HMMs and applications

Notes from Dr. Takis Benos

and DEKM book

Markov chains

- What is a Markov chain?

Markov chain of order *n* is a stochastic process of a series of outcomes, in which the probability of outcome *x* depends on the state of the previous *n* outcomes.

Markov chains (cntd)

- Markov chain (of *first* order) and the *Chain Rule*

$$P(\vec{x}) = P(X_L, X_{L-1}, \ldots, X_1) =$$

$$= P(X_L \mid X_{L-1}, \ldots, X_1) P(X_{L-1}, X_{L-2}, \ldots, X_1) =$$

$$= P(X_L \mid X_{L-1}, \ldots, X_1) P(X_{L-1} \mid X_{L-2}, \ldots, X_1) \ldots P(X_1) =$$

$$= P(X_L \mid X_{L-1}) P(X_{L-1} \mid X_{L-2}) \ldots P(X_2 \mid X_1) P(X_1) =$$

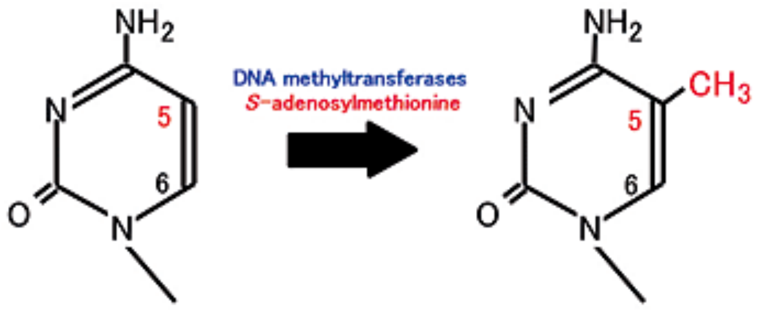
$$= P(X_1) \prod_{i=2}^{L} P(X_i \mid X_{i-1})$$

Chain rule: *P(A,B,C)=P(C|A,B) P(B|A) P(A)*

Application of Markov chains: CpG islands

- CG is relatively rare in the genome due to high mutation of methyl-CG to methyl-TG (or CA)
- Methylated CpG residues are often associated with house-keeping genes in the promoter and exon regions.
- Methyl-CpG binding proteins recruit histone deacetylases and are thus responsible for transcriptional repression.
- They have roles in gene silencing, genomic imprinting, and X-chromosome inactivation.

Methylation of Cytosine



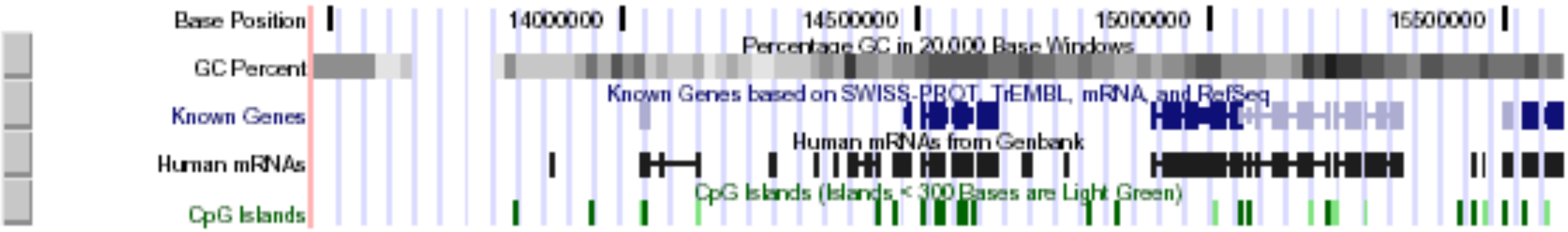
CpG islands and DNA Methylation

Largely confined to CpG dinucleotides

CpG islands - regions of more than 500 bp with CG content > 55%

denoted CpG to not confuse with CG base pair

Methylation often suppressed around genes, promoters


Base Position
14000000
14500000
15000000
15500000
Percentage GC in 20,000 Base Windows
GC Percent
Known Genes based on SWISS-PROT, TrEMBL, mRNA, and RefSeq
Known Genes
Human mRNAs from Genbank
Human mRNAs
CpG Islands (Islands < 300 Bases are Light Green)
CpG Islands

Can we predict CpG islands? – a good way of identifying potential gene regions as well! – But not so fast!!

Application of Markov chains: CpG islands

- Problem:

  Given two sets of sequences from the human genome, one with CpG islands and one without, can we calculate a model that can predict the CpG islands?

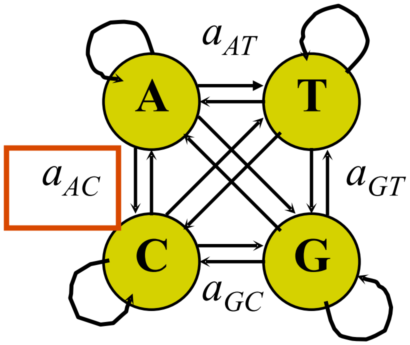
Sequence: $s = t \cdot t \cdot a \cdot c \cdot g \cdot g \cdot t$

0th-order: $P_0(s) = \mathrm{p}(t) \cdot \mathrm{p}(t) \cdot \mathrm{p}(a) \cdot \mathrm{p}(c) \cdot \mathrm{p}(g) \cdots = \prod_{i=1}^{N} \mathrm{p}(s_i)$

1st-order: $P_1(s) = \mathrm{p}(t) \cdot \mathrm{p}(t \mid t) \cdot \mathrm{p}(a \mid t) \cdot \mathrm{p}(c \mid a) \cdots = \mathrm{p}(s_1) \cdot \prod_{i=2}^{N} \mathrm{p}(s_i \mid s_{i-1})$

2nd-order: $P_2(s) = \mathrm{p}(tt) \cdot \mathrm{p}(a \mid tt) \cdot \mathrm{p}(c \mid ta) \cdot \mathrm{p}(g \mid ac) \cdots = \mathrm{p}(s_1 s_2) \cdot \prod_{i=3}^{N} \mathrm{p}(s_i \mid s_{i-2} s_{i-1})$

Application of Markov chains: CpG islands (cntd)



- A state for each of the four letters A,C, G, and T in the DNA alphabet
- ⟷: probability of a residue following another residue

+	A	C	G	T
A	.180	.274	.426	.120
C	.171	.368	.274	.188
G	.161	.339	.375	.125
T	.079	.355	.384	.182

Training Set:

- set of DNA sequences w/ known CpG islands

Derive two Markov chain models:

- '+' **model**: from the CpG islands
- '-' **model**: from the remainder of sequence

Transition probabilities for each model:

$$a^+_{st} = \frac{c^+_{st}}{\sum_{t'} c^+_{st'}}$$

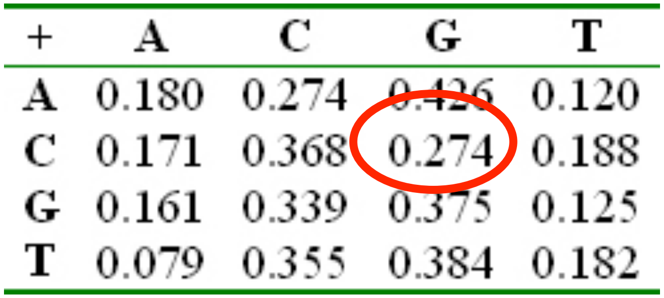
c^+_{st} is the number of times letter ***t*** followed letter ***s*** in the CpG islands

To use these models for discrimination, calculate the log-odds ratio:

$$S(x) = \log \frac{P(x|\text{model}+)}{P(x|\text{model}-)} = \sum_{i=1}^{L} \log \frac{a^+_{x_{i-1}x_i}}{a^-_{x_{i-1}x_i}}$$

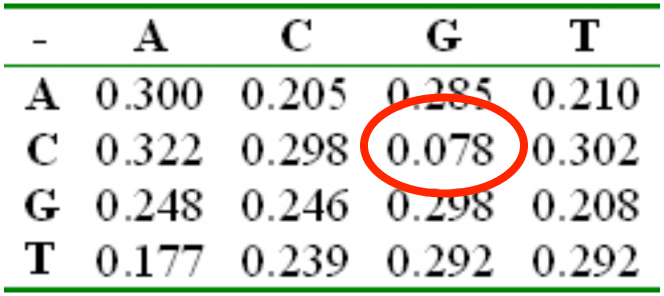
Application of Markov chains: CpG islands (cntd)

P(*t* | *s*,+)



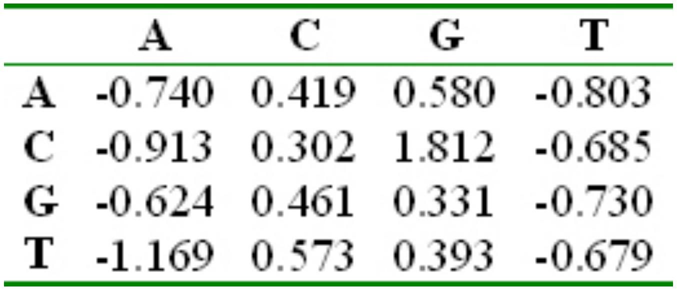
+	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182

P(*t* | *s*,-)



-	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

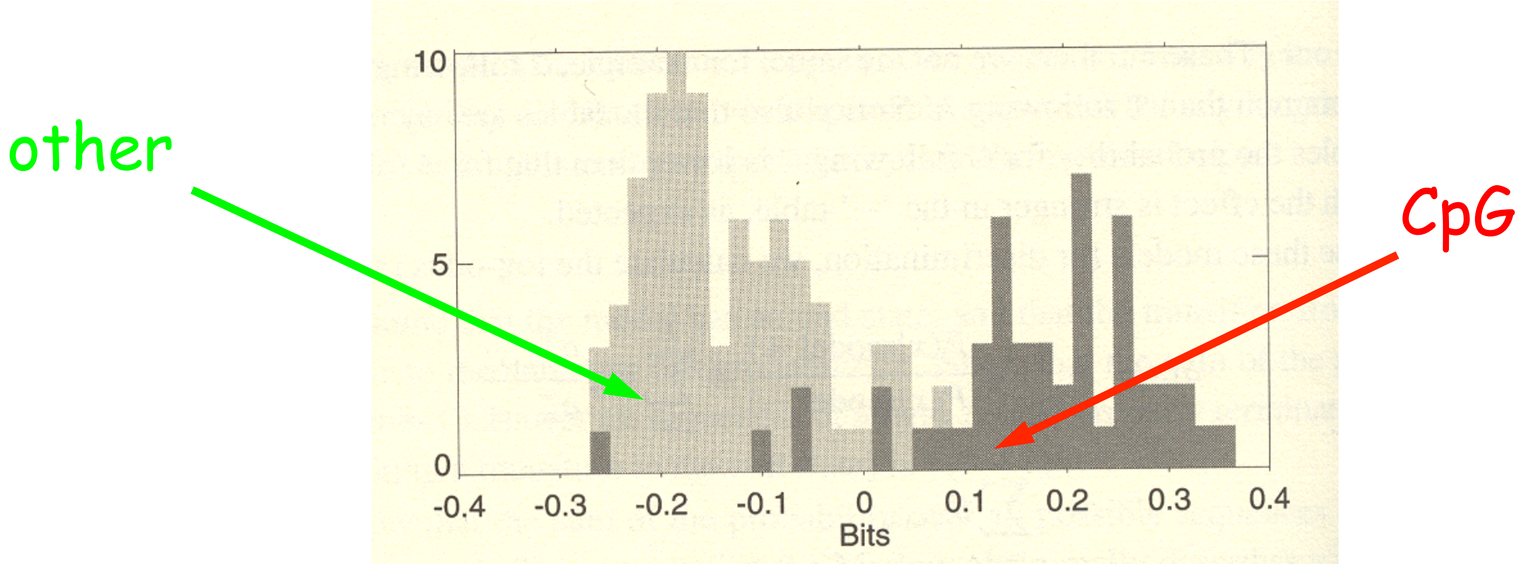
$\log_2(P(t\,|\,s,+)\,/\,P(t\,|\,s,-))$



	A	C	G	T
A	-0.740	0.419	0.580	-0.803
C	-0.913	0.302	1.812	-0.685
G	-0.624	0.461	0.331	-0.730
T	-1.169	0.573	0.393	-0.679

$$\boxed{\log_2 \frac{P(\vec{x}\,|+)}{P(\vec{x}\,|-)} = \sum_{i=1}^{L} \log_2 \frac{P(x_{i+1}\,|\,x_i,+)}{P(x_{i+1}\,|\,x_i,-)}}$$

Histogram of log-odd scores



other

CpG

Q1: Given a short sequence *x*, does it come from CpG island? (**Yes-No** question)

- Evaluate *S(x)*

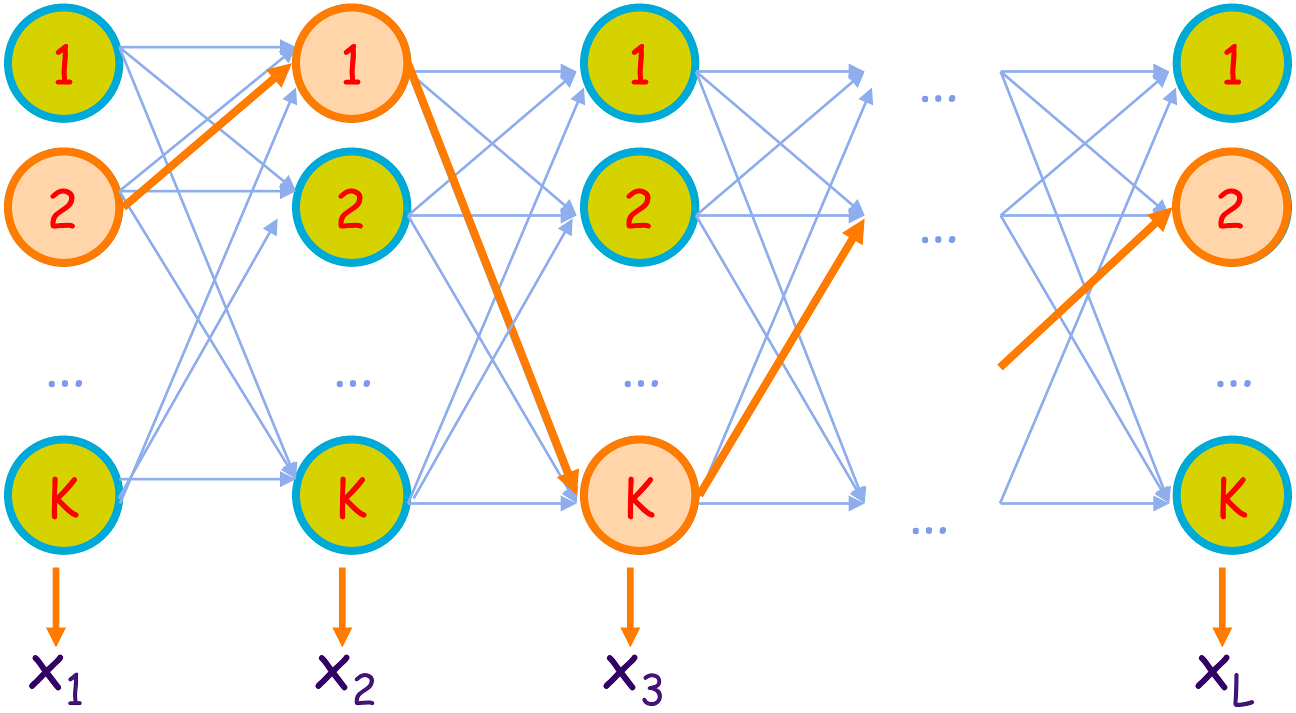
Q2: Given a long sequence *x*, how do we find CpG islands in it (**Where** question)?

- Calculate the log-odds score for a window of, say, 100 nucleotides around every nucleotide, plot it, and predict CpG islands as ones w/ positive values
- Drawbacks: Window size?

HMM: A parse of a sequence

Given a sequence $x = x_1 \ldots\ldots x_L$, and a HMM with K states,

A parse of x is a sequence of states $\pi = \pi_1, \ldots\ldots, \pi_L$



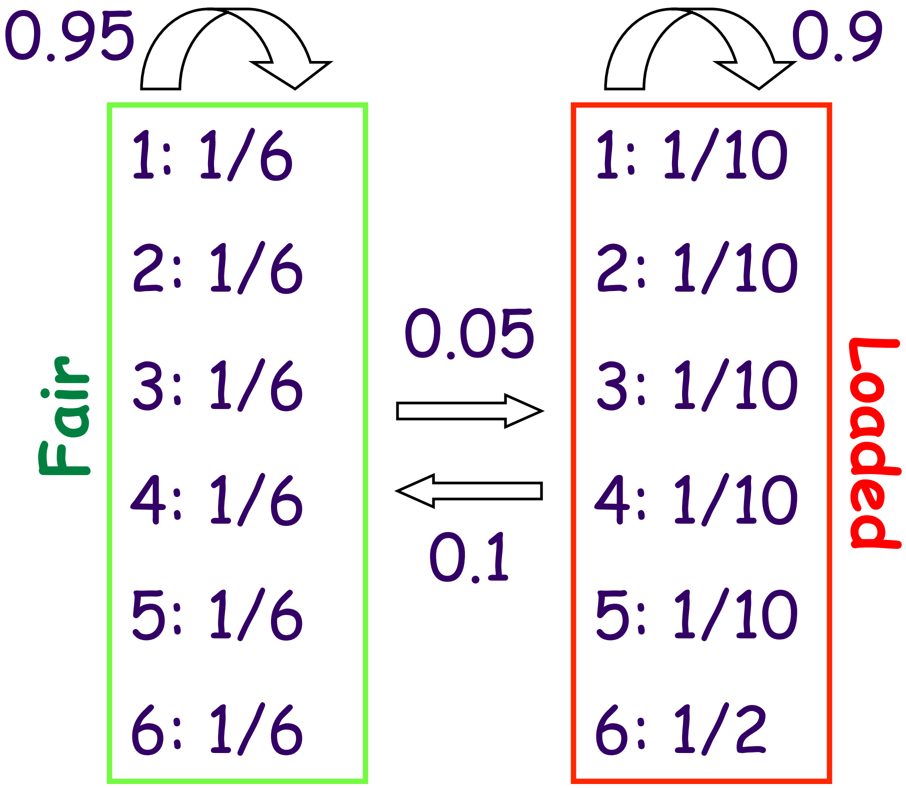
Hidden Markov Models (HMMs)

- What is a HMM?

  A Markov process in which the probability of an outcome depends also in a (hidden) random variable (*state*).

- Memory-less: future states affected only by current state
- We need:
  - ✓ Ω : alphabet of symbols (outcomes)
  - ✓ $\int$: set of states (hidden), each of which emits symbols
  - ✓ $A = (a_{kl})$: matrix of state transition probabilities
  - ✓ $E = (e_k(b)) = (P(x_i=b|\pi=k))$: matrix of emission probabilities

Example: the dishonest casino



0.95

Fair

1: 1/6

2: 1/6

3: 1/6

4: 1/6

5: 1/6

6: 1/6

0.05

0.1

0.9

Loaded

1: 1/10

2: 1/10

3: 1/10

4: 1/10

5: 1/10

6: 1/2

- ✓ Ω = {1, 2, 3, 4, 5, 6}
- ✓ $\mathcal{S}$ = {F, L}
- ✓ A : a_{FF}=0.95, a_{LL}=0.9,

  a_{FL}=0.05, a_{LF}=0.1
- ✓ E : $e_F(b)$=1/6 ($\forall\ b \in \Omega$),

  e_L("6")=1/2

  $e_L(b)$=1/10 (if $b \neq 6$)

Three main questions on HMMs

1. **Evaluation problem**

   GIVEN HMM M, sequence x
   FIND $P(x \mid M)$
   ALGOR. Forward $O(TN^2)$

2. **Decoding problem**

   GIVEN HMM M, sequence x
   FIND the sequence π of states that maximizes $P(\pi \mid x, M)$
   ALGOR. Viterbi, Forward-Backward $O(TN^2)$

3. **Learning problem**

   GIVEN HMM M, with unknown prob. parameters, sequence x
   FIND parameters $\theta = (\pi, e_{ij}, a_{kl})$ that maximize $P(x \mid \theta, M)$
   ALGOR. Maximum likelihood (ML), Baum-Welch (EM) $O(TN^2)$

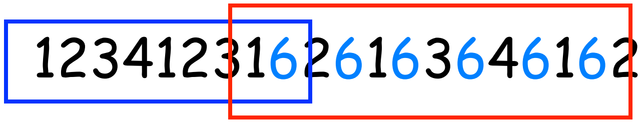
Problem 1: Evaluation

Find the likelihood a given sequence is generated by a particular model

E.g. Given the following sequence is it more likely that it comes from a ***Loaded*** or a ***Fair*** die?

123412316261636461623411221341

Problem 1: Evaluation (cntd)



123412316261636461623411221341

$$P(Data \mid F_1 ... F_{30}) = \prod_{i=1}^{30} a_{F,F} \cdot e_F(b_i) =$$

$$= 0.95^{29} \cdot (1/6)^{30} = 0.226 \cdot 4.52 \cdot 10^{-24} =$$

$$= 1.02 \cdot 10^{-24}$$

$$P(Data \mid L_1 ... L_{30}) = \prod_{i=1}^{30} a_{L,L} \cdot e_L(b_i) =$$

$$= (1/2)^{6} \cdot (1/10)^{24} \cdot 0.90^{29} = 1.56 \cdot 10^{-26} \cdot 0.047 =$$

$$= 7.36 \cdot 10^{-28}$$

What happens in a sliding window?

Three main questions on HMMs

✓ Evaluation problem

GIVEN HMM *M*, sequence *x*
FIND *P(x | M)*
ALGOR. Forward

1. **Decoding problem**

GIVEN HMM *M*, sequence *x*
FIND the sequence π of states that maximizes $P(\pi \mid x, M)$
ALGOR. Viterbi, Forward-Backward O(TN^2)

2. Learning problem

GIVEN HMM *M*, with unknown prob. parameters, sequence *x*
FIND parameters $\theta = (\pi, e_{ij}, a_{kl})$ that maximize $P(x \mid \theta, M)$
ALGOR. Maximum likelihood (ML), Baum-Welch (EM) O(TN^2)

Problem 2: Decoding

Given a point x_i in a sequence find its most probable state

E.g. Given the following sequence is it more likely that the 3rd observed "6" comes from a *Loaded* or a *Fair* die?

123412316261636461623411221341

The Forward Algorithm - derivation

- In order to calculate P(x_i) = probability of x_i, given the HMM, we need to sum over all possible ways of generating x_i:

$$P(x_i) = \sum_{\pi} P(x_i, \pi) = \sum_{\pi} P(x_i \mid \pi) \cdot P(\pi)$$

- To avoid summing over an exponential number of paths π, we first define the *forward probability*:

$$f_k(i) = P(x_1 \ldots x_i, \pi_i = k)$$

The Forward Algorithm – derivation (cntd)

- Then, we need to write the $f_k(i)$ as a function of the previous state, $f_l(i\text{-}1)$.

$$
\begin{aligned}
f_k(i) &= P(x_1,\ldots,x_{i-1},x_i,\pi_i = k) \\
&= \sum_{\pi_1,\ldots,\pi_{i-1}} P(x_1,\ldots,x_{i-1},\pi_1,\ldots,\pi_{i-1},\pi_i = k)\cdot e_k(x_i) \\
&= \sum_l \left(\underbrace{\sum_{\pi_1,\ldots,\pi_{i-2}} P(x_1,\ldots,x_{i-1},\pi_1,\ldots,\pi_{i-2},\pi_{i-1} = l)} \cdot a_{l,k} \right)\cdot e_k(x_i) \\
&= \sum_l P(x_1,\ldots,x_{i-1},\pi_{i-1} = l)\cdot a_{l,k}\cdot e_k(x_i) \\
&= e_k(x_i)\cdot \sum_l f_l(i-1)\cdot a_{l,k}
\end{aligned}
$$

Chain rule: *P(A,B,C)=P(C|A,B) P(B|A) P(A)*

The Forward Algorithm

We can compute *f_k(i)* for all *k*, *i*, using dynamic programming

Initialization:

$$f_0(0) = 1$$

$$f_k(0) = 0, \quad \forall k > 0$$

Iteration:

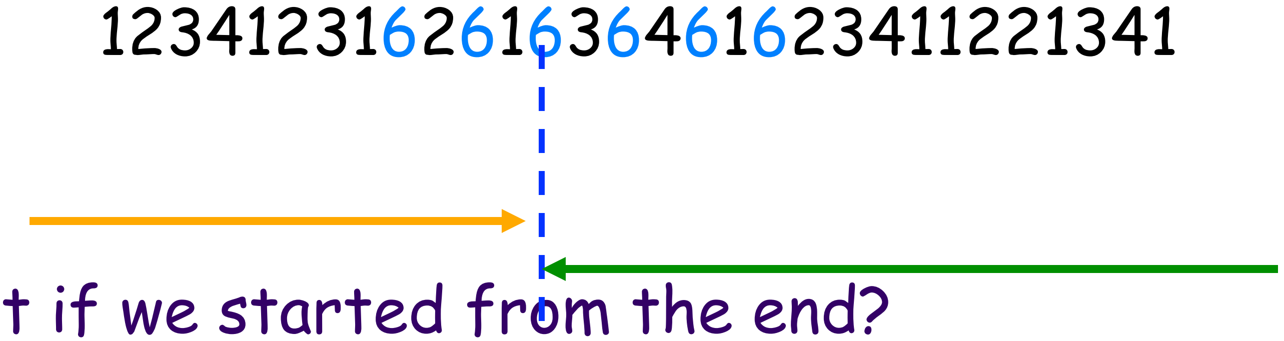
$$f_k(i) = e_k(x_i) \cdot \sum_l f_l(i-1) \cdot a_{l,k}$$

Termination:

$$P(\vec{x}) = \sum_k f_k(N) \cdot a_{k,0}$$

The Backward Algorithm

- Forward algorithm determines the most likely state *k* at position *i*, using the *previous* observations.



123412316261636461623411221341

- What if we started from the end?

The Backward Algorithm – derivation

- We define the *backward probability*:

$$\boxed{b_k(i) = P(x_{i+1},\ldots,x_N \mid \pi_i = k)}$$

$$= \sum_{\pi_{i+1},\ldots,\pi_N} P(x_{i+1},\ldots,x_N,\pi_{i+1},\ldots,\pi_N \mid \pi_i = k)$$

$$= \sum_l \sum_{\pi_{i+1},\ldots,\pi_N} P(x_{i+1},\ldots,x_N,\pi_{i+1} = l,\pi_{i+2},\ldots,\pi_N \mid \pi_i = k)$$

$$= \sum_l e_k(x_{i+1}) \cdot a_{k,l} \cdot \sum_{\pi_{i+2},\ldots,\pi_N} P(x_{i+2},\ldots,x_N,\pi_{i+2},\ldots,\pi_N \mid \pi_{i+i} = l)$$

$$= \sum_l b_l(i+1) \cdot a_{k,l} \cdot e_l(x_{i+1})$$

Chain rule: *P(A,B,C)=P(C|A,B) P(B|A) P(A)*

The Backward Algorithm

We can compute $b_k(i)$ for all k, i, using dynamic programming

Initialization: $$b_k(N) = a_{k,0}, \quad \forall k$$

Iteration: $$b_k(i) = \sum_l e_k(x_{i+1}) \cdot b_l(i+1) \cdot a_{k,l}$$

Termination: $$P(\vec{x}) = \sum_k b_k(1) \cdot a_{0,k} \cdot e_k(x_1)$$

Posterior probabilities of the dishonest casino data

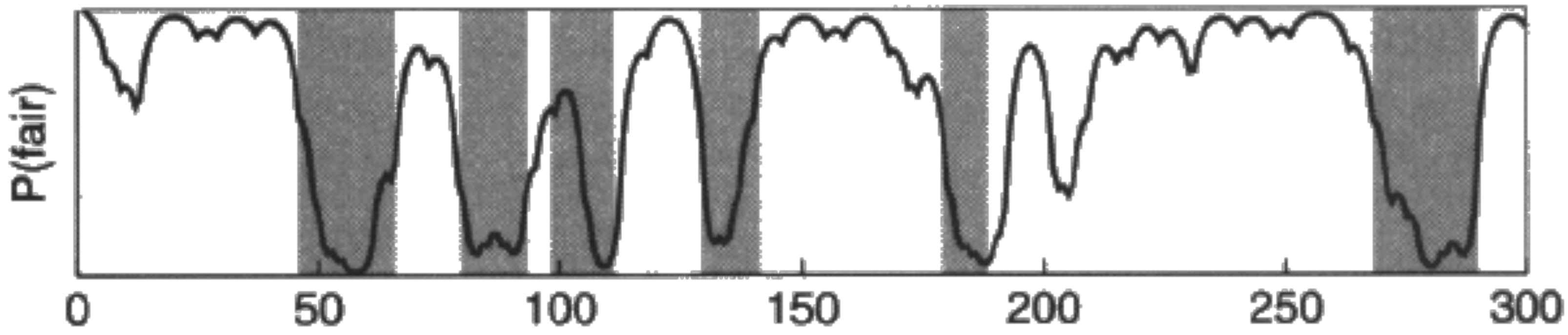
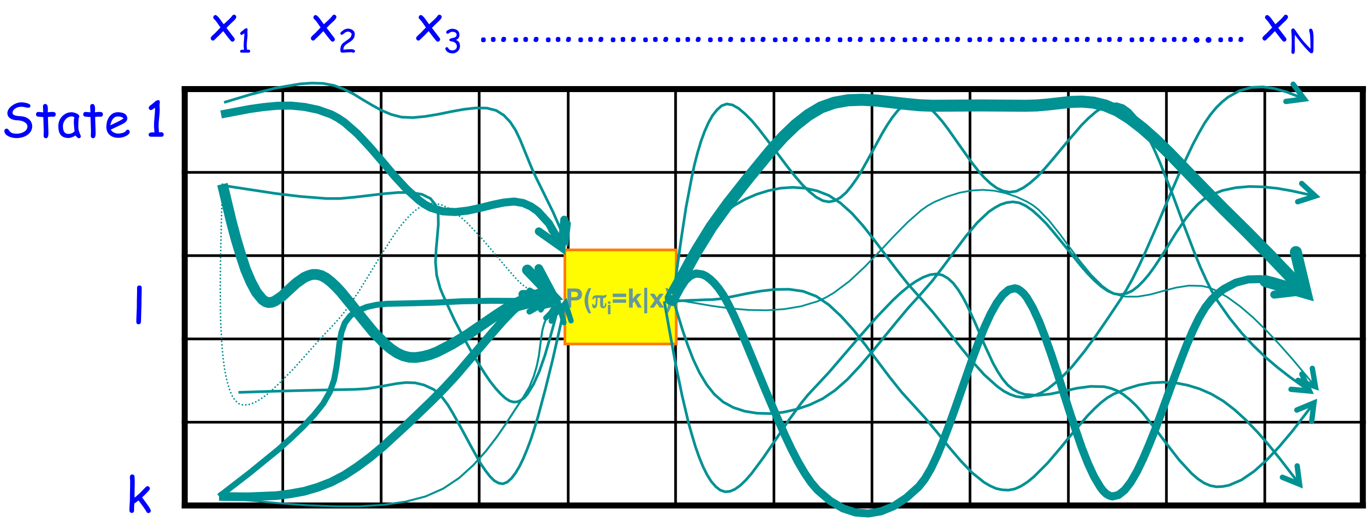


Figure 3.6 *The posterior probability of being in the state corresponding to the fair die in the casino example. The x axis shows the number of the roll. The shaded areas show when the roll was generated by the loaded die.*

Posterior Decoding



- *Posterior decoding* calculates the optimal path that explains the data.

- For each emitted symbol, *x_i*, it finds the most likely state that could produce it, based on the *forward* and *backward* probabilities.

The Viterbi Algorithm – derivation

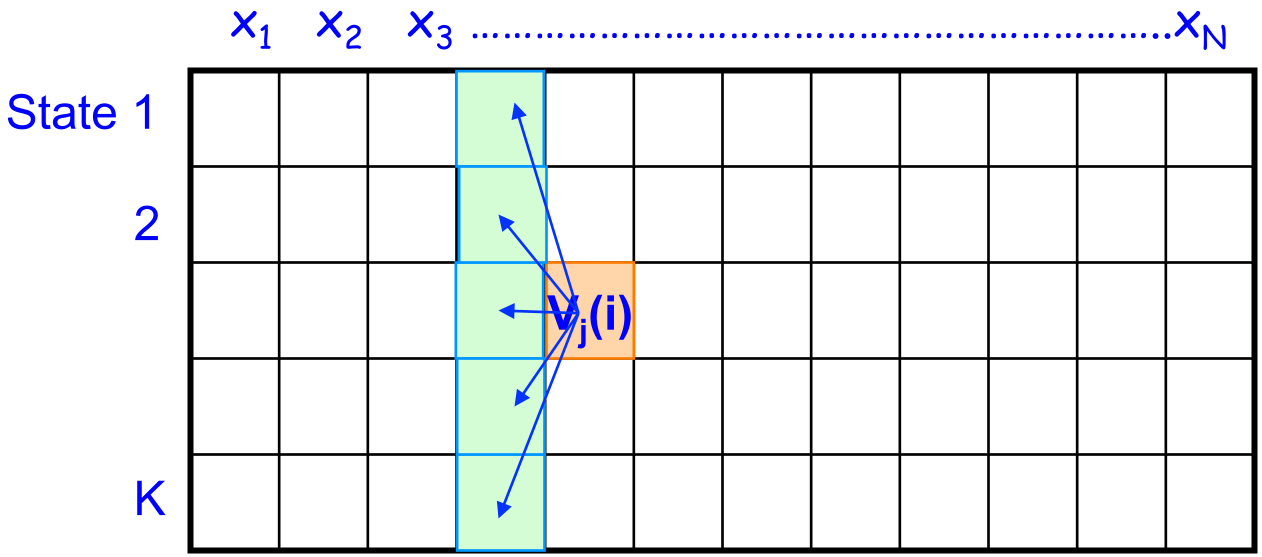
- We define:

$$V_k(i) = \max_{\{\pi_1,\ldots,\pi_{i-1}\}} P(x_1,\ldots,x_{i-1},\pi_1,\ldots,\pi_{i-1},\pi_i = k)$$

- Then, we need to write the $V_k(i)$ as a function of the previous state, $V_l(i\text{-}1)$.

$$V_k(i) = \ldots = e_k(x_i)\cdot \max_l\{a_{l,k}\cdot V_l(i-1)\}$$

The Viterbi Algorithm

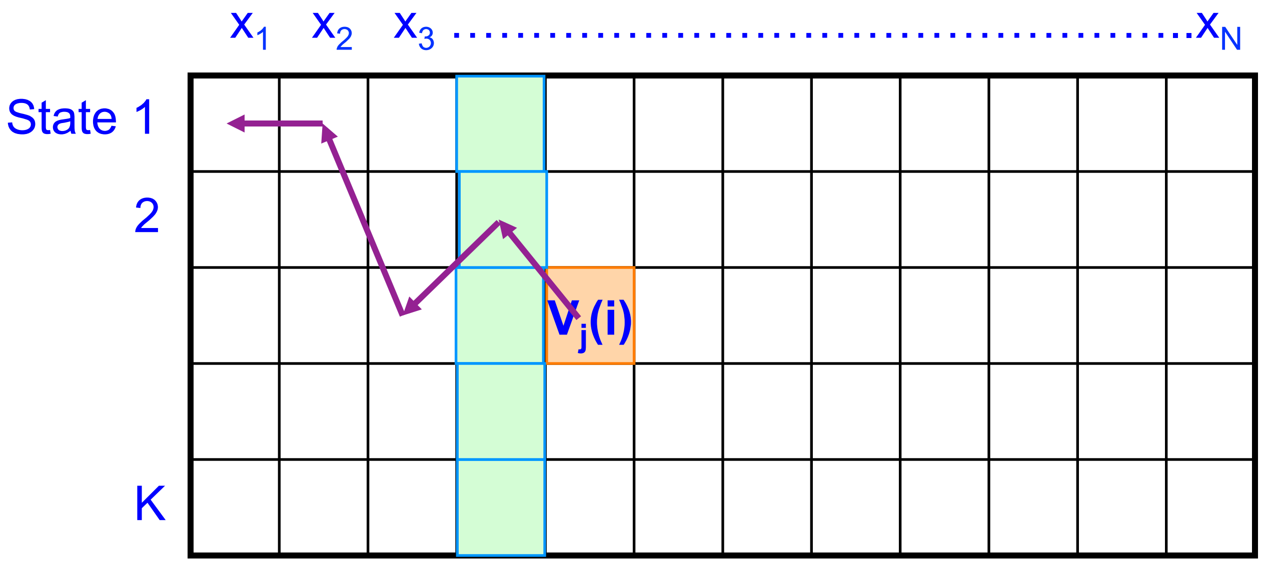


Similar to "aligning" a set of states to a sequence

Dynamic programming!

Viterbi decoding: traceback

The Viterbi Algorithm

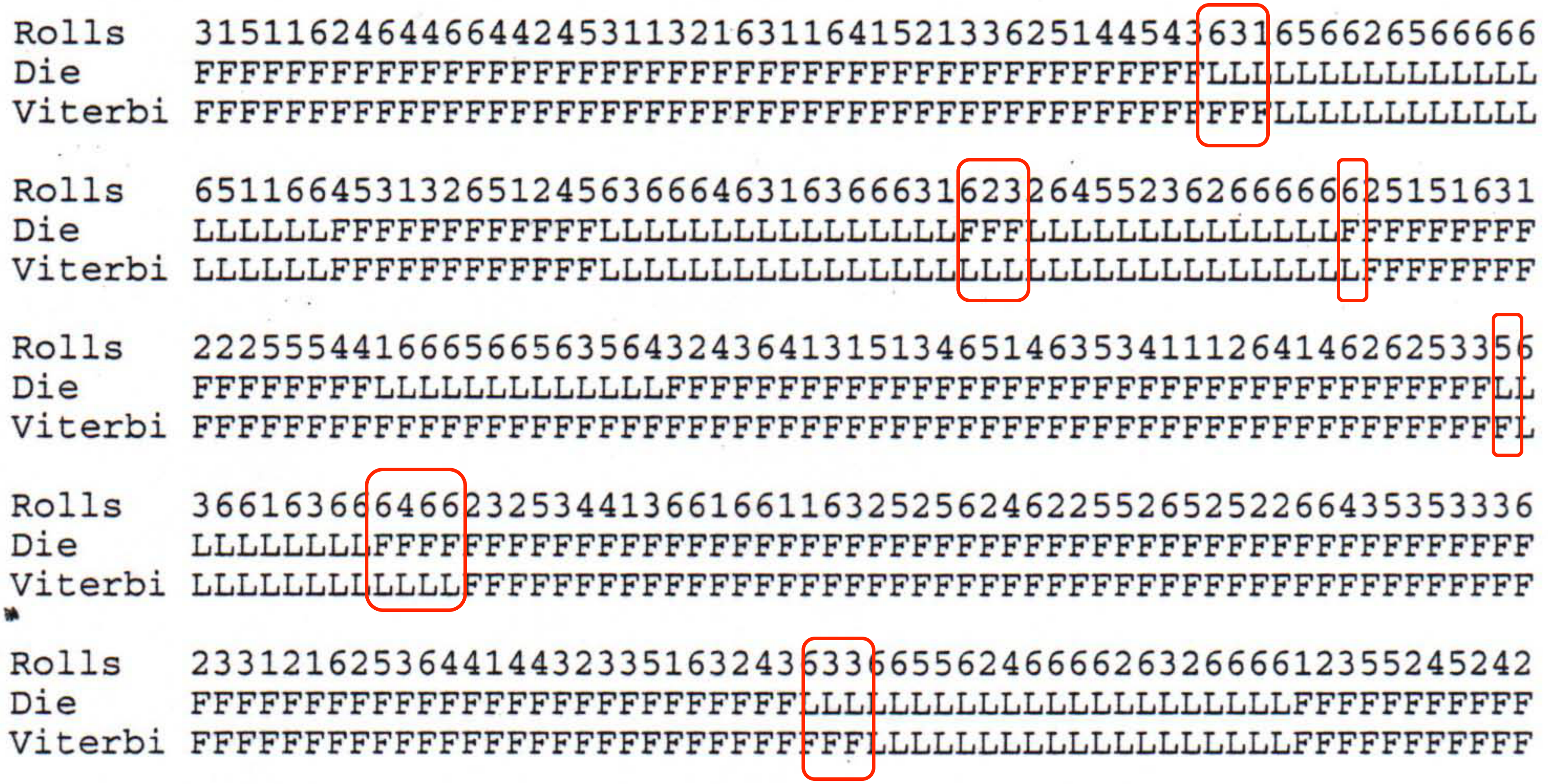


Similar to "aligning" a set of states to a sequence

Dynamic programming!

Viterbi decoding: traceback

Viterbi results



```
Rolls   315116246446644245311321631164152133625144543631656626566666
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLL

Rolls   651166453132651245636664631636663162326455236266666625151631
Die     LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLFFFFFFFFF
Viterbi LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF

Rolls   222555441666566563564324364131513465146353411126414626253356
Die     FFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls   366163666466232534413661661163252562462255265252266435353336
Die     LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls   233121625364414432335163243633665562466662632666612355245242
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFF
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLFFFFFFFFFF
```

Viterbi, Forward, Backward

VITERBI	FORWARD	BACKWARD
Initialization: $V_0(0) = 1$ $V_k(0) = 0$, for all $k > 0$	**Initialization:** $f_0(0) = 1$ $f_k(0) = 0$, for all $k > 0$	**Initialization:** $b_k(N) = a_{k0}$, for all k
Iteration: $V_l(i) = e_l(x_i)\ \mathbf{max}_k\ V_k(i-1)\ a_{kl}$	**Iteration:** $f_l(i) = e_l(x_i)\ \Sigma_k\ f_k(i-1)\ a_{kl}$	**Iteration:** $b_l(i) = \Sigma_k\ e_l(x_i+1)\ a_{kl}\ b_k(i+1)$
Termination: $P(x, \pi^*) = \mathbf{max}_k\ V_k(N)$	**Termination:** $P(x) = \Sigma_k\ f_k(N)\ a_{k0}$	**Termination:** $P(x) = \Sigma_k\ a_{0k}\ e_k(x_1)\ b_k(1)$
Time: $O(K^2N)$ Space: $O(KN)$	Time: $O(K^2N)$	Space: $O(KN)$

Three main questions on HMMs

✓ Evaluation problem

GIVEN HMM *M*, sequence *x*
FIND *P(x | M)*
ALGOR. Forward

✓ Decoding problem

GIVEN HMM *M*, sequence *x*
FIND the sequence π of states that maximizes $P(\pi \mid x, M)$
ALGOR. Viterbi, Forward-Backward

3. Learning

GIVEN HMM *M*, with unknown prob. parameters, sequence *x*
FIND parameters $\theta = (\pi, e_{ij}, a_{kl})$ that maximize $P(x \mid \theta, M)$
ALGOR. Maximum likelihood (ML), Baum-Welch (EM)

Problem 3: Learning

Given a model (structure) and data, calculate model's parameters

Two scenarios:

- Labeled data - Supervised learning

12341231	62616364616	23411221341
Fair	**Loaded**	**Fair**

- Unlabeled data - Unsupervised learning

123412316261636461623411221341

Two learning scenarios - examples

1. Supervised learning

   Examples:

   GIVEN: the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls

   GIVEN: a genomic region $x = x_1 \ldots x_{1,000,000}$ where we have good (experimental) annotations of the CpG islands

2. Unsupervised learning

   Examples:

   GIVEN: 10,000 rolls of the casino player, but we don't see when he changes dice

   GIVEN: a newly sequenced genome; we don't know how frequent are the CpG islands there, neither do we know their composition

TARGET: Update the parameters θ of the model to maximize $P(x|\theta)$

Supervised learning

- Given $x = x_1 \dots x_N$ for which the true state path $\pi = \pi_1 \dots \pi_N$ is known
  - Define:

    $A_{k,l}$ = # times state transition $k \rightarrow l$ occurs in π

    $E_k(b)$ = # times state k in π emits b in x

  - The maximum likelihood parameters θ *are:*

$$a_{k,l}^{ML} = \frac{A_{k,l}}{\sum_i A_{k,i}} \qquad e_k^{ML}(b) = \frac{E_k(b)}{\sum_c E_k(c)}$$

- Problem: overfitting (when training set is small for the model)

Overfitting

- **Example**
  - Given 10 casino rolls, we observe

$$\begin{aligned} \mathbf{x} &= 2,\ 1,\ 5,\ 6,\ 1,\ 2,\ 3,\ 6,\ 2,\ 3 \\ \pi &= F,\ F,\ F,\ F,\ F,\ F,\ F,\ F,\ F,\ F \end{aligned}$$

  - Then:

    $a_{FF} = 10/10 = 1.00$; $a_{FL} = 0/10 = 0$

    $e_F(1) = e_F(3) = 2/10 = 0.2$;

    $e_F(2) = 3/10 = 0.3$; $e_F(4) = 0/10 = 0$; $e_F(5) = e_F(6) = 1/10 = 0.1$

- **Solution:** add *pseudocounts*
  - Larger pseudocounts ⇒ strong prior belief (need a lot of data to change)
  - Smaller pseudocounts ⇒ just smoothing (to avoid zero probabilities)

Overfitting

- Example
  - Given 10 casino rolls, we observe

    x = 2, 1, 5, 6, 1, 2, 3, 6, 2, 3

    π = F, F, F, F, F, F, F, F, F, F

  - Then:

    a_{FF} = 11/12 = 0.92; a_{FL} = 1/12 = 0.08

    $e_F(1) = e_F(3)$ = 3/16 = 0.1875;

    $e_F(2)$ = 4/16 = 0.25; $e_F(4)$ = 1/16 = 0.0625; $e_F(5) = e_F(6)$ = 2/16 = 0.125

- Solution: add *pseudocounts*
  - Larger pseudocounts ⇒ strong prior belief (need a lot of data to change)
  - Smaller pseudocounts ⇒ just smoothing (to avoid zero probabilities)

Unsupervised learning - ML

- Given $x = x_1 \ldots x_N$ for which the true state path $\pi = \pi_1 \ldots \pi_N$ is unknown

  - **EXPECTATION MAXIMIZATION (EM) in a nutshell**

  0. Initialize the parameters θ of the model M
  1. Calculate the *expected* values of $A_{k,l}$, $E_k(b)$ based on the training data and current parameters
  2. Update θ according to $A_{k,l}$, $E_k(b)$ as in supervised learning
  3. Repeat #1 & #2 until convergence

  - In HMM training, we usually apply a special case of EM, called Baum-Welch Algorithm

The Baum-Welch (EM) algorithm simply put

- Recurrence:
  1. Estimate $A_{k,l}$ and $E_k(b)$ from $a_{k,l}$ and $e_k(b)$ overall all training sequences (E-step)
  2. Update $a_{k,l}$ and $e_k(b)$ using ML (M-step)
  3. Repeat steps #1, #2 with new parameters $a_{k,l}$ and $e_k(b)$

- Initialization:
  - Set A and E to pseudocounts (or priors)

- Termination: if *Δlog-likelihood < threshold* or Ntimes>max_times

The Baum-Welch algorithm

- Recurrence:

1. Calculate forward/backwards probs, $f_k(i)$ and $b_k(i)$, for each training sequence

2. E-step: Estimate the expected number of k→l transitions, $A_{k,l}$

$$A_{k,l} = \sum_i f_k(i) \cdot a_{k,l} \cdot e_l(x_{i+1}) \cdot b_l(i+1) / P(\vec{x} \mid \theta)$$

and the expected number of symbol b appearences in state k, $E_k(b)$

$$E_k(b) = \sum_{\{i \mid x_i = b\}} f_k(i) \cdot b_k(i) / P(\vec{x} \mid \theta)$$

3. M-step: Estimate new model parameters $a_{k,l}$ and $e_k(b)$ using ML across all training sequences
4. Estimate the new model's (log)likelihood to assess convergence

The Baum-Welch algorithm (cntd)

- Initialization: pick arbitrary model parameters
  - Set *A* and *E* to pseudocounts (or priors)
- Termination: if *Δlog-likelihood < threshold* or Ntimes>max_times

The Baum-Welch algorithm:

- is monotone
- guarantees convergence
- is a special case of EM
- has many local optima

An example of Baum-Welch
(thanks to Sarah Wheelan, JHU)

- I observe the dog across the street. Sometimes he is inside, sometimes outside.
- I assume that since he can not open the door himself, then there is another factor, hidden from me, that determines his behavior.
- Since I am lazy, I will guess there are only two hidden states, S_1 and S_2.

An example of Baum-Welch (cntd)

- One set of observations:
  - I-I-I-I-I-O-O-I-I-I
- Guessing two hidden states. I need to invent a transition and emission matrix.
  - Note: since Baum-Welch is an EM algorithm the better my initial guesses are the better the job I will do in estimating the true parameters

Day *k+1*

Day *k*

	S1	S2
S1	0.5	0.5
S2	0.4	0.6

	IN	OUT
S1	0.2	0.8
S2	0.9	0.1

An example of Baum-Welch (cntd)

- Let's assume initial values of:
  - $P(S_1) = 0.3$, $P(S_2) = 0.7$

- Example guess: if initial I-I came from S_1-S_2 then the probability is:
  $0.3 \times 0.2 \times 0.5 \times 0.9 = 0.027$

Day *k+1*

Day *k*

	S1	S2
S1	0.5	0.5
S2	0.4	0.6

	IN	OUT
S1	0.2	0.8
S2	0.9	0.1

An example of Baum-Welch (cntd)

- Now, let's estimate the transition matrix. Sequence I-I-I-I-I-O-O-I-I-I has the following events:
  - II, II, II, II, IO, OO, OI, II, II
- So, our estimate for S_1->S_2 transition probability is:
  - 0.285/2.4474 = 0.116
- Similarly, calculate the other three transition probs and normalize so they sum up to 1
- Update transition matrix

Seq	P(Seq) for S_1S_2	Best P(Seq)
II	0.027	0.3403 S_2S_2
II	0.027	0.3403 S_2S_2
II	0.027	0.3403 S_2S_2
II	0.027	0.3403 S_2S_2
IO	0.003	0.2016 S_2S_1
OO	0.012	0.0960 S_1S_1
OI	0.108	0.1080 S_1S_2
II	0.027	0.3403 S_2S_2
II	0.027	0.3403 S_2S_2
Total	**0.285**	**2.4474**

An example of Baum-Welch (cntd)

- Estimating initial probabilities:
  - Assume all sequences start with hidden state S_1, calculate best probability
  - Assume all sequences start with hidden state S_2, calculate best probability
  - Normalize to 1.

- Now, we have generated the updated transition, emission and initial probabilities. Repeat this method until those probabilities converge

The Baum-Welch algorithm

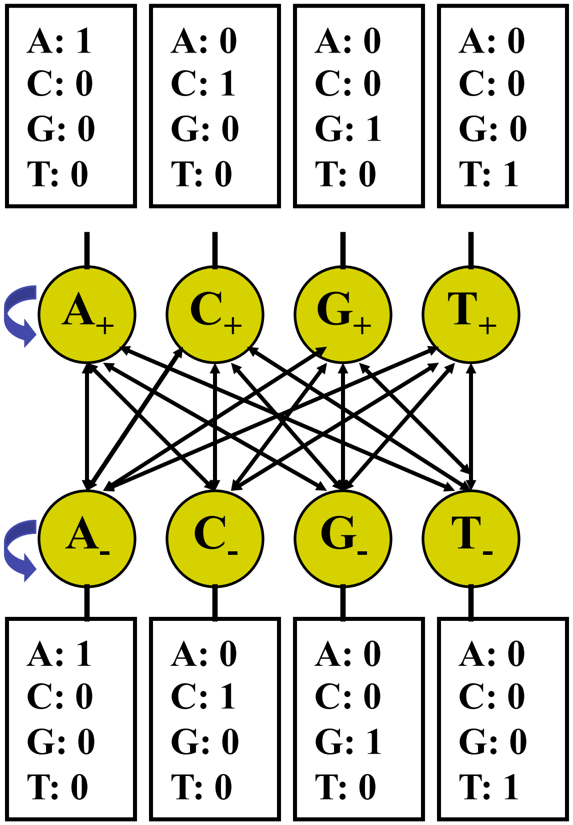
- Time complexity:
  - # iterations x $O(K^2N)$

- Guaranteed to increase the likelihood $P(x \mid \theta)$

- Not guaranteed to find globally optimal parameters
  - Converges to a local optimum, depending on initial conditions

- Too many parameters / too large model ⇒ Overtraining

Back to: HMM for CpG islands



A: 1
C: 0
G: 0
T: 0

A: 0
C: 1
G: 0
T: 0

A: 0
C: 0
G: 1
T: 0

A: 0
C: 0
G: 0
T: 1

A: 1
C: 0
G: 0
T: 0

A: 0
C: 1
G: 0
T: 0

A: 0
C: 0
G: 1
T: 0

A: 0
C: 0
G: 0
T: 1

Note: Each set ('+' or '-') has an additional set of transitions as in previous Markov chain

How do we find CpG islands in a sequence?

Build a single model that combines both Markov chains:

- **'+' states**: A_+, C_+, G_+, T_+
  - Emit symbols: A, C, G, T in CpG islands
- **'-' states**: A_-, C_-, G_-, T_-
  - Emit symbols: A, C, G, T in non-CpG islands

If a sequence CGCG is emitted by states (C_+,G_-,C_-,G_+), then:

$$P(CGCG) = a_{0,C_+} \times 1 \times a_{C_+,G_-} \times 1 \times a_{G_-,C_-} \times 1 \times a_{C_-,G_+} \times 1 \times a_{G_+,0}$$

In general, we DO NOT know the path. How to estimate the path?

What we have..

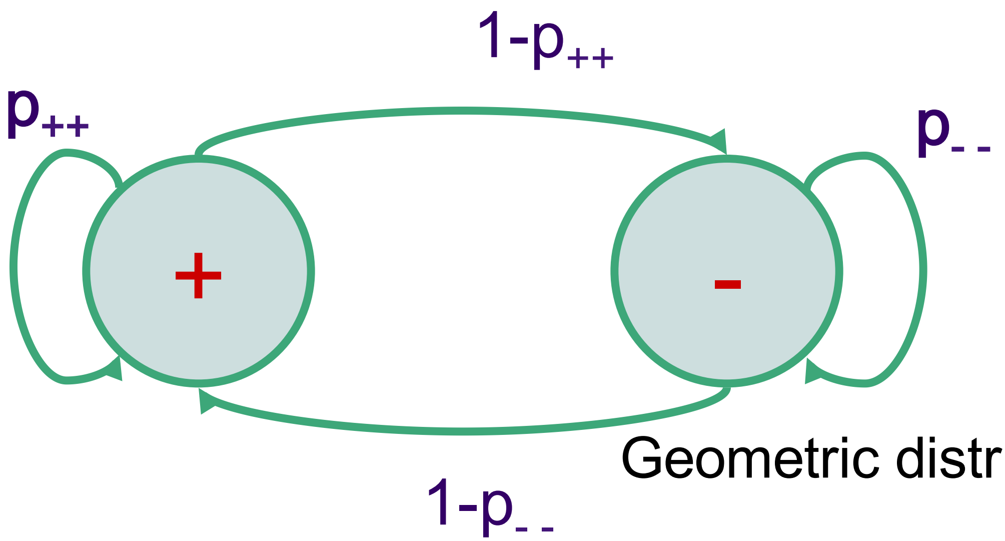
	A_+	C_+	G_+	T_+	A_-	C_-	G_-	T_-
A_+	.180	.274	.426	.120				
C_+	.171	.368	.274	.188				
G_+	.161	.339	.375	.125				
T_+	.079	.355	.384	.182				
A_-					.300	.205	.285	.210
C_-					.233	.298	.078	.302
G_-					.248	.246	.298	.208
T_-					.177	.239	.292	.292

Note: these transitions out of each state add up to one—no room for transitions between (+) and (-) states

Not a valid transition probability matrix nor a complete one!

A model of CpG Islands – Transitions

- What about transitions between (+) and (-) states?
- They affect
  - Avg. length of CpG island
  - Avg. separation between two CpG islands



Length distribution of region +:

$P(L=1) \Rightarrow +- = 1-p_{++}$:

$P(L=2) \Rightarrow ++- = p_{++}(1-p_{++})$

…

$P[L=\ell] = p_{++}^{\ell-1}(1-p_{++})$

$$\text{Geometric distribution, with mean } = \frac{1}{1-p_{++}}$$

Expected length of a state to continue in that state

What we have..

	A_+	C_+	G_+	T_+	A_-	C_-	G_-	T_-
A_+	.180	.274	.426	.120				
C_+	.171	.368	.274	.188				
G_+	.161	.339	.375	.125				
T_+	.079	.355	.384	.182				
A_-					.300	.205	.285	.210
C_-					.233	.298	.078	.302
G_-					.248	.246	.298	.208
T_-					.177	.239	.292	.292

$(1-\lambda_+) * freq(b_i)$

Now a valid transition probability matrix and a complete one!

λ_+

$(1-\lambda_-)*freq(b_i)$

λ_-

Another application: Profile HMMs

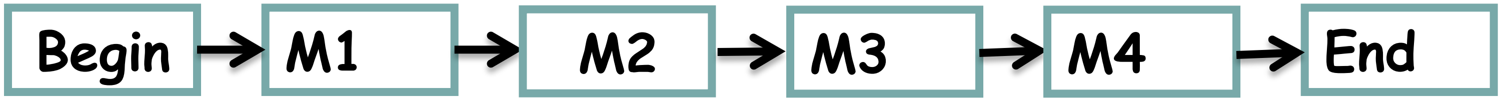
Profile HMMS (Haussler, 1993)

- Ungapped alignment of sequence *X* against profile *M*
  - e_i(a): probability of observing a at position *I*
  - $P(X \mid M) = \prod_{i=i,\dots,L} e_i(x_i)$
  - $Score(X \mid M) = \sum_{i=1,\dots,L} \log(e_i(x_i)/q_{x_i})$
- What about indels ?

Profile HMMs: “match” states

LEVK
LEIR
LEIK
LDVE

We make a single state HMM to represent above profile, using match states only



Deriving emission probabilities for the Match states

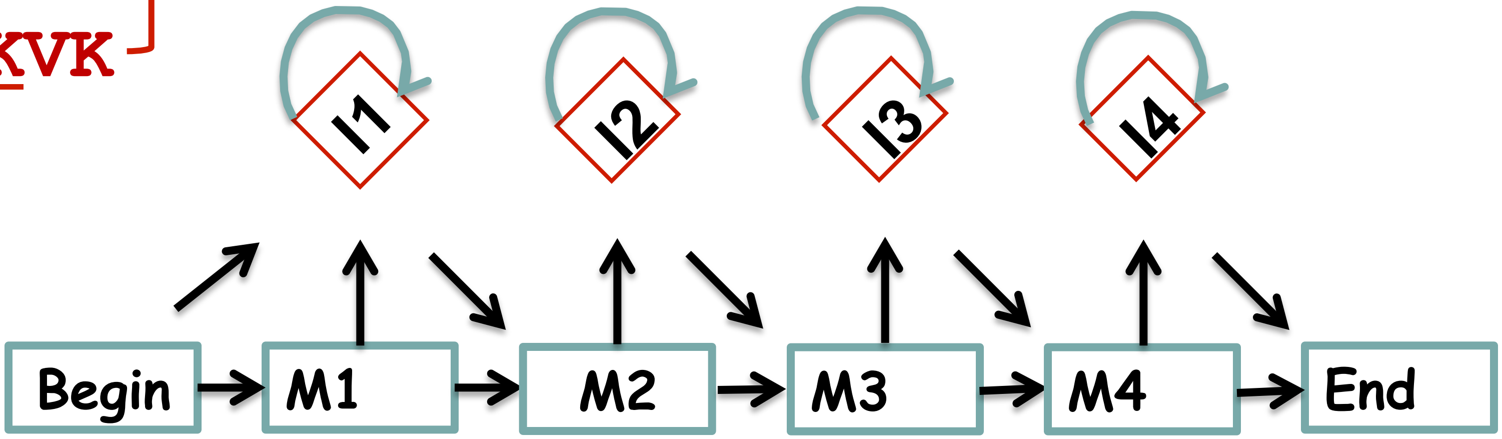
M1	M2	M3	M4
Pr(L)=1	Pr(E) = 3/4	Pr(V) = 1/2	Pr(R) = 1/4
	Pr(D) = 1/4	Pr(I) = 1/2	Pr(K) = 1/4
			Pr(E) = 1/4

Introducing "insert" states to the previous HMM

We want to know whether (for instance) the sequence LEKKVK is a good match to the HMM

```
LE--VK
LE--IR
LE--IK
LD--VE
LEKKVK
```

We know it should look like this in the end

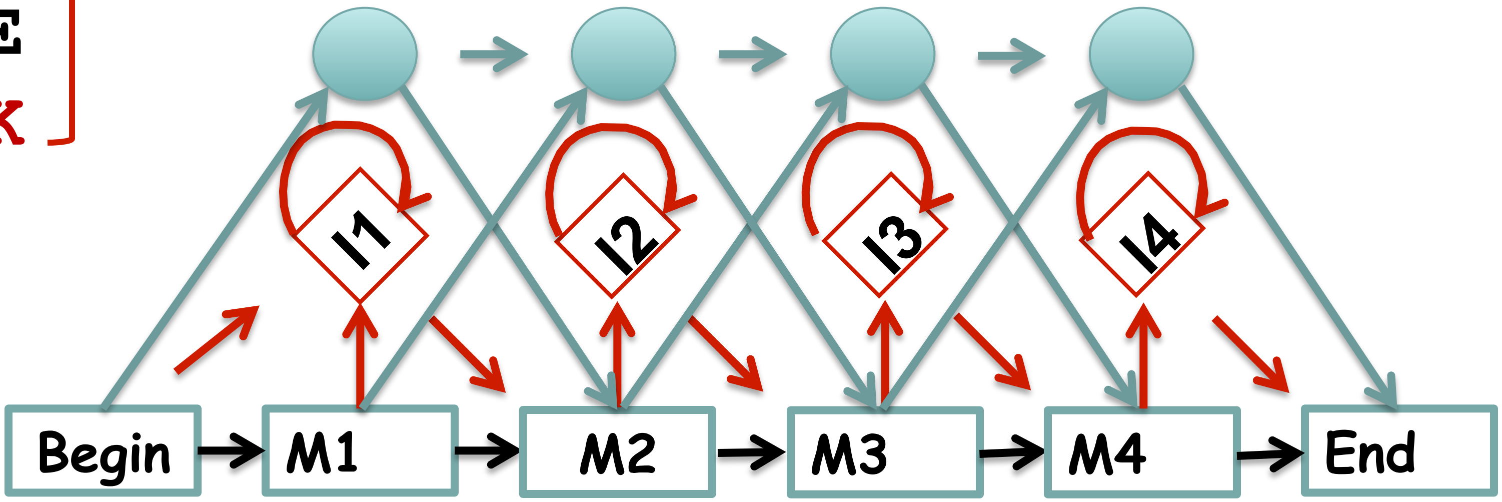


Introducing "delete" states to the previous HMM

We want to know whether (for instance) the sequence LEK is a good match to the HMM

```
LEVK
LEIR
LEIK
LDVE
LE-K
```

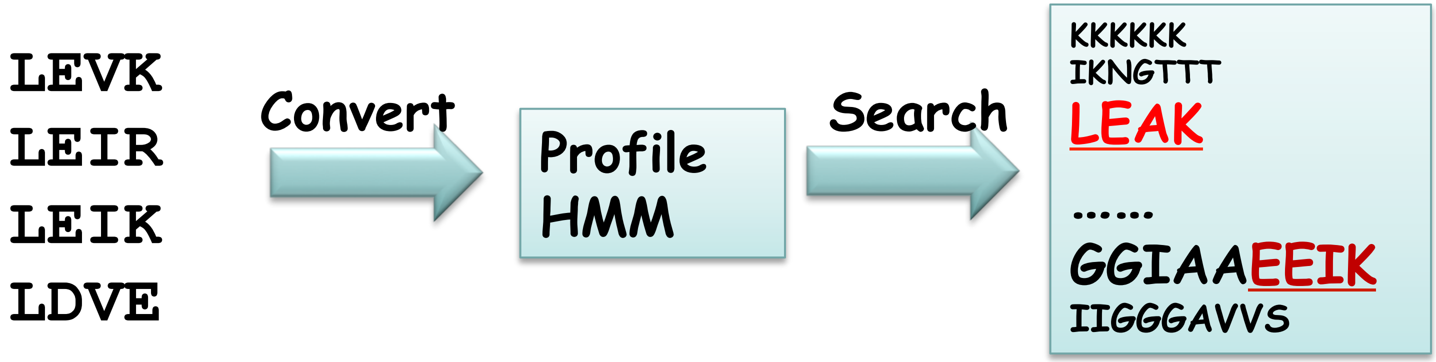
We know it should look like this in the end



Three main applications for profile HMMs

1. Find sequence homologs

- ie, we represent a sequence family by an HMM and use that to identify (“evaluate”) other related sequences



LEVK
LEIR
LEIK
LDVE

Convert

Profile HMM

Search

KKKKKK
IKNGTTT
LEAK
……
GGIAAEEIK
IIGGAVVS

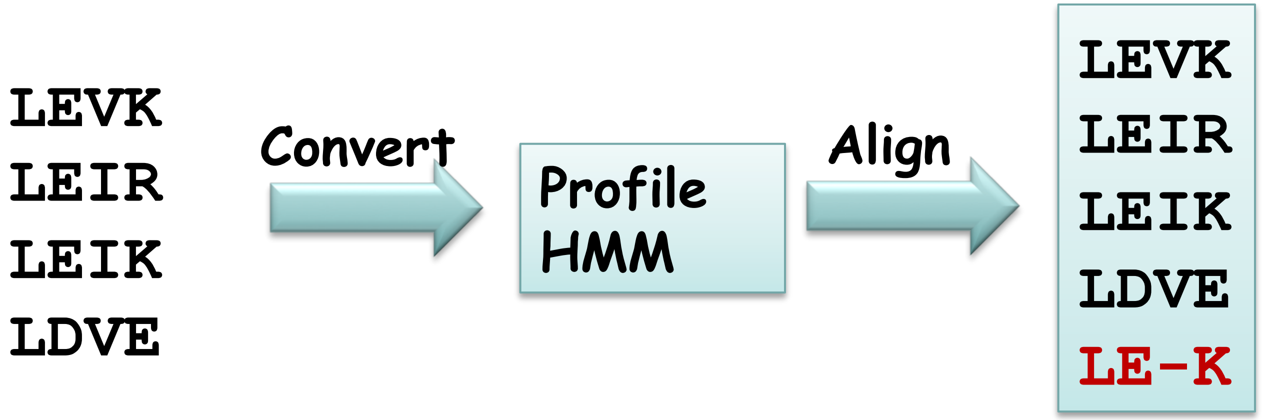
Evaluation: So Use Forward
Viterbi is OK too. $P(x, SP^* \mid \lambda)$

$$P(x \mid \lambda) = \sum_{\substack{AllPossibleParses \\ (SP^p; \#\ of\ possible\ p's = K^L)}} P(x, SP^p \mid \lambda) = \sum_{\substack{AllPossibleParses \\ (K^L\ \ Possibilities)}} \prod_{i=1}^{L} a(\pi_{i-1}, \pi_i) e(\pi_i, x_i)$$

Three main applications for profile HMM

2. Align a new sequence to the profile

- ie, we expnad our multiple sequence alignment



LEVK
LEIR
LEIK
LDVE

Convert

Profile HMM

Align

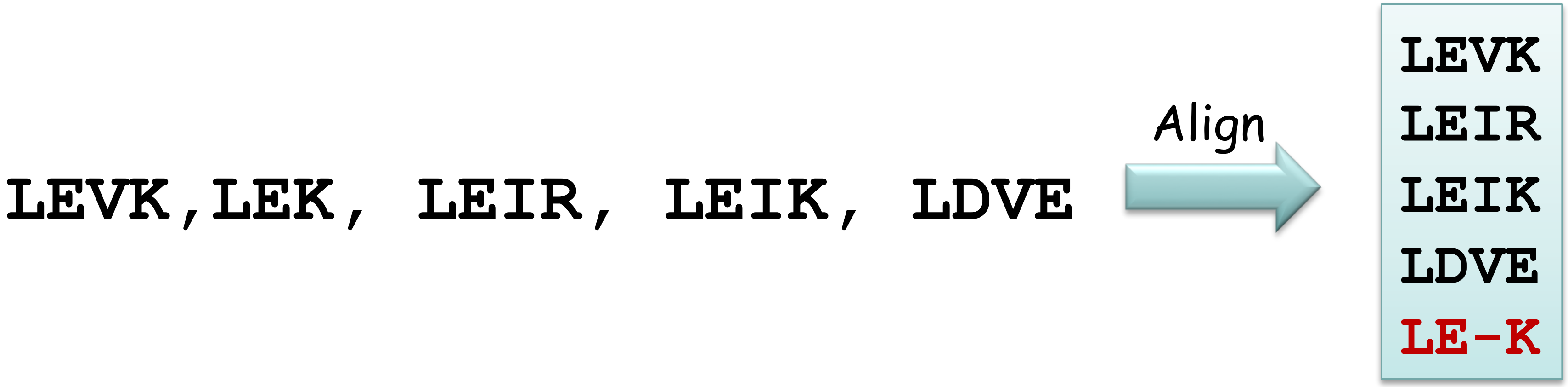
LEVK
LEIR
LEIK
LDVE
LE-K

This is Decoding: Use Viterbi

Three main applications for profile HMM

3. Align a set of sequences from scratch

- ie, we want to build a multiple sequence alignment of a set of "unaligned sequences"



LEVK,LEK, LEIR, LEIK, LDVE

Align

LEVK
LEIR
LEIK
LDVE
LE-K

This needs parameter estimation: use Baum-Welch

Making multiple sequence alignment from *unaligned sequences*

- Baum-Welch Expectation-maximization method
  - Start with a model whose length matches the average length of the sequences and with random output and transition probabilities.
  - Align all the sequences to the model.
  - Use the alignment to alter the output and transition probabilities
  - Repeat. Continue until the model stops changing
- By-product: It produced a multiple alignment

Acknowledgements

Some of the slides used in this lecture are adapted or modified slides from lectures of:

- Serafim Batzoglou, Stanford University
- Bino John, Dow Agrosciences
- Nagiza F. Samatova, Oak Ridge National Lab
- Sarah Wheelan, Johns Hopkins University
- Eric Xing, Carnegie-Mellon University

Theory and examples from the following books:

- T. Koski, "*Hidden Markov Models for Bioinformatics*", 2001, Kluwer Academic Publishers
- R. Durbin, S. Eddy, A. Krogh, G. Mitchison, "Biological Sequence Analysis", 1998, Cambridge University Press