Lecture 9

Markov Chain Models

Algorithms in Sequence Analysis

Mentoring Introductory Session

Marlies van Daalen & Anton Feenstra

Tomorrow (Tue 26 Nov) from 13.00-14.00 in NU-4A06.

Algorithms in Sequence Analysis

Practical assignment 3: HMM

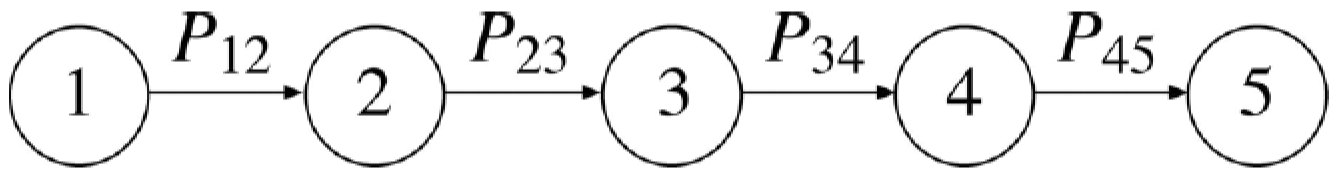
Hidden Markov Models (involves programming)

1. **Today:** ‘Paper’ exercise

2. **From tomorrow:** Programming:
    1. Viterbi algorithm
    2. Forward algorithm
    3. Forward-backward (Baum Welch) algorithm

Markov models

- A Markov model, also known as *Markov chain,* describes a sequence of events that occur one after another in a chain.
- Each event determines the probability of the next event. A Markov chain can be considered as a process that moves in one direction from one state to the next with a certain probability, which is known as *transition probability*


$$\begin{tikzcd} 1 \arrow[r, "P_{12}"] & 2 \arrow[r, "P_{23}"] & 3 \arrow[r, "P_{34}"] & 4 \arrow[r, "P_{45}"] & 5 \end{tikzcd}$$

A widely used machine learning approach: Markov models

Contents

Biological theme: gene prediction

Statistical theory behind Markov models: Bayesian statistics

For independent events X and Y:

$P(X, Y) = P(X) P(Y)$

Conditional probabilities - two events are not independent:

$P(Y \mid X)$ is **conditional probability** of Y given X

If X and Y are independent:

$P(Y \mid X) = P(Y)$ and $P(X \mid Y) = P(X)$

See Durbin et al., Chapter 1

Statistical theory behind Markov models: Bayesian statistics

Joint probabilities $P(X, Y)$ are then:

$P(X, Y) = P(Y \mid X)\, P(X)$ (if X is first) or

$P(X, Y) = P(X \mid Y)\, P(Y)$ (if Y is first)

If event X and Y are dependent and we know the conditional or joint probabilities, we get the **marginal probability**:

$P(X) = \Sigma_Y P(X, Y) = \Sigma_Y P(X \mid Y)\, P(Y)$ or

$P(Y) = \Sigma_X P(X, Y) = \Sigma_X P(Y \mid X)\, P(X)$

Setting $P(Y \mid X)\, P(X) = P(X \mid Y)\, P(Y)$ gives

$\mathbf{P(Y \mid X) / P(Y) = P(X \mid Y) / P(X)}$

or $\mathbf{P(Y \mid X) = P(X \mid Y)\, P(Y) / P(X)}$

See Durbin et al., Chapter 1

Now we really turn to Bayesian statistics

Suppose we have a number of models, where the model differences can be described by parameter *y*. Given a distribution of the parameters *y:*

P(*Y*) is probability of randomly selecting a model with parameter *Y* (where *Y* is a specific parameter out of the distribution *y*).

Each model produces observed data *X* depending on parameter *y*

Typically we want to use the observed data *X* to deduce the value *Y* from the model (sometimes we even may want to obtain the distribution *y*)

- We'll look at the Dishonest Casino example (in two slides)

Bayesian statistics

P(Y) is probability of randomly selecting a model with parameter Y.

prior probability

$$\mathbf{P}(Y \mid X) = \frac{\mathbf{P}(X \mid Y)\ \mathbf{P}(Y)}{\mathbf{P}(X)}\ ,$$

Posterior probability

Marginal probability

where P(Y | X) is the *posterior probability* given X, P(Y) is the *prior probability* of the model with parameter Y, P(X | Y) is the probability of obtaining data X given the model having parameter Y (*likelihood* of hypothesis that model has parameter Y), and P(X) is the *marginal probability* of X.

[…on an earlier slide we had

$$P(Y \mid X) / P(Y) = P(X \mid Y) / P(X) \ ]$$

Dishonest casino example

1% of dice in the casino is loaded, giving a chance of 50% for a six to come up. Loaded and unloaded dice are indistinguisible. There are two models: loaded or unloaded die.
Question: given you throw three 6s in a row with a given die, how likely is it that the die is loaded, so what is the chance for the unloaded die model?

$$\underbrace{P(D_{loaded} \mid \textit{3 sixes})}_{\text{posterior}} = \frac{P(\textit{3 sixes} \mid D_{loaded})\ \overbrace{P(D_{loaded})}^{\text{prior}}}{\underbrace{P(\textit{3 sixes})}_{\text{marginal}}} =$$

$$= \frac{0.5^3 * 0.01}{0.5^3 * 0.01 + (1/6)^3 * 0.99} = 0.21$$

So, the posterior probability of a dishonest die is 21-fold higher than its prior probability, given the model produces three 6s.

$P(D_{loaded} \mid \textit{4 sixes}) = 0.45$, $P(D_{loaded} \mid \textit{5 sixes}) = 0.71$

Example taken from *Biological Sequence Analysis* by Durbin, Eddy, Krogh and Mitchison (1998)

Bayesian parameter estimation

Baysian estimation can also be used to get values for continuous parameters, particularly if prior knowledge should be used to constrain estimates :

$$\underbrace{P(\theta|D)}_{\text{posterior}} = \frac{\overbrace{P(\theta)}^{\text{prior}}P(D|\theta)}{\underbrace{\int_{\theta'} P(\theta')P(D|\theta')}_{\text{marginal}}}$$

Since parameters are usually continuous rather than discrete quantities, the denominator is now an integral rather than a sum:

$$P(D) = \int_{\theta'} P(\theta')P(D|\theta')$$

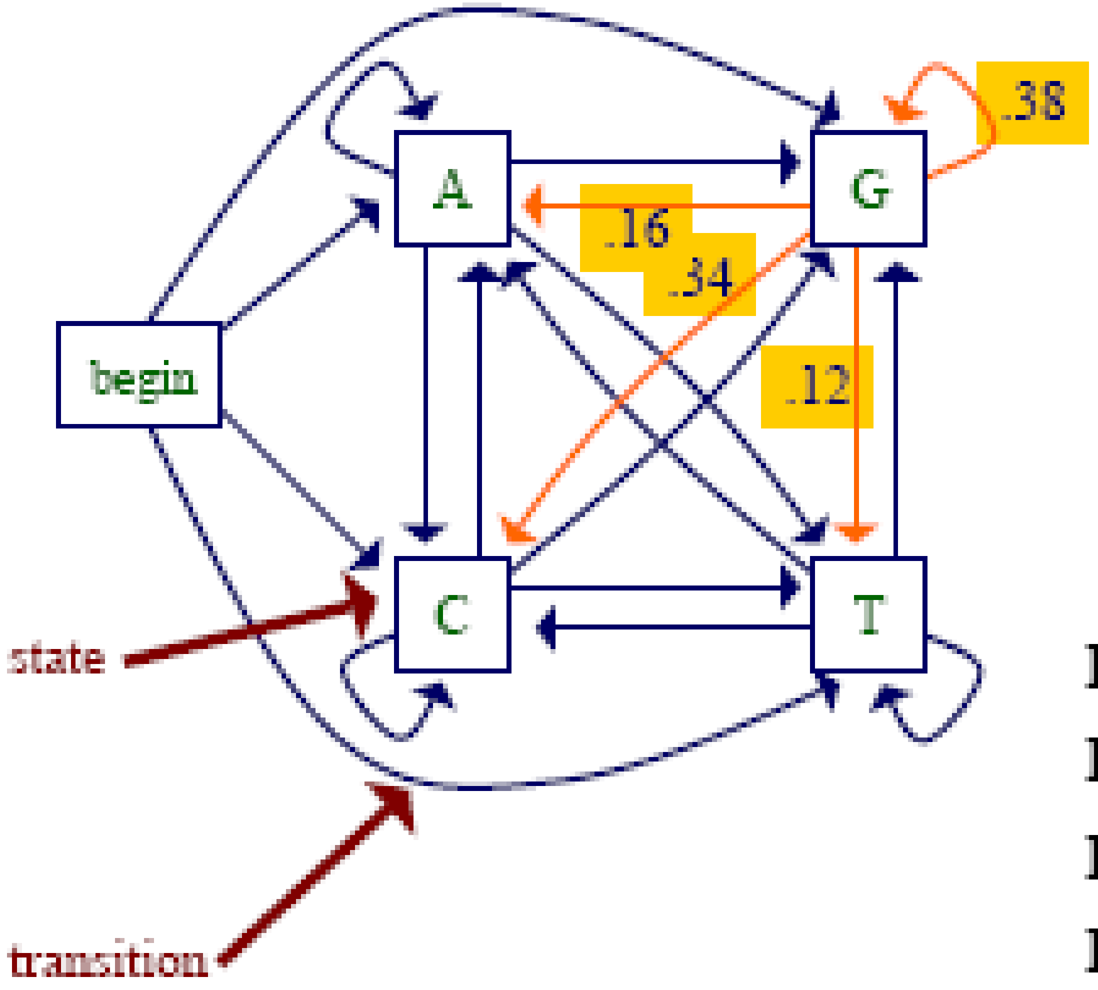
Think about how the equation for the previous die example would change if there were (many) more than two types of die

Biological Sequence Analysis by Durbin, Eddy, Krogh and Mitchison (1998) – Chapter 1

Some issues with Bayesian statistics

- Usually, the prior probability is assigned as a standard probability distribution
    - Often we don’t know the true distribution but as long as there are enough data we can still learn the proper posterior
- Since prior and likelihood are multiplied, some priors lead to posteriors having a standard form.
    - If this form has the same distribution as the prior, then the prior is *conjugate*
    - Conjugate distributions include the normal (or Gaussian); binomial and beta; multinomial and Dirichlet distributions.
- Importantly, conclusions from a Bayesian model depend critically on the prior probability used (this will come back during your HMM practical).

Markov Chain Models have conditional probabilities



transition probabilities

$$\Pr(x_i = a \mid x_{i-1} = g) = 0.16$$
$$\Pr(x_i = c \mid x_{i-1} = g) = 0.34$$
$$\Pr(x_i = g \mid x_{i-1} = g) = 0.38$$
$$\Pr(x_i = t \mid x_{i-1} = g) = 0.12$$

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
    - the transitions are given as a *N*N transition matrix*, where *N* is the number of states

Markov Chain Models

1. S – observations (i.e., the input to the model)
   - $x_1,\ldots,x_n$ – sequence of observations
2. Q - states
   - $\pi_1,\ldots,\pi_n$ – sequence of states
   - $f=(f_1,\ldots,f_N)^T$ - initial probability of states
3. $A = (a_{ij})$ – transition matrix

Markov Chain Models

calculating probabilities

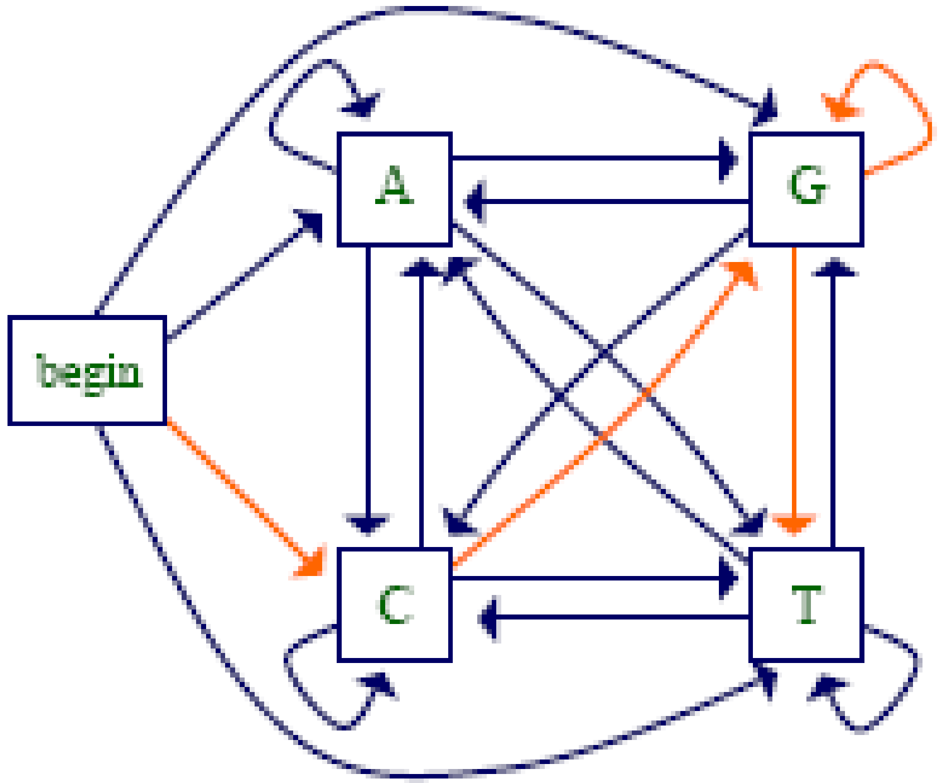
- given some sequence *x* of length *L*, we can ask how probable the sequence is given our model
- for any probabilistic model of sequences, we can write this probability as

$$\Pr(x) = \Pr(x_L, x_{L-1}, \ldots, x_1)$$
$$= \Pr(x_L | x_{L-1}, \ldots, x_1) \Pr(x_{L-1} | x_{L-2}, \ldots, x_1) \ldots \Pr(x_1)$$

- **key property** of a **1st order** Markov chain: the probability of each X_i depends only on X_{i-1} The previous state

$$\Pr(x) = \Pr(x_L | x_{L-1}) \Pr(x_{L-1} | x_{L-2}) \ldots \Pr(x_2 | x_1) \Pr(x_1)$$
$$= \Pr(x_1) \prod_{i=2}^{L} \Pr(x_i | x_{i-1})$$

Markov Chain Models



Calculate the probabilty of a sequence *X* by multiplying the transition probabilities along the associated path through the Markov model

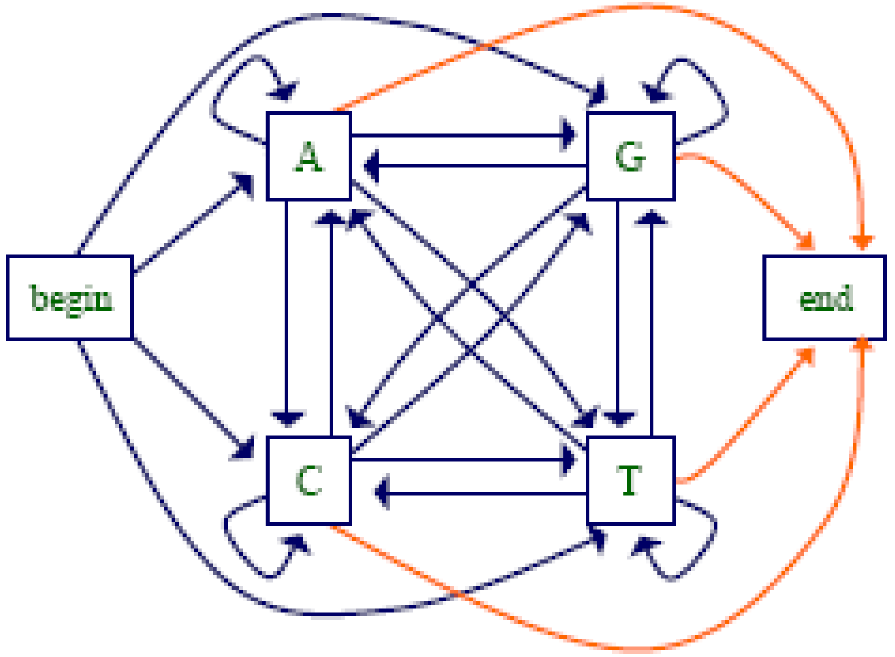
First order: P*(cggt)* = P(*c*|begin)P(*g*|*c*)P(*g*|*g*)P*(t*|*g*)

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

Transition from this state to end state is higher



Markov Chain Models

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

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

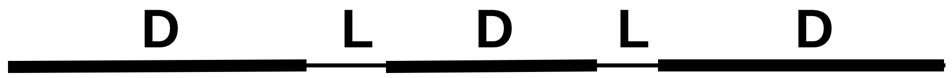
Similarly we can denote the probability of a sequence *x* as

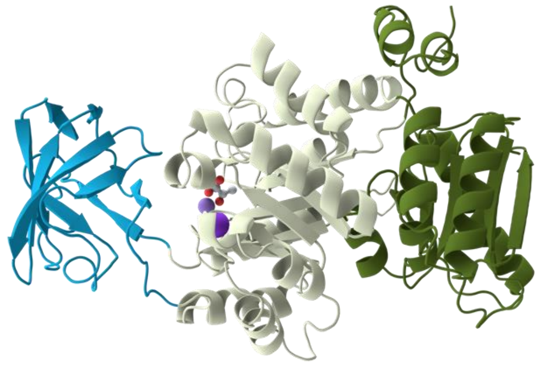
$$\Pr(x_1, x_2, .., x_n) = a_{\mathrm{B}x_i} \prod_{i=2}^{L} a_{x_{i-1}x_i} = \Pr(x_1) \prod_{i=2}^{L} \Pr(x_i \mid x_{i-1})$$

Where a_{Bx_i} represents the transition from the *begin* state to x_1

Third Course Assignment
Practical Hidden Markov Models

- You will construct your own HMM to predict domain/linker structures in protein sequences





- The best way to learn about an algorithm is by doing...

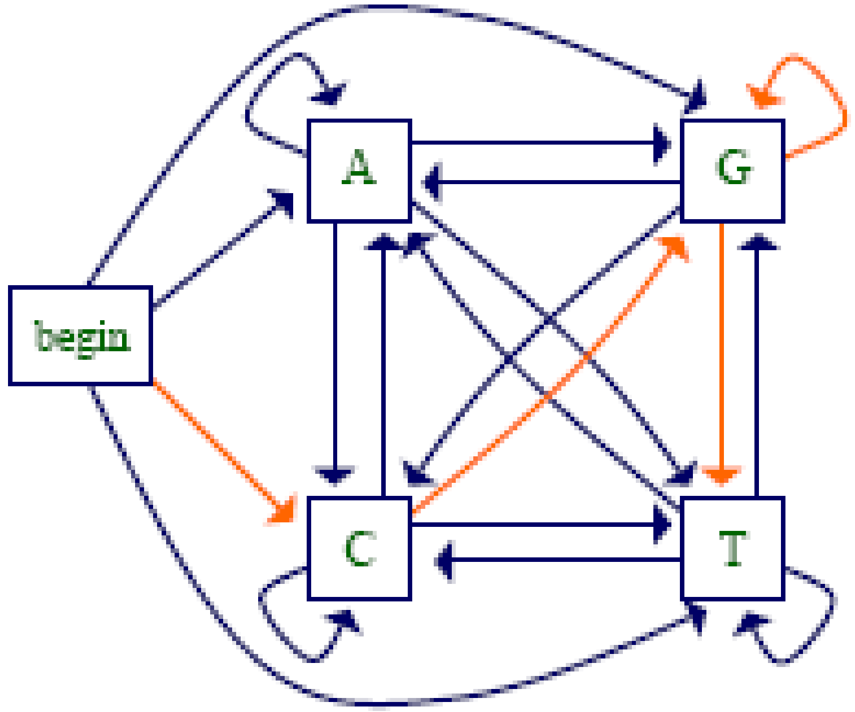
Preamble for your HMM practical: Hidden Markov Models (HMMs)

1. S – observations (*i.e.*, input to the model)
   - $x_1,\ldots,x_n$ – sequence of observations
2. Q - states
   - $\pi_1,\ldots,\pi_n$ – hidden sequence of states
   - $f=(f_1,\ldots,f_N)^T$ - initial probability of states
3. $A = (a_{ij})$ – transition matrix

and now extra

4. $E = (e_i(x))$ – emission probabilities

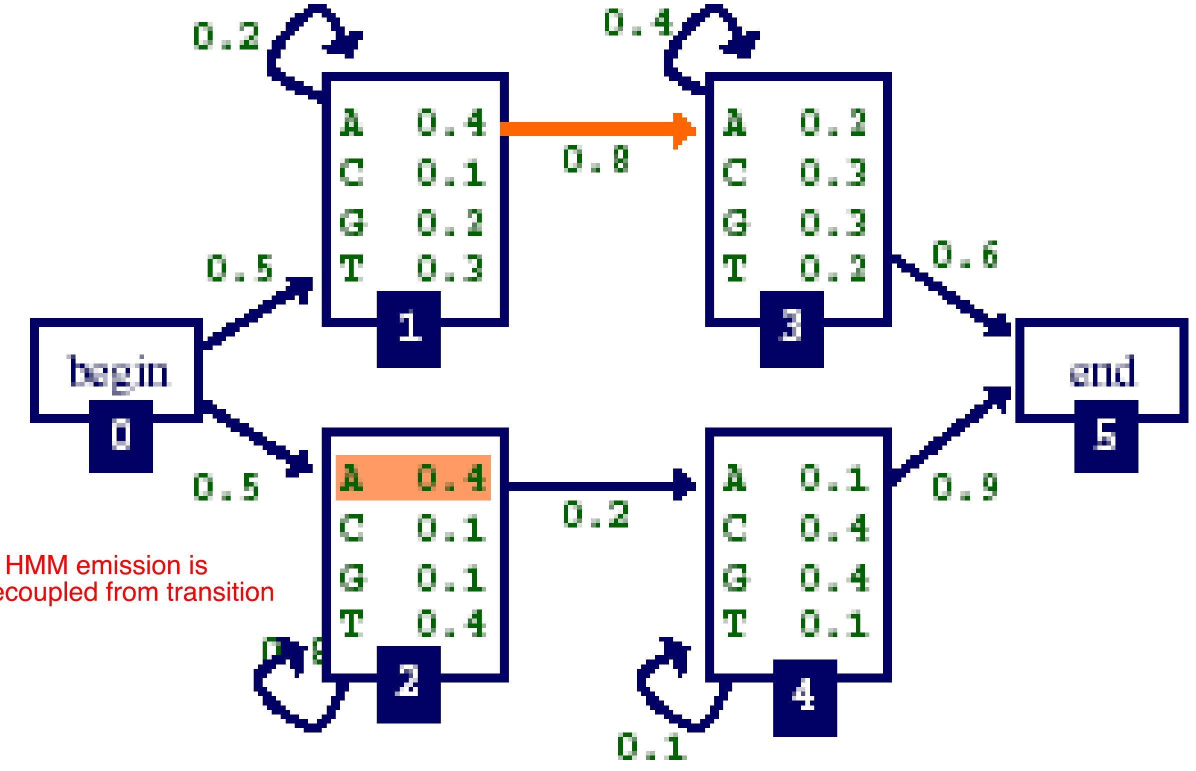
A Simple Markov Model



Symbol emitted is coupled to a state; there are only transition probabilities

A Simple HMM

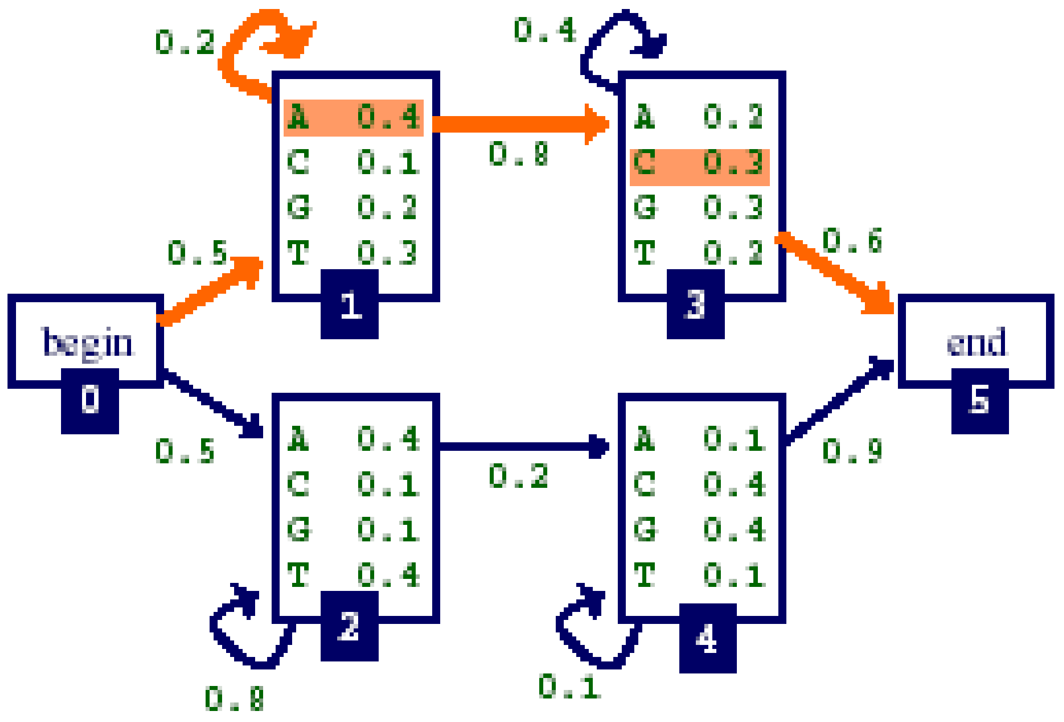
a_{13}	probability of a transition from state 1 to state 3
$e_2(A)$	probability of emitting character *A* in state 2



In HMM emission is decoupled from transition

Symbol emitted is **decoupled** from the state it is emitted by – this is the hidden aspect

What is the most probable path π of sequence AAC?



$$\Pr(AAC, \pi = \pi_0\pi_1\pi_1\pi_3\pi_5) = a_{01} \times e_1(A) \times a_{11} \times e_1(A) \times a_{13} \times e_3(C) \times a_{35}$$
$$= 0.5 \times 0.4 \times 0.2 \times 0.4 \times 0.8 \times 0.3 \times 0.6 = 0.002304$$

$$\Pr(AAC, \pi = \pi_0\pi_1\pi_3\pi_3\pi_5) = a_{01} \times e_1(A) \times a_{13} \times e_3(A) \times a_{33} \times e_3(C) \times a_{35}$$
$$= 0.5 \times 0.4 \times 0.8 \times 0.2 \times 0.4 \times 0.3 \times 0.6 = 0.002304$$
$$\Pr(AAC, \pi = \pi_0\pi_2\pi_2\pi_4\pi_5) = a_{02} \times e_2(A) \times a_{22} \times e_2(A) \times a_{24} \times e_4(C) \times a_{45}$$
$$= 0.5 \times 0.4 \times 0.8 \times 0.4 \times 0.2 \times 0.4 \times 0.9 = 0.004608$$

Many alternative paths are possible, the highest scoring should be selected

Given a HMM path, what is the score of a sequence generated by taking this path?

Probability that path is taken and sequence generated:

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

or

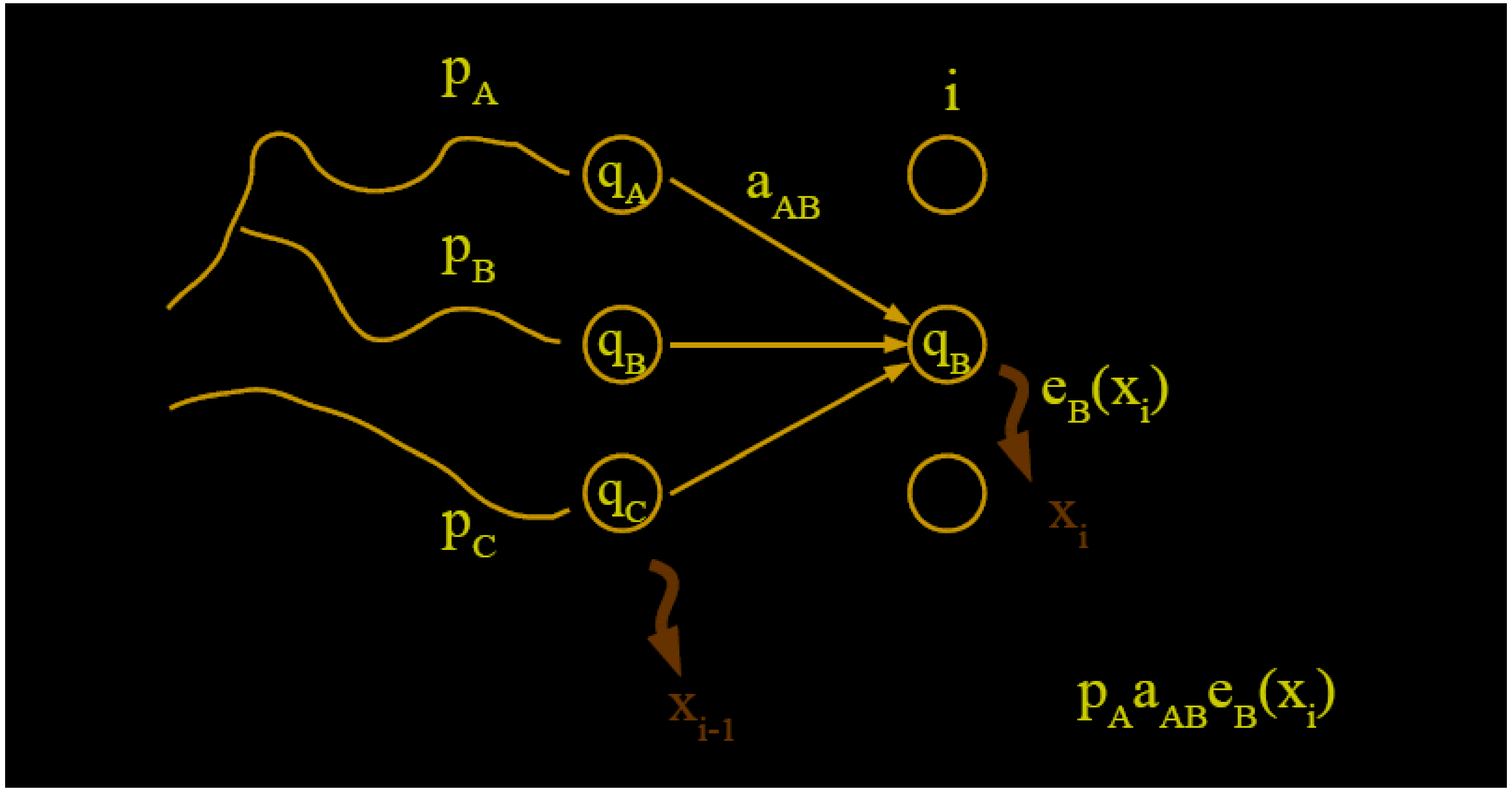
$$P(x,\pi) = a_{0\pi_1} \prod_{i=1}^{L} e_{\pi_i}(x_i) a_{\pi_i \pi_{i+1}}$$ (Durbin et al.)

(assuming begin (0) and end (π_{L+1}) states are the only silent states on path)

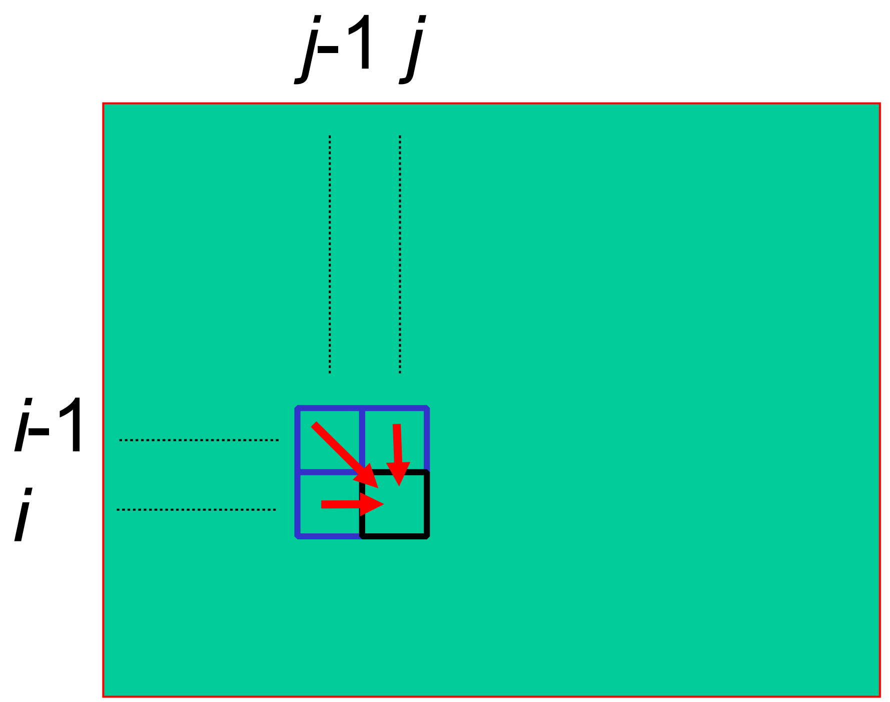
Finding the Most Probable Path: The Viterbi Algorithm

Viterbi – recursive step

What is the probability of the path which ends with q_A->q_B and emission E_B?



Recap: Dynamic Programming



For each cell you must check three cells that 'transition' into it.

With HMMs, it is possible that many more 'cells' (now called states) should be checked.

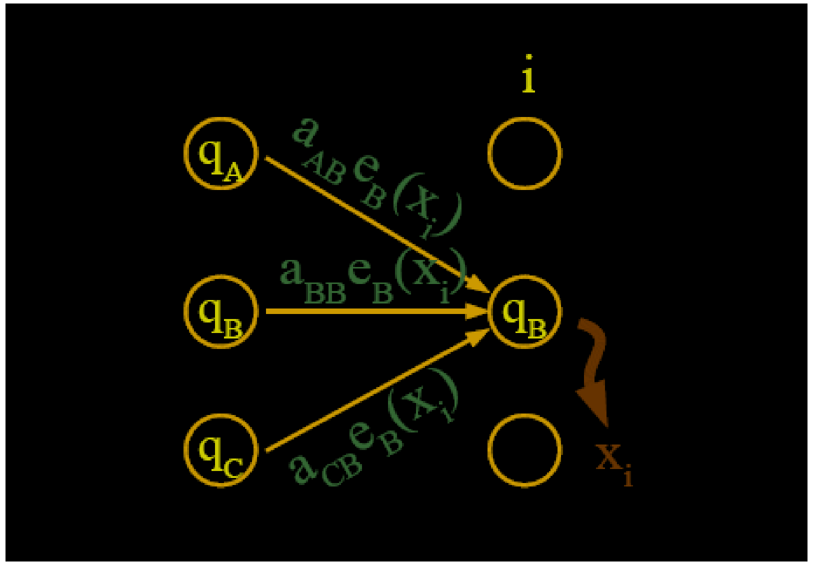
$$H(i,j) = Max \begin{cases} H(i-1,j-1) + s(i,j) & \text{diagonal } \searrow \\ H(i-1,j) - g & \text{vertical } \downarrow \\ H(i,j-1) - g & \text{horizontal } \rightarrow \end{cases}$$

This is a recursive formula

Finding the Most Probable Path: The Viterbi Algorithm

Viterbi – recursive step

What is the most probable path into state B in step i?



Many more than three paths can lead into state q_a…

Just as with DP, check all directions you can come from and select the one with the maximal score

If state q_B is indexed as *l*: $v_l(i) = e_l(x_i)\max_k[v_k(i-1)a_{kl}]$,

where transitions are checked between states all possible *k* and state *l* (the current state)

Finding the Most Probable Path: The Viterbi Algorithm

How to do the bookkeeping

•Andrew Viterbi used **Manhattan grid model** to solve the Decoding problem.

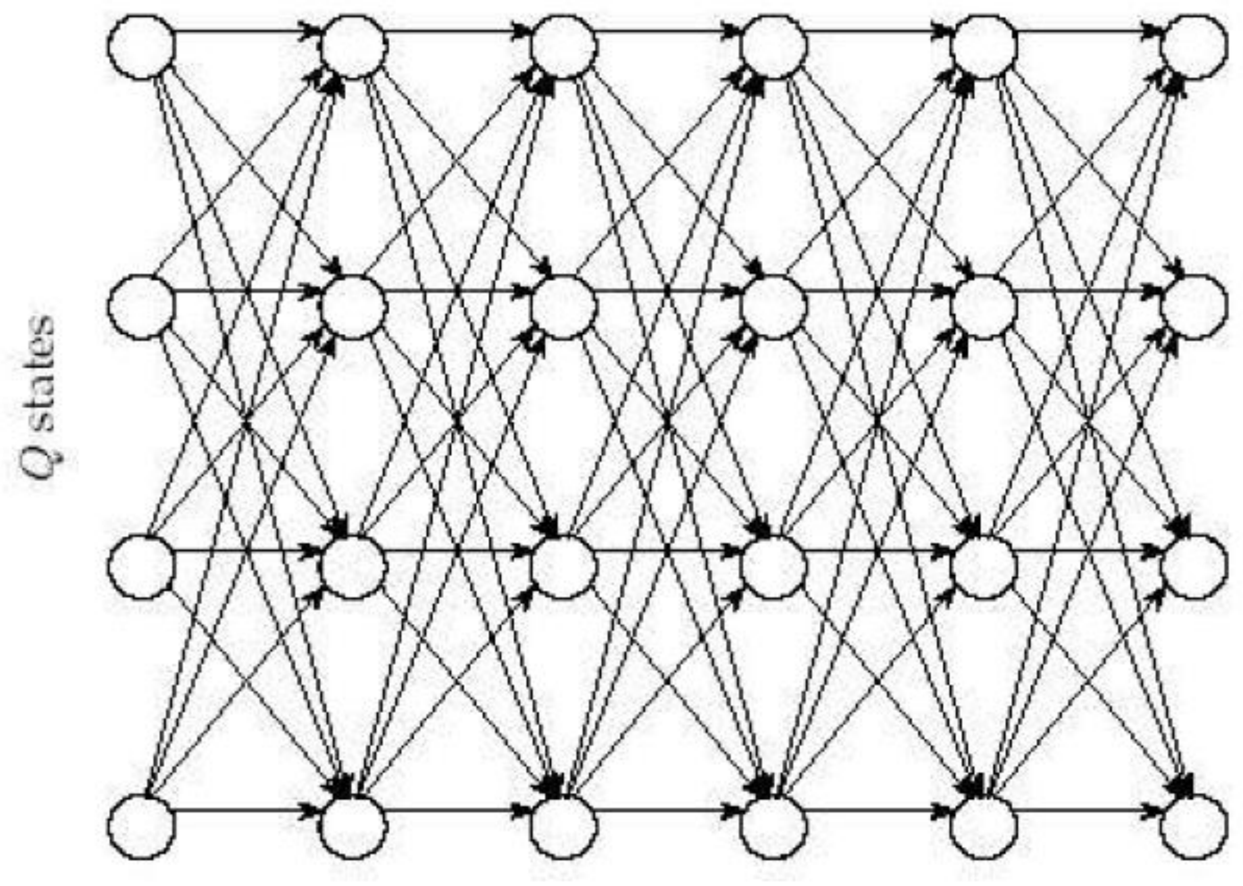
•Every choice of $\pi = \pi_1 \ldots \pi_n$ corresponds to a distinct path in the graph.

•Only valid direction in the graph is *eastward*.

•This graph has $N^2(n-1)$ edges, where N is number of states, and n the number of sequence symbols (L) plus the number of silent states

Finding the Most Probable Path: The Viterbi Algorithm

Manhattan grid model



Sequence $\longrightarrow$

This diagram is also called a **trellis**. Each column in the trellis shows all HMM states; it is executed from left to right (i.e. 'eastward') by going through the sequence of observations.

Execute the recursive formula:

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

Back to Markov Models - Biological Example: Methylation and CpG Islands

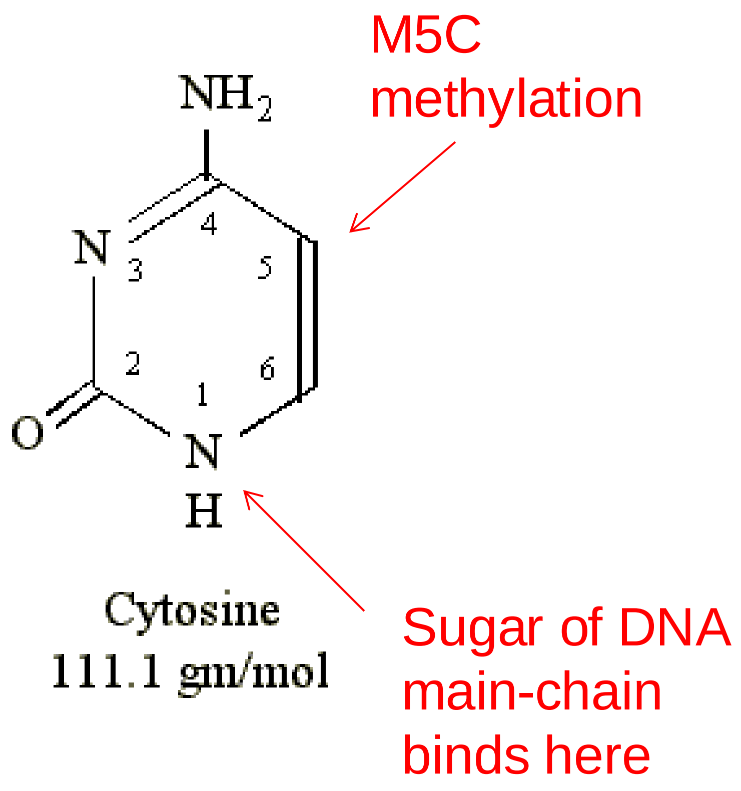
- CG dinucleotides (called CpG to avoid confusion with C-G base pairs) in nuclear DNA often undergo *methylation*, where a methyl (CH_3) “joins” the Cytosine (C).
- Methylated genes are usually *underexpressed* (‘silencing mark’).
  - CG dinucleotides are rarer in eukaryotic genomes than expected given the independent probabilities of C, G
  - In the human genome *CpG* (*CG*) is least frequent dinucleotide, because *C* in *CpG* is easily methylated and has the tendency to mutate into T afterwards
- However, methylation is suppressed around genes in a genome; unmethylated CGs are often found within *active gene regions* (this is condition and tissue dependent).
- As a result, ***CpG* appears more frequently within gene regions**: A **CpG island** is a short stretch of DNA in which the CpG frequency is higher than other regions; they are typically found with higher density around genes (particularly just upstream of Open Reading Frames (ORFs))

http://www.web-books.com/MoBio/Free/Ch7F2.htm

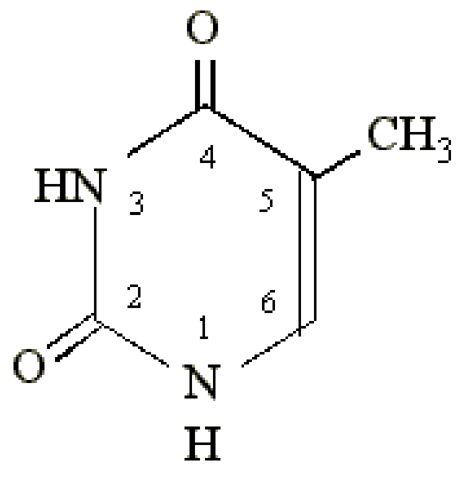
Biological Example: Methylation and CpG Islands

The enzyme methylating the C in CpG is M5C methyltransferase, also called C-5 cytosine-specific DNA methylase or C5 Mtase

- It specifically methylates the C-5 carbon of cytosines in DNA to produce C5-methylcytosine
- Some methylated cytosines become converted to thymine by deamination (at C-4)
  - Over evolutionary time scales, the methylated CG sequence will often be converted to the TG sequence (compromising the base pair the C can engage in).



Cytosine
111.1 gm/mol



Thymine
126.1 gm/mol

Example Application: gene prediction using CpG Islands

- **CpG islands**
  - Regions upstream of genes are richer in CG dinucleotides than elsewhere – *CpG islands*
  - Identifying the *CpG* islands in a genome is important as it represents 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

State: CpG island or not CpG island

Our example (see Durbin et al. textbook) includes using Maximum likelihood and Bayes' statistics, and feeding an input sequence to a Markov model (and also an HMM)

See Durbin et al. (course book) Chapter 3: Pp. 46 - 53

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:

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

Log likelihood of sequence given CpG state and sequence given null state.
Since it is the same sequence, the marginal P(sequence) cancels out

See Durbin et al. (course book) Chapter 3: Pp. 46 - 53

Markov Chains for Discrimination

- why use

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

- Bayes' rule tells us

$$\underset{\text{posterior}}{\Pr(CpG \mid x)} = \frac{\Pr(x \mid CpG)\,\overset{\text{prior}}{\Pr(CpG)}}{\Pr(x)}$$

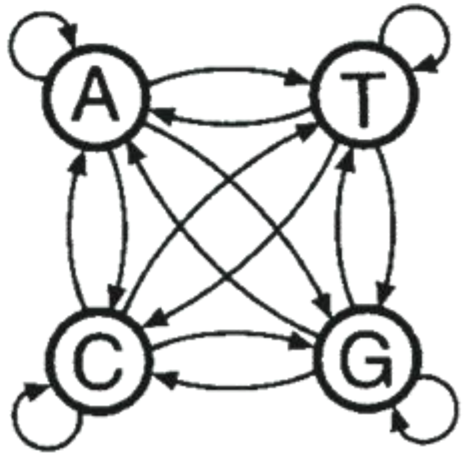
$$\Pr(null \mid x) = \frac{\Pr(x \mid null)\Pr(null)}{\Pr(x)}$$

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

CpG Islands

- We construct a Markov chain for CpG rich (+) and another for CpG poor (-) regions
- Using maximum likelihood estimates from 60K nucleotide training sets, we get two models

Max. likelihood: maximizes prob of data given parameters



Markov chain model, only transition probs

+	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

−	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

These dinucleotide frequency data are derived from genome sequences

Example taken from Chapter 3 in Durbin et al.'s course book

Ratio Test for CpC islands

- Given a sequence $X_1,\ldots,X_n$ we compute the likelihood ratio

In log domain multiplication becomes summation

$$S(x) = \log_2 \frac{P(x|\text{model}+)}{P(x|\text{model}-)} = \sum_{i=1}^{L} \log_2 \frac{a^{+}_{x_{i-1}x_i}}{a^{-}_{x_{i-1}x_i}} =$$

$$= \sum_{i=1}^{L} \beta_{x_{i-1}x_i}$$

Then we know how much is this sequence more likely to be a CpG island

β	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

Transition probabilities based on $\log_2$ values

Finding CpG islands: sequence fragment size

Simple Minded approach:

- Pick a window of size *N* (*N* = *100*, for example)
- Compute log-ratio for the sequence in the window, and classify based on that

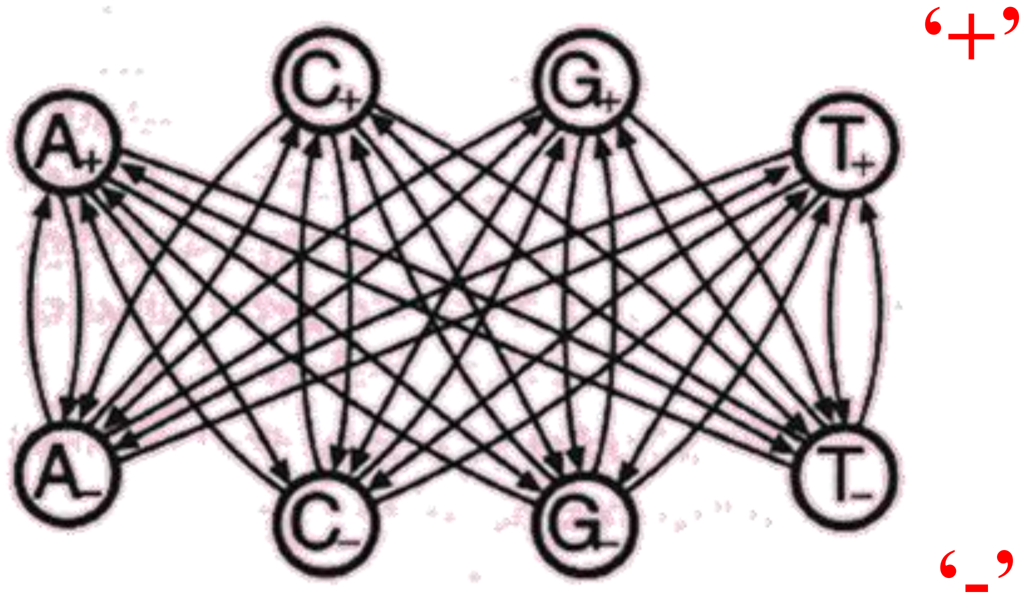
Problems:

- How do we select *N*? CpG islands have varying lengths…
- What do we do when the window intersects the boundary of a CpG island? We want to be able to jump from ‘+’ to ‘-’..

Alternative Approach: HMM model

- Build a single model that include "+" states and "-" states, so model can switch between CpG (+) and 'normal' (-)

In this HMM model you are not sure anymore in which state the sequence is when you observe an 'A'… (can be in '+' or '-').
Note that the transitions among '+' and among '-' states are left out for clarity in the figure.



- A state 'remembers' last nucleotide and the type of region
- A transition from a **"-"** state to a "**+**" corresponds to the start of a CpG island

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

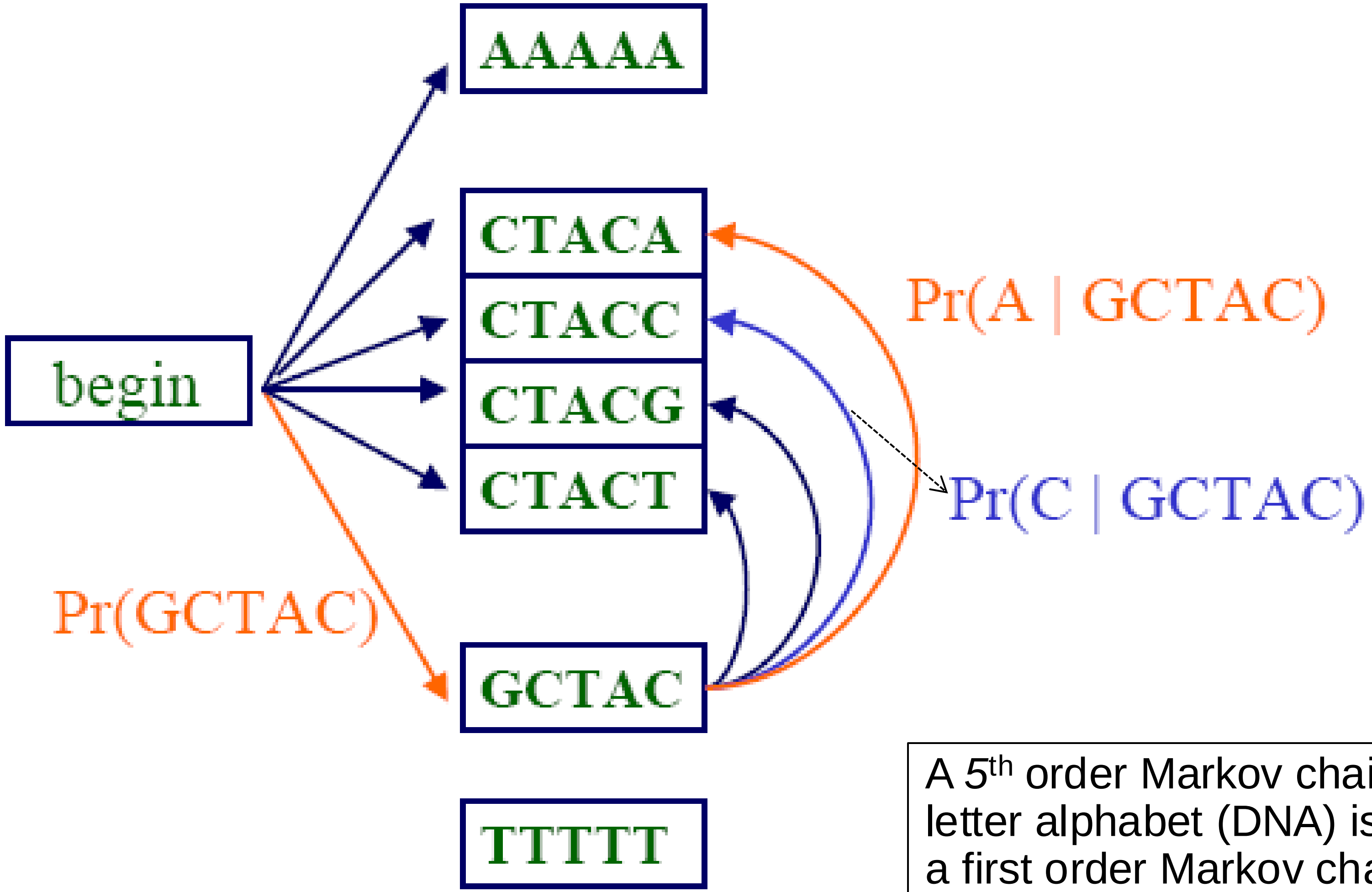
$$\Pr(x_i \mid x_{i-1}, x_{i-2}, \ldots, x_1) = \Pr(x_i \mid x_{i-1}, \ldots, x_{i-n})$$

Higher Order Markov Chains

- An *n*th order Markov chain over some alphabet is equivalent to a first order Markov chain over the alphabet of *n*-tuples
- Example: a 2nd order Markov model for DNA can be treated as a 1st order Markov model over alphabet:

  AA, AC, AG, AT, CA, CC, CG, CT, GA, GC, GG, GT, TA, TC, TG, and TT (i.e. all possible dinucleotides)

A Fifth Order Markov Chain



A 5th order Markov chain over a 4-letter alphabet (DNA) is equivalent to a first order Markov chain over the alphabet of *5*-tuples

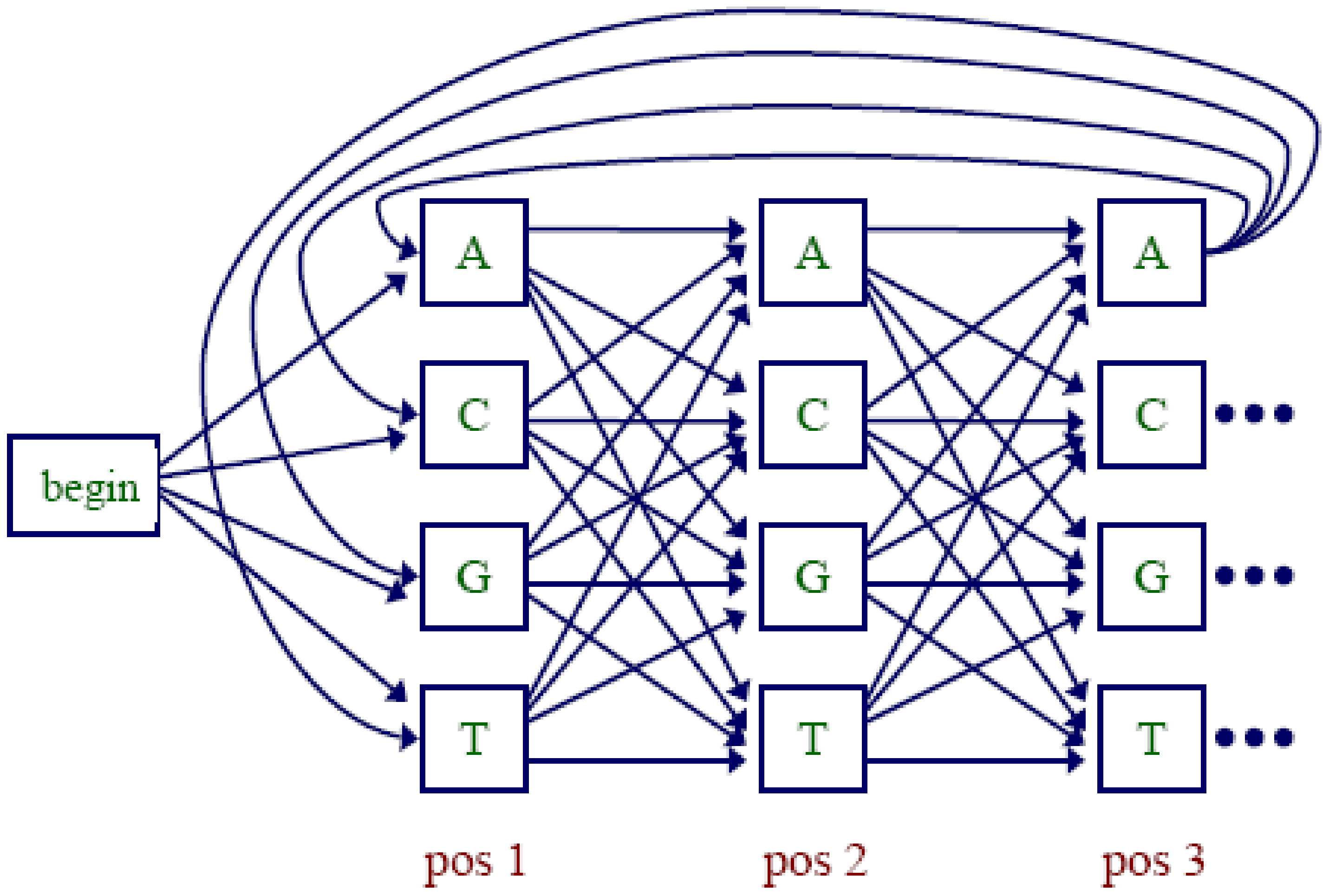
Inhomogeneous Markov Chains

- 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

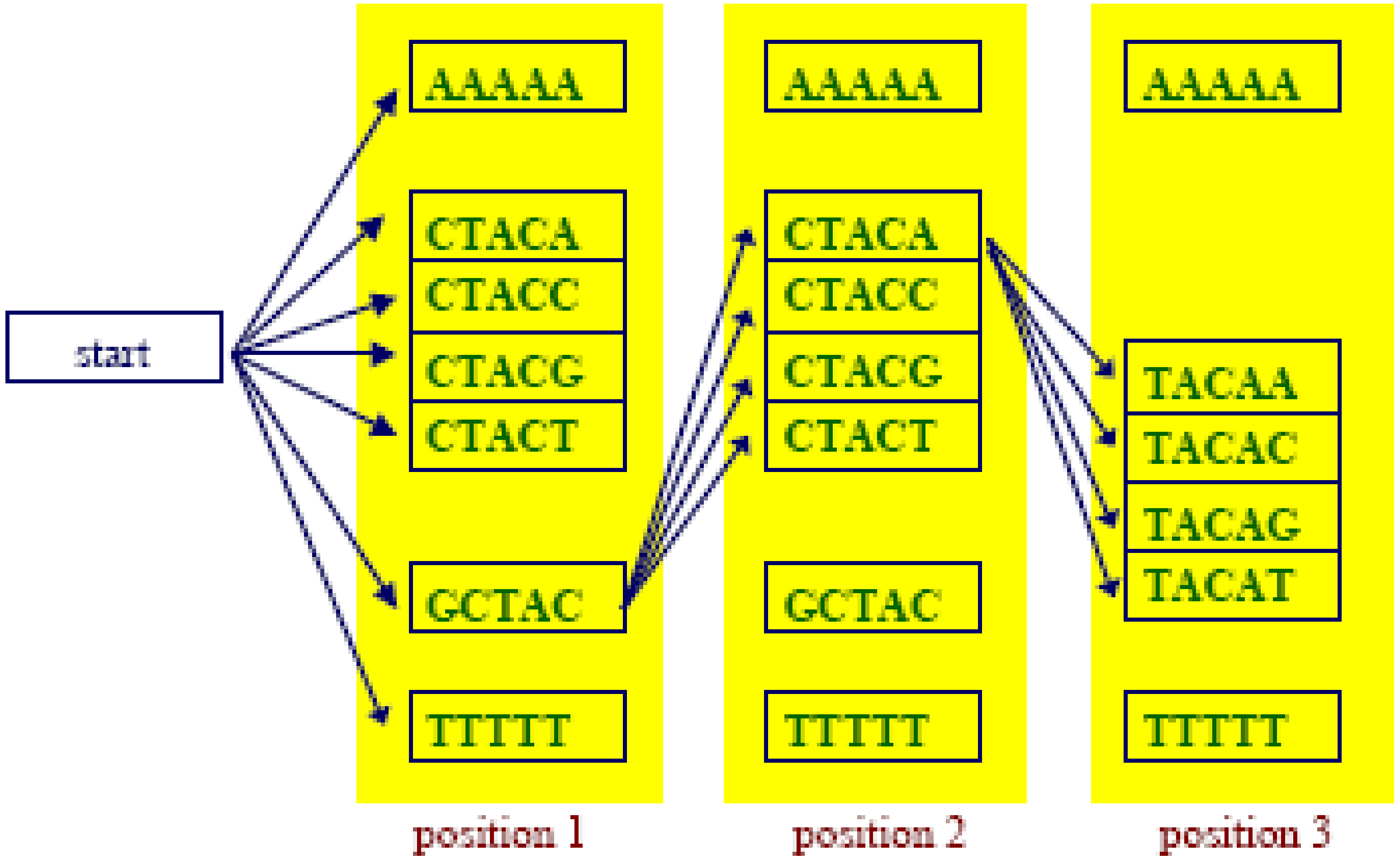
- Consider modeling codons in protein coding regions

Inhomogeneous Markov Chains



The probabilty of reaching the 'A' state may differ at position 1, 2 or 3, etc.

A Fifth Order Inhomogeneous Markov Chain



Selecting the Order of a Markov Chain Model

- Higher order models remember more "history"
- Additional history can have predictive value
- Example:
  - – predict the next word in this sentence fragment "…finish __" (up, it, first, last, …?)
  - – now predict it given more history
- "Fast guys finish __"

Selecting the Order of a Markov Chain Model (cnt.)

- However, the number of parameters we need to estimate grows exponentially with the order, so we need a large data set
  - for modeling DNA we need parameters for an nth order model, with $n \geq 5$ normally

- The higher the order, the less reliable we can expect our parameter estimates to be
  - estimating the parameters of a 2nd order homogenous Markov chain from the complete genome of E. Coli, we would see each word > 72,000 times on average
  - estimating the parameters of an 8th order chain, we would see each word ~ 5 times on average

- This means there is a trade-off between the advantage of more memory with higher orders and the disadvantage of less reliable parameter estimation

- Trade-off can be addressed using an **Interpolated Markov Model** (IMM)

Interpolated Markov Models (IMM)

- The IMM idea: manage this trade-off by interpolating among models of various orders
- *Simple* linear interpolation (mixing orders):

$$\begin{aligned}\Pr_{\mathrm{IMM}}(x_i \mid x_{i-1},\ldots,x_{i-n}) = {} & \lambda_0 \Pr(x_i) \\ & + \lambda_1 \Pr(x_i \mid x_{i-1}) \\ & \ldots \\ & + \lambda_n \Pr(x_i \mid x_{i-1},\ldots,x_{i-n})\end{aligned}$$

- where $\sum_i \lambda_i = 1$

Interpolated Markov Models (IMM)

- We can make the weights depend on the history
  – for a given order, we may have significantly more data to estimate some words as compared to others
- $$\begin{aligned}\Pr_{\text{IMM}}(x_i \mid x_{i-1},\ldots,x_{i-n}) &= \lambda_0 \Pr(x_i)\\ &+ \lambda_1(x_{i-1})\Pr(x_i \mid x_{i-1})\\ &\ldots\\ &+ \lambda_n(x_{i-1},\ldots,x_{i-n})\Pr(x_i \mid x_{i-1},\ldots,x_{i-n})\end{aligned}$$

Example: Gene Finding (again)

Search by Content

- Encoding a protein 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

- The latter is termed *codon preference*

- Codon preferences vary by species

Codon Preference in E. Coli

AA	codon	/100
Gly	GGG	1.89
Gly	GGA	0.44
Gly	GGU	52.99
Gly	GGC	34.55
Glu	GAG	15.68
Glu	GAA	57.20
Asp	GAU	21.63
Asp	GAC	43.26

Codon biases not only exist between various synonymous codons (coding for the same amino acid), but can also vary dramatically between organisms.

Some bacteria even have a different codon table, and so some of their codons may encode other amino acids than in higher organisms

Search by gene content

- 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 (= prokaryotic) genomes
  - uses 5th order inhomogenous Markov chain models
  - Will be used here as a base model to compare performance of the GLIMMER method that will be explained in some detail (later slides)

Prokaryote gene prediction

Gene Prediction using Markov Models

- Exploit the fact that oligonucleotide distributions in coding regions are different from those in noncoding regions.

- Because a protein-encoding gene is composed of nucleotides in triplets as codons, more effective Markov models are built in sets of three nucleotides, describing nonrandom distributions of trimers or hexamers, and so on.

- The parameters of a Markov model have to be trained using a set of sequences with known gene locations (i.e. coding or non-coding).
  - Once the parameters of the model are established, it can be used to compute the nonrandom distributions of trimers or hexamers in a new sequence to find regions that are compatible with the statistical profiles in the learning set.

Prokaryote gene prediction

Gene Prediction using Markov Models

- Statistical analyses have shown that pairs of codons (or *dipeptides* at the protein level) tend to correlate. The frequency of six unique nucleotides appearing together in a coding region *i* is much higher than by random chance.

Fifth-order: memory of previous 5 nucleotides, so it calculates probability of hexamers

- Therefore, a fifth-order Markov model, which calculates the probability of hexamer bases, can detect nucleotide correlations found in coding regions more accurately and is in fact most often used.

- A potential problem of using a fifth-order (or even higher order) Markov chain is that if there are not enough hexamers, which happens in short gene sequences, the method's efficacy may be limited.

Prokaryote gene prediction by GLIMMER

Gene Prediction using Markov Models

- To cope with this limitation, a variable-length ***interpolated Markov model (IMM)***, has been developed, called **Glimmer**.

interpolation

- The IMM method samples the number of sequence patterns for the various orders of the Markov models, with *k* ranging from 1 to 8 (dimers to ninemers)
    - It uses a weighting scheme which places less weight on rare k-mers and more weight on more frequent k-mers (for a given order)

- The probability of the final model is the sum of probabilities of all weighted k-mers.
    - In other words, this method has more flexibility in using Markov models depending on the amount of data available. Higher-order models are used when there is a sufficient amount of data and lower-order models are used when the amount of data is smaller.

Prokaryote gene prediction by GLIMMER

- **GLIMMER** (Gene Locator and Interpolated Markov ModelER, www.tigr.org/softlab/glimmer/glimmer.html)

- Computation consists of two steps: model building and gene prediction.
  - Model building involves training by the input sequence, which optimizes the parameters of the model.

- Glimmer also has a variant, GlimmerM, for eukaryotic gene prediction (more difficult to predict than bacterial genes).

The GLIMMER System

- Salzberg et al., 1998
- System for identifying genes in bacterial genomes
- Uses 8th order, inhomogeneous, interpolated Markov chain models
- The model starts at 8th order (i.e. nonamers) and for calculating the weight l_n , it checks whether for a given order n there are sufficient observations to train the model, otherwise it goes to order n-1

IMMs in GLIMMER

- How does GLIMMER determine the values?
- First, let us express the IMM probability calculation recursively:

$$\Pr_{\text{IMM,n}}(x_i \mid x_{i-1},\ldots,x_{i-n}) =$$
$$\lambda_n(x_{i-1},\ldots,x_{i-n})\Pr(x_i \mid x_{i-1},\ldots,x_{i-n}) +$$
$$[1-\lambda_n(x_{i-1},\ldots,x_{i-n})]\Pr_{\text{IMM,n-1}}(x_i \mid x_{i-1},\ldots,x_{i-n+1})$$

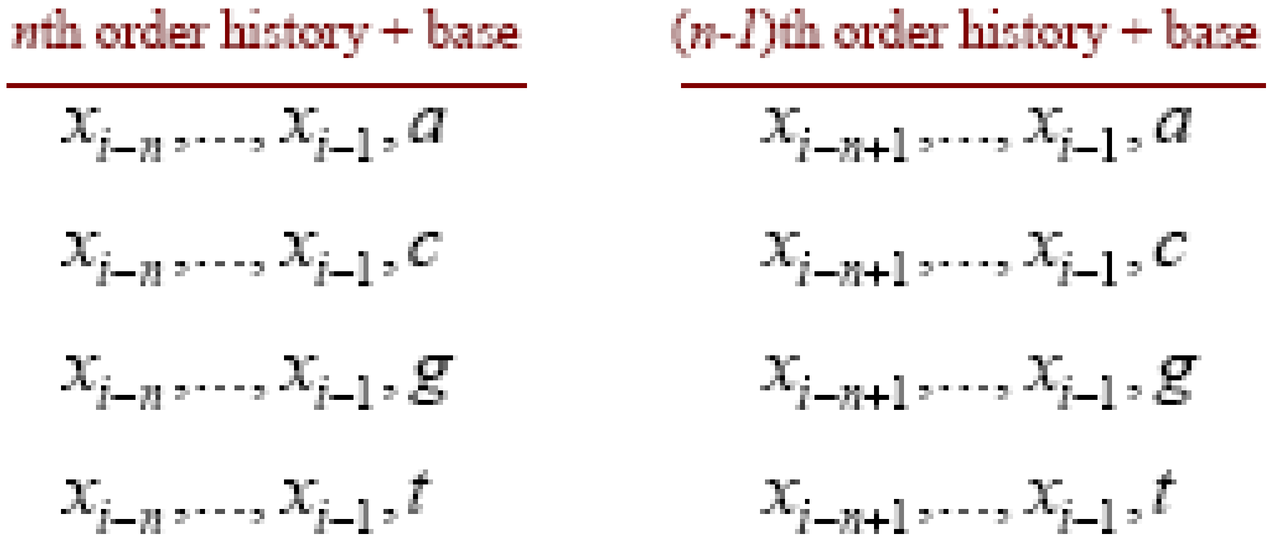
- let $c(x_{i-1},\ldots,x_{i-n})$ be the number of times we see the history $x_{i-1},\ldots,x_{i-n}$ in our training set

$$\lambda_n(x_{i-1},\ldots,x_{i-n}) = 1 \quad \text{if} \quad c(x_{i-1},\ldots,x_{i-n}) > 400$$

IMMs in GLIMMER

- If we see $x_{i-1} \ldots x_{i-n}$ more than 400 times, we use it to the full
- If we haven't seen $x_{i-1} \ldots x_{i-n}$ more than 400 times, then compare nth order and (n-1)th order history::

Less weight to rare sequences



nth order history + base	(n-1)th order history + base
$x_{i-n}, \ldots, x_{i-1}, a$	$x_{i-n+1}, \ldots, x_{i-1}, a$
$x_{i-n}, \ldots, x_{i-1}, c$	$x_{i-n+1}, \ldots, x_{i-1}, c$
$x_{i-n}, \ldots, x_{i-1}, g$	$x_{i-n+1}, \ldots, x_{i-1}, g$
$x_{i-n}, \ldots, x_{i-1}, t$	$x_{i-n+1}, \ldots, x_{i-1}, t$

..sequence goes from left to right here…

- Use a statistical test (χ^2) to get a value d ($0 < d < 1$) indicating confidence that the distributions represented by the two sets of counts (nth vs. (n-1)th order) are different

[$\chi 2 = ((O - E)^2/E)$]. A value of $d \geq 0.5$ is taken to indicate that the distributions are sufficiently different (so nth order is used).

IMMs in GLIMMER

- putting it all together

use only nth order

$$\lambda_n(x_{i-1},\ldots,x_{i-n}) = \begin{cases} 1 & \text{if } c(x_{i-1},\ldots,x_{i-n}) > 400 \\ d \times \dfrac{c(x_{i-1},\ldots,x_{i-n})}{400} & \text{else if } d \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

where $d \in (0,1)$

weigh in n-order model based on χ^2 score when comparing nth-order with n-1th-order Markov model (preceding slide)

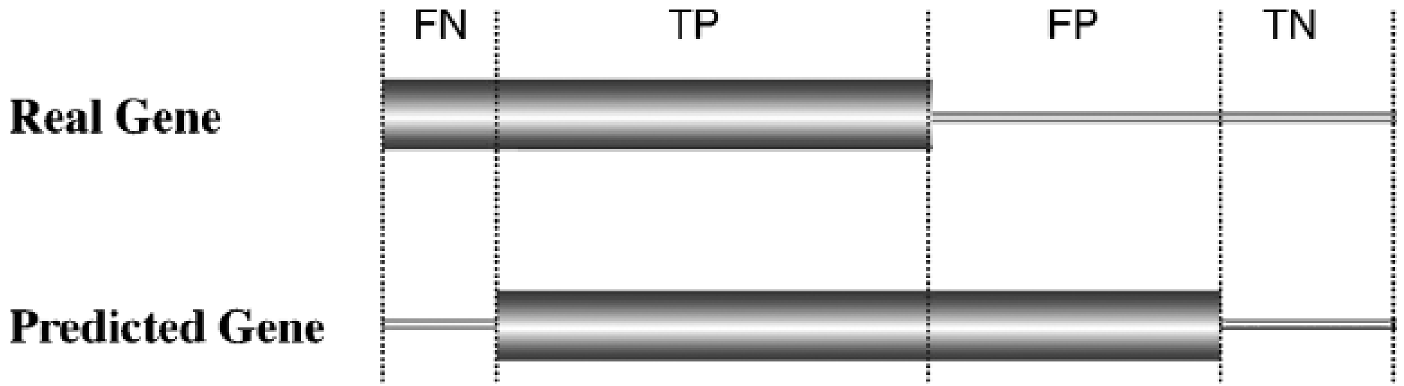
.. if n-order MCM is similar to n-1-order, then forget about n-order model and use the smaller (n-1) order to capture more observations.

Evaluation:

The GLIMMER method versus 5th order Markov model (GeneMark) as reference model

- 8th order IMM compared against 5th order Markov model to show improvement
- Trained on 1168 genes (ORFs really)
- Tested on 1717 annotated (more or less known) genes

Prokaryote gene prediction - prediction success

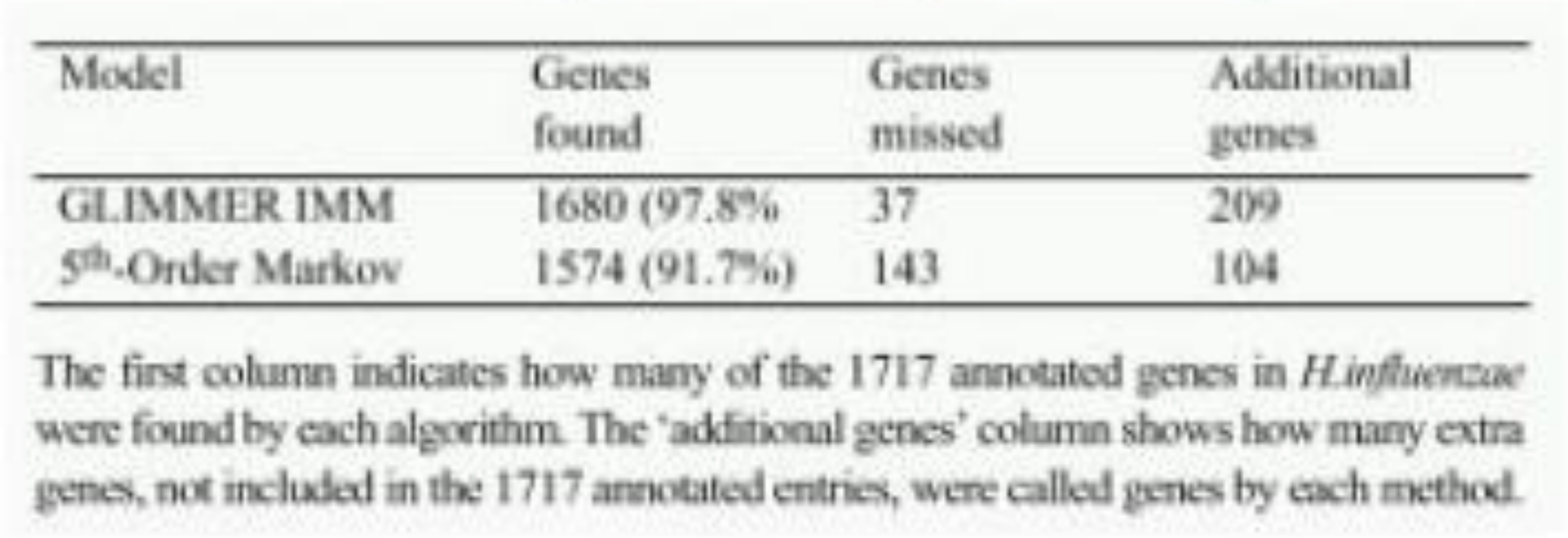


FN – false negative, TP – true positive, FP – false positive, TN – true negative

Sensitivity: *Sn = TP/(TP + FN)*
Specificity *Sp = TN/(TN + FP)*
PPV: *PPV = TP/(TP + FP)* -- *Positive Predictive Value*

Matthews correlation: $MCC = \dfrac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

GLIMMER results



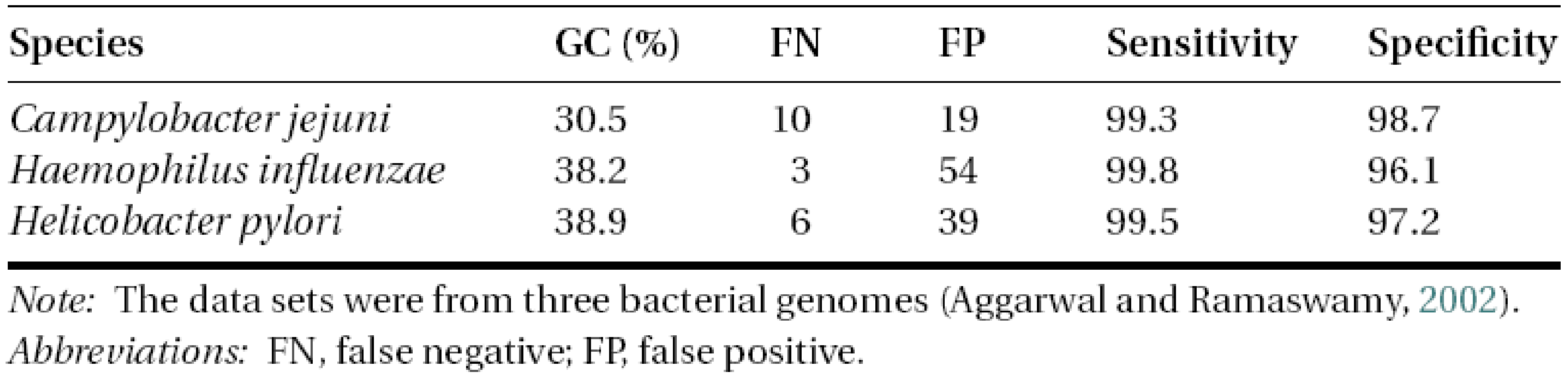
	TP	FN	FP
Model	Genes found	Genes missed	Additional genes
GLIMMER IMM	1680 (97.8%	37	209
5th-Order Markov	1574 (91.7%)	143	104

The first column indicates how many of the 1717 annotated genes in *H.influenzae* were found by each algorithm. The 'additional genes' column shows how many extra genes, not included in the 1717 annotated entries, were called genes by each method.

GLIMMER $\text{sensitivity} = \frac{1680}{1680+37} = 0.978$ PPV $= \frac{1680}{1680+209} = 0.889$

5th Order $\text{sensitivity} = \frac{1574}{1574+143} = 0.917$ PPV $= \frac{1574}{1574+104} = 0.938$

GLIMMER results



Species	GC (%)	FN	FP	Sensitivity	Specificity
Campylobacter jejuni	30.5	10	19	99.3	98.7
Haemophilus influenzae	38.2	3	54	99.8	96.1
Helicobacter pylori	38.9	6	39	99.5	97.2

Note: The data sets were from three bacterial genomes (Aggarwal and Ramaswamy, 2002).
Abbreviations: FN, false negative; FP, false positive.

Background text

Further refinement of bacterial gene prediction methods

- The methods shown on preceding slides have been shown to be reasonably successful in finding genes in a genome. The common problem is imprecise prediction of **translation initiation sites** because of inefficient identification of ribosomal binding sites. This problem can be remedied by identifying the ribosomal binding site associated with a start codon. A number of algorithms have been developed solely for this purpose.

- **RBSfinder** is one such algorithm. RBSfinder (ftp://ftp.tigr.org/pub/software/RBSfinder/) is a UNIX program that uses the prediction output from Glimmer and searches for the Shine–Delgarno sequences in the vicinity of predicted start sites. If a high-scoring site is found by the intrinsic probabilistic model, a start codon is confirmed; otherwise the program moves to other putative translation start sites and repeats the process.

Further refinement of bacterial gene prediction methods?

Background text

- GLIMMER: 8th order IMM

- RBSfinder = GLIMMER + finding Shine–Delgarno sequence (RBS)

- *Question:* How about combining this with CpG island prediction?

- Could all this be combined in a single Markov model?

Wrapping up

- Conditional probabilities and Bayesian statistics primer
- Markov chain models
  - Transition probabilities
- Preamble: Hidden Markov models (HMMs)
  - Transition probabilities and emission probabilities
- HMMs: Finding the most probable (optimal) path
  - The Viterbi algorithm
- Markov Model example: CpG prediction
  - Maximum likelihood with pseudo-counting, Markov approach and HMM approach
- Higher Order, Inhomogeneous and Interpolated Markov Models
  - Example of 8th-order interpolated Markov Model (IMM) for bacterial gene prediction: Glimmer

References

- Seminal review by Lawrence R. Rabiner (1989)
- bioalgorithms.info
- Wikipedia
- Course book *Biological Sequence Analysis*
- Book *Understanding Bioinformatics*

Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids by Richard Durbin, Sean R. Eddy, Anders Krogh, Graeme Mitchison, Cambridge University Press, ISBN 0521 62971 3 Pbk

APPENDIX

A next slides comprise some biological background information in prokaryotic genetics, including how this type of information can be used to predict coding genes

Prokaryote gene

Background text

- Prokaryotes, which include bacteria and Archaea, have relatively small genomes with sizes ranging from 0.5 to 10Mbp (1Mbp=10^6 bp).

- The gene density in prokaryotic genomes is high, with more than 90% of a genome sequence containing coding sequence.

- There are very few repetitive sequences. Each prokaryotic gene is composed of a single contiguous stretch of ORF coding for a single protein or RNA with no interruptions within a gene (no splicing).

Prokaryote gene

Background text

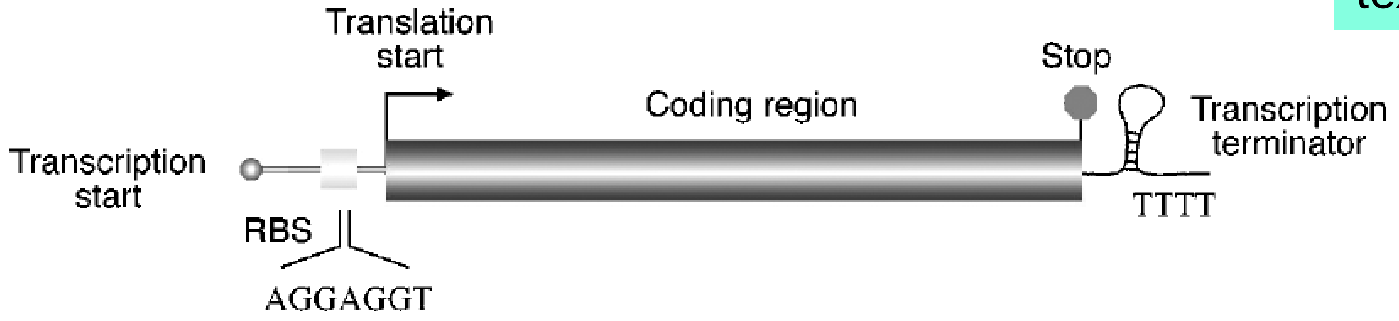


Figure 8.1: Structure of a typical prokaryotic gene structure. *Abbreviation:* RBS, ribosome binding site.

- The majority of genes have a start codon ATG (or AUG in mRNA) coding for methionine. Occasionally, GTG and TTG are used as alternative start codons, both also coding for methionine.

- Because there may be multiple ATG, GTG, or TTG codons in a frame, the presence of these codons at the beginning of the frame does not necessarily give a clear indication of the translation initiation site.

- To help gene identification, other features associated with translation are used: One such feature is the **ribosomal binding site (RBS)** , also called the ***Shine-Delgarno sequence***, a stretch of purine-rich sequence complementary to 16S rRNA in the ribosome. It is located immediately downstream of the transcription initiation site and slightly upstream of the translation start codon. In many bacteria, it has a consensus motif of AGGAGGT. Identification of the ribosome binding site can help locate the start codon.

Prokaryote gene

Background text

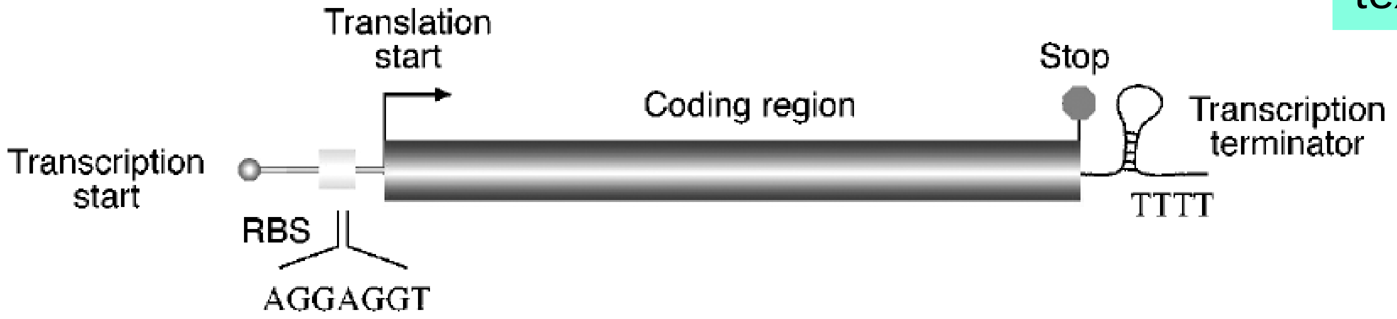


Figure 8.1: Structure of a typical prokaryotic gene structure. *Abbreviation:* RBS, ribosome binding site.

- At the end of the protein coding region is a stop codon that causes translation to stop. There are three possible stop codons, identification of which is straightforward.

- Many prokaryotic genes are transcribed together as one operon. The end of the operon is characterized by a transcription termination signal called *rho-independent terminator.* The terminator sequence has a distinct stem-loop secondary structure followed by a string of Ts. Identification of the terminator site, in conjunction with promoter site identification, can sometimes help in gene prediction.

Prokaryote gene prediction

how to predict an ORF by hand

Background text

- Perform conceptual translation in all six possible frames (three frames forward and three frames reverse). Because a stop codon occurs in about every twenty codons by chance in a noncoding region, a frame longer than 30 codons without interruption by stop codons is suggestive of a gene coding region (threshold is normally set even higher at 50 or 60 codons).

- The putative frame is further manually confirmed by the presence of other signals such as a start codon and Shine–Delgarno sequence.

- After translating the putative ORF into the most likely protein sequence, it may then be used to search against a protein database. Detection of homologs from this search is probably the strongest indicator of a protein-coding frame.

Prokaryote gene prediction

how to predict an ORF computationally

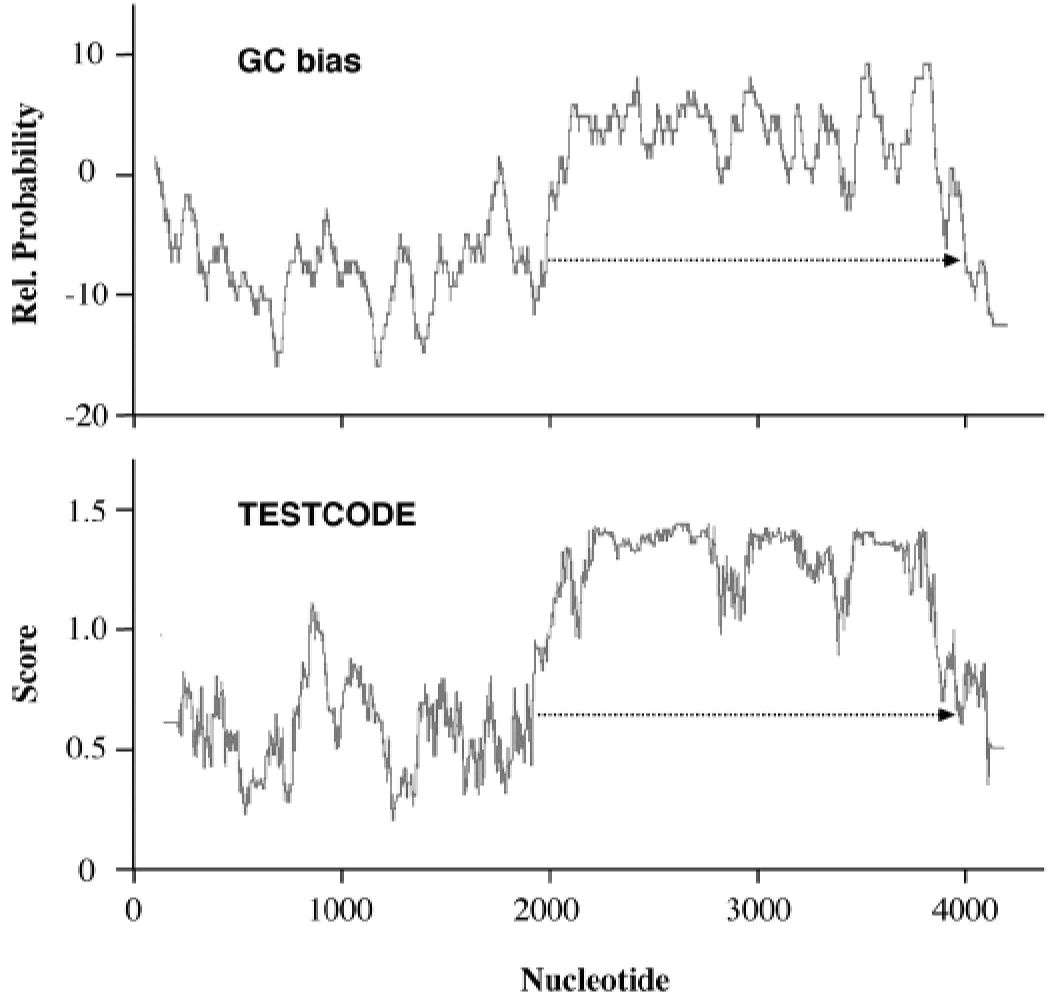
Background text

Examine nonrandomness of nucleotide distribution

- **GC bias**: the third position of a codon has a preference to use G or C over A or T in a coding sequence. By plotting the GC composition at this position, regions with values significantly above the random level can be identified, which are indicative of the presence of ORFs. In practice, the statistical patterns are computed for all six possible frames.
- Relating to GC bias, the method TESTCODE (implemented in the commercial GCG package) exploits the fact that the third codon nucleotides in a coding region tend to repeat themselves. By plotting the repeating patterns at this position, coding and noncoding regions can be differentiated. The results of the methods GC Bias and TESTCODE are often consistent.
- These two early methods are often used in conjunction to confirm the results of each other (next slide).

Prokaryote gene prediction

Background text



Coding frame detection of a bacterial gene using either the GC bias or the TESTCODE method. Both result in similar identification of a reading frame (*dashed arrows).*