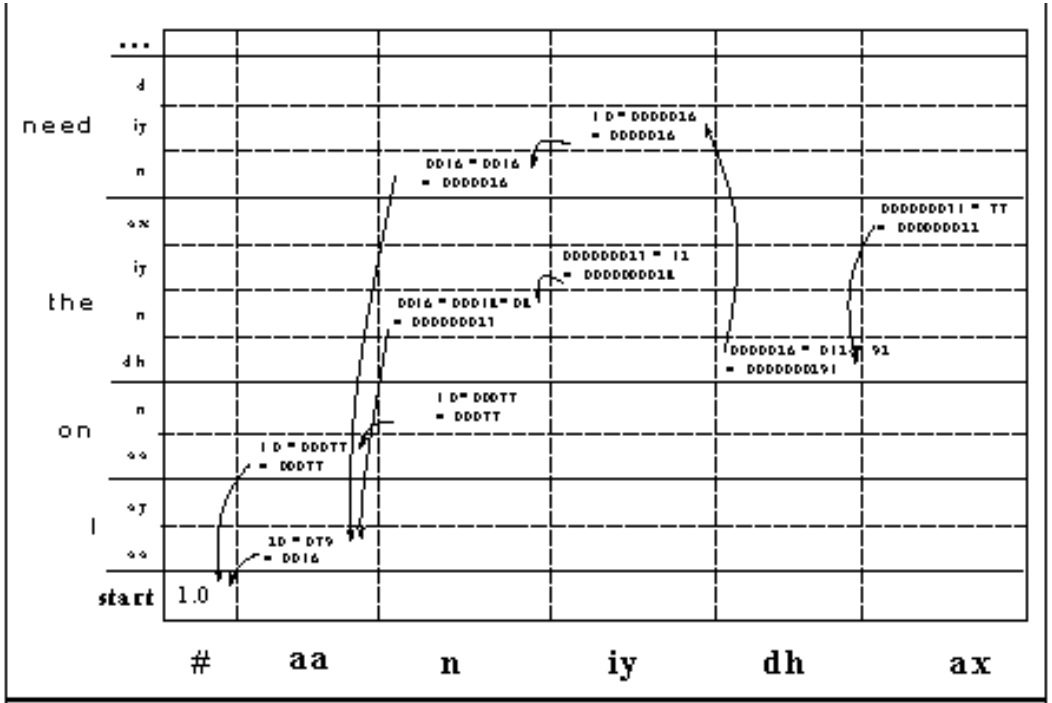


Figure 7.10 The entries in the individual state columns for the Viterbi algorithm. Each cell keeps the probability of the best path so far and a pointer to the previous cell along that path. Backtracing from the successful last word (*the*), we can reconstruct the word sequence *I need the*.

HMM – Posterior Decoding

Durch Kombinieren der Forward und Backward Algorithmen können wir, wenn die ganze Sequenz $\mathbf{V}^T$ bekannt ist, die gesamte Wahrscheinlichkeit berechnen, dass das System zum Zeitpunkt t im Zustand ω_i war:

$$P\left(\omega_i\left(t\right)|\mathbf{V}^T\right)=\frac{\alpha_i\left(t\right)\beta_i\left(t\right)}{P\left(\mathbf{V}^T\right)}$$

wobei: $P\left(\mathbf{V}^T\right)=\alpha\left(T+1\right)=\beta\left(0\right)=\sum_i\alpha_i\left(t\right)\beta_i\left(t\right)$

(was ist wohl $\alpha\left(T+1\right)$und $\beta\left(0\right)$?)

Antwort:

$$\alpha(T+1) = \alpha_{c+1}(T+1) = \left[\sum_i \alpha_i(T)\, a_{i,\text{end}}\right]; a_{i,\text{end}} = a_{i,c+1}$$

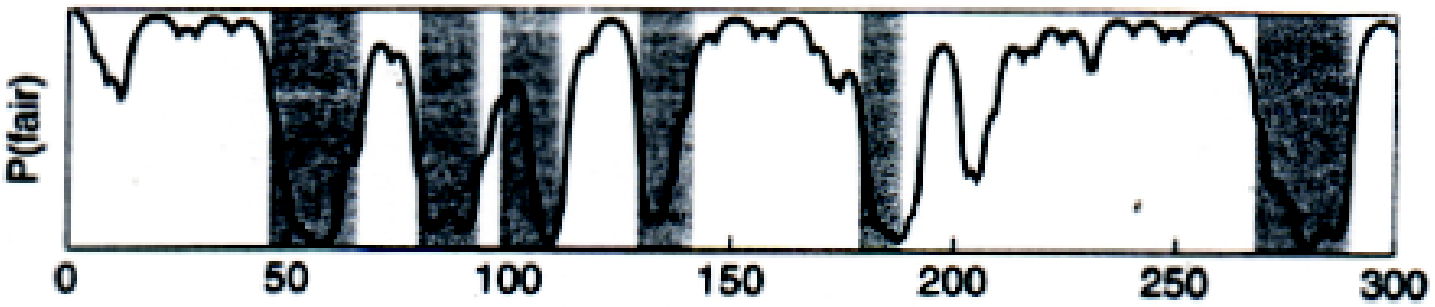
$a_{i,\text{end}}$ ist die Wahrscheinlichkeit für Zustand ω_i der letzte Zustand einer Sequenz zu sein.

$$\beta(0) = \beta_0(0) = \sum_j \beta_j(1)\, a_{0j} b_{jk}\left(v(1) = v_k\right)$$

a_{0j} ist die Wahrscheinlichkeit, dass eine Sequenz mit Zustand ω_j beginnt

HMM – Posterior Decoding

Beispiel: Dishonest Casino



3. Lernproblem:

Wie bestimmt man aus Trainingsdaten die a_{ij}, b_{jk} ?

Forward-Backward- oder Baum-Welch- Algorithmus (Leonard E. Baum und Lloyd R. Welch, 1970)

Hidden Markov Models – Lernen

Problem wäre einfach, wenn zusätzlich zu den $\mathbf{V}^{T(l)}$ auch die Sequenzen der versteckten Zustände $\boldsymbol{\omega}^{T(l)}$ bekannt wären:

- Auszählen für alle $\mathbf{V}^{T(l)}$ und alle t :
  - Anzahl A_{ij} der Übergänge von $\omega_i(t) \rightarrow \omega_j(t+1)$
  - Anzahl B_{jk} der sichtbaren Zustände $v_k(t)$ bei verstecktem Zustand $\omega_j(t)$
- Dann Maximum Likelihood Schätzung:

$$a_{ij} = \frac{A_{ij}}{\sum_q A_{iq}} \qquad b_{jk} = \frac{B_{jk}}{\sum_l B_{jl}}$$

HMM – Baum-Welch

Nun der Forward-Backward- oder Baum-Welch-Algorithmus:

Der Algorithmus ist ein Expectation-Maximization-Algorithmus:

Wir fangen mit einem beliebigen Modell (a_{ij} , b_{jk}) an und berechnen die Erwartungswerte von A_{ij} und B_{jk} unter der Voraussetzung, dass dieses Modell stimmt.
Dann berechnen wir daraus iterativ neue a_{ij} und b_{jk}.

HMM – Baum-Welch

Für die Wahrscheinlichkeit eines Übergangs vom Zustand $\omega_i(t-1)$ in den Zustand $\omega_j(t)$ erhalten wir (Sequenz $\mathbf{V}^T$):

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1)\,a_{ij}b_{jk}\left(v(t)=v_k\right)\beta_j(t)}{P\left(\mathbf{V}^T \mid \boldsymbol{\theta}\right)} = \frac{\gamma'_{ij}(t)}{P\left(\mathbf{V}^T \mid \boldsymbol{\theta}\right)}$$

Wobei $P\left(\mathbf{V}^T \mid \boldsymbol{\theta}\right)$ die Wahrscheinlichkeit ist, dass das Modell die sichtbare Sequenz $\mathbf{V}^T$ auf irgendeinem Pfad erzeugt hat.

HMM – Baum-Welch

Dann bekommen wir, wenn wir eine Menge von Trainingssequenzen $\mathbf{V}^{T(l)}$ betrachten, als Erwartungswert (das hochgestellte, in Klammern stehende l bezeichnet die Nummer der Trainingssequenz):

$$A_{ij} = \sum_{l} \sum_{t=1}^{T(l)+1} \gamma_{ij}^{(l)}(t)$$

$$= \sum_{l} \frac{1}{P\left(\mathbf{V}^{T(l)} \mid \boldsymbol{\theta}\right)} \sum_{t=1}^{T(l)+1} \alpha_{i}^{(l)}(t-1)\, a_{ij} b_{jk} \left(v^{(l)}(t) = v_k\right) \beta_{j}^{(l)}(t)$$

$$= \sum_{l} \frac{1}{P\left(\mathbf{V}^{T(l)} \mid \boldsymbol{\theta}\right)} \sum_{t=1}^{T(l)+1} \gamma'^{(l)}_{ij}(t)$$

HMM – Baum-Welch

Die Gesamtwahrscheinlichkeit für $\mathbf{V}^{T(l)}$ kann man schreiben als

$$P\left(\mathbf{V}^{T(l)} \mid \boldsymbol{\theta}\right) = \sum_i \sum_j \gamma'^{(l)}_{ij}(t) = \frac{1}{T(l)+1} \sum_i \sum_j \sum_{t=1}^{T(l)+1} \gamma'^{(l)}_{ij}(t)$$

Schreiben wir für $\sum_{t=1}^{T(l)+1} \gamma'^{(l)}_{ij}(t) = A_{ij}^{(l)}$, so erhalten wir:

$$A_{ij} = \sum_l \frac{T(l)+1}{\sum_m \sum_n A_{mn}^{(l)}} A_{ij}^{(l)}$$

HMM – Baum-Welch

Die Maximum likelihood Schätzung für das neue a_{ij} ist:

$$\hat{a}_{ij} = \frac{A_{ij}}{\sum_q A_{iq}}$$

Betrachten wir nur *eine* Folge (mehrere können u. U. zu einer zusammengefasst werden), so können wir durch $P\left(\mathbf{V}^{T(l)} \mid \boldsymbol{\theta}\right)$ kürzen:

$$\hat{a}_{ij} = \frac{A_{ij}}{\sum_q A_{iq}} = \frac{A_{ij}^{(1)}}{\sum_q A_{iq}^{(1)}}$$

HMM – Baum-Welch

Schätzung der b_{jk}:
Wir berechnen die den Erwartungswert für die Anzahl der Fälle, in denen sich das System im Zustand ω_j befindet:

$$\sum_t P\left(\omega_j(t) \mid \mathbf{V}^T, \boldsymbol{\theta}\right) = \frac{1}{P\left(\mathbf{V}^T \mid \boldsymbol{\theta}\right)} \sum_t \alpha_j(t)\beta_j(t)$$

$$\left(= \frac{1}{P\left(\mathbf{V}^T \mid \boldsymbol{\theta}\right)} \sum_t \sum_m \alpha_m(t-1) a_{mj} b_{jk}\left(v(t) = v_k\right) \beta_j(t) \right.$$

$$\left. = \frac{1}{P\left(\mathbf{V}^T \mid \boldsymbol{\theta}\right)} \sum_t \sum_m \gamma'_{mj}(t) \right)$$

HMM – Baum-Welch

Dann schränken wir ein auf visible state v_k :

$$\sum_t P\left(\omega_j(t), v_k(t) \mid \mathbf{V}^T, \boldsymbol{\theta}\right) = \sum_{t; v(t)=v_k} \alpha_j(t)\beta_j(t)$$

und erhalten:

$$B_{jk} = \sum_l \frac{1}{P\left(\mathbf{V}^{T(l)} \mid \boldsymbol{\theta}\right)} \sum_{t; v(t)=v_k} \alpha_j^{(l)}(t)\beta_j^{(l)}(t)$$

$$= \sum_l \frac{1}{P\left(\mathbf{V}^{T(l)} \mid \boldsymbol{\theta}\right)} \sum_{t; v(t)=v_k} \sum_m \gamma'_{mj}(t)$$

Wieder schreiben wir abkürzend:

$$B_{jk}^{(l)} = \sum_{t; v(t)=v_k} \alpha_j^{(l)}(t)\beta_j^{(l)}(t) = \sum_{t; v(t)=v_k} \sum_m \gamma'_{mj}(t)$$

HMM – Baum-Welch

Die neuen b_{jk} berechnen wir wieder wie im Fall bekannter innerer Zustände (s. o.):

$$\hat{b}_{jk} = \frac{B_{jk}}{\sum_q B_{jq}}$$

Auch hier können wir wieder den Fall betrachten, dass nur *eine* Folge $\mathbf{V}^{T(1)}$ (mehrere können u. U. zu einer zusammengefasst werden) gelernt werden soll, und können dann durch $P\left(\mathbf{V}^{T(1)} \mid \boldsymbol{\theta}\right)$ kürzen:

$$\hat{b}_{jk} = \frac{B_{jk}}{\sum_q B_{jq}} = \frac{B_{jk}^{(1)}}{\sum_q B_{jq}^{(1)}}$$

HMM – Baum-Welch

Anschließend wird so lange iteriert, bis

- die Parameter a_{ij} und b_{jk} sich nicht mehr stark ändern, oder
- Die Wahrscheinlichkeit, dass die Trainingsseqenzen vom Modell erzeugt wurden, einen Mindestwert überschreitet.