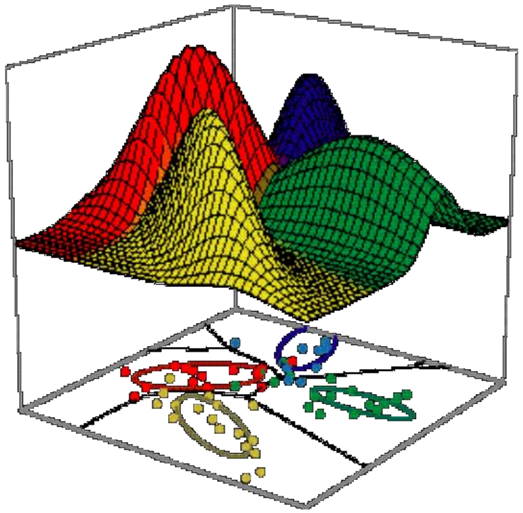
Part 11: Hidden Markov Models



Markov Chains*

Introduction to HMMs

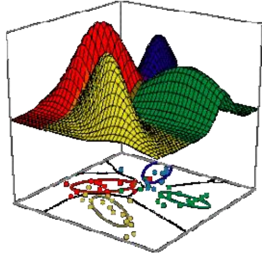
HMM Details*

Sample applications

Expectation Maximization

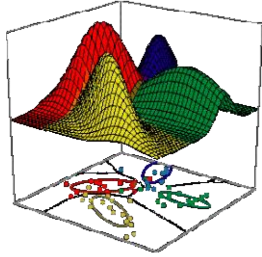
Some materials in these slides were taken from Pattern Classification (2nd ed) by R. O. Duda, P. E. Hart and D. G. Stork, John Wiley & Sons, 2000, **Chapter 3.9-3.10**. and *Jaap Heringa, “Bioinformatics data analysis and tools,” Centre for Integrative Bioinformatics VU, vrije Universiteit amsterdam, 2005.

Pattern Classification in Bioinformatics



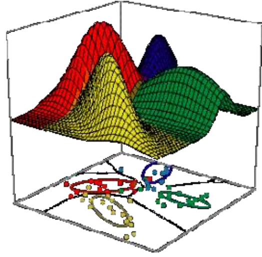
- We are often faced with a small number of ‘solved’ examples in biology
  - ‘wet lab’ experiments are hard & expensive
  - E.g. ~30K solved protein structures, but >100Billion base pairs in Genbank
- Wish to extrapolate/generalize from these few examples to create new knowledge
- Often searching for patterns in the data
  - Create a pattern to represent a ‘class’
  - Search for that pattern in new data to apply annotation

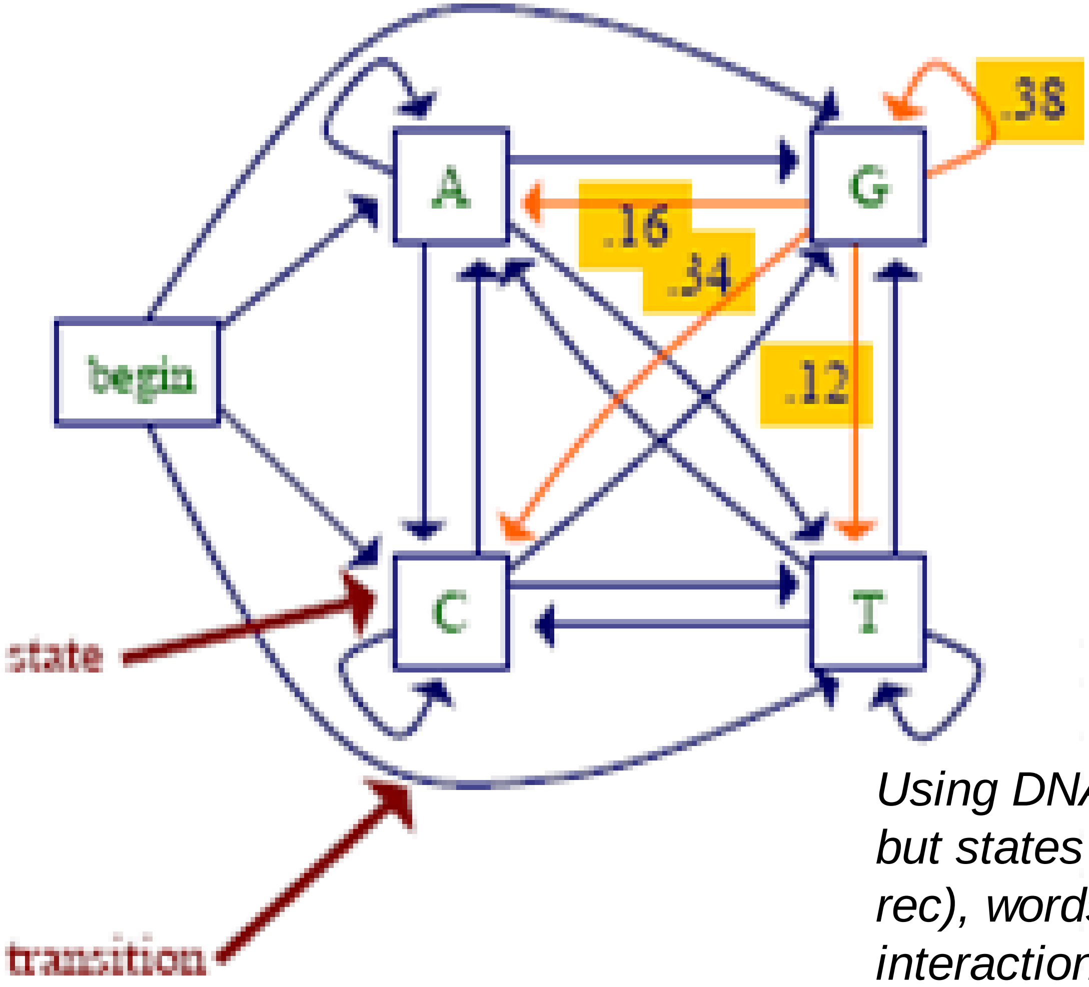
Pattern Classification in Bioinformatics



- Data and patterns are often not clear cut
- When we want to make a method to recognize a pattern (e.g. a sequence motif), we have to learn from the data
  - e.g. maybe there are differences between sequences that have the pattern and those that do not
- This leads to *Data mining* and *Machine learning*
- Hidden Markov models are a powerful tool for pattern recognition/classification.
  - Especially well suited to problems where the input data is a *sequence* of some kind
    - E.g. protein sequence, phonemes in speech, time series data

Markov Chain Models





Transition Probabilities:

$\Pr(x_i=a \mid x_{i-1}=g)=0.16$

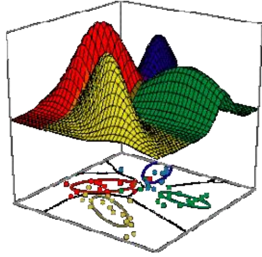
$\Pr(x_i=c \mid x_{i-1}=g)=0.34$

$\Pr(x_i=g \mid x_{i-1}=g)=0.38$

$\Pr(x_i=t \mid x_{i-1}=g)=0.12$

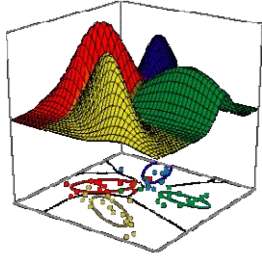
Using DNA seq analysis in this chapter, but states could be phonemes (speech rec), words (predictive keyboard), interactions with a webserver (bot ID), etc.

Markov Chain Models



- a Markov chain model is defined by:
  - a set of states
    - some states *emit* symbols
    - other states (e.g. the *begin* state) are *silent*
  - a set of transitions with associated probabilities
    - the transitions emanating from a given state define a distribution over the possible next states

Markov Chain Models



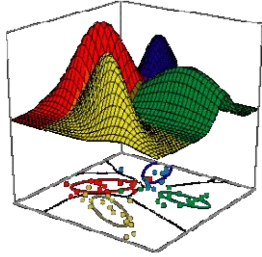
- Given some sequence *x* of length *L*, we can ask how probable the sequence is given our model
- For any probabilistic model of sequences, we can write this probability as

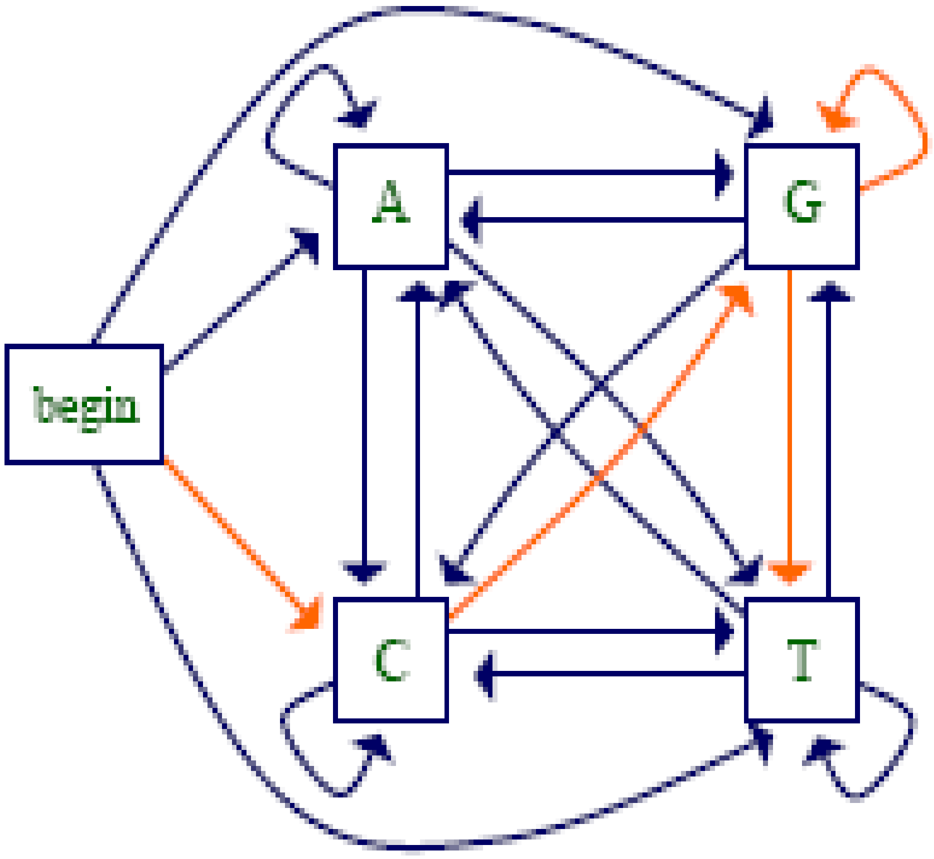
$$
\begin{aligned}
P(x) &= P(x_L, x_{L-1}, \ldots, x_1) \\
&= P(x_L \mid x_{L-1}, \ldots, x_1) P(x_{L-1} \mid x_{L-2}, \ldots, x_1) \ldots P(x_1)
\end{aligned}
$$

- Key property of a (1st order) Markov chain: the probability of each x_i depends only on x_{i-1}

$$
\begin{aligned}
P(x) &= P(x_L \mid x_{L-1}, \ldots, x_1) P(x_{L-1} \mid x_{L-2}, \ldots, x_1) \ldots P(x_1) \\
&= P(x_1) \prod_{i=2}^{L} P(x_i \mid x_{i-1})
\end{aligned}
$$

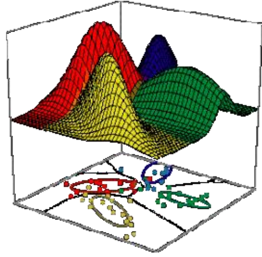
Markov Chain Models





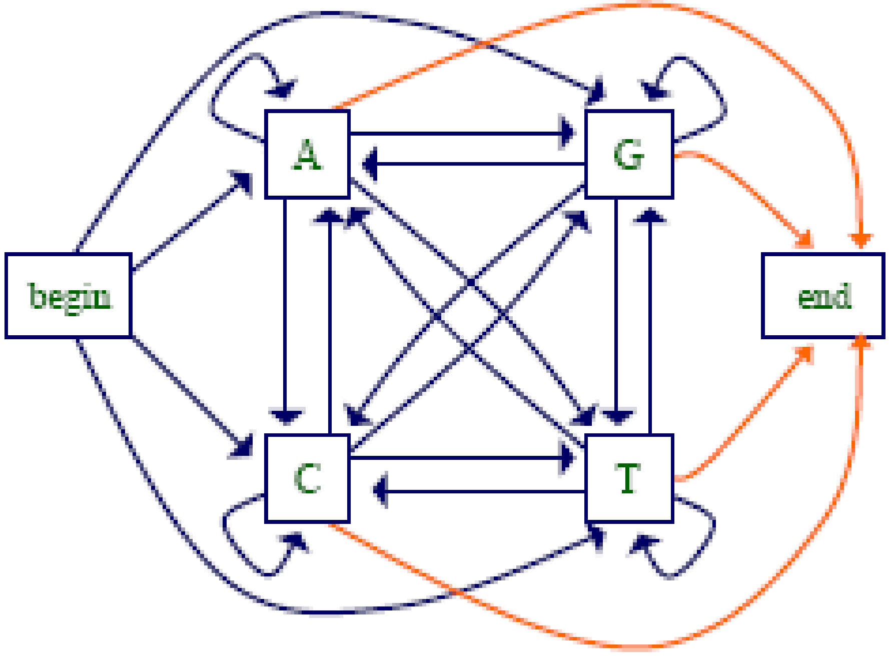
$$\Pr(cggt) = \Pr(c)\Pr(g|c)\Pr(g|g)\Pr(t|g)$$

Markov Chain Models

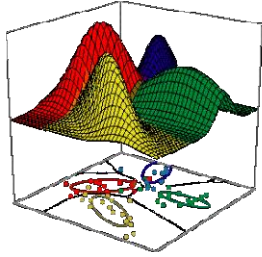


Can also have an *end* state, allowing the model to represent:

- Sequences of different lengths
- Preferences for sequences ending with particular symbols



Markov Chain Models



The transition parameters can be denoted by $a_{x_{i-1}x_i}$ where

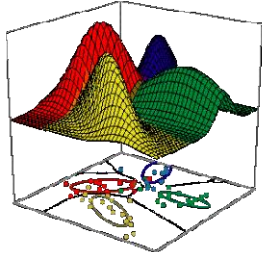
$$a_{x_{i-1}x_i} = \Pr(x_i \mid x_{i-1})$$

Similarly we can denote the probability of a sequence *x* as

$$P(x) = a_{Bx_1} \prod_{i=2}^{L} a_{x_{i-1}x_i} = P(x_1) \prod_{i=2}^{L} P(x_i \mid x_{i-1})$$

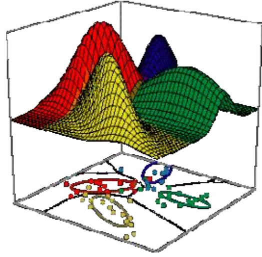
Where $a_{\mathrm{B}xi}$ represents the transition from the *begin* state

Example Application



- CpG islands
  - CG dinucleotides are rarer in eukaryotic genomes than expected given the independent probabilities of C, G
  - In human genome *CpG* (*CG*) is least frequent dinucleotide, because *C* in *CpG* is easily methylated and has the tendency to mutate into T afterwards.
  - Methylation is suppressed around genes in a genome: *CpG* appears more frequently within these regions, called *CpG* islands.
  - Particularly, the regions upstream of genes are richer in CG dinucleotides than elsewhere – *CpG islands*
  - Identifying the *CpG* islands in a genome is important.
    - useful evidence for finding genes
- Could predict CpG islands with Markov chains
  - one to represent CpG islands
  - one to represent the rest of the genome

Estimating the Model Parameters

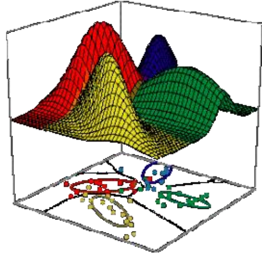


- Given some data (e.g. a set of sequences from CpG islands), how can we determine the probability parameters of our model?
- One approach: *maximum likelihood estimation*
  - given a set of data D
  - set the parameters θ to maximize

$$\Pr(D \mid \theta)$$

  - i.e. make the data D look likely under the model

Maximum Likelihood Estimation



- Suppose we want to estimate the parameters Pr(a), Pr(c), Pr(g), Pr(t)

- And we're given the sequences:

  accgcgctta

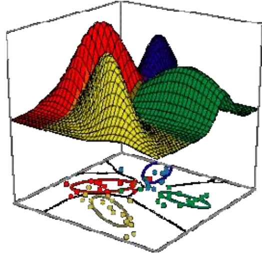
  gcttagtgac

  tagccgttac

- Then the maximum likelihood estimates are:

Pr(a) = 6/30 = 0.2	Pr(g) = 7/30 = 0.233
Pr(c) = 9/30 = 0.3	Pr(t) = 8/30 = 0.267

Maximum Likelihood Estimation



- suppose instead we saw the following sequences

  gccgcgcttg

  gcttggtggc

  tggccgttgc

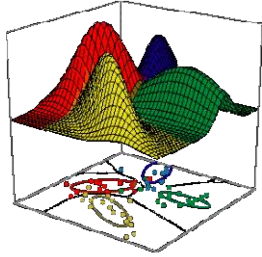
- then the maximum likelihood estimates are

$$\Pr(a) = \frac{0}{30} = 0 \qquad \Pr(g) = \frac{13}{30} = 0.433$$

$$\Pr(c) = \frac{9}{30} = 0.3 \qquad \Pr(t) = \frac{8}{30} = 0.267$$

do we really want to set this to 0?

A Bayesian Approach



- instead of estimating parameters strictly from the data, we could start with some prior belief for each
- for example, we could use *Laplace estimates*

$$\Pr(a) = \frac{n_a + 1}{\sum_i (n_i + 1)}$$

pseudocount

- using Laplace estimates with the sequences

gccgcgcttg

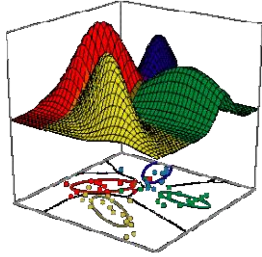
gcttggtggc

tggccgttgc

$$\Pr(a) = \frac{0+1}{34}$$

$$\Pr(c) = \frac{9+1}{34}$$

A Bayesian Approach



- a more general form: *m-estimates*

$$\Pr(a) = \frac{n_a + \overbrace{p_a}^{\text{prior probability of } a} m}{\left(\sum_i n_i\right) + \underbrace{m}_{\text{number of “virtual” instances}}}$$

- with m=8 and uniform priors

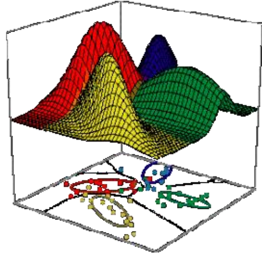
gccgcgcttg

gcttggtggc

tggccgttgc

$$\Pr(c) = \frac{9 + 0.25 \times 8}{30 + 8} = \frac{11}{38}$$

Markov Chains for Discrimination

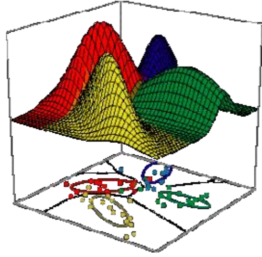


- suppose we want to distinguish CpG islands from other sequence regions
- given sequences from CpG islands, and sequences from other regions, we can construct
  - a model to represent CpG islands
  - a *null model* to represent the other regions
- can then score a test sequence by:

$$score(x) = \log \frac{\Pr(x \mid \text{CpG model})}{\Pr(x \mid \text{null model})}$$

Scaling by probability that a null model would generate the observed sequence is important to normalize for different sequence lengths….

Markov Chains for Discrimination

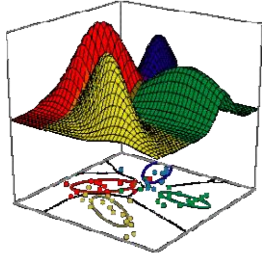


- parameters estimated for CpG and null models

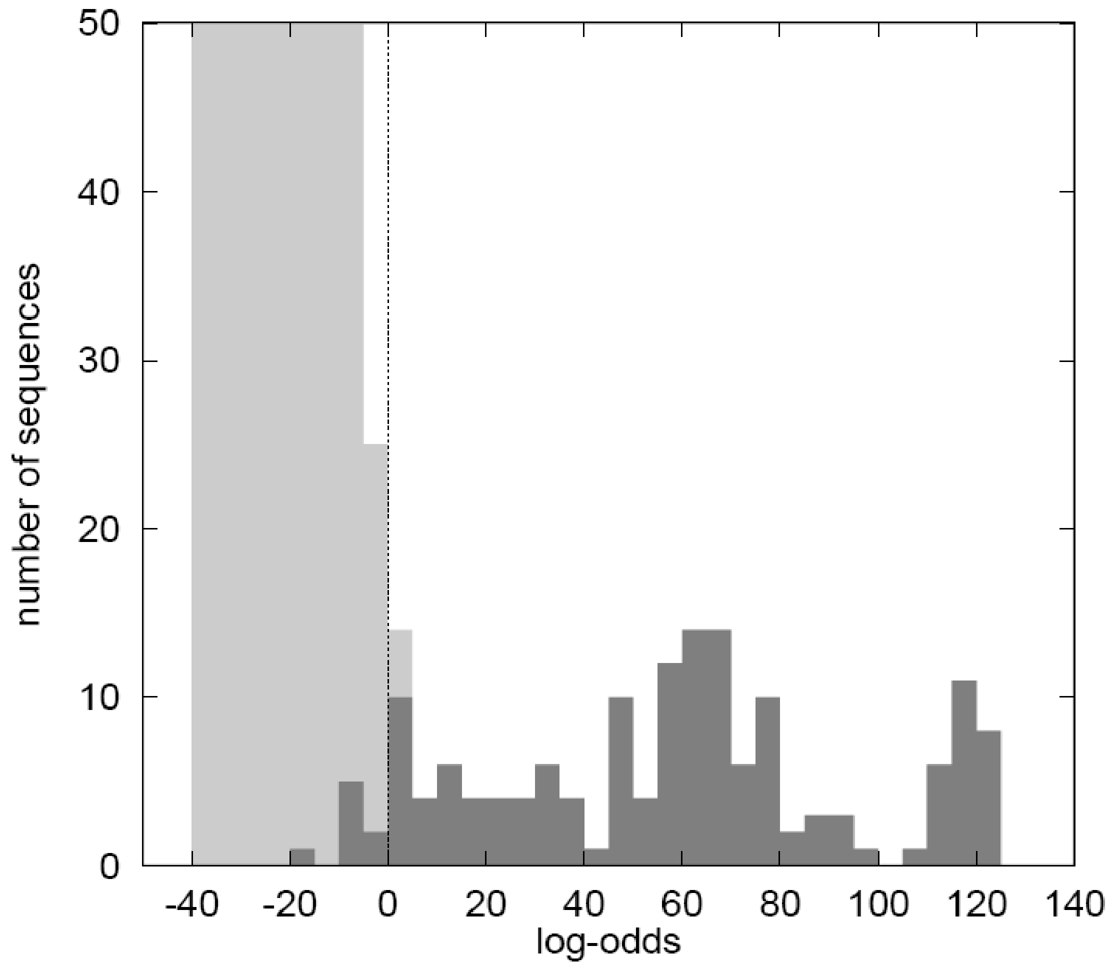
+	A	C	G	T
A	.18	.27	.43	.12
C	.17	.37	.27	.19
G	.16	.34	.38	.12
T	.08	.36	.38	.18

-	A	C	G	T
A	.30	.21	.28	.21
C	.32	.30	.08	.30
G	.25	.24	.30	.21
T	.18	.24	.29	.29

These dinucleotide frequency data are derived from genome sequences



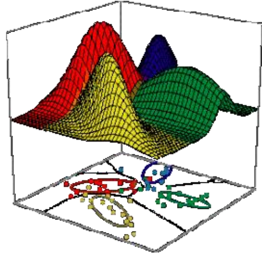
Markov Chains for Discrimination



- Light bars represent negative sequences
- Dark bars represent positive sequences
- The actual figure here is not from a CpG island discrimination task…

Figure from A. Krogh, "An Introduction to Hidden Markov Models for Biological Sequences" in Computational Methods in Molecular Biology, Salzberg et al editors, 1998.

Markov Chains for Discrimination



- why use

$$score(x) = \log \frac{\Pr(x \mid CpG)}{\Pr(x \mid null)}$$

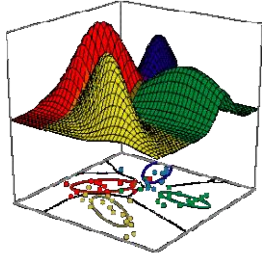
- Bayes' rule tells us

$$\Pr(CpG \mid x) = \frac{\Pr(x \mid CpG)\Pr(CpG)}{\Pr(x)}$$

$$\Pr(null \mid x) = \frac{\Pr(x \mid null)\Pr(null)}{\Pr(x)}$$

- if we're not taking into account priors, then just need to compare $\Pr(x \mid CpG)$ and $\Pr(x \mid null)$

Higher Order Markov Chains



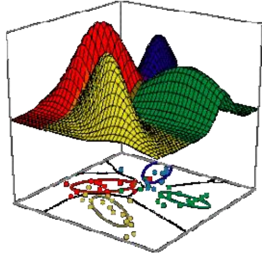
- the Markov property specifies that the probability of a state depends only on the probability of the previous state
- but we can build more "memory" into our states by using a higher order Markov model
- in an *n*th order Markov model

$$\Pr(x_i \mid x_{i-1}, x_{i-2}, \ldots, x_1) = \Pr(x_i \mid x_{i-1}, \ldots, x_{i-n})$$

Inhomogeneous Markov Chains

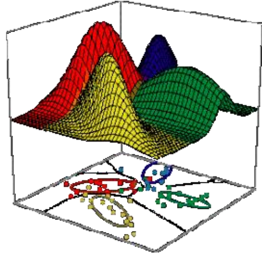
• In the Markov chain models we have considered so far, the probabilities do not depend on where we are in a given sequence
• In an *inhomogeneous* Markov model, we can have different distributions at different positions in the sequence – approaches an HMM…

Application: Gene Finding



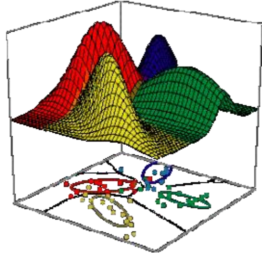
- Protein encoding affects the statistical properties of a DNA sequence
  - some amino acids are used more frequently than others (Leu more popular than Trp)
  - different numbers of codons for different amino acids (Leu has 6, Trp has 1)
  - for a given amino acid, usually one codon is used more frequently than others
  - This is termed *codon preference*
  - Codon preferences vary by species
- Therefore, expect coding DNA sequence to have different properties than non-coding.

Application: Gene Finding



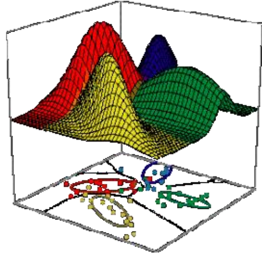
- Common way to search by content
  - build Markov models of coding & noncoding regions
  - apply models to ORFs (Open Reading Frames)
    or fixed-sized windows of sequence
- GeneMark [Borodovsky et al.]
  - popular system for identifying genes in bacterial genomes
  - uses 5th order inhomogenous Markov chain models

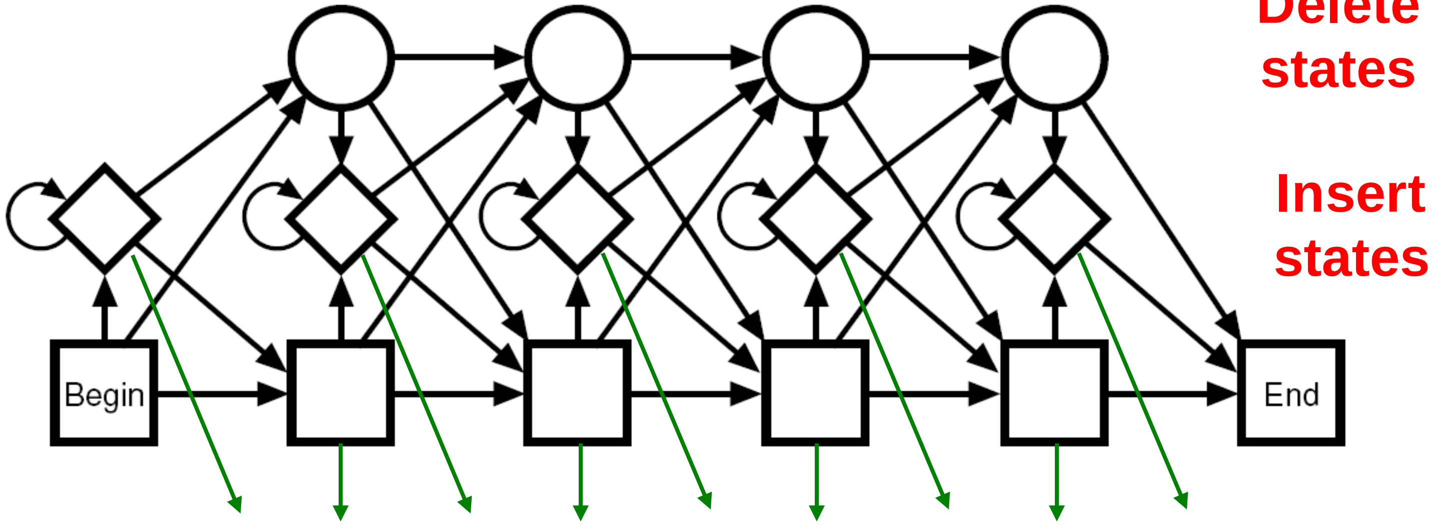
Hidden Markov Models



- Extension of Markov Chains (MC)
- Separate concepts of:
  - 'what state we are in' ← 'Hidden' part (MC)
  - 'what symbol do we observe' ← 'Observed' part
- Emission probabilities vary by position
- Permit insertions/deletions
  - i.e. states with no emitted symbol
  - Can naturally represent a family of sequences with different lengths.

Hidden Markov Topology

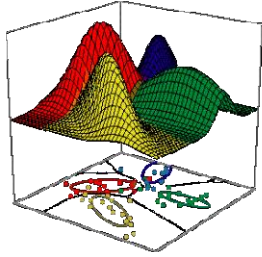




Each state (except Start, End, and Delete States) emits a symbol, based on a probability distribution

Adapted from A. Krogh (1998). “An Introduction to Hidden Markov Models for Biological Sequences”, in Computational Methods in Molecular Biology, Salzberg et al Editors.

HMM vs. MC vs. Linear Motif

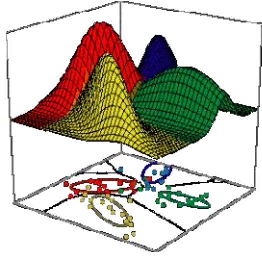


- Assume we have 5 examples of a binding site:

```
A C A - - - A T G
T C A A C T A T C
A C A C - - A G C
A G A - - - A T C
A C C G - - A T C
```

- We want to find a pattern to represent this binding site, then search genome for more examples.

HMM vs. MC vs. Linear Motif



A	C	A	-	-	-	A	T	G
T	C	A	A	C	T	A	T	C
A	C	A	C	-	-	A	G	C
A	G	A	-	-	-	A	T	C
A	C	C	G	-	-	A	T	C

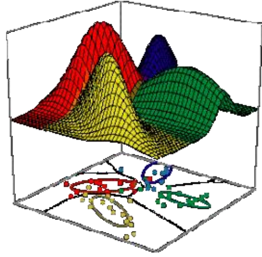
- We could represent this using a **linear motif**:

[AT] [CG] [AC] [ACGT-]&#42;A [TG] [GC]

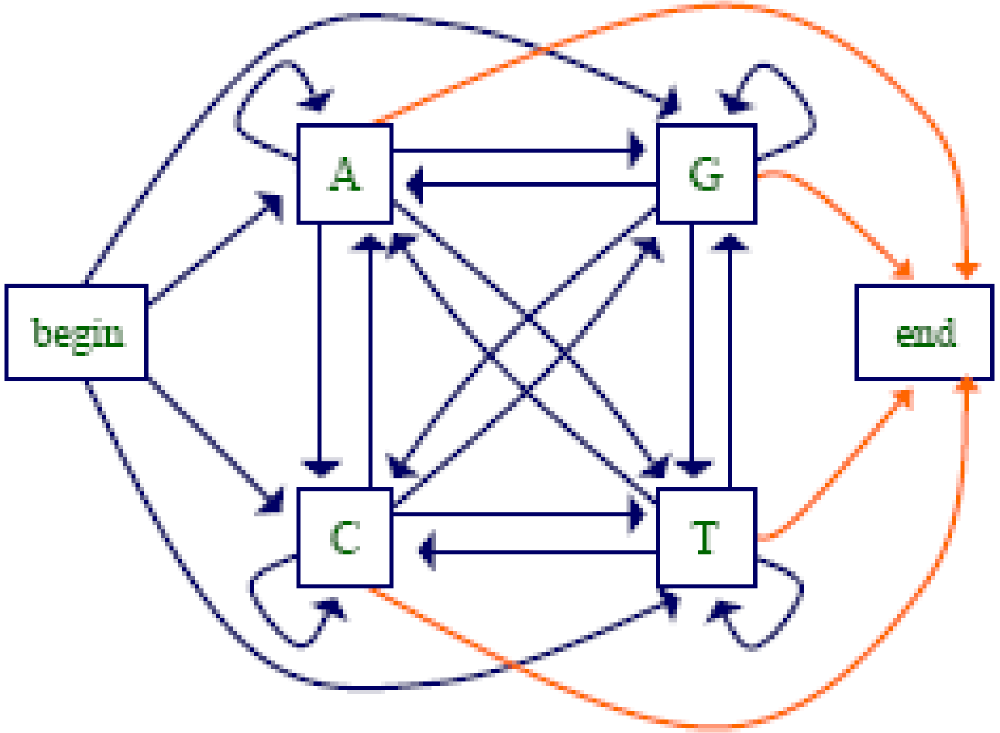
[AT] [CG] [AC] [ACGT-](3) A [TG] [GC]

  - This generates ~3600 possible valid sequences
  - Some more likely than others, but not captured by motif.

HMM vs. MC vs. Linear Motif

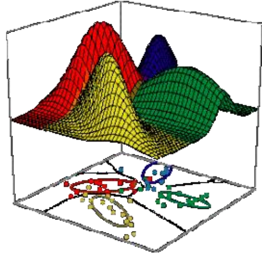


- We could represent this using a **Markov Chain**:



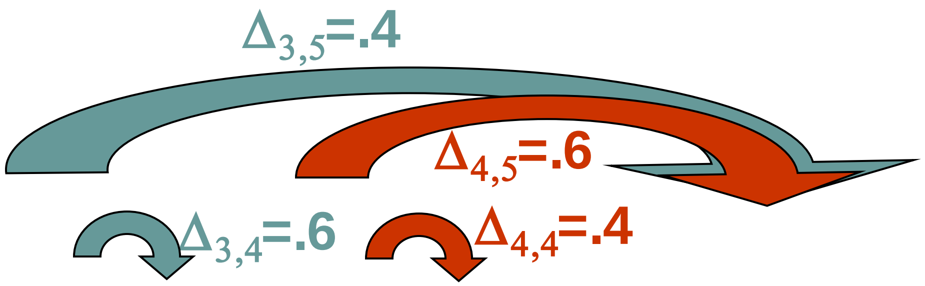
- Compute: P(C|A) = 5/45 = 0.11, P(C|G) = …
- Can now capture likelihood of a given sequence matching the model, but have lost *position-dependence* of probabilities.

HMM vs. MC vs. Linear Motif



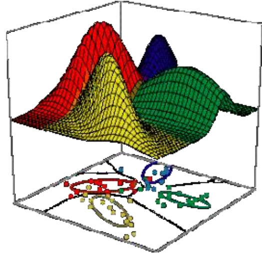
- We could represent this using a **hidden Markov Model**:

Transition:

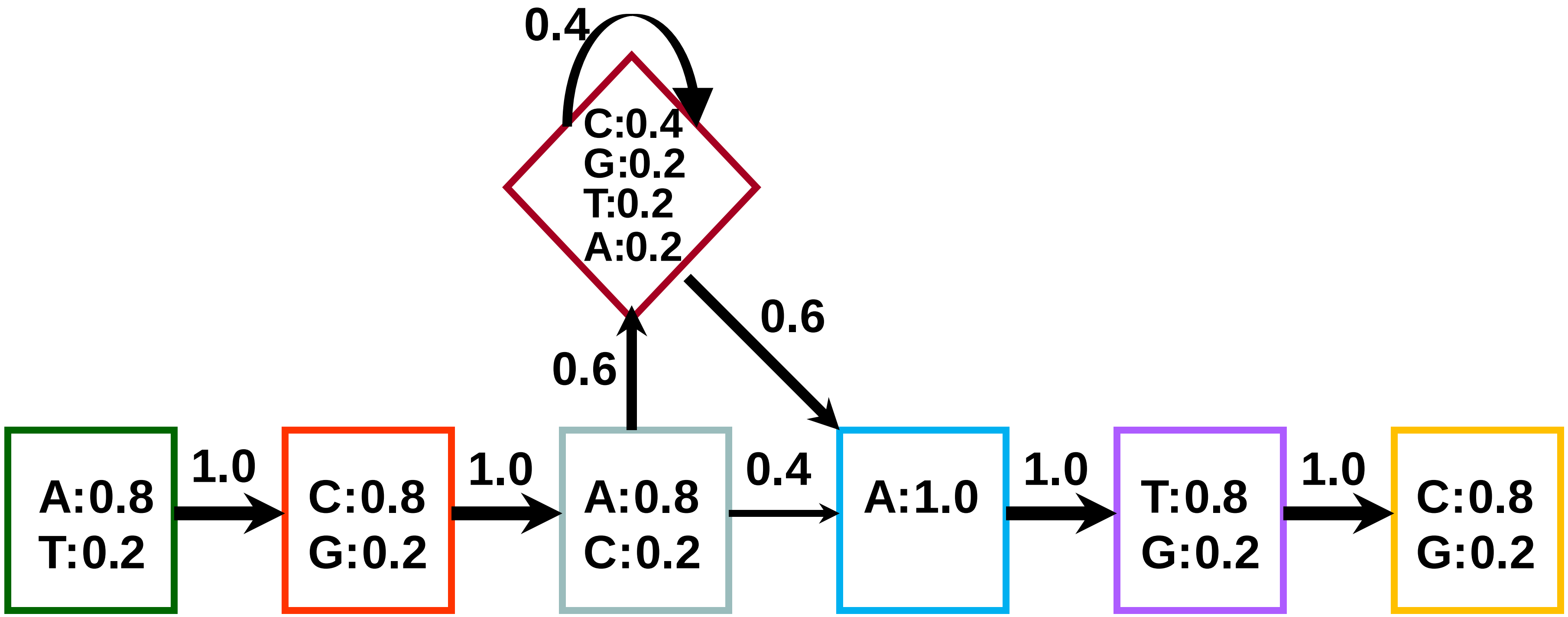


State:	1	2	3	4			5	6	7
	A	C	A	–	–	–	A	T	G
	T	C	A	A	C	T	A	T	C
	A	C	A	C	–	–	A	G	C
	A	G	A	–	–	–	A	T	C
	A	C	C	G	–	–	A	T	C
Emission:	p(A)=.8 p(T)=.2	p(C)=.8 p(G)=.2	p(A)=.8 p(C)=.2	p(A)=.2 p(C)=.4 p(T)=.2 p(G)=.2			p(A)=1	p(T)=.8 p(G)=.2	p(C)=.8 p(G)=.2

HMM vs. MC vs. Linear Motif



- Resulting **hidden Markov Model**:

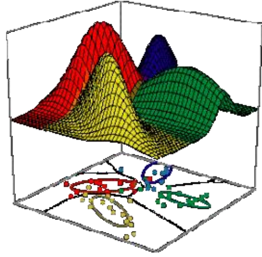


- Probability of observing: ACAC--ATC *Ignore dashes…*

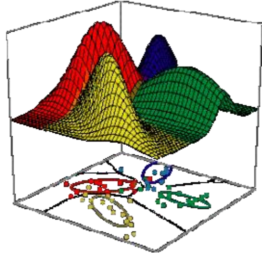
P(ACAC--ATC)=0.8x1.0 x 0.8x1.0 x 0.8x0.6 x 0.4x0.6 x 1.0x1.0 x 0.8x1.0 x 0.8 = 0.0047

- Now have probability (MC) and position-dependence (Motif)

Hidden Markov Models (formally…)

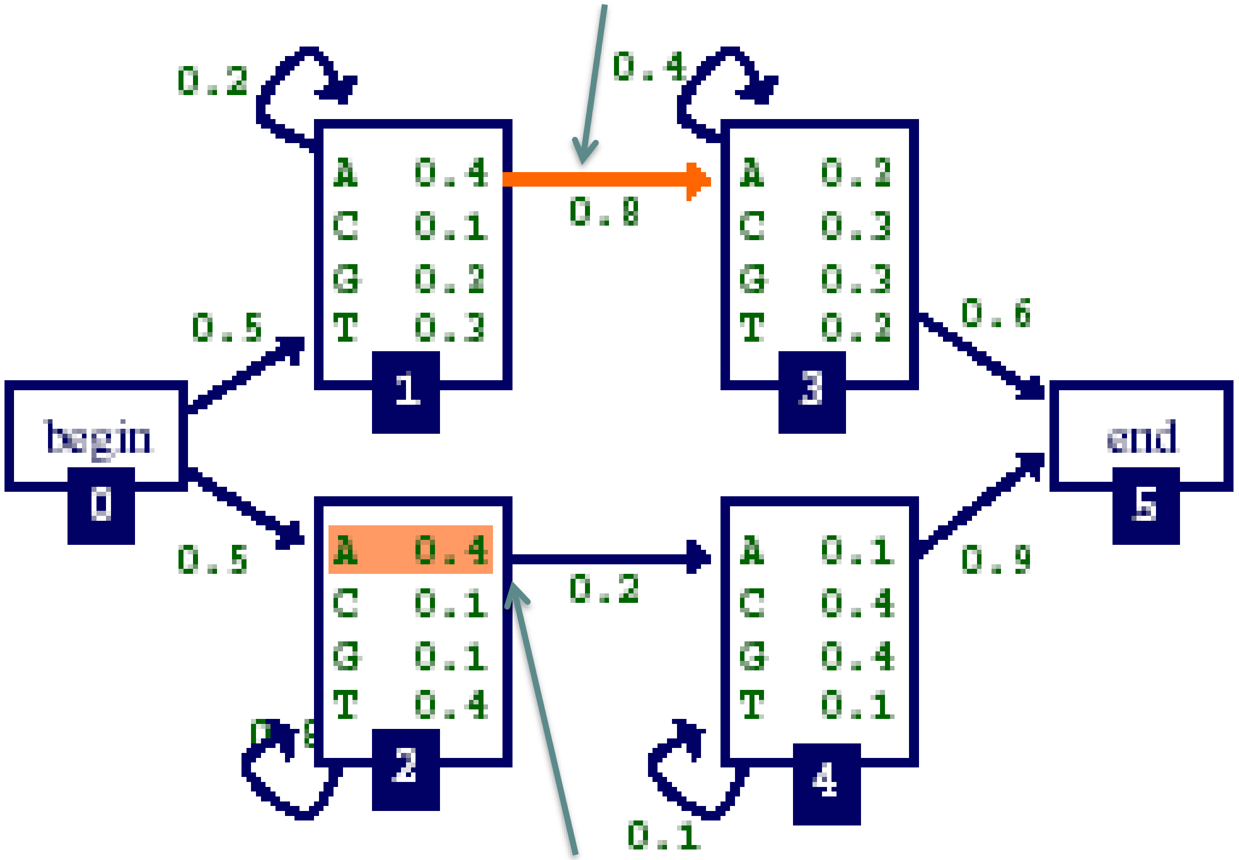


1. Σ - observations
   - $x_1,\ldots,x_L$ – sequence of observations
2. Q - states
   - $\pi_1,\ldots,\pi_n$ – hidden sequence of states
   - $\phi=(\phi_1,\ldots,\phi_N)^T$ - initial probability of states
3. $A = (a_{ij})$ – transition matrix
   - Prob of moving to state j given that we are in state i
4. $E = (e_i(x))$ – emission probabilities
   - Prob of emitting symbol x given that we are in state i



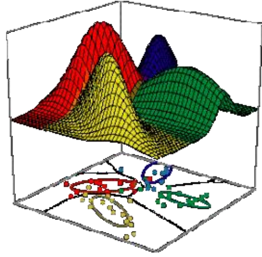
A Simple HMM

a_{13} = probability of a transition from state 1 to state 3



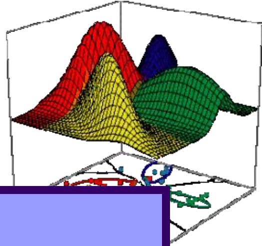
$e_2(A)$ = probability of emitting character *A* in state 2

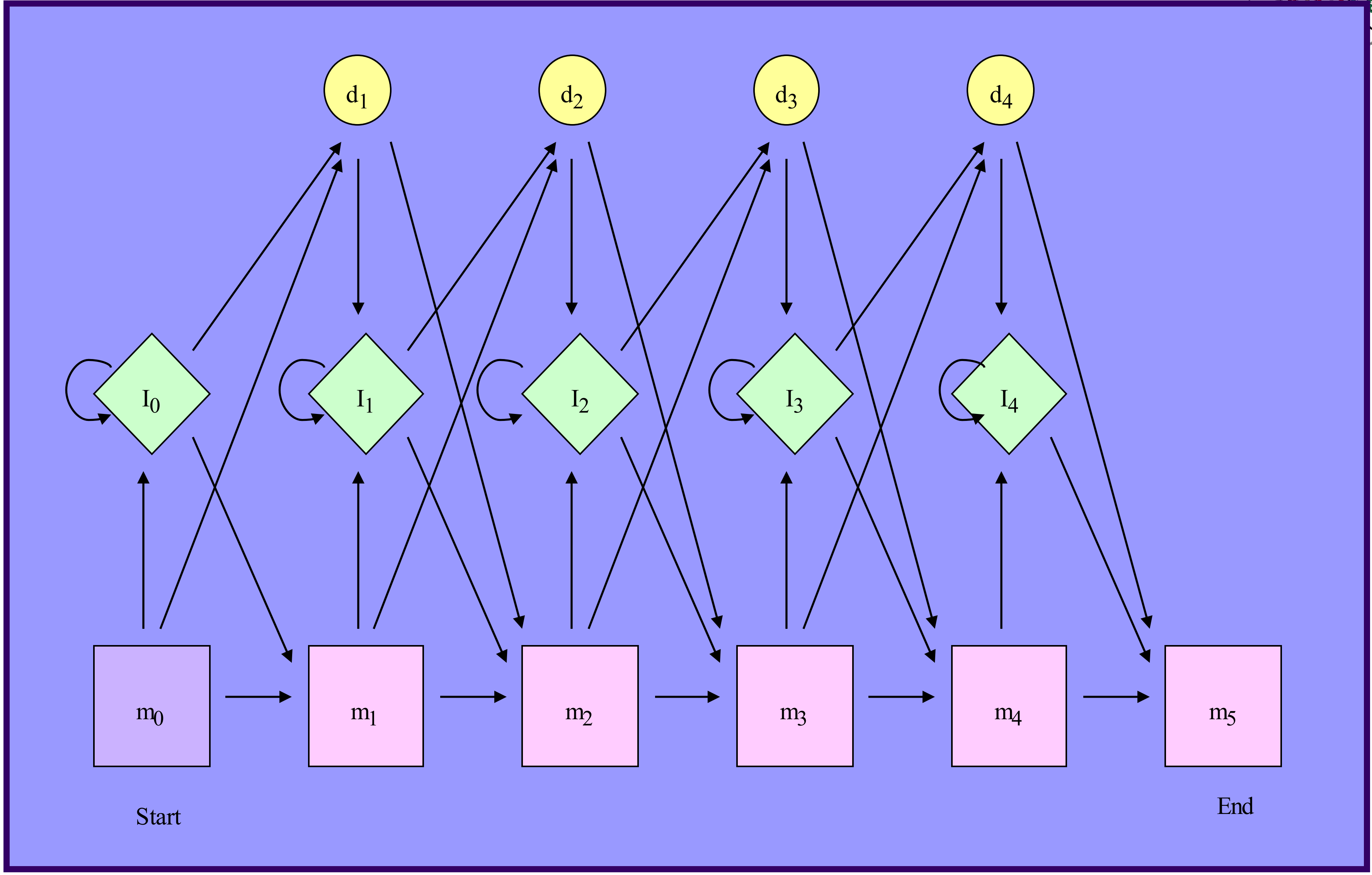
The Hidden State



- We will distinguish between the *observed* parts of a problem and the *hidden* parts
- In the Markov models we have considered previously, it is clear which state accounts for each part of the observed sequence
- In the model above (preceding slide), there are multiple states that could account for each part of the observed sequence
  - this is the hidden part of the problem
  - states are decoupled from sequence symbols

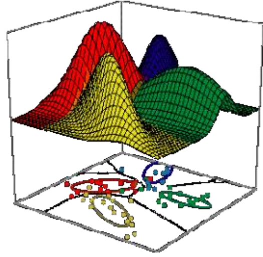
Profile HMM: m=match state, I-insert state, d=delete state; go from left to right. I and m states output amino acids; d states are 'silent".





Model for protein sequence alignment with insertions and deletions

The Parameters of an HMM



- as in Markov chain models, we have transition probabilities

$$a_{kl} = \Pr(\pi_i = l \mid \pi_{i-1} = k)$$

probability of a transition from state *k* to *l*

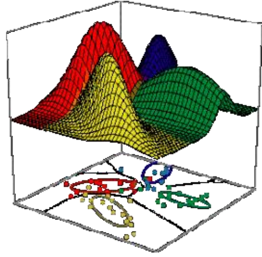
π represents a path (sequence of states) through the model

- since we've decoupled states and characters, we might also have emission probabilities

$$e_k(b) = \Pr(x_i = b \mid \pi_i = k)$$

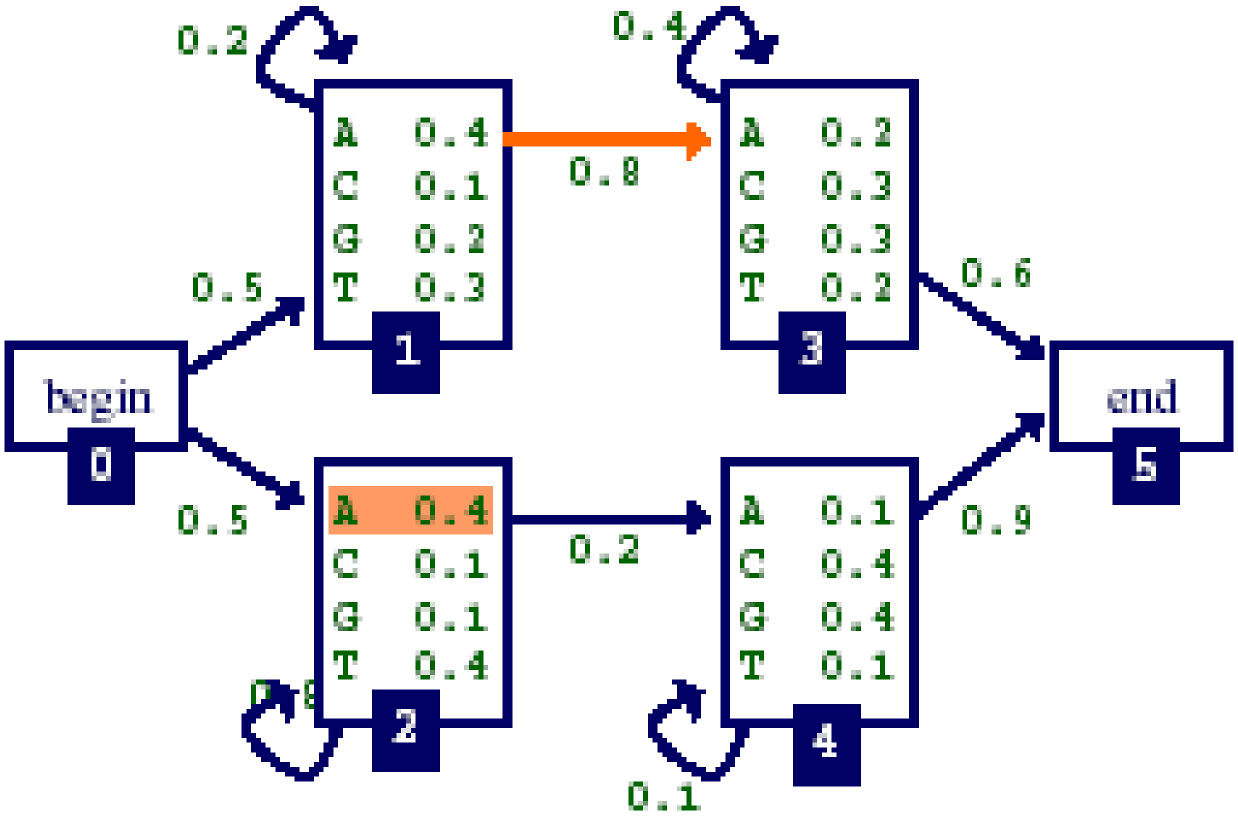
probability of emitting character b in state *k*

A Simple HMM *(again)*

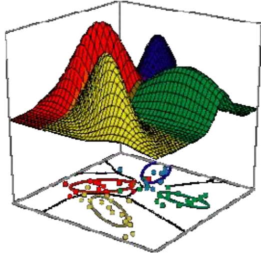


a_{13} = probability of a transition from state 1 to state 3

$e_2(A)$ = probability of emitting character *A* in state 2



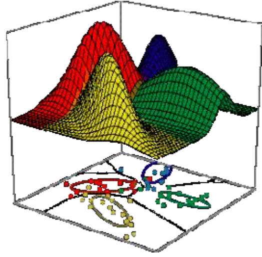
Three Important Questions
(asked 3 ways)



- 1) How likely is a given sequence? *(e.g. motif searching)*
  the Forward algorithm
- 2) What is the most probable "path" for generating a given sequence? *(e.g. multiple sequence alignment)*
  the Viterbi algorithm
- 3) How can we learn the HMM parameters given a set of sequences? *(e.g. training an HMM from a family of seq)*
  the Forward-Backward (Baum-Welch) algorithm

Three Important Questions

(asked 3 ways – Duda terminology)



Once we have an HMM, there are three problems of interest.

(1) The Evaluation Problem

Given an HMM and a sequence of observations, what is the probability that the observations are generated by the model?

(2) The Decoding Problem

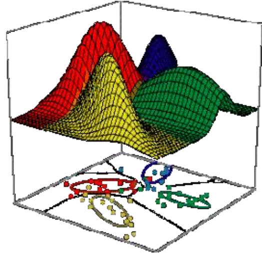
Given a model and a sequence of observations, what is the most likely state sequence in the model that produced the observations?

(3) The Learning Problem

Given a model and a sequence of observations, how should we adjust the model parameters in order to maximize the likelihood of observing the data

Learning problem must be solved, if we want to train an HMM for the subsequent use of recognition tasks.

Three Important Questions
(asked 3 ways)

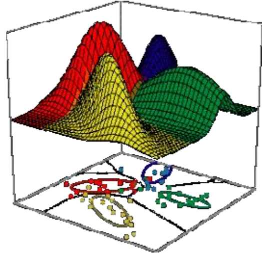


i) Given A+E what is the probability of $(x_1, ..., x_n)$?
Forward algorithm

ii) Given A+E+$(x_1, ..., x_n)$ what is $(\pi_1, ..., \pi_n)$?
Viterbi algorithm

iii) Given $(x_1, ..., x_n)$ what is A+E?
Baum-Welch algorithm

iv) Given A+E+$(x_1, ..., x_n)$ what is the most probable state at step i? **forward-backward algorithm**



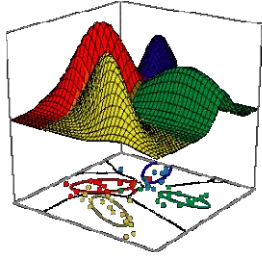
The First Question

- 1) How likely is a given sequence?

$\longrightarrow$ Forward algorithm

- 2) What is the most probable “path” for generating a given sequence?
- 3) How can we learn the HMM parameters given a set of sequences?

How Likely is a Given Sequence?
(for one path through the model)



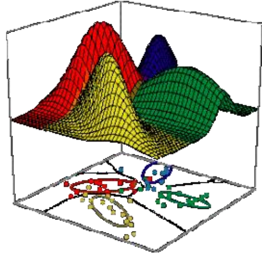
- The probability that the path is taken and the sequence is generated:

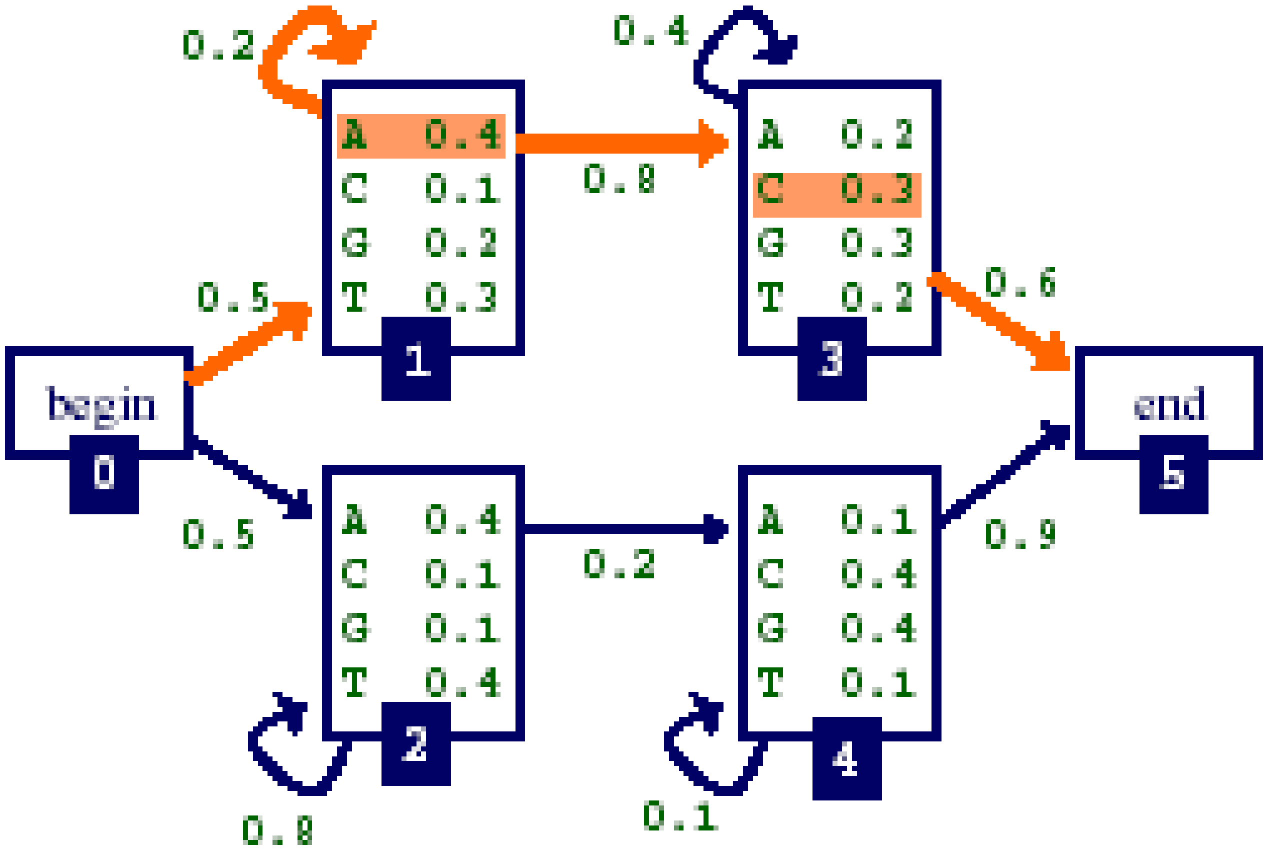
$$\Pr(x_1,\ldots,x_L,\pi_0,\ldots,\pi_N) = a_{0\pi_1}\prod_{i=1}^{L} e_{\pi_i}(x_i)\,a_{\pi_i\pi_{i+1}}$$

  - (assuming begin/end are the only silent states on path)

How Likely is a Given Sequence?

(for one path through the model)



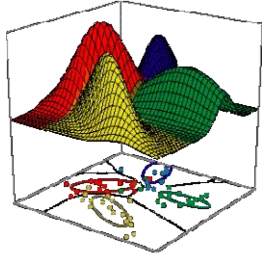


$$P(AAC,\pi) = a_{01} \bullet e_1(A) \bullet a_{11} \bullet e_1(A) \bullet a_{13} \bullet e_3(C) \bullet a_{35}$$

$$= 0.5 \bullet 0.4 \bullet 0.2 \bullet 0.4 \bullet 0.8 \bullet 0.3 \bullet 0.6$$

How Likely is a Given Sequence?

(for ALL paths through the model)



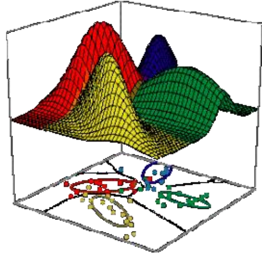
The probability over *all possible* paths is:

$$\Pr(x_1,\ldots,x_L) = \sum_{\pi} \Pr(x_1,\ldots,x_L,\underbrace{\pi_0,\ldots,\pi_N}_{\pi})$$

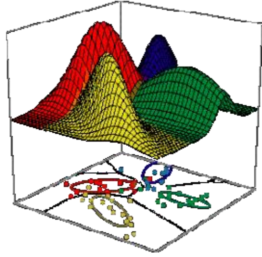
• but the number of paths can be exponential in the length of the sequence...

• the Forward algorithm enables us to compute this efficiently

How Likely is a Given Sequence: The Forward Algorithm

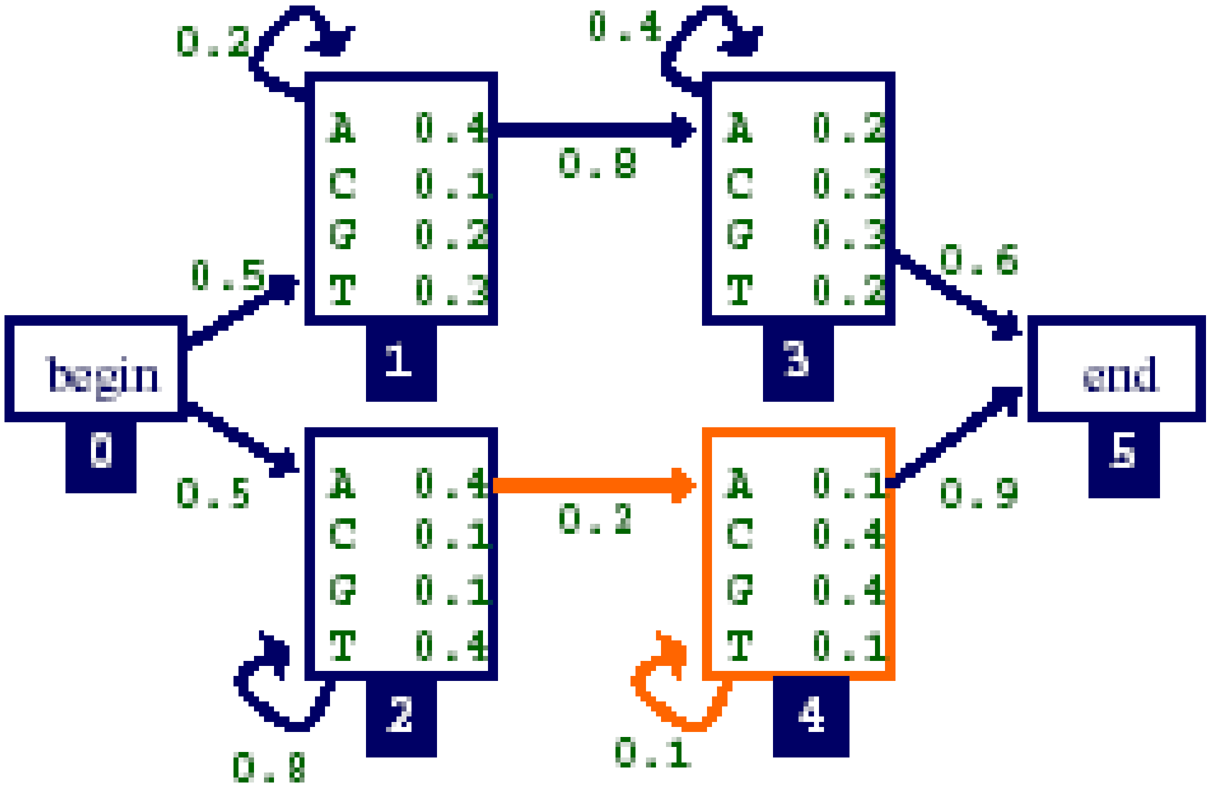


- Define $f_k(i)$ to be the probability of being in state *k at step i*
  - Having observed the first *i* characters of *x*, what is prob of being in state *k?*
- We want to compute $f_N(L)$, the probability of being in the END state having observed all of *x*
- We can define this recursively



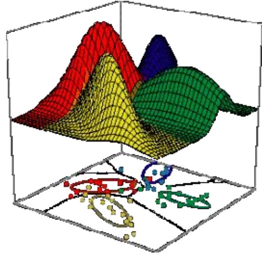
How Likely is a Given Sequence:

Because of the Markov property, we don't have to explicitly enumerate every path – use dynamic programming instead



e.g. compute $f_4(i)$ using $f_2(i\text{-}1)$ and $f_4(i\text{-}1)$

The forward algorithm



- Initialisation:

  $f_0(0) = 1$ (start),

  $f_k(0) = 0$ (for all other silent states, k)

  probability that we're in start state and have observed 0 characters from the sequence

- Recursion:

  $f_l(i) = e_l(i)\Sigma_k f_k(i\text{-}1)a_{kl}$ (emitting states),

  $f_l(i) = \Sigma_k f_k(i)a_{kl}$ (silent states)

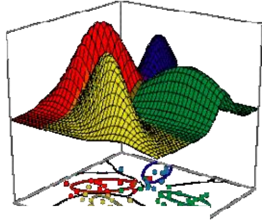
- Termination:

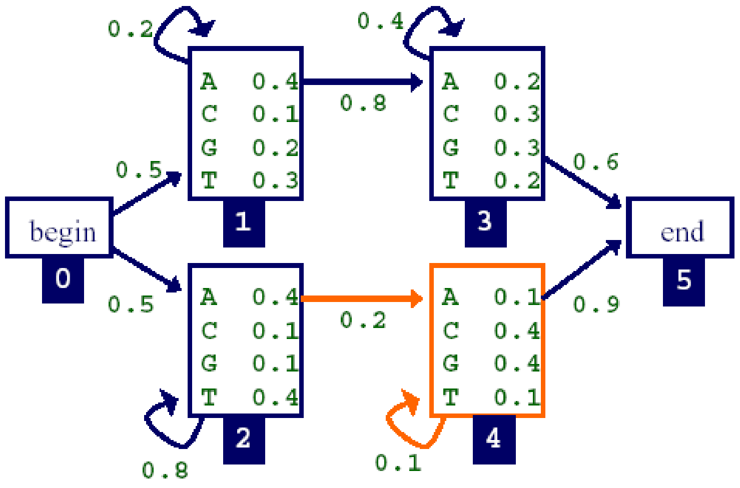
  $\Pr(x) = \Pr(x_1\ldots x_L) = f_N(L) = \Sigma_k f_k(L)a_{kN}$

  probability that we are in the end state and have observed the entire sequence

Recall: $f_k(i)$=prob in state k after emitting i symbols

Forward algorithm example





- given the sequence x = TAGA
- initialization

$$f_0(0) = 1 \qquad f_1(0) = 0 \quad \cdots \quad f_5(0) = 0$$

- computing other values

$$f_1(1) = e_1(T) \times (f_0(0) \times a_{01} + f_1(0) a_{11}) =$$
$$0.3 \times (1 \times 0.5 + 0 \times 0.2) = 0.15$$

$$f_2(1) = 0.4 \times (1 \times 0.5 + 0 \times 0.8)$$

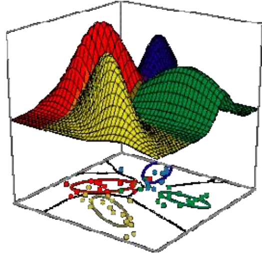
$$f_1(2) = e_1(A) \times (f_0(1) \times a_{01} + f_1(1) a_{11}) =$$
$$0.4 \times (0 \times 0.5 + 0.15 \times 0.2)$$

• • •

$$\Pr(TAGA) = f_5(4) = (f_3(4) \times a_{35} + f_4(4) a_{45})$$

..all the time calculate over all possible ways to get to a considered state

Three Important Questions

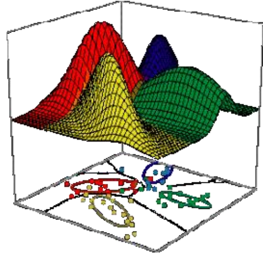


- 1) How likely is a given sequence?
- 2) What is the most probable "path" for generating a given sequence?

$\longrightarrow$ Viterbi algorithm

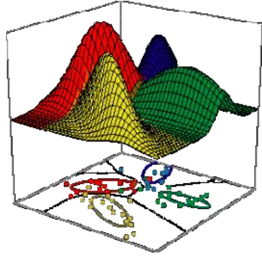
- 3) How can we learn the HMM parameters given a set of sequences?

Finding the Most Probable Path: The Viterbi Algorithm



- Define $v_k(i)$ to be the probability of the most probable path accounting for the first i characters of x and ending in state k
- We want to compute $v_N(L)$, the probability of the most probable path accounting for all of the sequence and ending in the end state
  - Can be defined recursively
  - Use dynamic programming to find $v_N(L)$ efficiently

Finding the Most Probable Path: The Viterbi Algorithm



Initialisation:

$v_0(0) = 1$ (start), $v_k(0) = 0$ (for all non-silent states)

Recursion for emitting states (i =1…L):

$$v_l(i) = e_l(x_i) \max_k \left[v_k(i-1) a_{kl} \right]$$

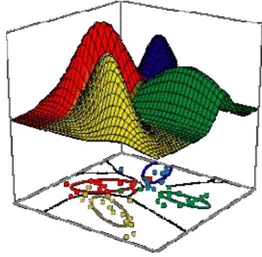
$$\mathrm{ptr}_l(i) = \arg\max_k \left[v_k(i-1) a_{kl} \right]$$

Recursion for silent states:

$$v_l(i) = \max_k \left[v_k(i) a_{kl} \right]$$

$$\mathrm{ptr}_l(i) = \arg\max_k \left[v_k(i) a_{kl} \right]$$

Finding the Most Probable Path: The Viterbi Algorithm



- termination:

$$\Pr(x,\pi)=\max_k\left(v_k(L)a_{kN}\right)$$

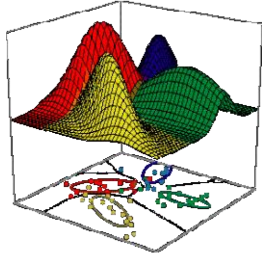
What is the probability of the most probable path to get here?

$$\pi_{\mathrm{L}}=\operatorname*{argmax}_k\left(v_k(L)a_{kN}\right)$$

From which state did we come, in the most probable path?

- traceback: follow pointers back starting at π_{L}

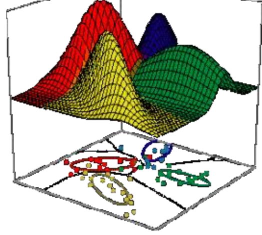
Three Important Questions



- 1) How likely is a given sequence?
- 2) What is the most probable “path” for generating a given sequence?
- 3) How can we learn the HMM parameters given a set of sequences?

$\longrightarrow$ The Baum-Welch Algorithm

The Learning Problem

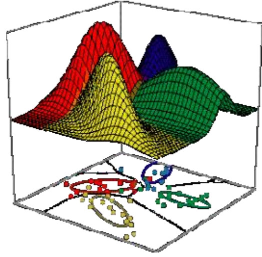


Generally, the learning problem is how to adjust the HMM parameters, so that the given set of observations (called the training set) is represented by the model in the best way for the intended application.

Thus it would be clear that the “quantity” we wish to optimize during the learning process can be different from application to application. In other words there may be several optimization criteria for learning, out of which a suitable one is selected depending on the application.

There are two main optimization criteria found in the literature; Maximum Likelihood (ML) and Maximum Mutual Information (MMI).

The Learning Task

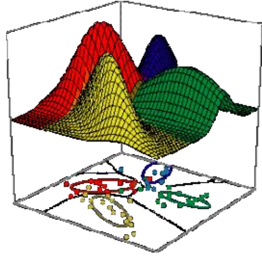


- Given:
  - a model
  - a set of sequences (the training set)

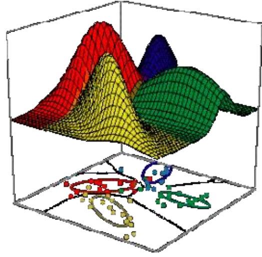
- Do:
  - find the most likely parameters to explain the training sequences

- The goal is to find a model that *generalizes* well to sequences we haven't seen before

Learning Parameters



- If we know the state path for each training sequence, learning the model parameters is simple
  - – no hidden state during training
  - – count how often each transmission & emission is used
  - – normalize/smooth to get probabilities a_{ij} & $e_i(x)$
  - – process just like it was for Markov chain models
- If we don’t know the path for each training sequence, how can we determine the counts?
  - – key insight: estimate the counts by considering every path weighted by its probability

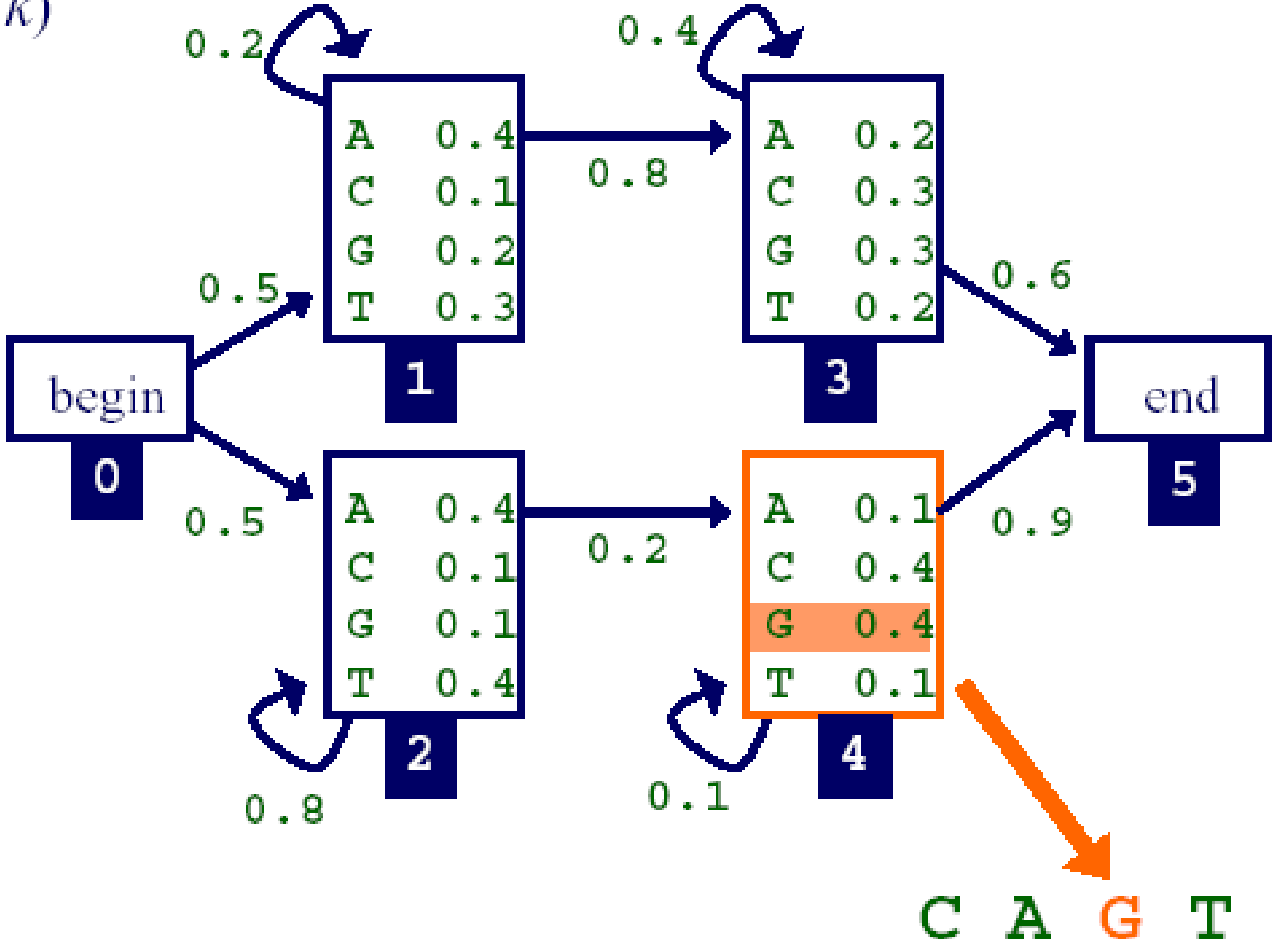


Learning Parameters: The Baum-Welch Algorithm

- An EM (expectation maximization) approach, a forward-backward algorithm
- Algorithm sketch:
  - – initialize parameters of model
  - – iterate until convergence
- Calculate the *expected* number of times each transition or emission is used
- Adjust the parameters to *maximize* the likelihood of these expected values
- Baum-Welch has as important feature that it always converges

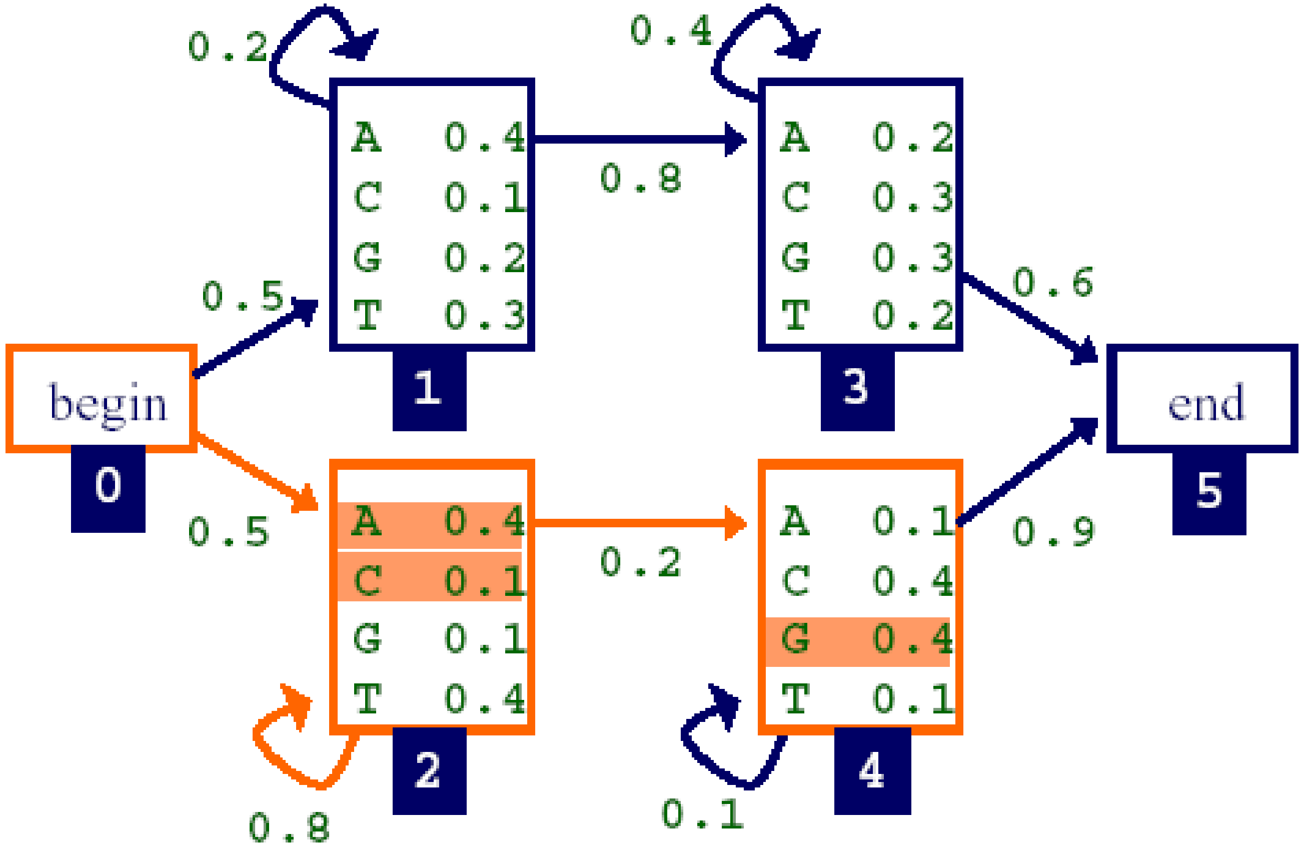
The Expectation step

- we want to know the probability of producing sequence x with the i th symbol being produced by state k (for all x, i and k)



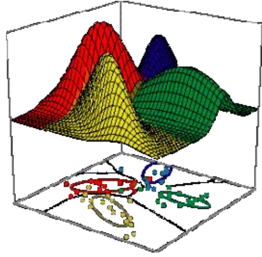
The Expectation step

- the forward algorithm gives us $f_k(i)$, the probability of being in state *k* having observed the first *i* characters of *x*

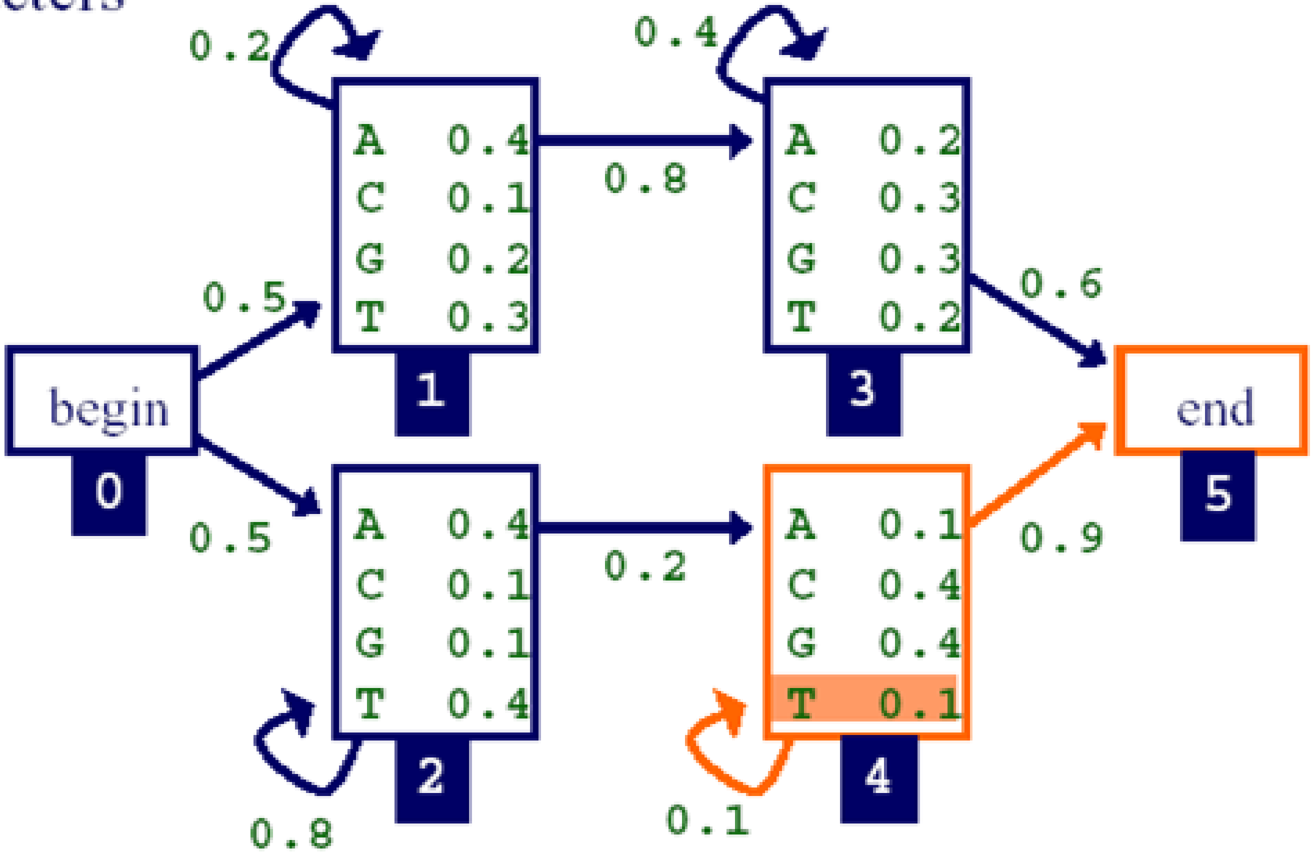


C A G T

The Expectation step



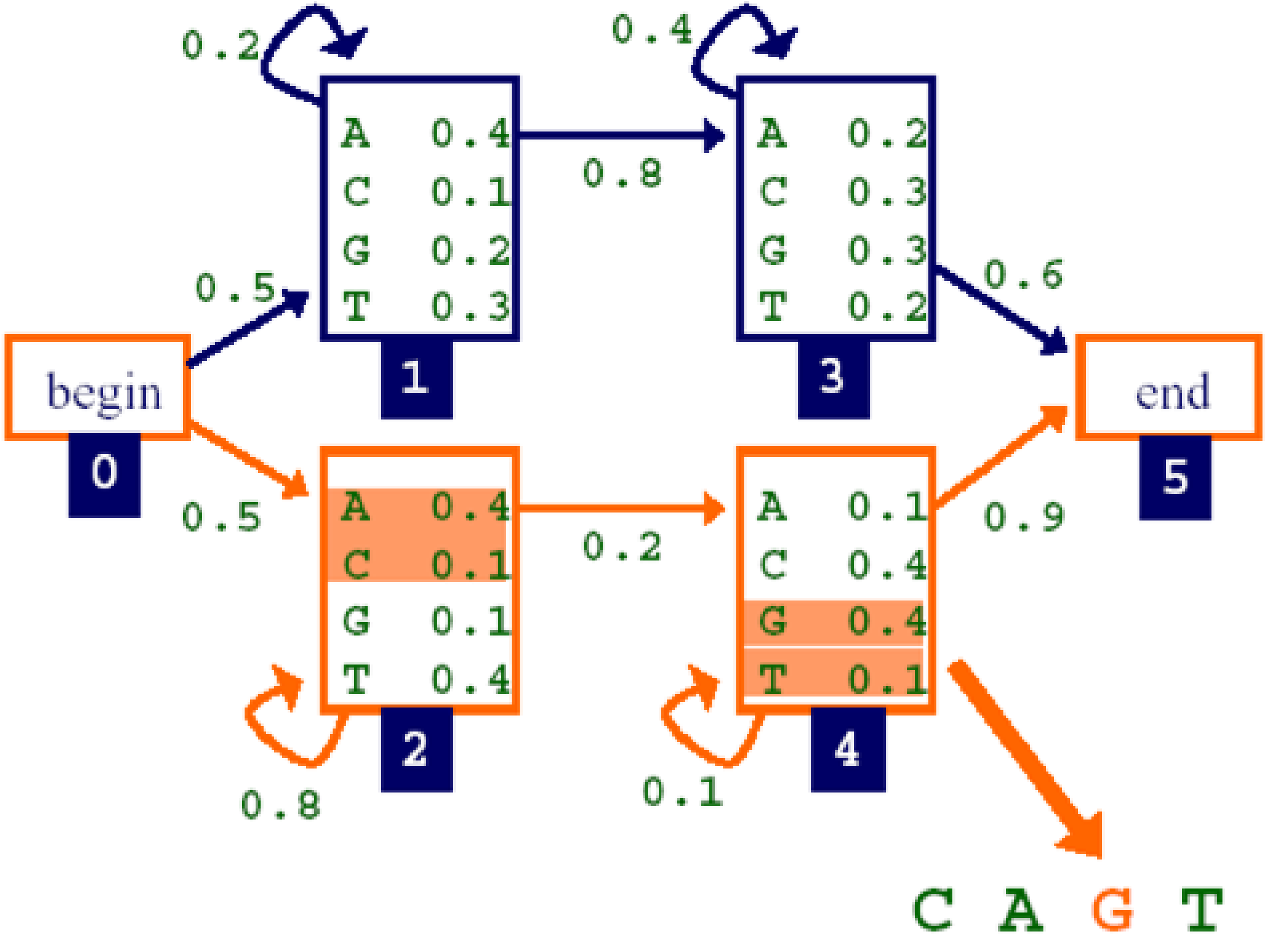
- the *backward algorithm* gives us $b_k(i)$, the probability of observing the rest of x, given that we're in state *k* after *i* characters



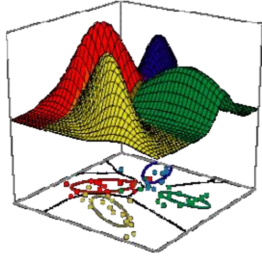
C A G T

The Expectation step

Putting the forward and backward algorithms together, we can compute the probability of producing sequence x with the ith symbol being produced by the kth state



The Expectation step

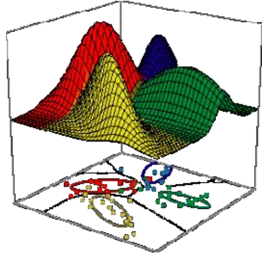


- First, we need to know the probability of the i^{th} symbol being produced by state k, given sequence x:

$$\Pr(\pi_i = k \mid x)$$

- Given this we can compute our expected counts for state transitions & character emissions

The Expectation step



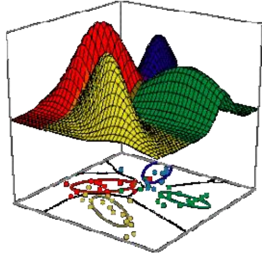
- the probability of ~~of~~ producing x with the i th symbol being produced by state k is

$$\Pr(\pi_i = k, x) = \Pr(x_1 \ldots x_i, \pi_i = k) \times \Pr(x_{i+1} \ldots x_L \mid \pi_i = k)$$

- the first term is $f_k(i)$, computed by the forward algorithm

- the second term is $b_k(i)$, computed by the backward algorithm

The Backward Algorithm



- initialization:

$$b_k(L) = a_{kN}$$

for states with a transition to *end* state

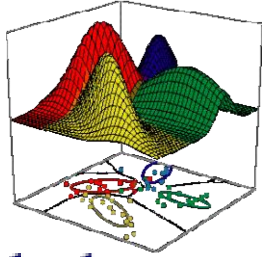
- recursion ($i = L \ldots 1$):

$$b_k(i) = \sum_l \begin{cases} a_{kl} b_l(i), & \text{if } l \text{ is silent state} \\ a_{kl} e_l(x_{i+1}) b_l(i+1), & \text{otherwise} \end{cases}$$

- termination:

$$\Pr(x) = \Pr(x_1 \ldots x_L) = \sum_l \begin{cases} a_{0l} b_l(0), & \text{if } l \text{ is silent state} \\ a_{0l} e_l(x_1) b_l(1), & \text{otherwise} \end{cases}$$

The Expectation step

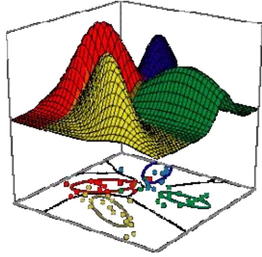


- now we can calculate the probability of the *i* th symbol being produced by state *k*, given *x*

$$\Pr(\pi_i = k \mid x) = \frac{\Pr(\pi_i = k, x)}{\Pr(x)}$$

$$\boxed{= \frac{f_k(i) b_k(i)}{\Pr(x)}}$$

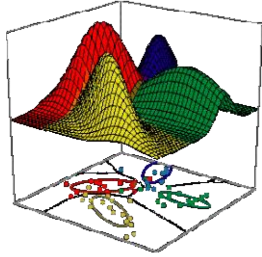
The Expectation step



- now we can calculate the expected number of times letter c is emitted by state k

$$n_{k,c} = \underbrace{\sum_{x}}_{\text{sum over sequences}} \frac{1}{\Pr(x)} \underbrace{\sum_{\{i|x_i=c\}}}_{\text{sum over positions where } c \text{ occurs in x}} f_k(i) b_k(i)$$

The Expectation step



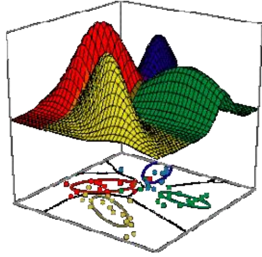
- and we can calculate the expected number of times that the transition from *k* to *l* is used

$$n_{k\to l} = \sum_x \frac{\sum_i f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)}{\Pr(x)}$$

- or if *l* is a silent state

$$n_{k\to l} = \sum_x \frac{\sum_i f_k(i) a_{kl} b_l(i)}{\Pr(x)}$$

The Maximization step



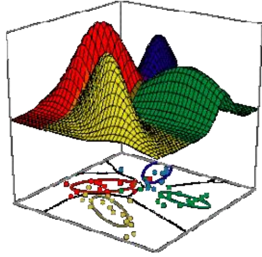
- Let $n_{k,c}$ be the expected number of emissions of c from state k for the training set
- estimate new emission parameters by:

$$e_k(c) = \frac{n_{k,c}}{\sum_{c'} n_{k,c'}}$$

..make it a probability through dividing by the total number of emissions out of state *k*

- just like in the simple case
- but typically we'll do some "smoothing" (e.g. add pseudocounts)

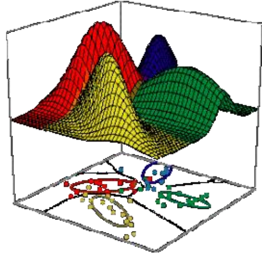
The Maximization step



- let $n_{k \to l}$ be the expected number of transitions from state k to state l for the training set
- estimate new transition parameters by:

$$a_{kl} = \frac{n_{k \to l}}{\sum_m n_{k \to m}}$$

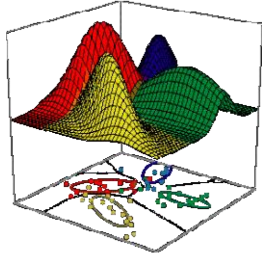
..make it a probability through dividing by the total number of transitions out of state *k*



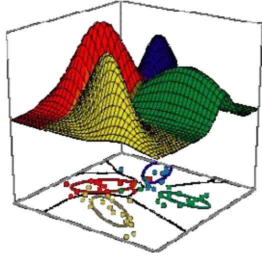
The Baum-Welch Algorithm

- Initialize parameters of model
- Iterate until convergence
  - – calculate the *expected* number of times each transition or emission is used
  - – adjust the parameters to *maximize* the likelihood of these expected values
- This algorithm will converge to a *local* maximum (in the likelihood of the data given the model)
- Usually in a fairly small number of iterations

HMMs in Bioinformatics

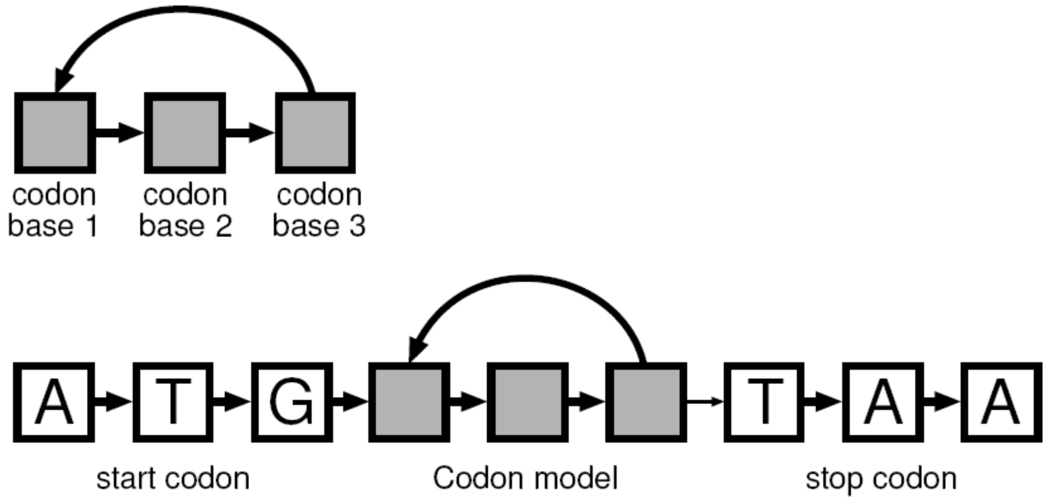


- Used extensively in Prokaryotic and Eukaryotic Gene Prediction
- Used to create Sequence Profiles and to classify sequences into families
- Used in Multiple Sequence Alignment
- Used in detecting and modeling protein domains or modules
- Used in recognizing protein topology

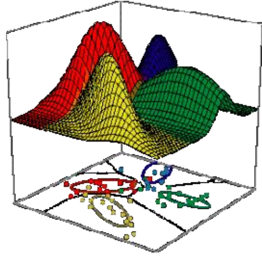


HMM for gene finding

- Simplest model:



HMM for gene finding



- More realistic

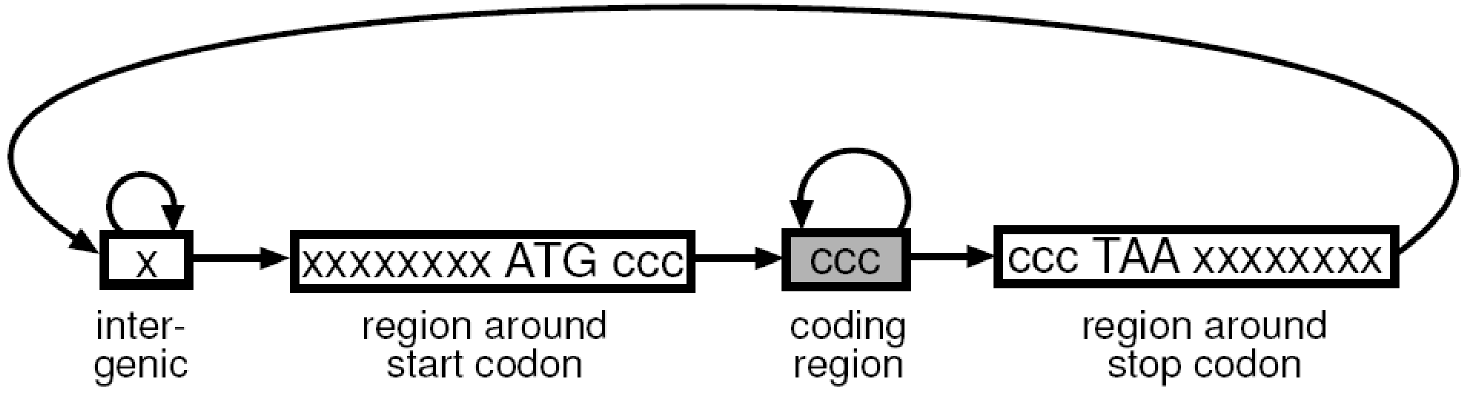
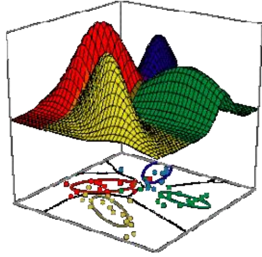


Figure 4.11: A hidden Markov model for unspliced genes. In this drawing an ‘x’ means a state for non-coding DNA, and a ‘c’ a state for coding DNA. Only one of the three possible stop codons are shown in the model of the region around the stop codon.

Figure from A. Krogh, An Introduction to Hidden Markov Models for Biological Sequences



HMM for Eukaryotic Gene Finding

- ‘coding region’ HMM permits splicing in 3 frames

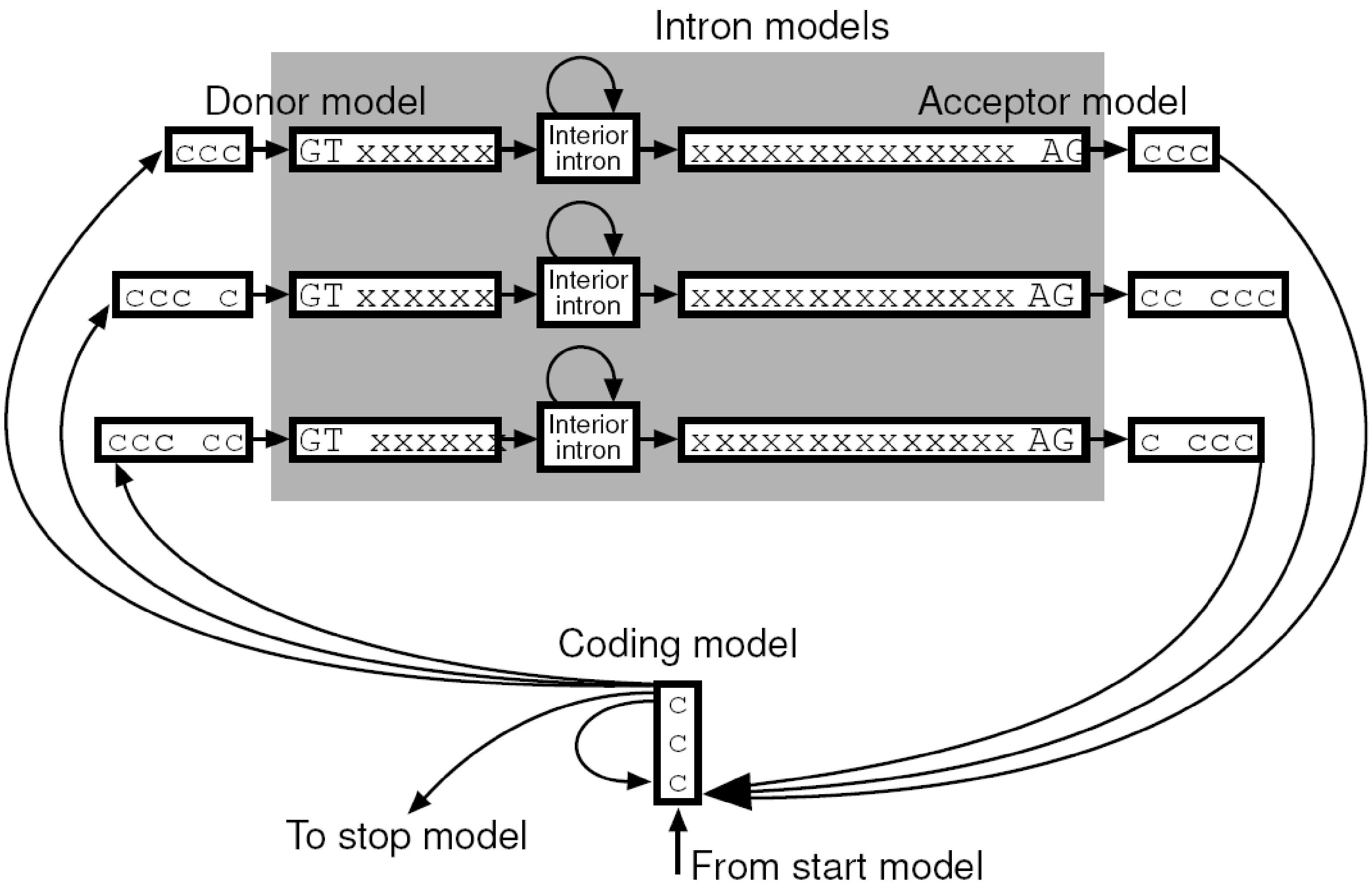
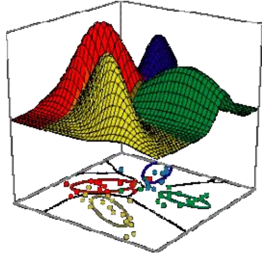
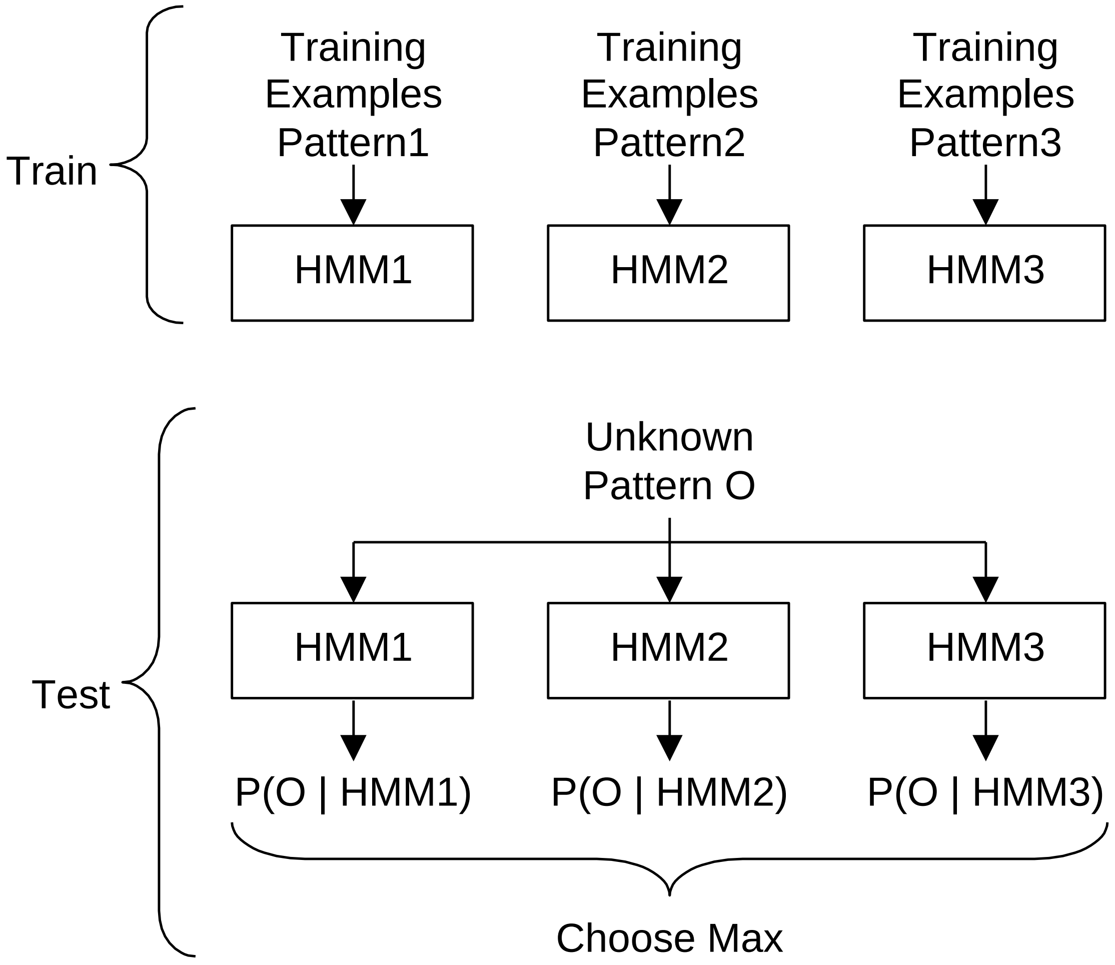


Figure from A. Krogh, An Introduction to Hidden Markov Models for Biological Sequences

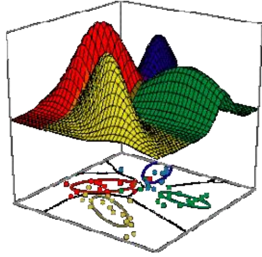
HMM-based homology searching





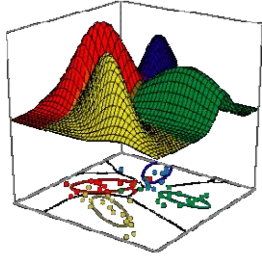
A.D.C. Chan, “What is a HMM tutorial,” 2000.

HMM-based homology searching

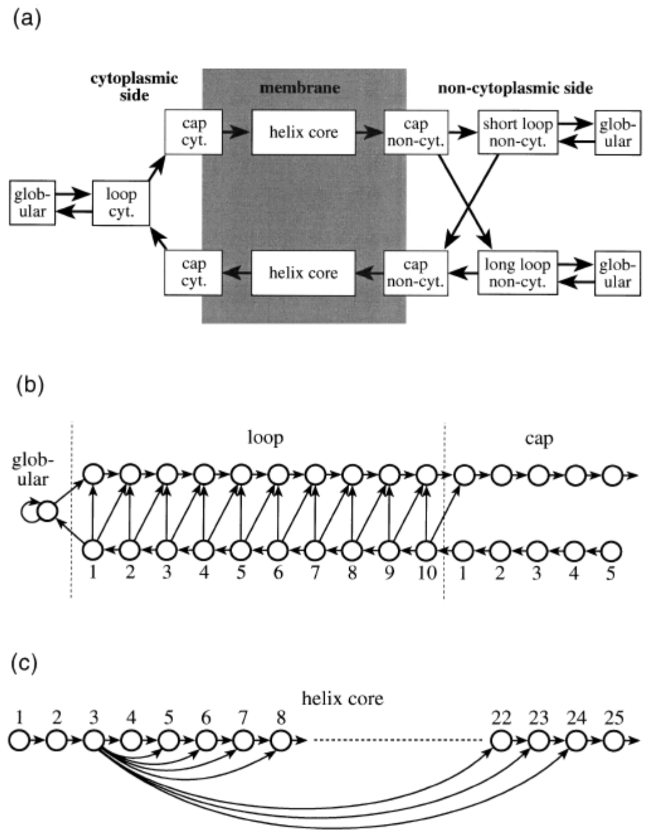


- Most widely used HMM-based profile searching tools currently are SAM-T2K (Karplus *et al.*, 1998) and HMMER2 (Eddy, 1998)
- formal probabilistic basis and consistent theory behind gap and insertion scores
- More powerful than linear motifs since you can capture likelihood of a sequence satisfying motif.
- Pfam is a database of HMM’s for different protein families. (http://pfam.sanger.ac.uk/)

Other applications of HMM



- Transmembrane helix predictions (TMHMM)



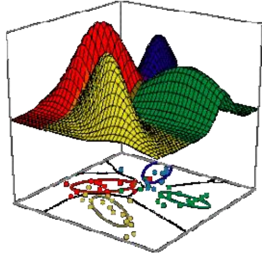
Each box is an HMM

Short loop model

Helix core model (len = 5-26)

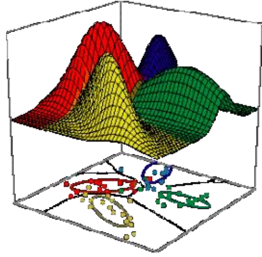
From Krogh et al, “Predicting transmembrane helix topology with a HMM,” JMB 305:567-580, 1997

Other applications of HMM



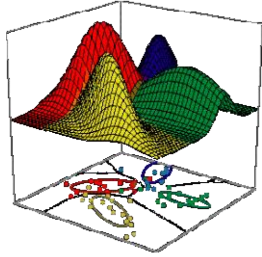
- Protein secondary structure prediction
  - SAM-T06: http://www.soe.ucsc.edu/compbio/HMM-apps/HMM-applications.html)
- Protein family prediction (Krough and others)
- Combine HMM and ANN
  - ANN does not handle sequence length variation well…
  - Fit HMM to data, then pass model parameters as inputs to ANN
- Speech processing (first ‘big’ application)

Key References



- L.R. Rabiner and B.H. Juang (1986). “An Introduction to Hidden Markov Models,” IEEE ASSP Magazine, p 4-15.
- L. R. Rabiner (1989). "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proceedings of the IEEE, vol 77, no 2, 257-287.
- A. Krogh (1998). “An Introduction to Hidden Markov Models for Biological Sequences”, in Computational Methods in Molecular Biology, Salzberg et al Editors, p45-63, Elsevier.

Expectation-Minimization (Duda 3.9)



- Generalization of Baum-Welch algorithm
- Iterative parameter estimation technique when some variables are missing/latent/unobservable
  - Simplify the problem by assuming these variables are known/observed, estimate parameters, then maximizing likelihood of assumed values given updated parameters.
- Has several applications:
  - Duda et al illustrates the use of EM to extend ML to estimate parameters governing a distribution where some samples have missing data
  - Also used in machine vision, psychometrics, natural language processing, medical image reconstruction.