Algorithmique pour la bioinformatique

Chaînes de Markov cachées

Nicolas Lartillot

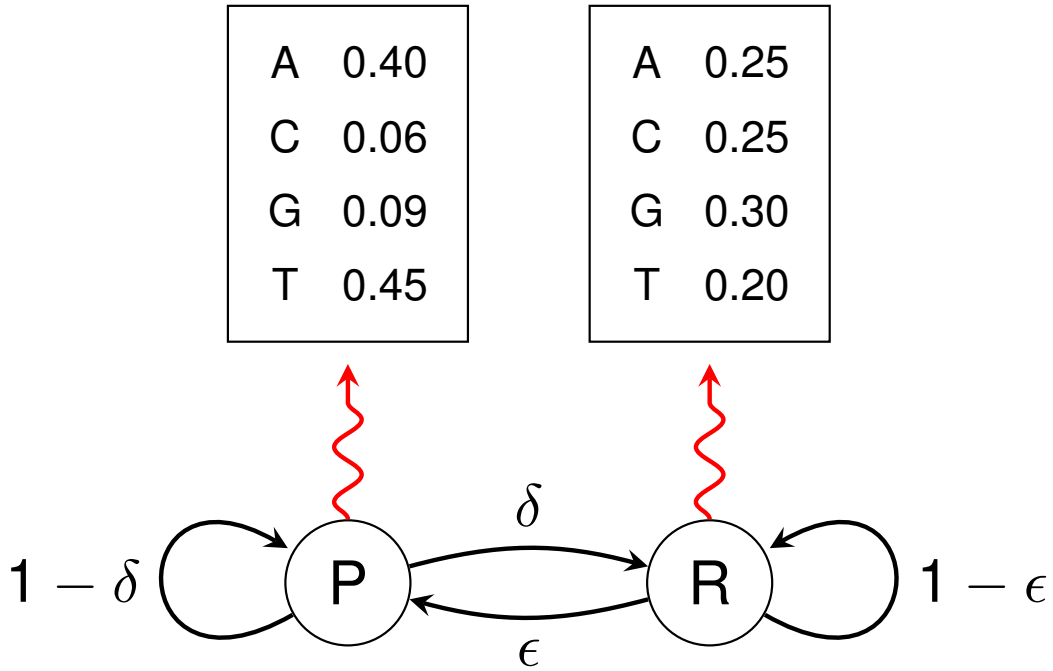
January 2017

1. Introduction

2. Hidden Markov Models
   - Formal definition
   - Most likely path: Viterbi algorithm
   - Summing over paths: Baum-Welsch algorithm
   - Learning HMM parameters and structure

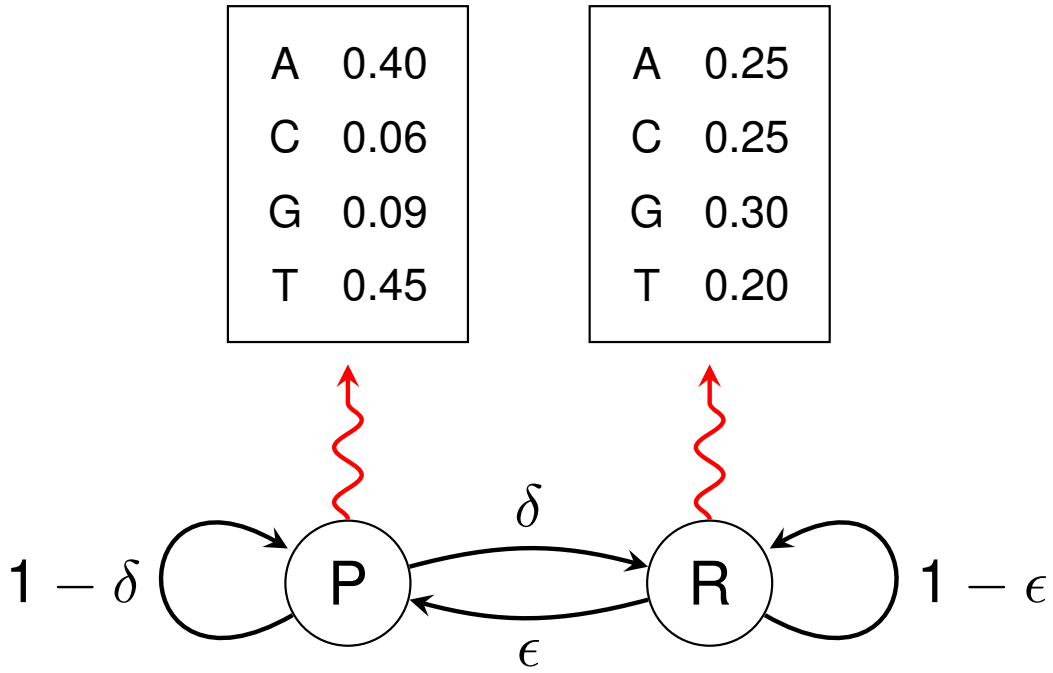
3. Examples
   - Predicting secondary structure of proteins
   - Pairwise alignment
   - Multiple alignment: profile HMMs

Model of alternating GC-rich and GC-poor regions




```
observed sequence (S)    ACTAGAATGGCGGCCAGACGATTACATA..
  hidden sequence (H)    PPPPPPPPPRRRRRRRRRRRPPPPPPPP..
```

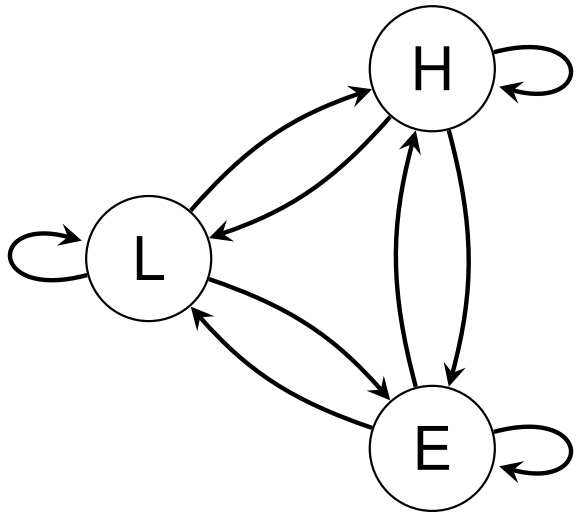
Emission and transition probabilities



transition probabilities ($\delta = \epsilon = 0.001$)

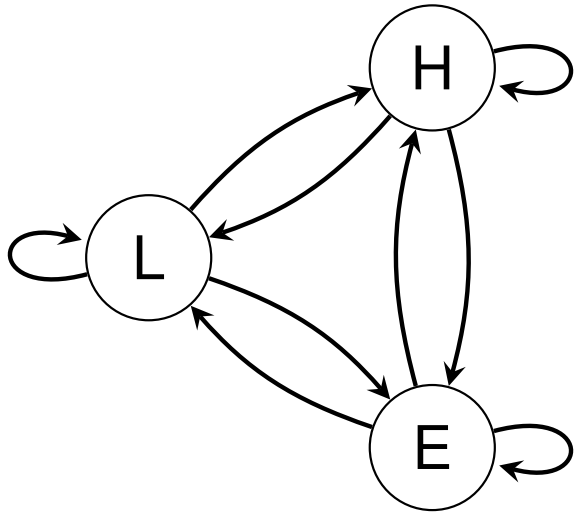
	P	R
P	0.999	0.001
R	0.001	0.999

Secondary structure



- 3 hidden states: L (loop), H (Helix) and E (Extended beta-strand)
- in each state, 20 emission probabilities (amino-acids)

Secondary structure



transition probabilities between hidden states

	L	H	E
L	0.85	0.07	0.08
H	0.05	0.94	0.01
E	0.06	0.01	0.93

$$\mathbf{q} = (q_{kl})_{k,l=L,H,E}$$

emission probabilities

	A	C	...	Y
L	0.05	0.07	...	0.08
H	0.03	0.03	...	0.12
E	0.10	0.11	...	0.03

$$\mathbf{e} = (e_k(x))_{k=L,H,E,\ x=A,C,\ldots,Y}$$

A Hidden Markov chain M characterized by

- an alphabet of hidden states $s_1, s_2, \ldots, s_K$
- an alphabet of observable symbols $v_1, v_2, \ldots, v_P$
- initial probabilities (over hidden states) π_k (vector of dim K)
- transition probabilities (between hidden states) q_{kl} ($K \times K$ matrix)
- emission probabilities $e_k(v_p)$ (K vectors of dim P)

A realization: 2 parallel series of random variables

- the hidden state path $h = (h_t)_{t=1..T}$
- the observed sequence of emitted symbols: $x = (x_t)_{t=1..T}$

Joint probability (hidden and observed states)

$$p(x, h \mid M) = \pi(h_1) e_{h_1}(x_1) \left[\prod_{t=1}^{T-1} q_{h_t h_{t+1}} e_{h_{t+1}}(x_{t+1}) \right]$$

Probability of paths and emissions

Joint probability (hidden and observed states)

$$
\begin{aligned}
p(x, h \mid M) &= \pi(h_1) e_{h_1}(x_1) \left[\prod_{t=1}^{T-1} q_{h_t h_{t+1}} e_{h_{t+1}}(x_{t+1}) \right] \\
&= \left[\pi(h_1) \prod_{t=1}^{T-1} q_{h_t h_{t+1}} \right] \left[e_{h_1}(x_1) \prod_{t=1}^{T-1} e_{h_{t+1}}(x_{t+1}) \right] \\
&= p(h \mid M)\, p(x \mid h, M)
\end{aligned}
$$

Joint probability (hidden and observed states)

$$p(x, h \mid M) = p(x \mid h, M)\, p(h \mid M)$$

Marginal probability of observed sequence

(sum over all paths)

$$p(x \mid M) = \sum_h p(x, h \mid M)$$

Posterior probability of hidden states (Bayes theorem)

$$p(h \mid x, M) = \frac{p(x \mid h, M)\, p(h \mid M)}{p(x \mid M)}$$

$$= \frac{p(x, h \mid M)}{p(x \mid M)}$$

The main algorithmic problems of HMM

paths

given an observed sequence x

- find the most likely hidden path (maximize $p(h \mid x, M)$)
- integrate probability over all paths (compute $p(x \mid M)$)

learning HMMs

- estimating parameters (transition and emission probabilities)
- choosing structure (how many hidden states, etc)

two types of learning

supervised: given annotated examples (x, h pairs)

unsupervised: given non annotated examples (x only)

Most likely path: Viterbi algorithm

$h = (h_t)_{t=1..T}$: the sequence of hidden state (hidden path)

$x = (x_t)_{t=1..T}$: the observed sequence of emitted symbols

$p(x, h \mid M)$: the joint probability

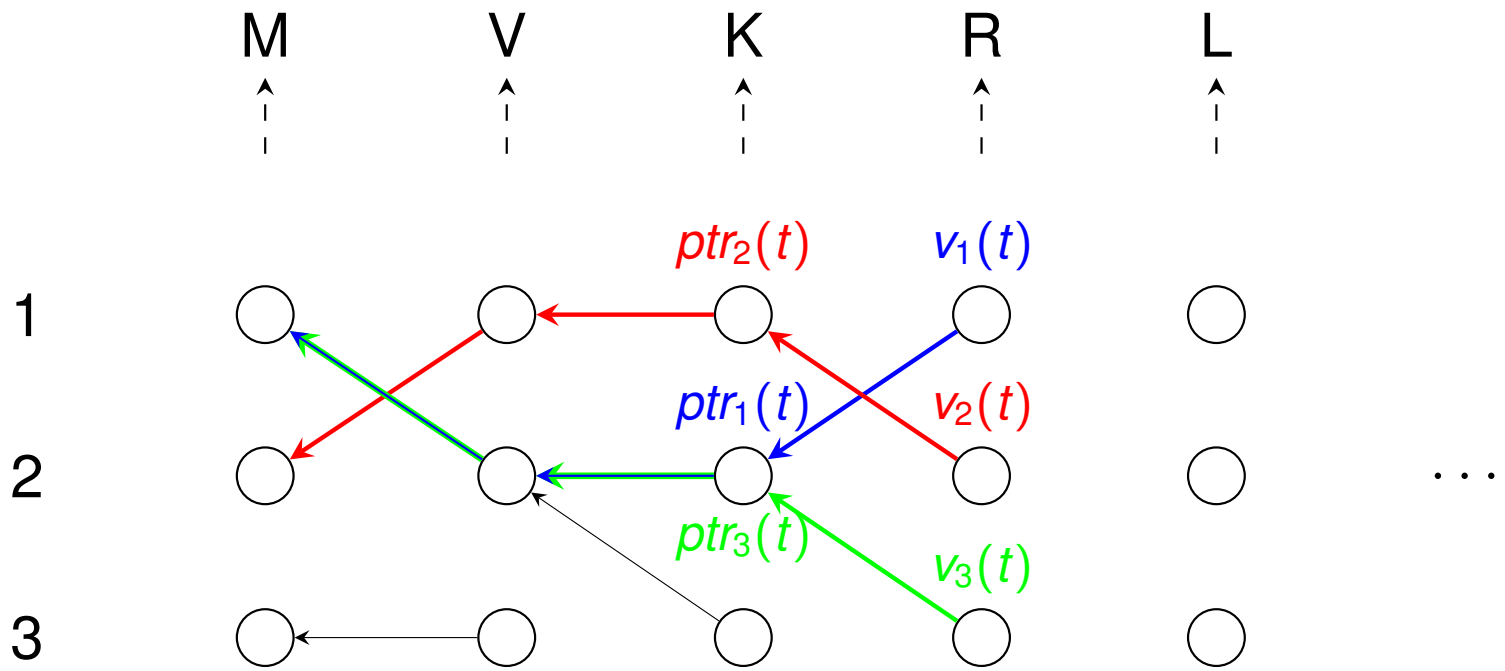
question

Given $x = (x_t)_{t=1..T}$, find $\hat{h} = (\hat{h}_t)_{t=1..T}$ such that:

$$\hat{h} = \max_h p(x, h \mid M)$$

- maximum is over K^T possible paths
- cannot be computed by brute force searching
- but exhaustive search is possible using dynamic programming

Viterbi recursion



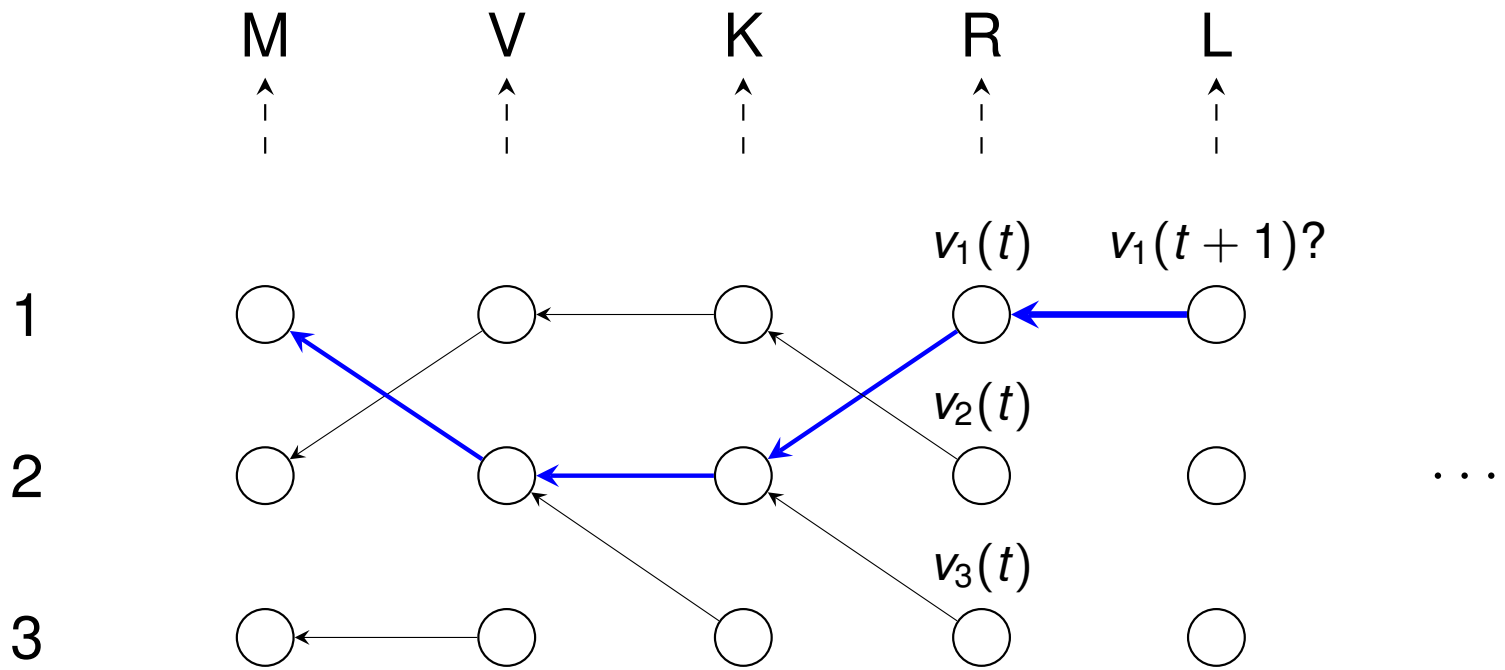
Definition

$v_k(t)$: probability of the most probable path ending in state k at time t.

$$v_k(t) = \max_{h_1,\ldots,h_{t-1}} p(h_1,\ldots,h_{t-1},h_t=k,x_1,\ldots,x_t \mid M)$$

$ptr_k(t)$: hidden state at time $t-1$ of this most probable path

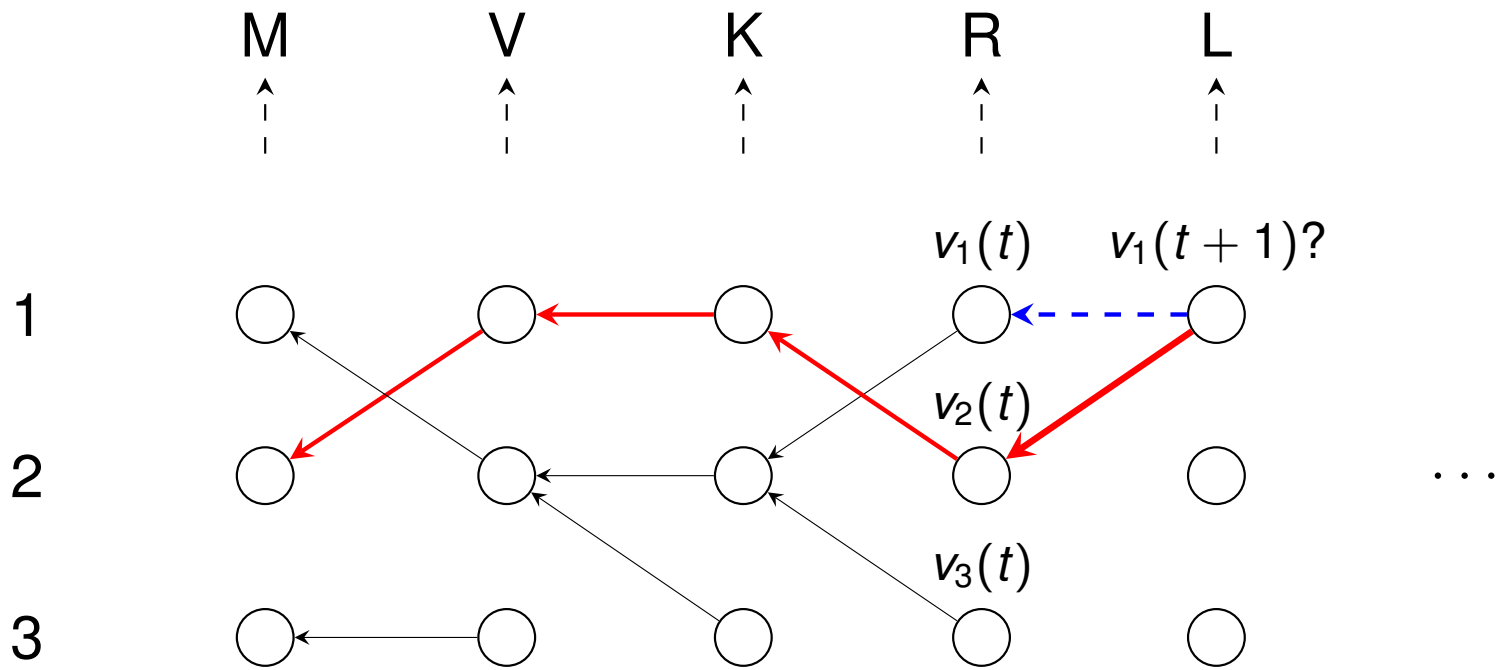
Viterbi recursion



recursion

$$v_1(t+1) \stackrel{?}{=} v_1(t) q_{11} e_1(L)$$

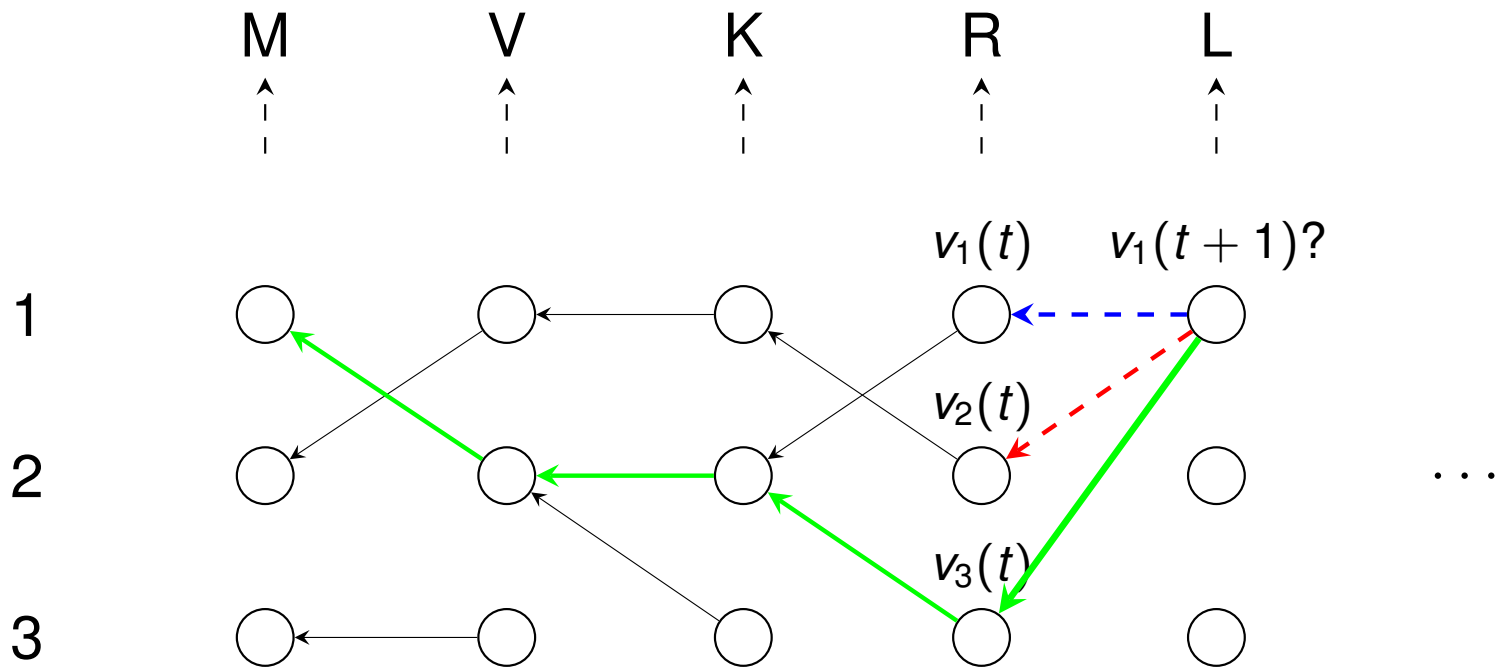
Viterbi recursion



recursion

$$v_1(t+1) \stackrel{?}{=} v_2(t) q_{21} e_1(L)$$

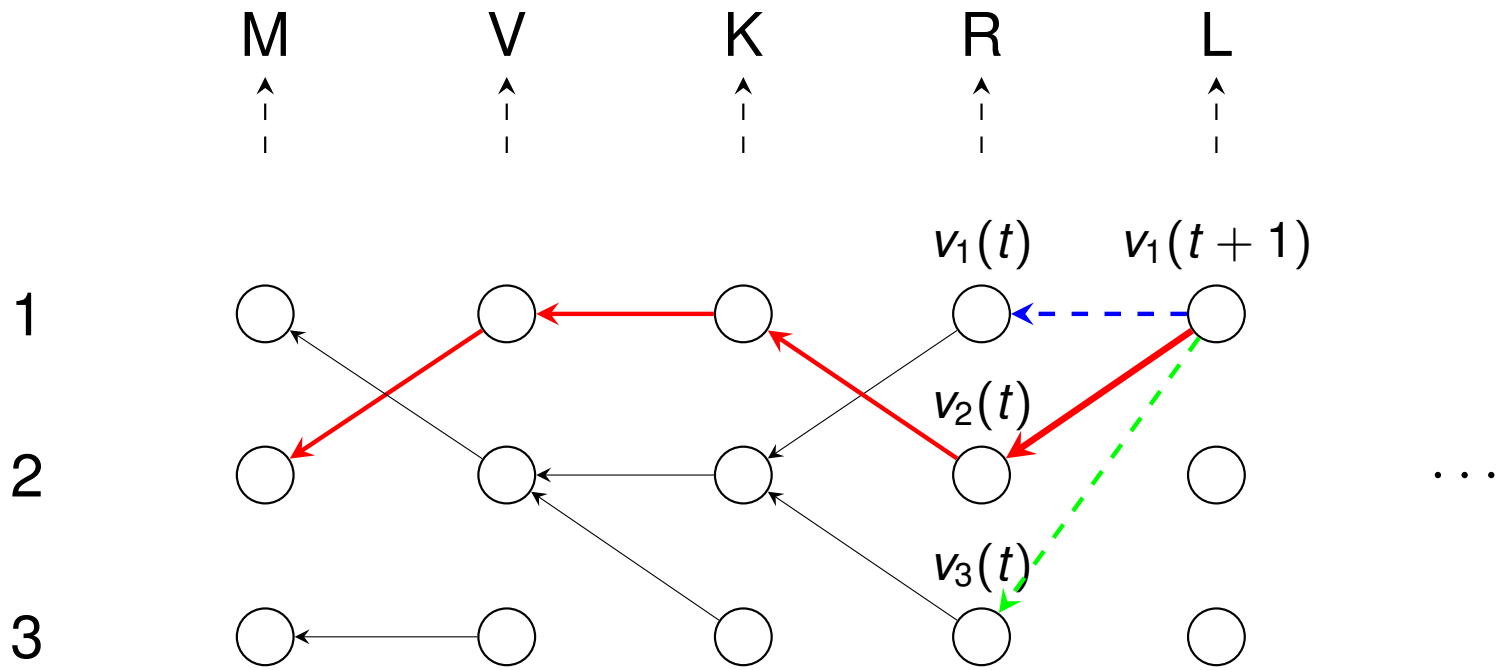
Viterbi recursion



recursion

$$v_1(t+1) \stackrel{?}{=} v_3(t) q_{31} e_1(L)$$

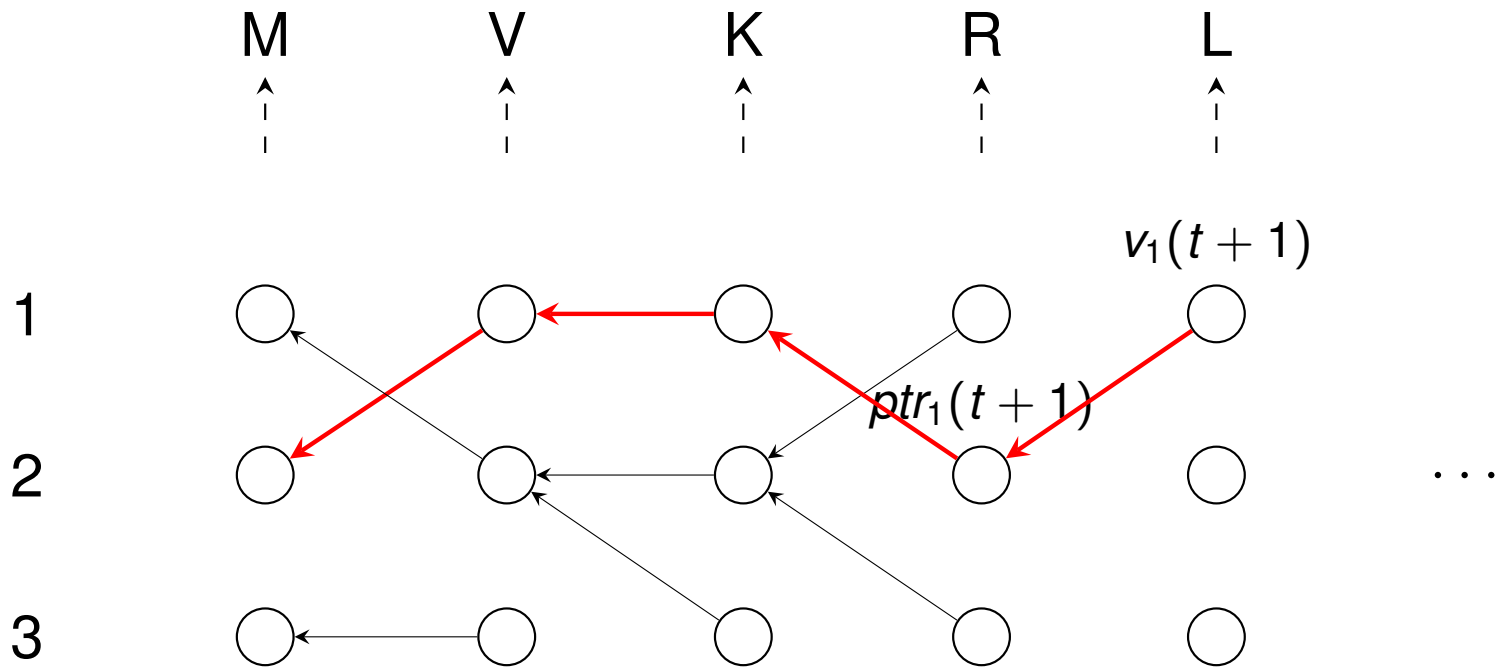
Viterbi recursion



recursion

$$v_1(t+1) = \max_{k=1,2,3} v_k(t) q_{k1} e_1(x_{t+1})$$

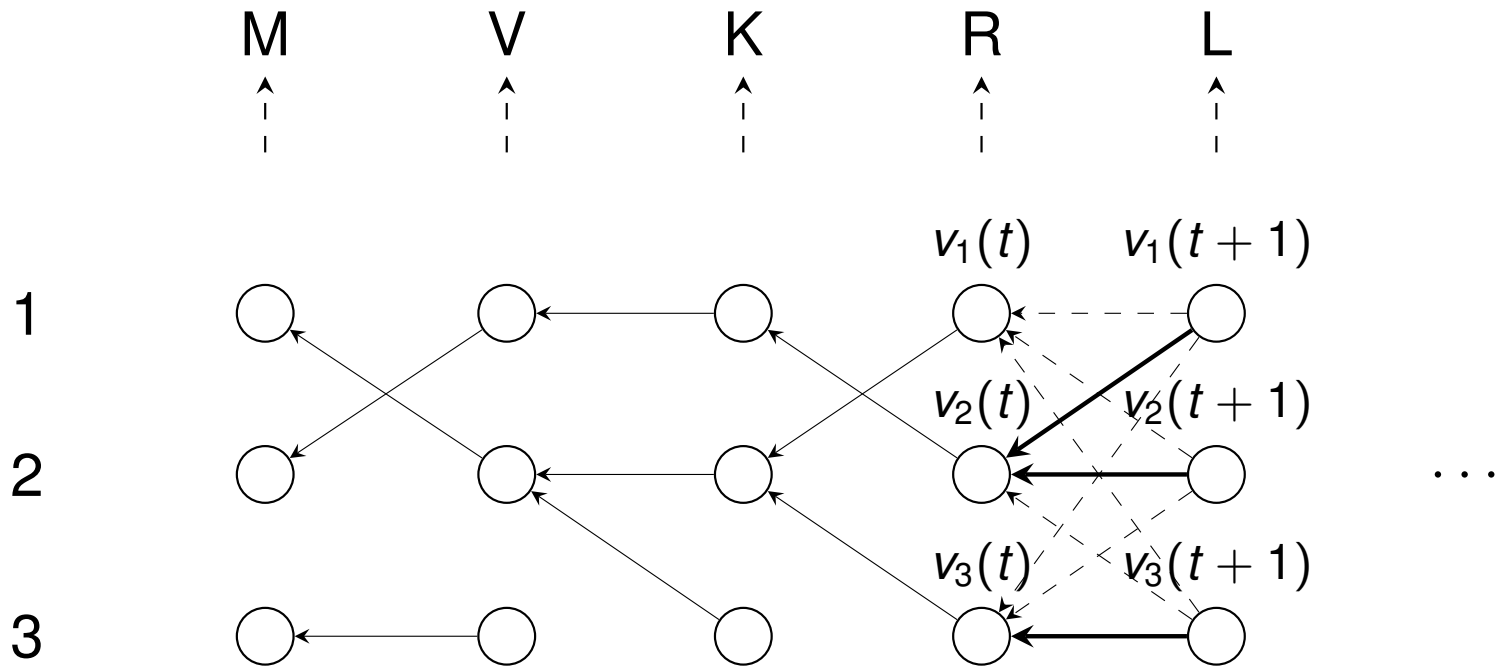
Viterbi recursion



recursion

$$ptr_1(t+1) = \arg \max_{k=1,2,3} v_k(t) q_{k1} e_1(x_{t+1})$$

Viterbi recursion



recursion

$$v_l(t+1) = \max_k v_k(t) q_{kl} e_l(x_{t+1})$$

Viterbi algorithm

initialization $v_0(0) = 1$, $v_k(0) = 0$ for $k < 0$.

recursion $t = 0 \dots T-1$

$$v_l(t+1) = e_l(x_{t+1}) \max_k v_k(t) q_{kl}$$

$$ptr_l(t+1) = argmax_k v_k(t) q_{kl}$$

termination

$$p(x, \hat{h} \mid M) = \max_k v_k(T)$$

$$\hat{h}_T = argmax_k v_k(T)$$

traceback $t = T-1 \dots 1$

$$\hat{h}_t = ptr_{t+1}(\hat{h}_{t+1})$$

Complexity of Viterbi algorithm

compute a product of probabilities

- at each time step $t = 1..T$
- for each state $k = 1..K$ at time t
- and for each state $l = 1..K$ at time $t + 1$

Complexity

$$TK^2$$

- linear in T: very efficient, even for long genomic sequences
- quadratic in K: should keep model simple (low number of hidden states)

Integrating over all paths. Baum Welsch algorithm

$h = (h_t)_{t=1..T}$: the sequence of hidden state (hidden path)

$x = (x_t)_{t=1..T}$: the observed sequence of emitted symbols

$p(x, h \mid M)$: the joint probability

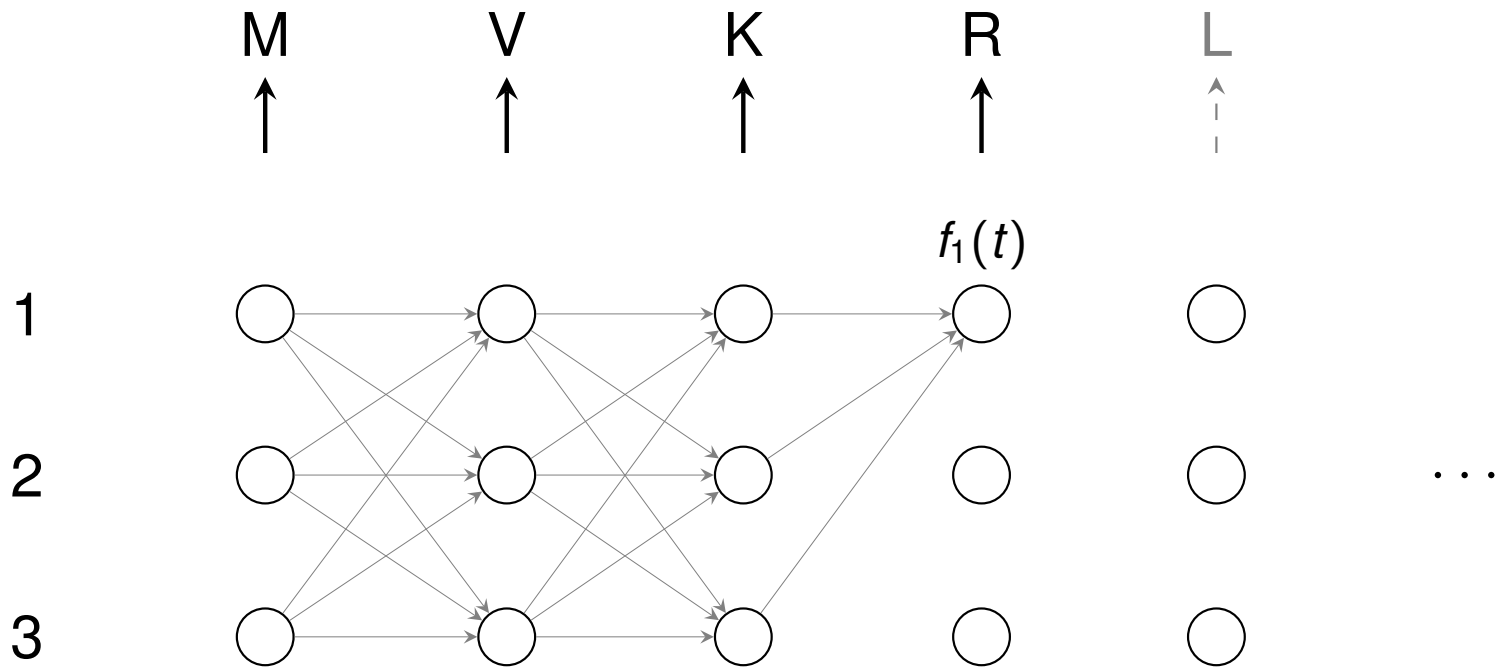
question

Given $x = (x_t)_{t=1..T}$, compute

$$p(x \mid M) = \sum_h p(x, h \mid M)$$

- sum is over K^N possible paths
- again, exhaustive sum is possible using dynamic programming (forward and backward algorithms)

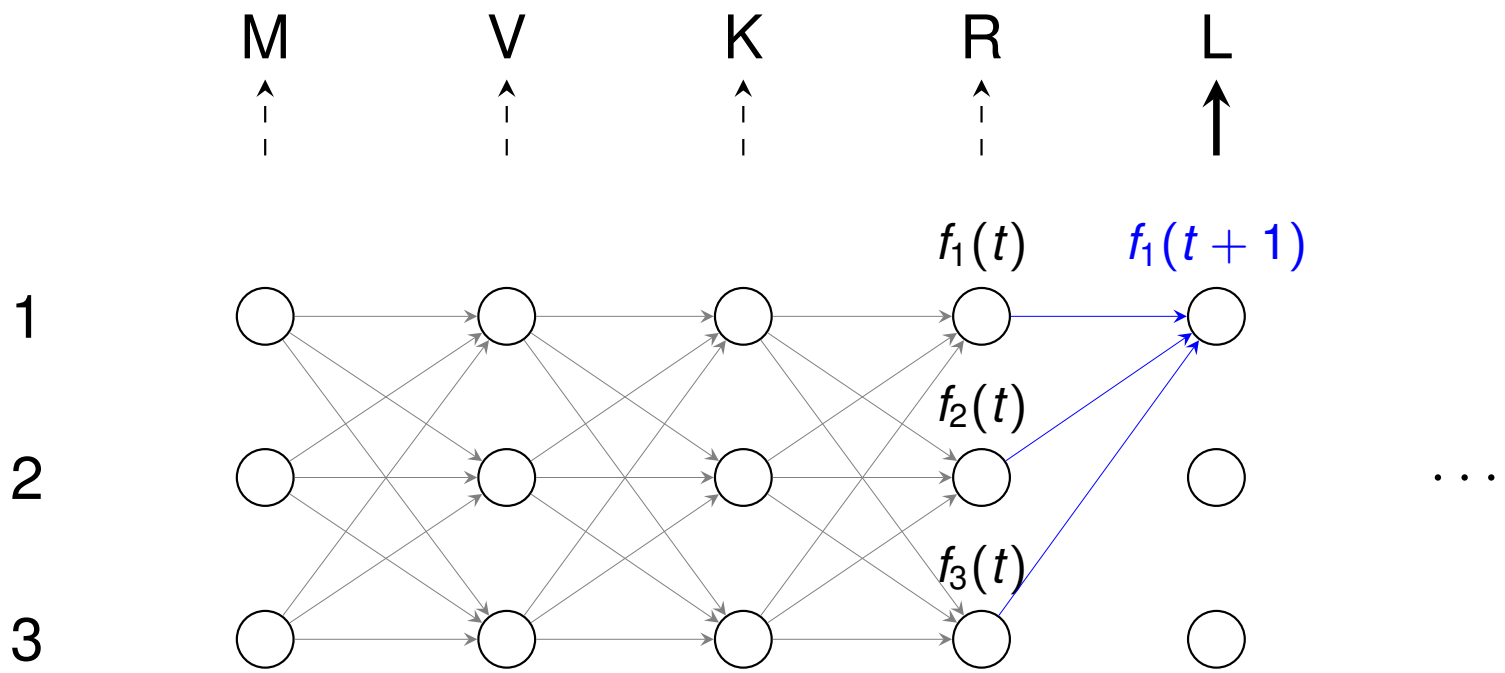
Forward algorithm



definition

$$f_k(t) = p(x_1, \ldots, x_t, h_t = k)$$

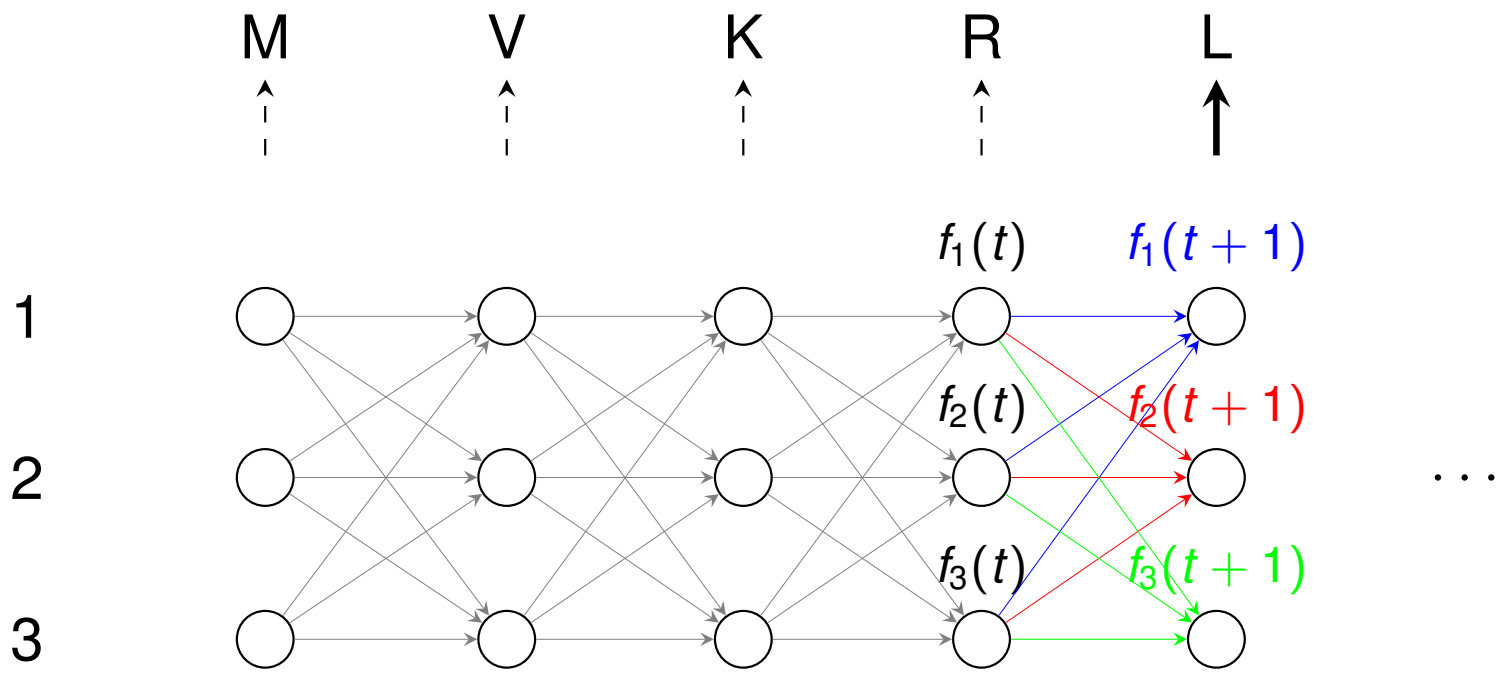
Forward algorithm



recursion

$$f_l(t+1) = \sum_k f_k(t) q_{kl} e_l(x_{t+1})$$

Forward algorithm



recursion

$$f_l(t+1) = \sum_k f_k(t) q_{kl} e_l(x_{t+1})$$

Forward algorithm

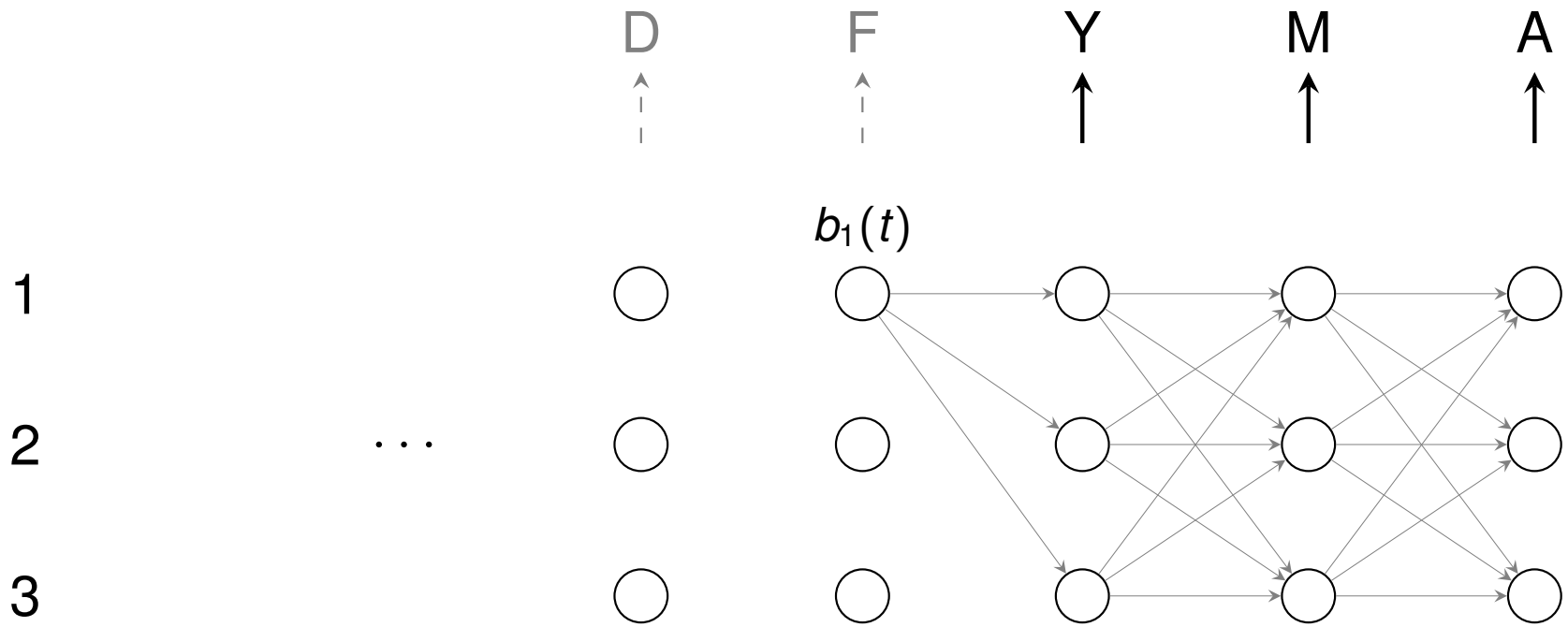
initialization $f_0(0) = 1$, $f_k(0) = 0$ for $k > 0$.

recursion $t = 0 \dots T - 1$

$$f_l(t+1) = \sum_k f_k(t) q_{kl} e_l(x_{t+1})$$

termination $p(x \mid M) = \sum_k f_k(T)$

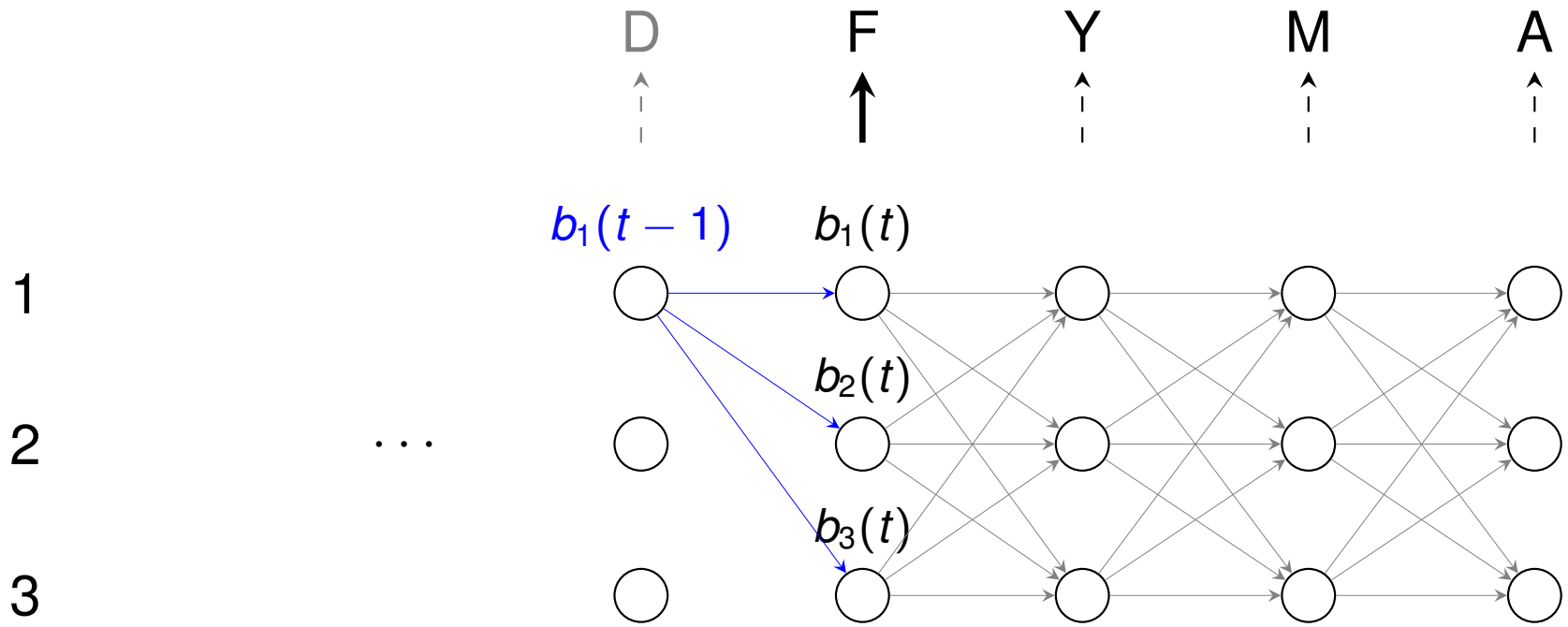
Backward algorithm



definition

$$b_k(t) = p(x_{t+1}, \ldots, x_T \mid h_t = k)$$

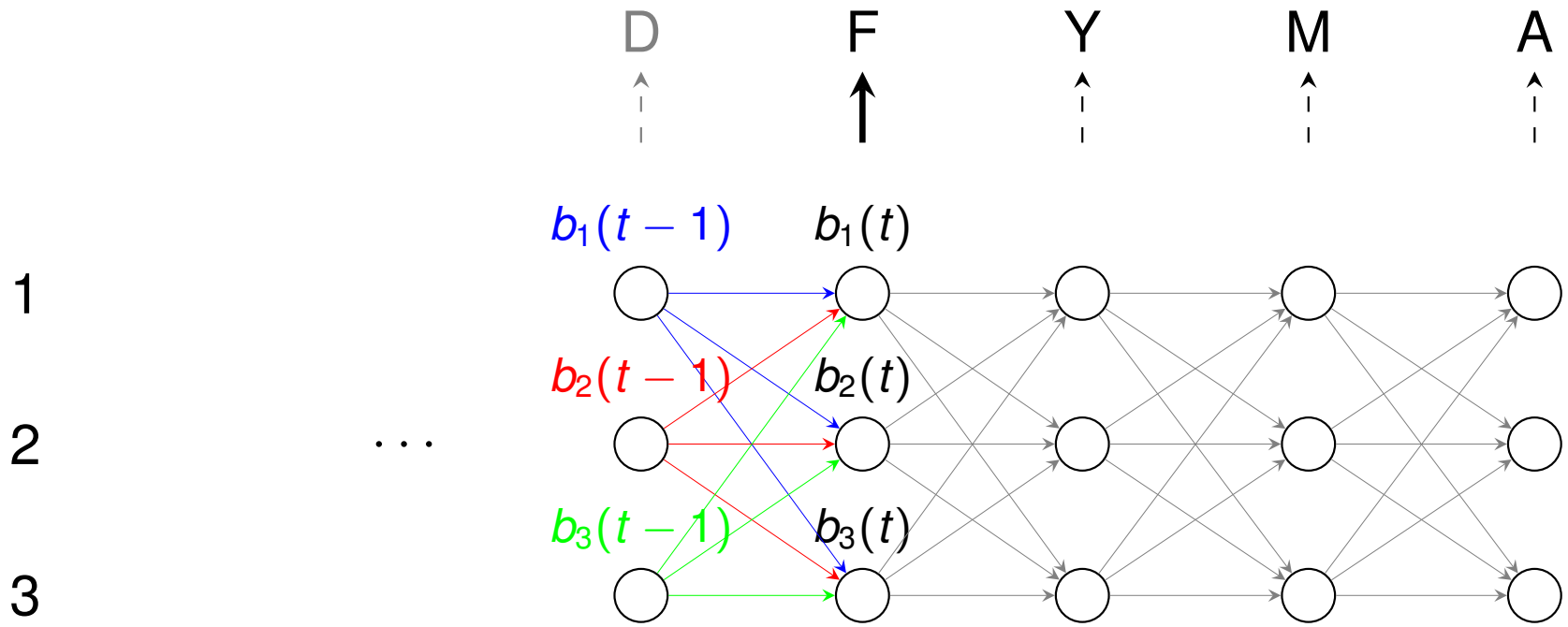
Backward algorithm



recursion

$$b_k(t-1) = \sum_l q_{kl} e_l(x_t) b_l(t)$$

Backward algorithm



recursion

$$b_k(t-1) = \sum_l q_{kl} e_l(x_t) b_l(t)$$

Backward algorithm

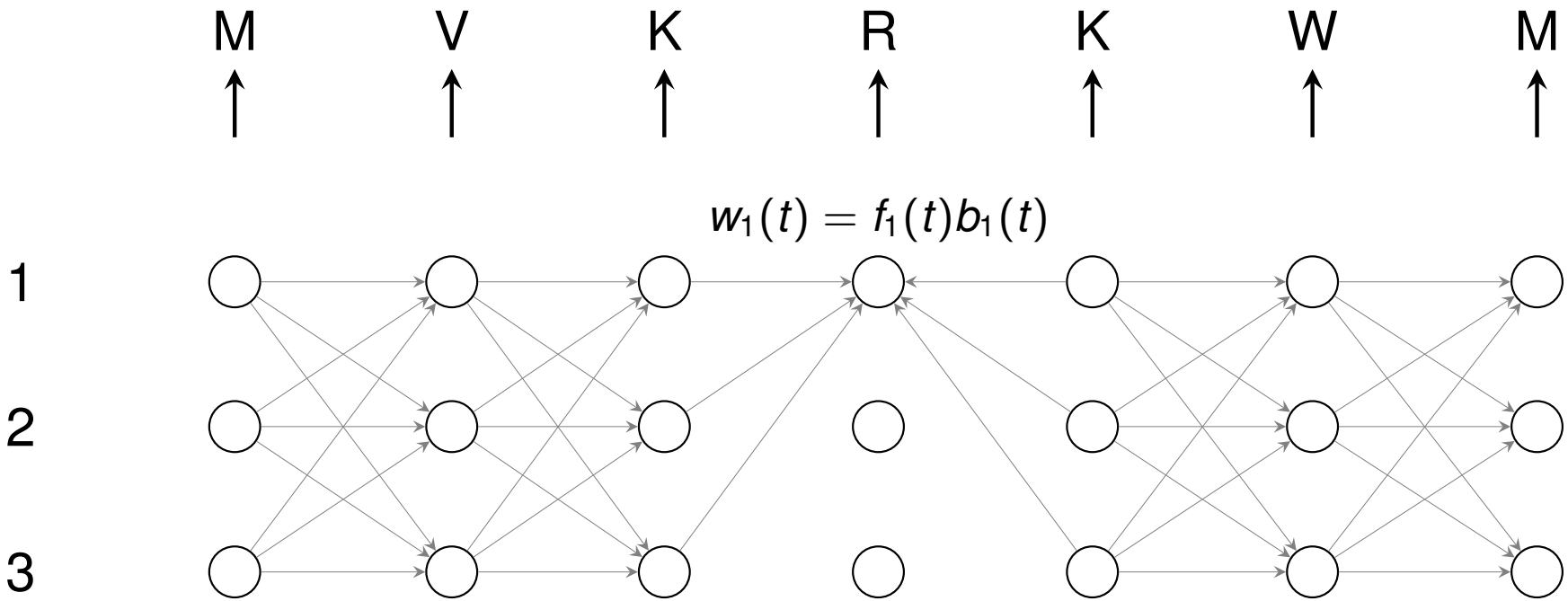
initialization $b_k(L) = q_{k0}$.

recursion $t = T \ldots 2$

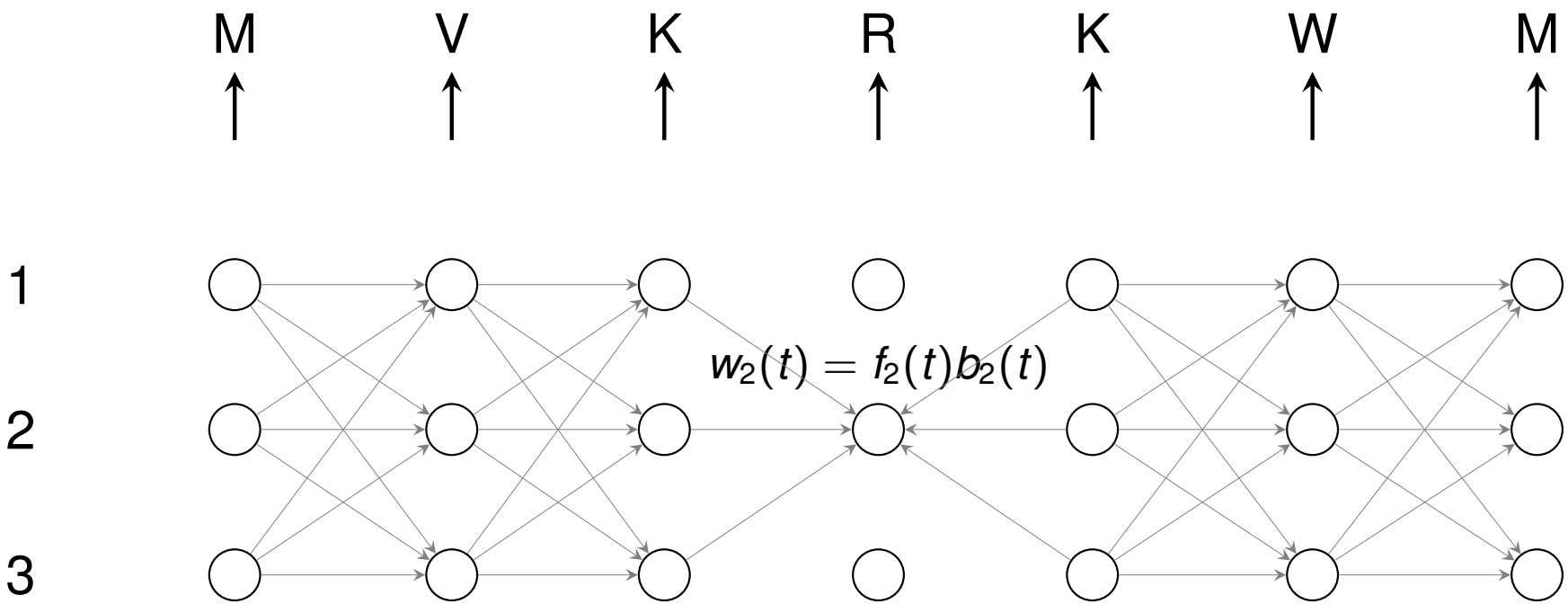
$$b_k(t-1) = \sum_l e_l(x_t) b_l(t) q_{kl}$$

termination $p(x \mid M) = \sum_l q_{0l} e_l(x_1) b_l(1)$

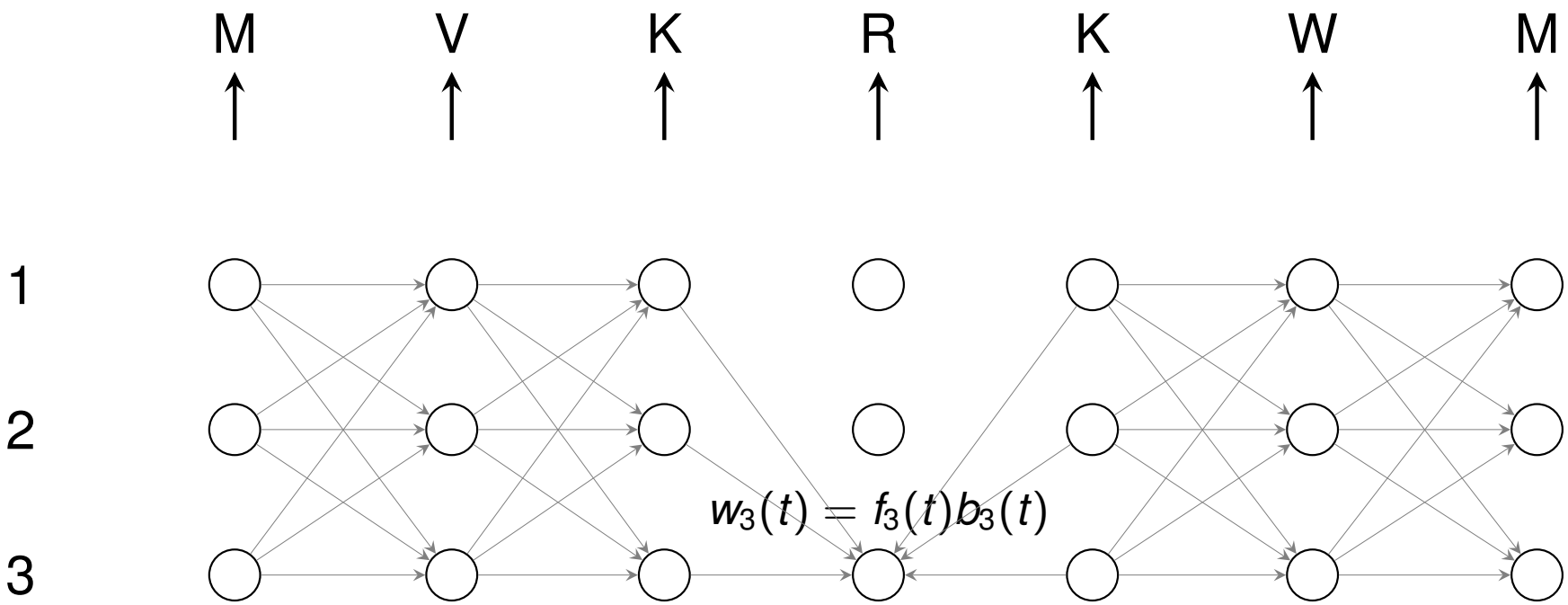
Posterior decoding



Posterior decoding



Posterior decoding



Posterior decoding

$$
\begin{aligned}
w_k(t) = \sum_{h|h_t=k} p(x,h) &= p(x_1,\ldots,x_t,h_t=k)\, p(x_{t+1},\ldots,x_T \mid h_t=k) \\
&= f_t(k)\, b_t(k) \\
p(h_t=k \mid x) &= \frac{w_k(t)}{\sum_l w_l(t)}
\end{aligned}
$$

Learning HMM parameters

question

Given a training database of (x, h) annotated pairs learn the parameters of the HMM.

just compute

N_{kl} total number of transitions from k to l in database

N_k total number of transitions from k: $N_k = \sum_l N_{kl}$

M_{kp} total number of emissions of symbol p when in state k

M_k total number of emissions in state k: $M_k = \sum_p M_{kp}$

ML estimate

$$q_{kl} = \frac{N_{kl}}{N_k}, \qquad e_k(s_p) = \frac{M_{kp}}{M_k}$$

Learning HMM parameters

question

Given a training database of (x, h) annotated pairs learn the parameters of the HMM.

just compute

N_{kl} total number of transitions from k to l in database

N_k total number of transitions from k: $N_k = \sum_l N_{kl}$

M_{kp} total number of emissions of symbol p when in state k

M_k total number of emissions in state k: $M_k = \sum_p M_{kp}$

Bayes estimate (add pseudocounts)

$$q_{kl} = \frac{N_{kl}+n_l}{N_k+n}, \qquad e_k(s_p) = \frac{M_{kp}+m_p}{M_k+m}$$

Unsupervised training

Viterbi training (approximate)

- start from rough parameter estimates
- use Viterbi algorithm to infer paths
- compute N_{kl} and M_{kp} on viterbi paths
- estimate parameters based on N_{kl} and M_{kp}
- iterate

Unsupervised training

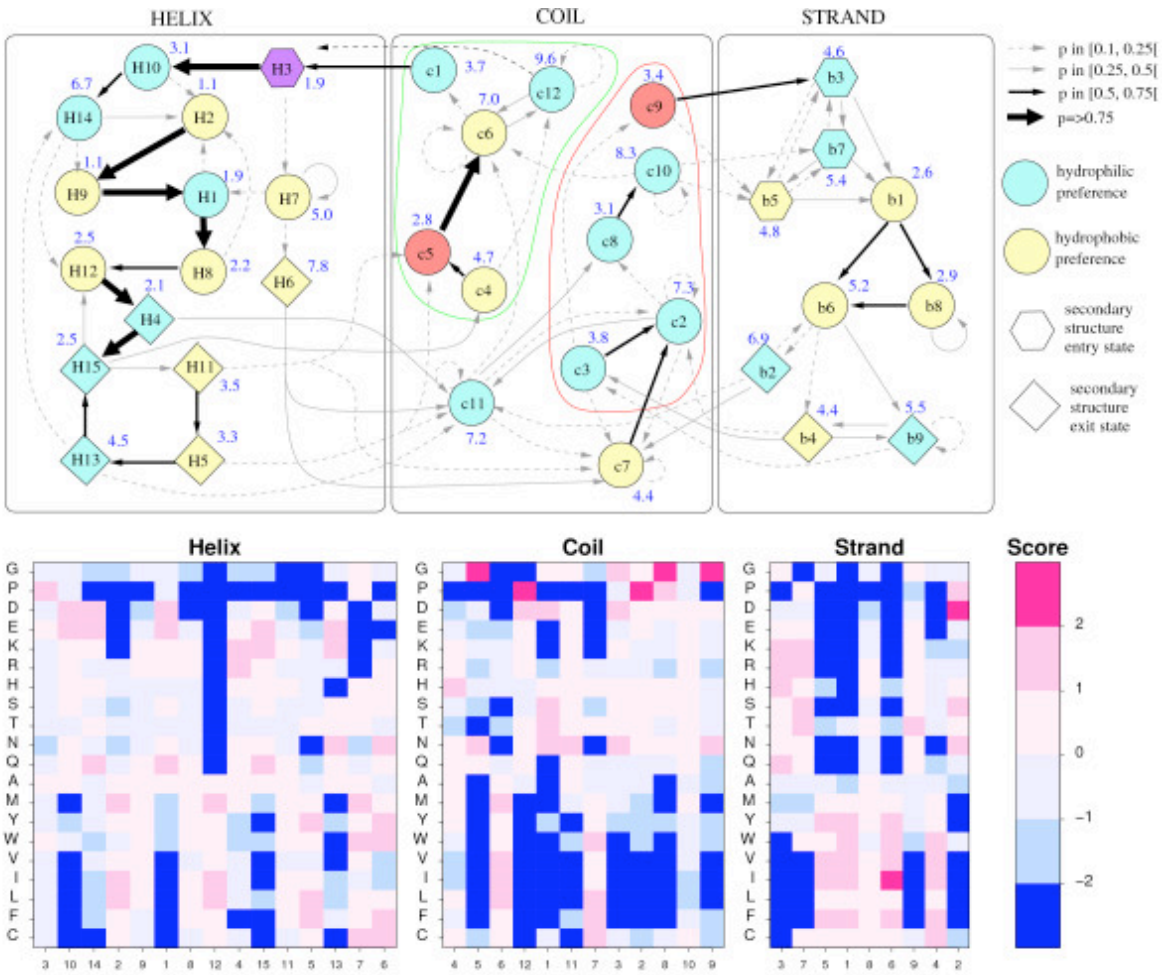
Viterbi training (approximate)

- start from rough parameter estimates
- use Viterbi algorithm to infer paths
- compute N_{kl} and M_{kp} on viterbi paths
- estimate parameters based on N_{kl} and M_{kp}
- iterate

Baum Welsch training (exact: EM)

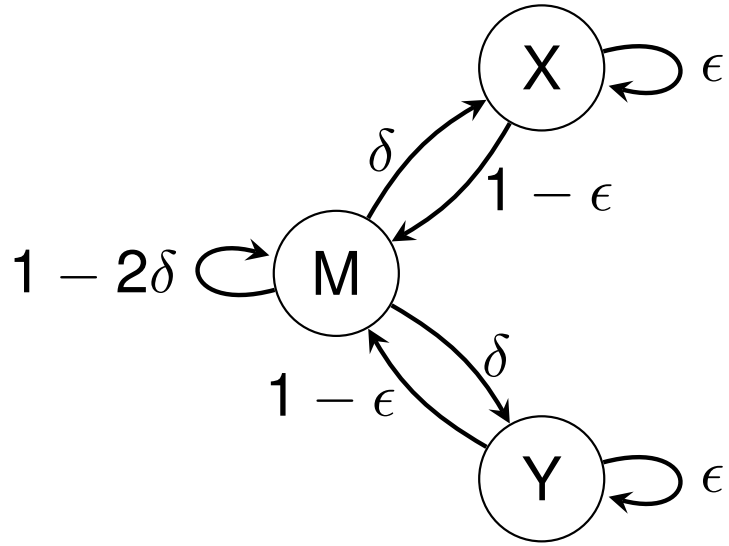
- start from rough parameter estimates
- use Baum Welsch algorithm to compute *expectations* of N_{kl} and M_{kp} over all paths
- estimate parameters based on those expectations $\overline{N}_{kl}$ and $\overline{M}_{kp}$
- iterate

Predicting secondary structure



Martin et al 2006

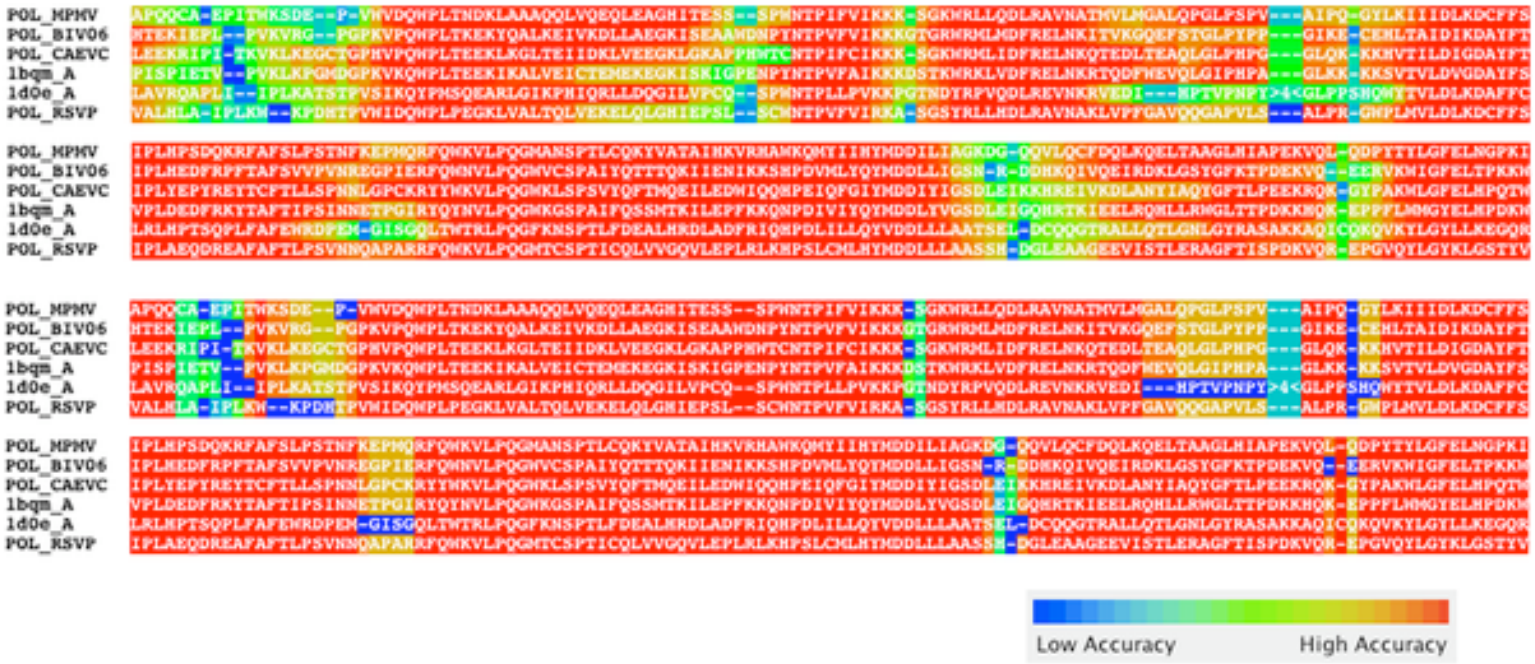
Pair HMM for pairwise alignment




```
        sequence X      ACAGAAT--ATT
        sequence Y      A-AGGATACAT-
hidden  sequence        MXMMMMMYYMMX
```

- gap opening with probability δ
- gap extension with probability ϵ
- emission probability (1 base in state *X* or *Y*, 2 bases in state *M*)

Posterior decoding (FSA)



- for each pair of positions, compute posterior probability of being aligned (i.e. jointly emitted by a match state)
- put in same colums positions that have high match probability

Bradley et al PLOS Compute Biol 2009.

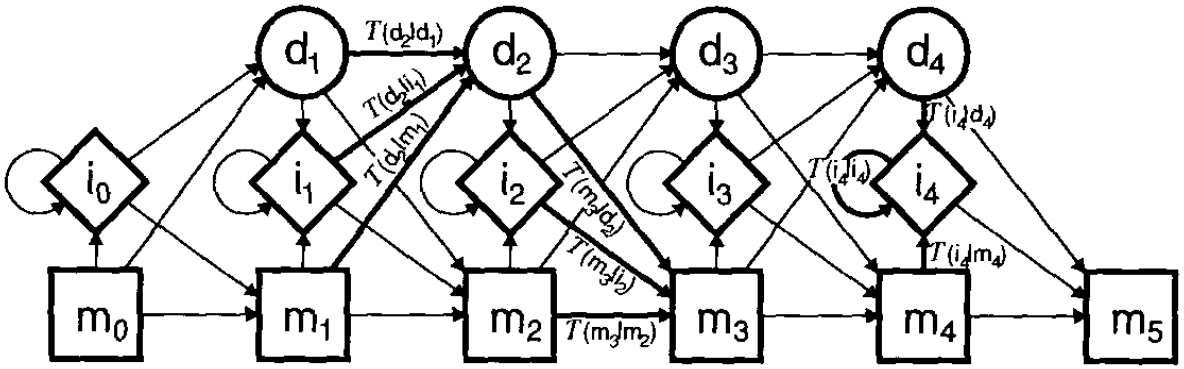
```
Helix                 AAAAAAAAAAAAAAAA   BBBBBBBBBBBBBBBBBCCCCCCCCCCC    DDDDDDDEE
HBA_HUMAN  ---------VLSPADKTNVKAAWGKVG--HAGEYGAEALERMFLSFPTTKTYFPHF-DLS-----HGSA
HBB_HUMAN  --------VHLTPEEKSAVTALWGKV----NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNP
MYG_PHYCA  ---------VLSEGEWQLVLHVWAKVEA--DVAGHGQDILIRLFKSHPETLEKFDRFKHLKTEAEMKASE
GLB3_CHITP ---------ILSADQISTVQASFDKVKG-------DPVGILYAVFKADPSIMAKFTQFAG-KDLESIKGTA
GLB5_PETMA PIVDTGSVAPLSAAEKTKIRSAWAPVYS--TYETSGVDILVKFFTSTPAAQEFFPKFKGLTTADQLKKSA
LGB2_LUPLU --------GALTESQAALVKSSWEEFNA--NIPKHTHRFFILVLEIAPAAKDLFS-FLK-GTSEVPQNNP
GLB1_GLYDI ---------GLSAAQRQVIAATWKDIAGADNGAGVGKDCLIKFLSAIPQMAAVFG-FSG----AS---DP

Helix       EEEEEEEEEEEEEEEEEE          FFFFFFFFFFFF  FFGGGGGGGGGGGGGGGGGG
HBA_HUMAN  QVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL--RVDPVNFKLLSHCLLVTLAAHLPAE
HBB_HUMAN  KVKAHGKKVLGAFSDGLAHL---D--NLKGTFATLSELHCDKL--HVDPENFRLLGNVLVCVLAHHFGKE
MYG_PHYCA  DLKKHGVTVLTALGAILKK----K-GHHEAELKPLAQSHATKH--KIPIKYLEFISEAIIHVLHSRHPGD
GLB3_CHITP PFETHANRIVGFFSKIIGEL--P---NIEADVNTFVASHKPRG---VTHDQLNNFRAGFVSYMKAHT--D
GLB5_PETMA DVRWHAERIINAVNDAVASM--DDTEKMSMKLRDLSGKHAKSF--QVDPQYFKVLAAVIADTVAAG----
LGB2_LUPLU ELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG---VADAHFPVVKEAILKTIKEVVGAK
GLB1_GLYDI GVAALGAKVLAQIGVAVSHL--GDEGKMVAQMKAVGVRHKGYGNKHIKAQYFEPLGASLLSAMEHRIGGK

Helix       HHHHHHHHHHHHHHHHHHHHHHHHHH
HBA_HUMAN  FTPAVHASLDKFLASVSTVLTSKYR------
HBB_HUMAN  FTPPVQAAYQKVVAGVANALAHKYH------
MYG_PHYCA  FGADAQGAMNKALELFRKDIAAKYKELGYQG
GLB3_CHITP FA-GAEAAWGATLDTFFGMIFSKM-------
GLB5_PETMA -----DAGFEKLMSMICILLRSAY-------
LGB2_LUPLU WSEELNSAWTIAYDELAIVIKKEMNDAA---
GLB1_GLYDI MNAAAKDAWAAAYADISGALISGLQS-----
```

- gaps are not uniformly distributed along sequences
- conserved blocks, alternating with regions of more variable length
- corresponds to structured / loop regions of the conformation

Profile HMM



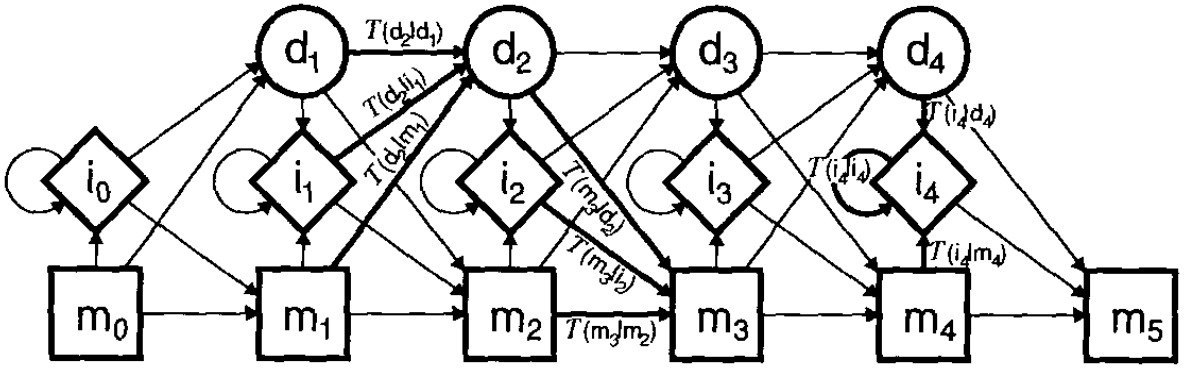
trained on seed alignment:

```
FQDN-R
FK-N-R
FK-E-R
FK-EVR
........
012.3.45
```

applied on new sequence

```
AEFWQ-RAI
...........
0ii1i2d4ii5
```

Profile HMM

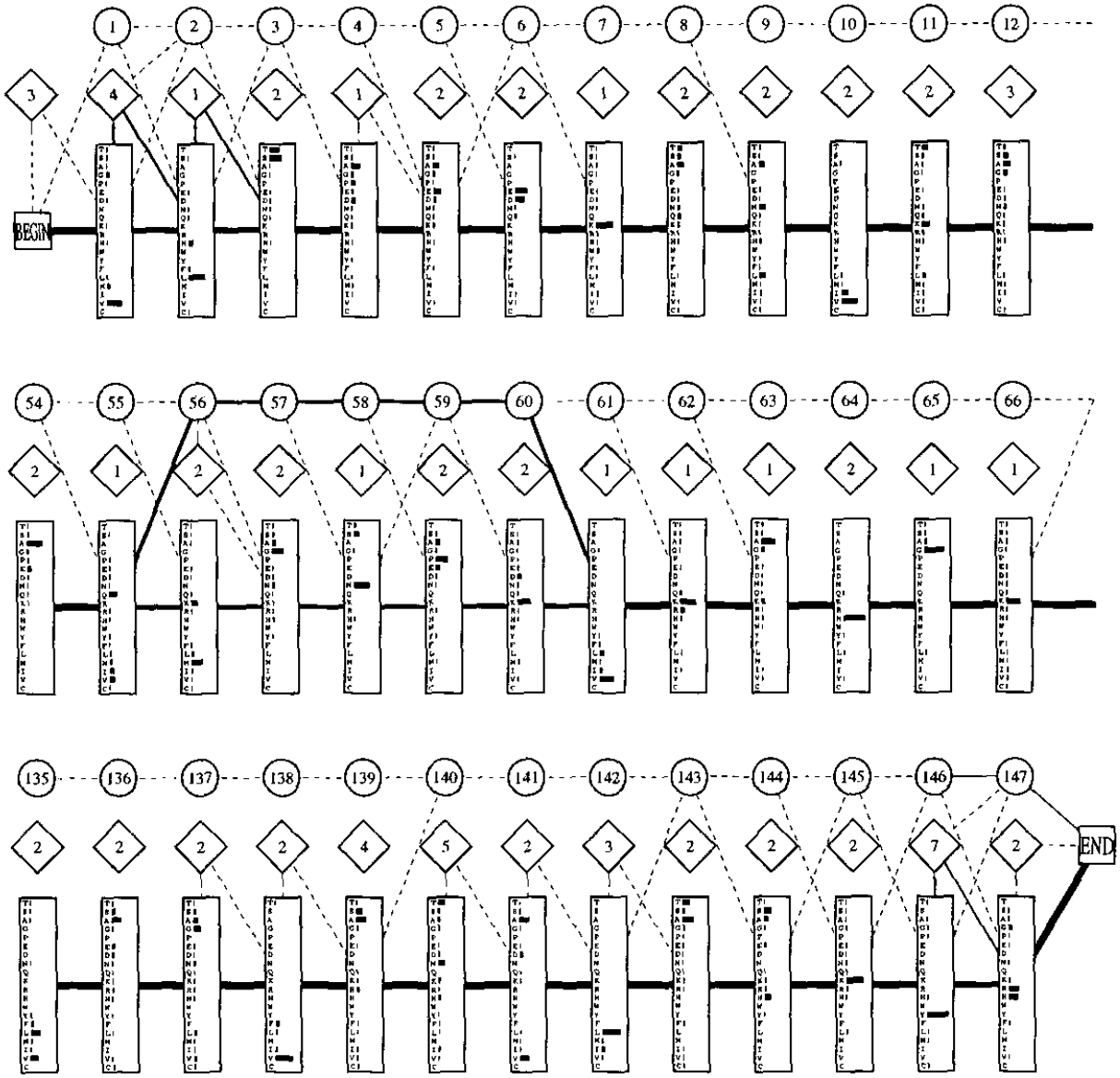


overall procedure

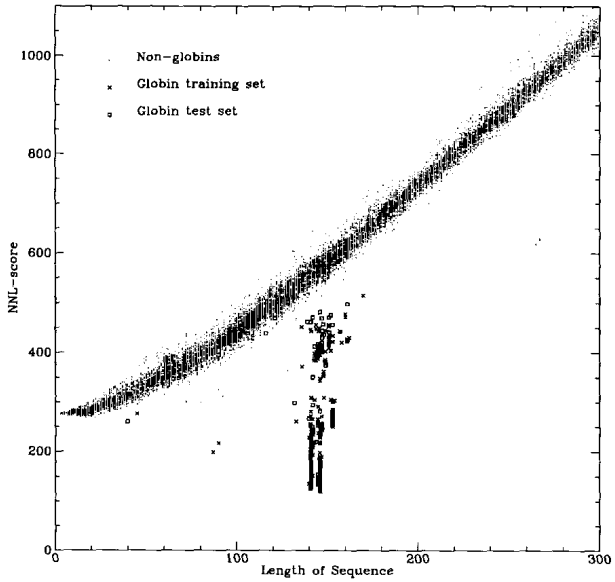
for each protein family

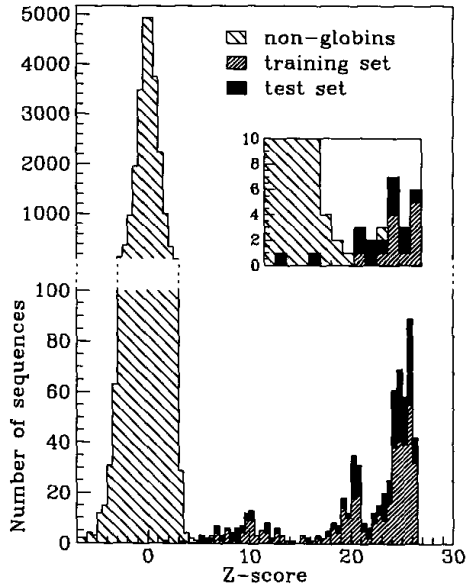
- make a seed alignment (based on superposition of 3D structures)
- train a profile HMM on seed alignment
- homologues should have higher $p(s \mid M)$: using Baum Welsch, search for homologues in databases.
- using Viterbi (or posterior decoding): align homologues with seed alignment

exemple: globin family



exemple: globin family





Krogh et al 1994