HMM(Hidden Markov Model)

Hyopil Shin

1. Finite State Machines

(1) Mealy Machines

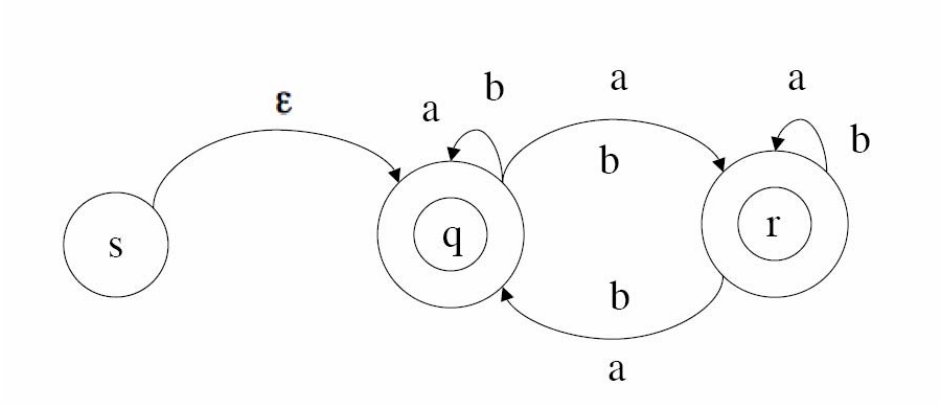


Figure 1. Mealy Machines

(2) Moor Machines

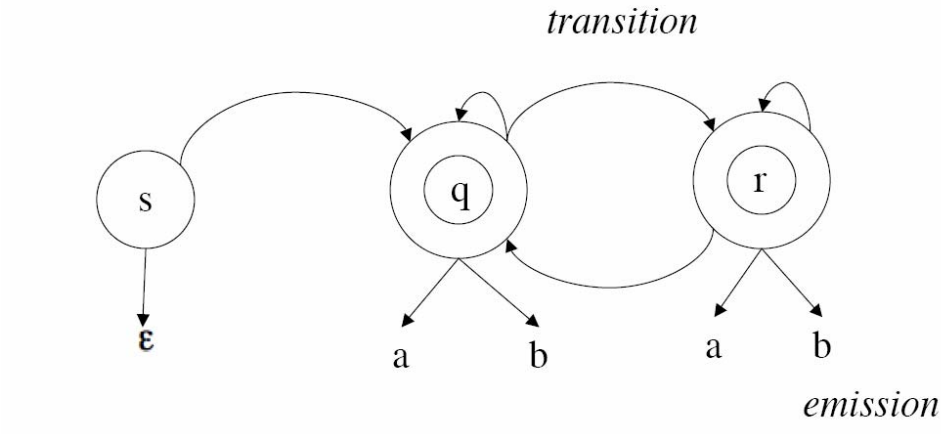


Figure 2. Moor Machines

2. Probabilistic FSMs

- Each transition is associated with a *transition probability*
- Each emission is associated with an *emission probability*
- Two conditions:
  - All outgoing transition arcs from a state must sum to 1
  - All emission arcs from a state must sum to 1

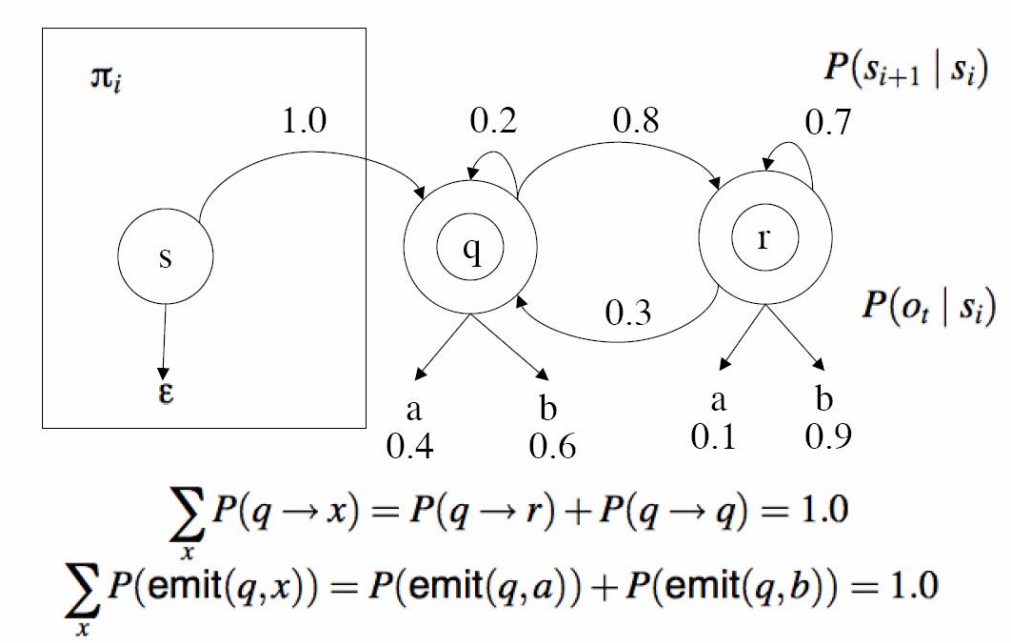


Figure 3. Probabilistic FSMs

3. Markov Model

- Markov processes/chains/models were first developed by Andrei A. Markov (a student of Chebyshev).
- Their first use was actually for a linguistic purpose – modeling the letter sequences in works of Russian literature (Markov 1913)

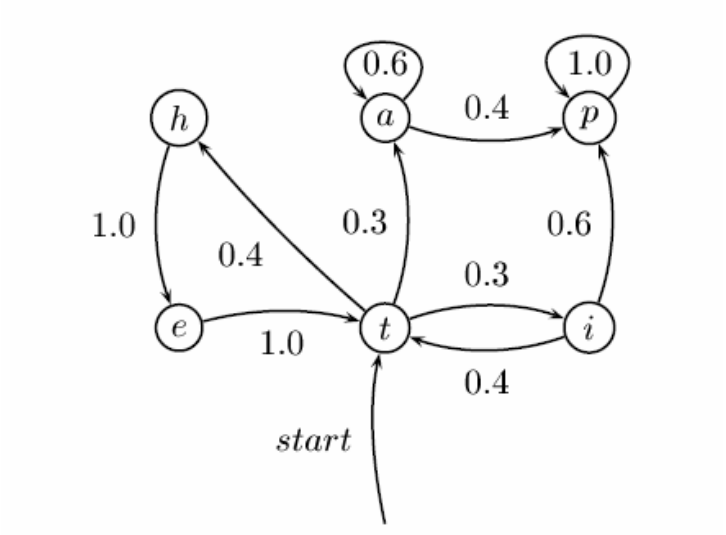


Figure 4. A Markov Model

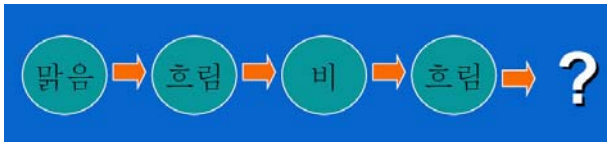
- Markov Assumption

$$P(X_t = s_i \mid \ldots, X_{t-1} = s_j)$$

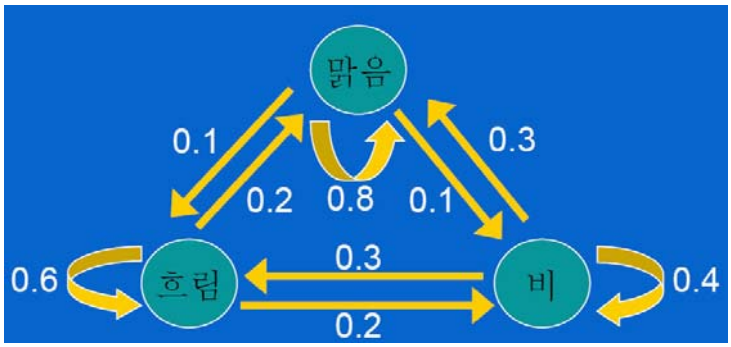
- Stationary(time invariant) Distribution

$$P(X_t = s_i \mid X_{t-1} = s_j) = P(X_{t+l} = s_i \mid X_{t+l-1} = s_j)$$

- Markov Model 예



내일의 날씨는?



오늘 날씨 \ 내일날씨	맑음	흐림	비
맑음	0.8	0.1	0.1
흐림	0.2	0.6	0.2
비	0.3	0.3	0.4

• Length of the Observation sequence : T

• Observable states :
1, 2, …, N

• Observed sequence :
$O_1, O_2, \ldots, O_t, \ldots O_T$

• Markov property
$P(O_{t+1} = i \mid O_1, O_2, \ldots, O_{t-1}, O_t) = P(O_{t+1} = i \mid O_t)$

$P(O_1, O_2, \ldots, O_T)$
$= P(O_1)P(O_2|O_1)P(O_3|O_1, O_2)\ldots P(O_T|O_1, \ldots, O_{T-1})$
$= P(O_1)P(O_2|O_1)P(O_3|O_2)\ldots P(O_T|O_{T-1})$

P(맑음,흐림,비)
= P(맑음)P(흐림|맑음)P(비|흐림)
= 1.0 x 0.1 x 0.2 = 0.02

4. Hidden Markov Model

• There are n states $s_1, \cdots, s_i, \cdots, s_n$
• The emissions are observed (input data)
• Observation sequence O=$(o_1, \cdots, o_t, \cdots, o_T)$
• The states are not directly observed (hidden)
• Data does not directly tell us which state X_t is linked with observation o_t

$$X_t \in \{s_1, \ldots, s_n\}$$

4.1. Markov Chains vs. HMMs

Given an observation sequence

O=(o_1, …, o_t, …, o_T)

• An *n*th order Markov Chain computes the probability $P(=(o_1, \cdots, o_t, \cdots, o_T)$

• An HMM computes the probability $P(X_1, \cdots, X_{T+1},, o_1, \cdots, o_t, \cdots, o_T)$ where the state sequence is *hidden*

- HIDDEN MARKOV MODELS (HMMs) have been the mainstay of the statistical modeling used in modern speech recognition systems. Despite their limitations, variants of HMMs are still the most widely used technique in that domain, and are generally regarded as the most successful.
- An *HMM* is nothing more than a probabilistic function of a Markov process.
- In an HMM, you don't know the state sequence that the model passes through, but only some probabilistic function of it.
- Crazy soft Drink Machine: it can be in two states, cola preferring (CP) and iced tea preferring (IP), but it switches between them randomly after each purchase.

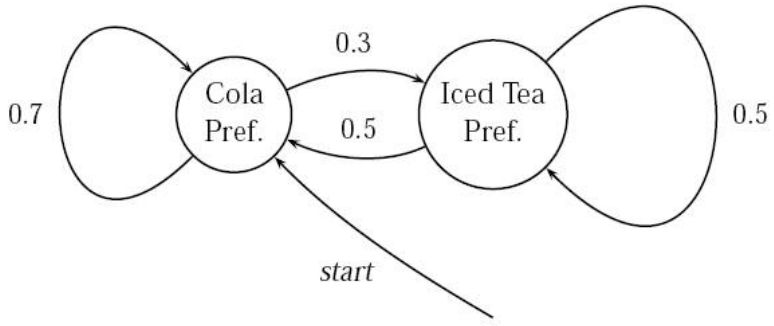


Figure 5. Crazy Soft Drink Machine

Output probability given from the state

	cola	iced tea (ice_t)	lemonade(lem)
CP	0.6	0.1	0.3
IP	0.1	0.7	0.2

What is the probability of seeing the output sequence {lem, ice_t} if the machine always starts off in the cola preferring state?

Solution: We need to consider all paths that might be taken through the HMM, and then to sum over them. We know the machine starts in state

CP. There are then four possibilities depending on which of the two states the machine is in at the other two time instants. So the total probability is:

0.7* 0.3* 0.7* 0.1+ 0.7* 0.3 *0.3* 0.1 +0.3* 0.3* 0.5* 0.7+ 0.3* 0.3* 0.5* 0.7 = 0.084

4.2. Why use HMMs?

- HMMs are useful when one can think of underlying events probabilistically generating surface events. One widespread use of this is tagging –assigning parts of speech (or other classifiers) to the words in a text.
- A simple illustration of how we can use HMMs is in generating parameters for linear interpolation of nn-gram models.
- one way to estimate the probability of a sentence:

  P(Sue drank her beer before the meal arrived)

  was with an n-gram model, such as a trigram model, but that just using an n-gram model with fixed n tended to suffer because of data sparseness. one idea of how to smooth -gram estimates was to use linear interpolation of -gram estimates for various , for example:

$$P_{\text{li}}(w_n|w_{n-1},w_{n-2}) = \lambda_1 P_1(w_n) + \lambda_2 P_2(w_n|w_{n-1}) + \lambda_3 P_3(w_n|w_{n-1},w_{n-2})$$

- The question, then, is how to set the parameters λ_i. While we could make reasonable guesses as to what parameter values to use (and we know that together they must obey the stochastic constraint $\sum_i \lambda_i = 1$), it seems that we should be able to find the optimal values automatically.
- we build an HMM with four states for each word pair, one for the basic word pair, and three representing each choice of n-gram model for calculating the next transition.
- Note how this HMM assigns the same probabilities as the earlier equation: there are three ways for w^c to follow $w^a w^b$ and the total probability of seeing w^c next is then the sum of each of the n-gram probabilities that adorn the arcs multiplied by the corresponding parameter .

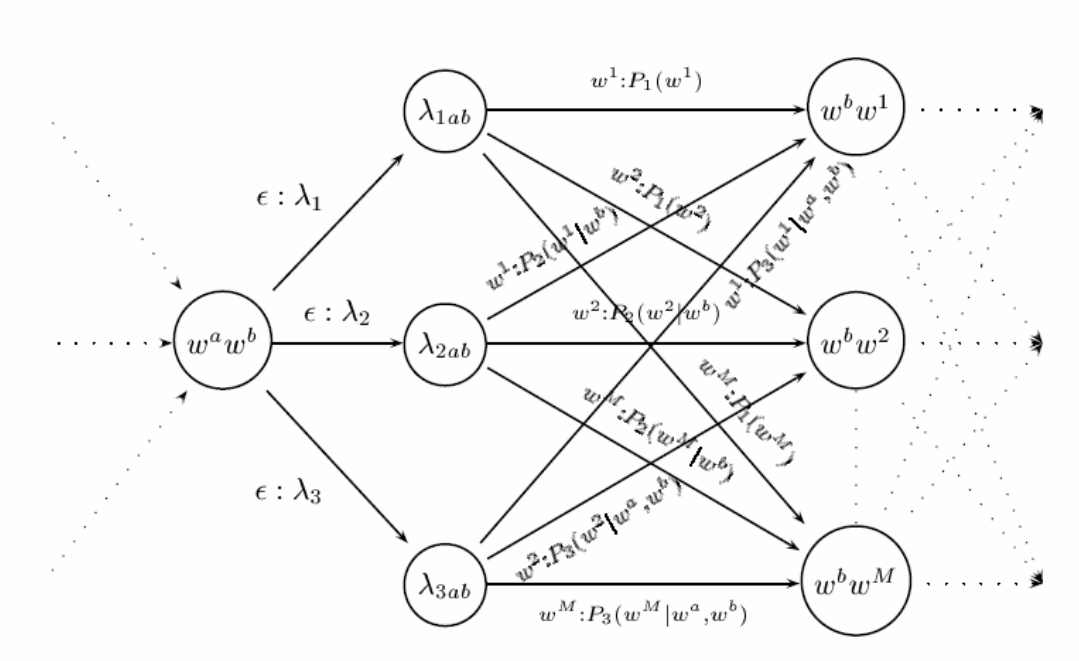


Figure 6. HMM for a linearly interpolated language model

4.3. Notation for HMM

Set of states	$S = \{s_1, \ldots s_N\}$
Output alphabet	$K = \{k_1, \ldots, k_M\} = \{1, \ldots, M\}$
Intial state probabilities	$\Pi = \{\pi_i\}, i \in S$
State transition probabilities	$A = \{a_{ij}\}, i, j \in S$
Symbol emission probabilities	$B = \{b_{ijk}\}, i, j \in S, k \in K$
State sequence	$X = (X_1, \ldots, X_{T+1}) \quad X_t : S \mapsto \{1, \ldots, N\}$
Output sequence	$O = (o_1, \ldots, o_T) \quad o_t \in K$

4.4. Three Problems for HMM

1. Probability evaluation

Given a model $\mu = (A, B, \Pi)$, how do we efficiently compute how likely a certain observation is, that is $P(O|\mu)$?

2. Optimal state sequence(decoding)

Given the observation sequence O and a model _, how do we choose a state sequence (X_1,…, X_{T+1}) that best explains the observations?

3. Parameter estimation(training)

Given an observation sequence O, and a space of possible models found by varying the model parameters $\mu = (A, B, \Pi)$, how do we find the model that best explains the observed data?

1' Finding the probability of an observation

For any state sequence $X = (X_1, \ldots, X_{T+1})$,

$$\begin{aligned} P(O|X,\mu) &= \prod_{t=1}^{T} P(o_t|X_t, X_{t+1}, \mu) \\ &= b_{X_1 X_2 o_1} b_{X_2 X_3 o_2} \cdots b_{X_T X_{T+1} o_T} \end{aligned}$$

and,

$$P(X|\mu) = \pi_{X_1} a_{X_1 X_2} a_{X_2 X_3} \cdots a_{X_T X_{T+1}}$$

Now,

$$P(O, X|\mu) = P(O|X,\mu)P(X|\mu)$$

Therefore,

$$\begin{aligned} P(O|\mu) &= \sum_{X} P(O|X,\mu)P(X|\mu) \\ &= \sum_{X_1 \cdots X_{T+1}} \pi_{X_1} \prod_{t=1}^{T} a_{X_t X_{t+1}} b_{X_t X_{t+1} o_t} \end{aligned}$$

➔ very expensive

The forward procedure

$$\alpha_i(t) = P(o_1 o_2 \cdots o_{t-1}, X_t = i|\mu)$$

Trellis algorithms: Closeup of the computation of forward probabilities at one node. The forward probability $\alpha_j(t+1)$ is calculated by summing the product of the probabilities on each incoming arc with the forward probability of the originating node.

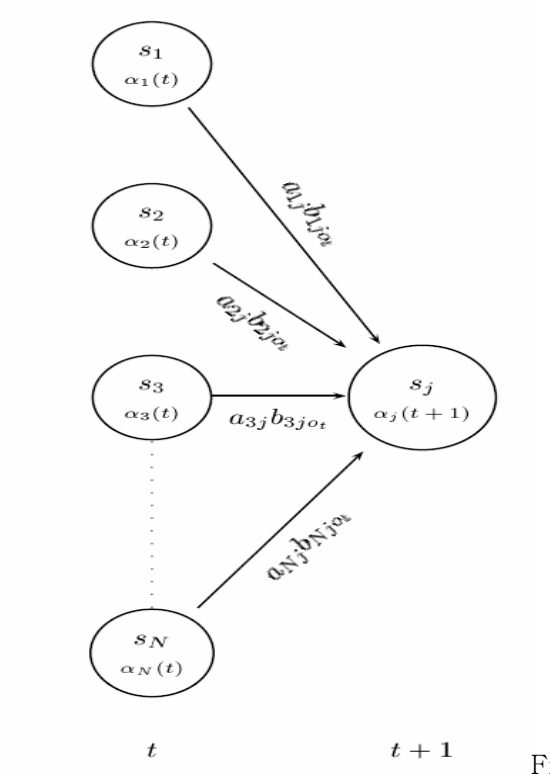


Figure 7. Trellis algorithms

1. Initialization

$$\alpha_i(1) = \pi_i, \quad 1 \le i \le N$$

2. Induction

$$\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{ijo_t}, \quad 1 \le t \le T, 1 \le j \le N$$

3. Total

$$P(O|\mu) = \sum_{i=1}^{N} \alpha_i(T+1)$$

The backward procedure

$$\beta_i(t) = P(o_t \cdots o_T | X_t = i, \mu)$$

1. Initialization

$$\beta_i(T+1) = 1, \quad 1 \le i \le N$$

2. Induction

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_{ij o_t} \beta_j(t+1), \quad 1 \le t \le T, 1 \le i \le N$$

3. Total

$$P(O|\mu) = \sum_{i=1}^{N} \pi_i \beta_i(1)$$

Combining them

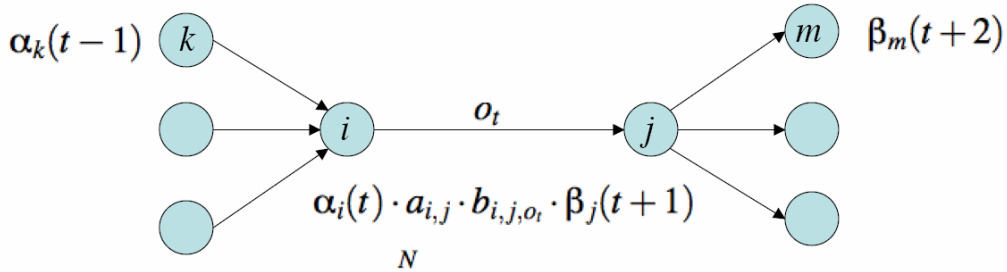
$$\begin{aligned} P(O, X_t = i|\mu) &= P(o_1 \cdots o_T, X_t = i|\mu) \\ &= P(o_1 \cdots o_{t-1}, X_t = i, o_t \cdots o_T|\mu) \end{aligned}$$

$$\begin{aligned} &= P(o_1 \cdots o_{t-1}, X_t = i|\mu) P(o_t \cdots o_T|o_1 \cdots o_{t-1}, X_t = i, \mu) \\ &= P(o_1 \cdots o_{t-1}, X_t = i|\mu) P(o_t \cdots o_T|X_t = i, \mu) \\ &= \alpha_i(t)\beta_i(t) \end{aligned}$$

Therefore:

$$P(O|\mu) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t), \quad 1 \le t \le T+1$$

Forward-backward Algorithm



$$\alpha_i(t) = \sum_{k=1}^{N} a_{k,i} \cdot b_{k,i,o_{t-1}} \cdot \alpha_k(t-1)$$

$$\beta_j(t+1) = \sum_{m=1}^{N} a_{j,m} \cdot b_{j,m,o_{t+1}} \cdot \beta_m(t+2)$$

$$P(o_1, \ldots, o_T) = \sum_{i=1}^{N} \alpha_i(T+1) = \sum_{i=1}^{N} \pi_i \cdot \beta_i(1)$$

2'. Finding the best sequence state (Decoding)

Viterbi algorithm

$$\arg\max_X P(X|O, \mu)$$

1. Initialization

   $\delta_j(1) = \pi_j, \quad 1 \le j \le N$

2. Induction

   $\delta_j(t+1) = \max_{1 \le i \le N} \delta_i(t) a_{ij} b_{ijo_t}, \quad 1 \le j \le N$

   Store backtrace

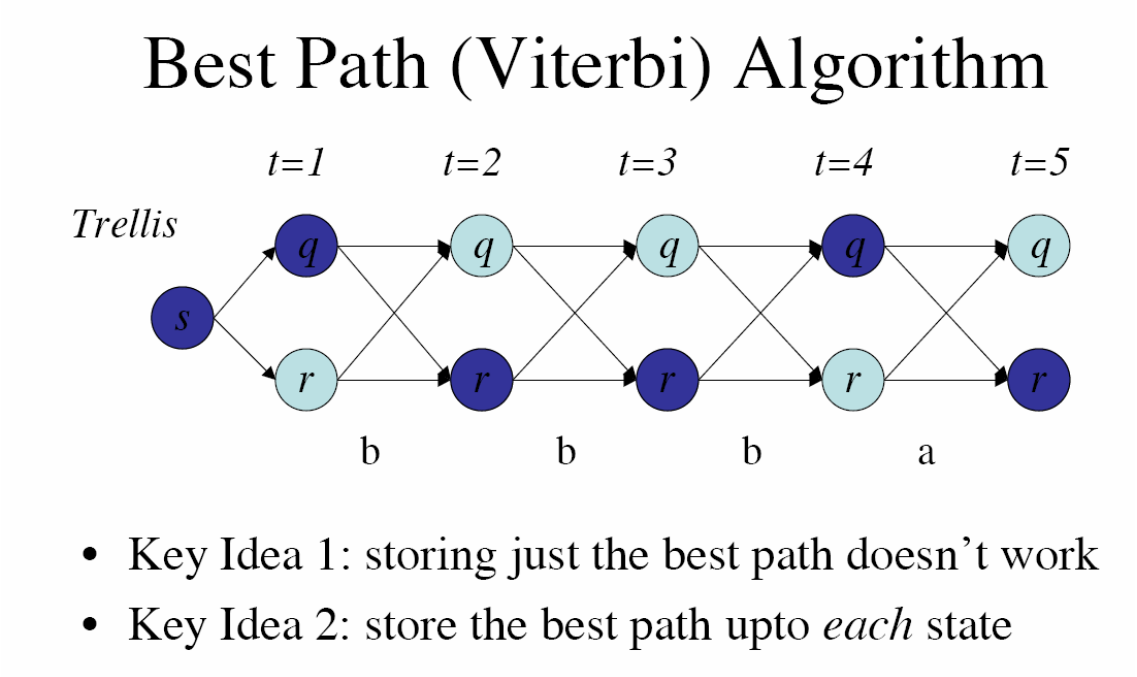
   $\psi_j(t+1) = \arg\max_{1 \le i \le N} \delta_i(t) a_{ij} b_{ijo_t}, 1 \le j \le N$

3. Termination and path readout (by backtracking). The most likely state sequence is worked out from the right backwards:

$$\hat{X}_{T+1} = \arg\max_{1 \le i \le N} \delta_i(T+1)$$

$$\hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1)$$

$$P(\hat{X}) = \max_{1 \le i \le N} \delta_i(T+1)$$

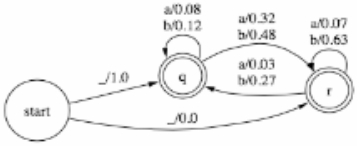


4.5. Example

a. HMM

	p(...|q)	p(...|r)	p(...|start)
p(a|...)	0.4	0.1	0
p(b|...)	0.6	0.9	0
p(e|...)	0	0	1
p(q|...)	0.2	0.3	1
p(r|...)	0.8	0.7	0

	p(...|q)	p(...|r)
p(a,q|...)	0.08	0.03
p(b,q|...)	0.12	0.27
p(a,r|...)	0.32	0.07
p(b,r|...)	0.48	0.63
totals	1	1



input = bbba	e	b	bb	bbb	bbba
α(q)	1	0.12	0.144	0.11448	0.018036
α(r)	0	0.48	0.36	0.29592	0.057348
P(bbba)	1	0.6	0.504	0.4104	0.075384

input = bbba	bbba	bba	ba	a	e
β(q)	0.075384	0.0936	0.096	0.4	1
β(r)	0.1094715	0.13365	0.171	0.1	1

β(q)	0.075384
P(bbba)	0.075384

input = bbba	e	b	bb	bbb	bbba	totals	new values
p(a,q|q)	1	0	0	0	0.0191404	1.0191404	0.40119112
p(b,q|q)	1	0.01787966	0.02200573	0.07289398	0	1.11277937	0.4380527
p(a,r|q)	0	0	0	0	0.0765616	0.0765616	0.03013896
p(b,r|q)	0	0.10212034	0.15679083	0.07289398	0	0.33180516	0.13061722
p(a,q|r)	1	0	0	0	0.02282235	1.02282235	0.25052988
p(b,q|r)	1	0.16091691	0.12378223	0.42395415	0	1.7086533	0.41851717
p(a,r|r)	0	0	0	0	0.05325215	0.05325215	0.01304357
p(b,r|r)	0	0.53613181	0.51446991	0.24730659	0	1.29790831	0.31790938
					total for q	2.54028653	1
					total for r	4.0826361	1

after iteration

	p(...|q)	p(...|r)
p(a,q|...)	0.401191121	0.250529884
p(b,q|...)	0.438052698	0.418517167
p(a,r|...)	0.030138964	0.01304357
p(b,r|...)	0.130617217	0.317909379
totals	1	1

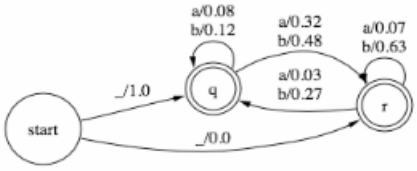
before iterations

	p(...|q)	p(...|r)
p(a,q|...)	0.08	0.03
p(b,q|...)	0.12	0.27
p(a,r|...)	0.32	0.07
p(b,r|...)	0.48	0.63
totals	1	1

b. Viterbi

	p(...|q)	p(...|r)	p(...|start)
p(a|...)	**0.4**	**0.1**	**0**
p(b|...)	**0.6**	**0.9**	**0**
p(e|...)	**0**	**0**	**1**
p(q|...)	**0.2**	**0.3**	**1**
p(r|...)	**0.8**	**0.7**	**0**

	p(...|q)	p(...|r)
p(a,q|...)	**0.08**	**0.03**
p(b,q|...)	**0.12**	**0.27**
p(a,r|...)	**0.32**	**0.07**
p(b,r|...)	**0.48**	**0.63**
totals	**1**	**1**



input = bbba	e	b	bb	bbb	bbba
best(q)	1	0.12	0.1296	0.081648	0.00653184
best(r)	0	0.48	0.3024	0.190512	0.02612736
	q	qq	qqq	qrqq	qrrqq
	1	0.12	0.0144	0.015552	0.00653184
	r	rq	qrq	qrrq	qrrrq
	0	0	0.1296	0.081648	0.00571536
		rr	qrr	qrrr	qrrrr
		0	0.3024	0.190512	0.01333584
		qr	qqr	qrqr	qrrqr
		0.48	0.0576	0.062208	0.02612736