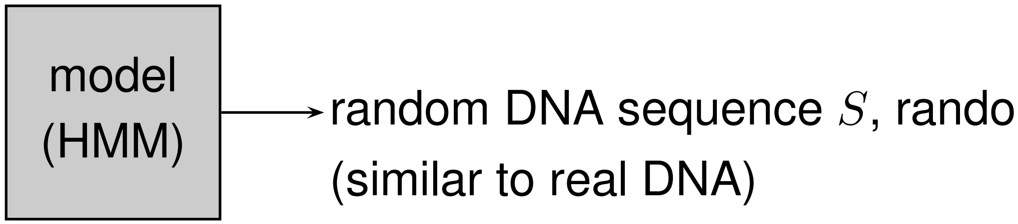
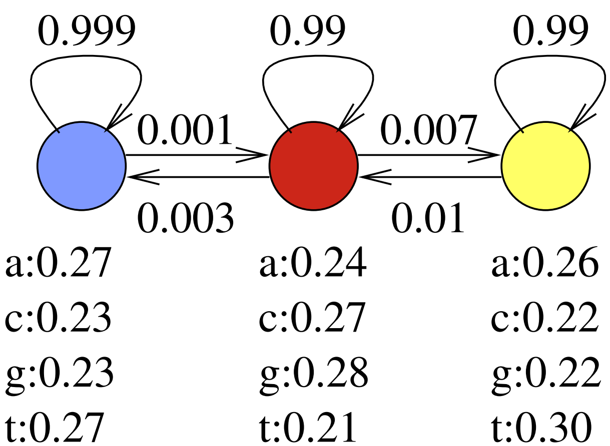
Recall: HMM (hidden Markov model, skrytý Markovov model)



$\Pr(S, A)$ – probability that the model generates pair (S, A).



Assume the model starts in the blue state

$$\Pr(\text{acag}) = 0.27 \cdot 0.001 \cdot 0.27 \cdot 0.99 \cdot 0.24 \cdot 0.99 \cdot 0.28 = 4.8 \cdot 10^{-6}$$

$$\Pr(\text{acag}) = 0.27 \cdot 0.999 \cdot 0.23 \cdot 0.999 \cdot 0.27 \cdot 0.999 \cdot 0.23 = 0.0038$$

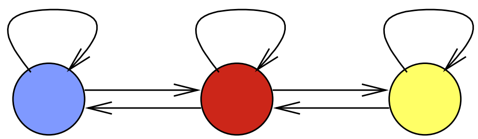
Another toy example: weather

- Period of low atmospheric pressure: mostly raining
- Period of high atmospheirc pressure: mostly sunny

Each period typically lasts several days

Exercise: Represent by an HMM

Recall: Parameters of HMMs (notation)



Sequence $S = S_1, \ldots, S_n$

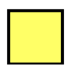
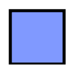
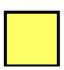
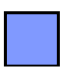
Annotation $A = A_1, \ldots, A_n$

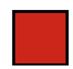
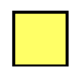
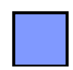
Model parameters:

Transition probability $a(u, v) = \Pr(A_{i+1} = v | A_i = u)$,

Emission probability $e(u, x) = \Pr(S_i = x | A_i = u)$,

Starting probability $\pi(u) = \Pr(A_1 = u)$.

a			
	0.99	0.007	0.003
	0.01	0.99	0
	0.001	0	0.999

e	a	c	g	t
	0.24	0.27	0.28	0.21
	0.26	0.22	0.22	0.30
	0.27	0.23	0.23	0.27

The resulting probability:

$$\Pr(A, S) = \pi(A_1)e(A_1, S_1) \prod_{i=2}^{n} a(A_{i-1}, A_i)e(A_i, S_i)$$

Viterbi algorithm

For a given HMM and sequence S,

find the most probable annotation (state path)

$A = \arg\max_A \Pr(A, S) = \arg\max_A \Pr(A \mid S)$

Any ideas?

Recall our example:

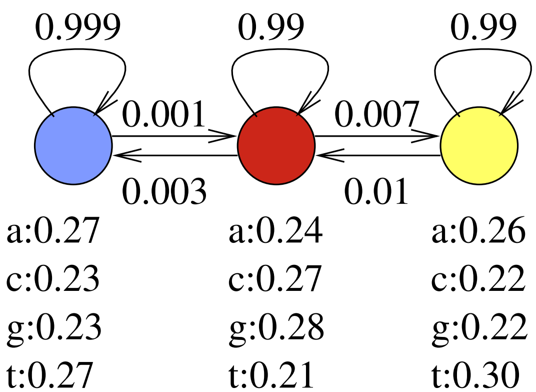
$$\Pr(\text{acag}) = 0.27 \cdot 0.001 \cdot 0.27 \cdot 0.99 \cdot 0.24 \cdot 0.99 \cdot 0.28 = 4.8 \cdot 10^{-6}$$

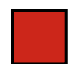
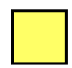
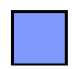
$$\Pr(\text{acag}) = 0.27 \cdot 0.999 \cdot 0.23 \cdot 0.999 \cdot 0.27 \cdot 0.999 \cdot 0.23 = 0.0038$$

Viterbi algorithm

Find the most probable state path $A = \arg\max_A \Pr(A, S)$

Subproblem $V[u,i]$: probability of the most probable state path generating $S_1 S_2 \ldots S_i$ and ending in state u



$V[u,i]$	a	c	a	g
				
				
				

Viterbi algorithm

Subproblem $V[u, i]$: probability of the most probable state path generating $S_1 S_2 \dots S_i$ and ending in state u

Recurrence?

$V[u, 1] =$

$V[u, i] =$

Recall notation:

Sequence $S = S_1, \dots, S_n$, annotation $A = A_1, \dots, A_n$

Transition probability $a(u, v) = \Pr(A_{i+1} = v | A_i = u)$,

Emission probability $e(u, x) = \Pr(S_i = x | A_i = u)$,

Starting probability $\pi(u) = \Pr(A_1 = u)$.

$\Pr(A, S) = \pi(A_1) e(A_1, S_1) \prod_{i=2}^{n} a(A_{i-1}, A_i) e(A_i, S_i)$

Viterbi algorithm

Subproblem $V[u,i]$: probability of the most probable state path generating $S_1 S_2 \ldots S_i$ and ending in state u

Recurrence:

$V[u,1] = \pi_u \cdot e_{u,S_1}$

$V[u,i] = \max_w V[w,i-1] \cdot a_{w,u} \cdot e_{u,S_i}$

Algorithm, final answer, running time?

Recall notation:

Sequence $S = S_1, \ldots, S_n$, annotation $A = A_1, \ldots, A_n$

Transition probability $a(u,v) = \Pr(A_{i+1} = v | A_i = u)$,

Emission probability $e(u,x) = \Pr(S_i = x | A_i = u)$,

Starting probability $\pi(u) = \Pr(A_1 = u)$.

$\Pr(A,S) = \pi(A_1) e(A_1, S_1) \prod_{i=2}^{n} a(A_{i-1}, A_i) e(A_i, S_i)$

Viterbi algorithm (overview)

Goal: Find the most probable state path $A = \arg\max_A \Pr(A, S)$

Subproblem $V[u, i]$: probability of the most probable state path generating $S_1 S_2 \dots S_i$ and ending in state u

Recurrence:

$V[u, 1] = \pi_u \cdot e_{u,S_1}$

$V[u, i] = \max_w V[w, i-1] \cdot a_{w,u} \cdot e_{u,S_i}$

Algorithm:

Initialize $V[*, 1]$

for $i = 2 \dots n$ (n=length of S)

  for $u = 1 \dots m$ (m =number of states)

    compute $V[u, i]$, keep best w in $B[u, i]$

Maximum $V[u, n]$ over all u is $\max_A \Pr(A, S)$

Retrieve the full path using matrix B

Dynamic programming in $O(nm^2)$ time

Second problem: overall probability of S

Viterbi computes $\arg\max_A \Pr(A, S)$

Now we want $\Pr(S) = \sum_A \Pr(A, S)$

Usefull e.g. to compare different models, which is more likely to produce S

Any ideas?

Recall our example:

$$\Pr(\text{acag}) = 0.27 \cdot 0.001 \cdot 0.27 \cdot 0.99 \cdot 0.24 \cdot 0.99 \cdot 0.28 = 4.8 \cdot 10^{-6}$$

$$\Pr(\text{acag}) = 0.27 \cdot 0.999 \cdot 0.23 \cdot 0.999 \cdot 0.27 \cdot 0.999 \cdot 0.23 = 0.0038$$

Forward algorithm (dopredný algoritmus)

Computes overall probability that the model emits S $\Pr(S) = \sum_A \Pr(A, S)$

Subproblem $F[u, i]$: probability that in i steps we generate $S_1, S_2, \ldots S_i$ and end in state u.

$F[u, i] = \Pr(A_i = u \wedge S_1, S_2, \ldots, S_i) =$
$\sum_{A_1,\ldots,A_{i-1},A_i=u} \Pr(A_1, A_2, ..., A_i \wedge S_1, S_2, ..., S_i)$

Recurrence?

$F[u, 1] =$

$F[u, i] =$

Recall Viterbi recurrence:

$V[u, 1] = \pi_u \cdot e_{u,S_1}$

$V[u, i] = \max_w V[w, i-1] \cdot a_{w,u} \cdot e_{u,S_i}$

Forward algorithm

Computes overall probability that the model emits S $\Pr(S) = \sum_A \Pr(A, S)$

Subproblem $F[u, i]$: probability that in i steps we generate $S_1, S_2, \ldots S_i$ and end in state u.

Recurrence:

$$F[u, 1] = \pi_u \cdot e_{u,S_1}$$
$$F[u, i] = \sum_w F[w, i-1] \cdot a_{w,u} \cdot e_{u,S_i}$$

Recall Viterbi recurrence:

$$V[u, 1] = \pi_u \cdot e_{u,S_1}$$
$$V[u, i] = \max_w V[w, i-1] \cdot a_{w,u} \cdot e_{u,S_i}$$

Forward algorithm

Computes overall probability that the model emits S $\Pr(S) = \sum_A \Pr(A, S)$

Subproblem $F[u, i]$: probability that in i steps we generate $S_1, S_2, \ldots S_i$ and end in state u.

Recurrence:

$$F[u, 1] = \pi_u \cdot e_{u,S_1}$$
$$F[u, i] = \sum_w F[w, i-1] \cdot a_{w,u} \cdot e_{u,S_i}$$

Result? $\Pr(S) =$

Running time?

Forward algorithm

Computes overall probability that the model emits S $\Pr(S) = \sum_A \Pr(A, S)$

Subproblem $F[u, i]$: probability that in i steps we generate $S_1, S_2, \ldots S_i$ and end in state u.

Recurrence:

$$F[u, 1] = \pi_u \cdot e_{u,S_1}$$

$$F[u, i] = \sum_w F[w, i-1] \cdot a_{w,u} \cdot e_{u,S_i}$$

Result $\Pr(S) = \sum_u F[u, n]$

Running time $O(nm^2)$

Third problem: probability that S_i was generated in state u

$$\Pr(A_i = u \mid S) = \frac{\Pr(A_i = u, S)}{\Pr(S)}$$
$$\Pr(A_i = u, S) = \sum_{A : A_i = u} \Pr(A, S)$$

Compute this by a combination of forward and backward algorithms

$F[u, i]$: probability that in i steps we generate $S_1, S_2, \ldots S_i$ and end in state u.

$B[u, i]$: probability that if we start at u at position i, we will generate $S_{i+1} \ldots, S_n$ in the next steps

$$\Pr(A_i = u, S) = F[u, i] \cdot B[u, i]$$

Backward algorithm (spätný algoritmus)

Forward algorithm $F[u,i]$: probability that in i steps we generate $S_1, S_2, \ldots S_i$ and end in state u.

$$F[u,1] = \pi_u \cdot e_{u,S_1}$$
$$F[u,i] = \sum_w F[w,i-1] \cdot a_{w,u} \cdot e_{u,S_i}$$

Backward algorithm $B[u,i]$: probability that if we start at u at position i, we will generate $S_{i+1} \ldots, S_n$ in the next steps

How to compute $B[u,i]$?

Backward algorithm (spätný algoritmus)

Forward algorithm $F[u,i]$: probability that in i steps we generate $S_1, S_2, \ldots S_i$ and end in state u.

$$F[u,1] = \pi_u \cdot e_{u,S_1}$$
$$F[u,i] = \sum_w F[w,i-1] \cdot a_{w,u} \cdot e_{u,S_i}$$

Backward algorithm $B[u,i]$: probability that if we start at u at position i, we will generate $S_{i+1} \ldots, S_n$ in the next steps

$$B[u,n] = 1$$
$$B[u,i] = \sum_w B[w,i+1] \cdot a_{u,w} \cdot e_{w,S_{i+1}}$$

Exercise: How to use matrix B to compute $\Pr(S)$?

Posterior decoding

Using forward/backward we can compute

$\Pr(A_i = u \mid S)$ for each u and i (posterior probabilities of states)

in $O(nm^2)$ overall time

We can also select A such that $A_i = \max_i \Pr(A_i = u \mid S)$

Advantage: This takes into account suboptimal state paths

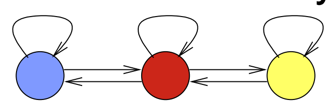
Disadvantage: $\Pr(A \mid S)$ can be zero or very low

Another option: use posterior probabilities to assign confidence to parts of prediction from Viterbi

Recall: Finding genes with HMMs

model (HMM) → random DNA sequence S, randor (similar to real DNA)

$\Pr(S, A)$ – probability that the model generates pair (S, A).

- **Determine states and transitions of the model:** by hand based on your knowledge about the gene structure 
- **Parameter training:** emission and transition probabilities are determined based on the real sequences with known genes (**training set**)
- **Use:** for a new sequence S, find the most probable annotation
  $A = \arg\max_A \Pr(A|S)$
  Viterbi algorithm in $O(nm^2)$ (dynamic programming)

Parameter training

- States and allowed transitions typically manually
- Probabilities of transition, emission, starting usually automatically from training adata
- More complex models with more parameters need more training data
  Otherwise **overfitting**: model fits training data very well but behaves poorly on unseen examples
- To test acurracy of the model use a separate testing set not used for training.

HMM parameter training from annotated sequences

Input: state diagram of the model and a training set of sequences and state paths $(S^{(1)}, A^{(1)}), (S^{(2)}, A^{(2)}), \ldots$

Goal: choose parameters maximalizing their likelihood in the model $\arg\max_{a,e,\pi} \prod_i \Pr(S^{(i)}, A^{(i)} | a, e, \pi)$

This is achieved by using observed frequencies

For example $a_{u,v}$: find all occurrences of state u and find out how often is it followed by v

HMM parameter training from unannotated sequences

Input: state diagram of the model and a training set of sequences $S^{(1)}, S^{(2)}, \ldots$, state paths $A^{(1)}$ unknown

Goal: choose parameters maximalizing their likelihood in the model $\arg\max_{a,e,\pi} \prod_i \Pr(S^{(i)}|a,e,\pi)$

Baum-Welch algorithm (version of expectation maximization, EM).

Iterative heuristic algorithm improving parameters until convergence.

Each iteration forward and backward algorithms