6.047/6.878/HST.507
Computational Biology: Genomes, Networks, Evolution

Lecture 05

Hidden Markov Models
Part II

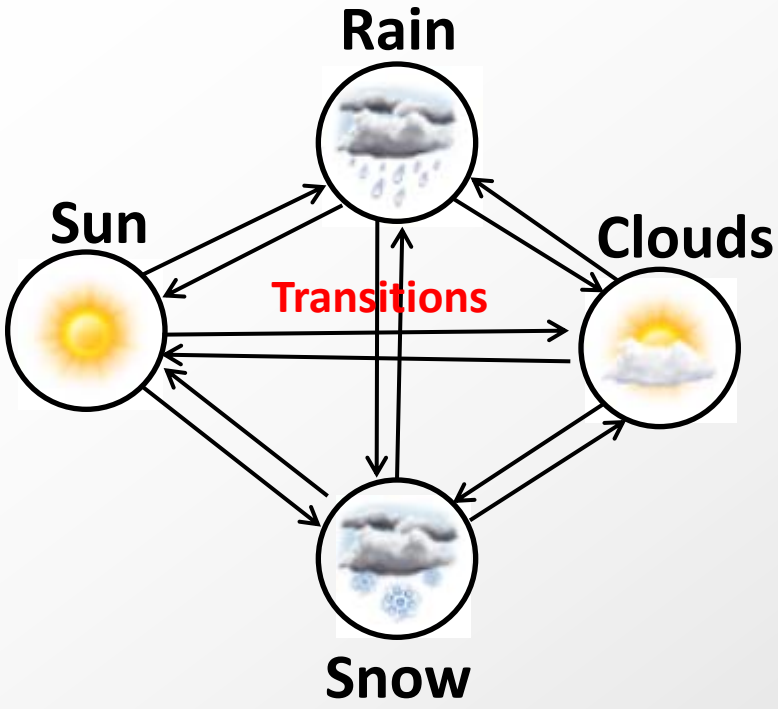
Module 1: Aligning and modeling genomes

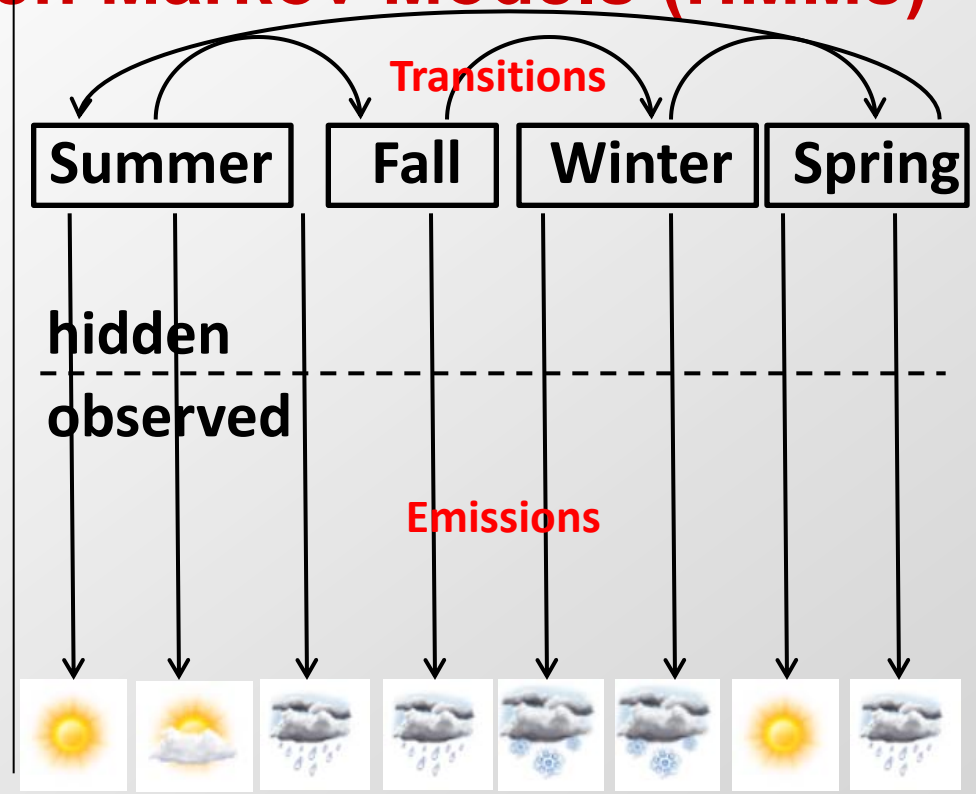
- Module 1: Computational foundations
  - Dynamic programming: exploring exponential spaces in poly-time
  - Introduce Hidden Markov Models (HMMs): Central tool in CS
  - HMM algorithms: Decoding, evaluation, parsing, likelihood, scoring
- This week: Sequence alignment / comparative genomics
  - Local/global alignment: infer nucleotide-level evolutionary events
  - Database search: scan for regions that may have common ancestry
- Next week: Modeling genomes / exon / CpG island finding
  - Modeling class of elements, recognizing members of a class
  - Application to gene finding, conservation islands, CpG islands

Goals for today: HMMs, part II

1. Review: Basics and three algorithms from last time
   - Markov Chains and Hidden Markov Models
   - Calculating likelihoods $P(x,\pi)$ (algorithm 1)
   - Viterbi algorithm: Find $\pi^* = \text{argmax}_\pi P(x,\pi)$ (alg 3)
   - Forward algorithm: Find $P(x)$, over all paths (alg 2)
2. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures (GENSCAN), chromatin (ChromHMM)
3. Posterior decoding: Another way of 'parsing'
   - Find most likely state π_i, sum over all possible paths
4. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params

Markov chains and Hidden Markov Models (HMMs)



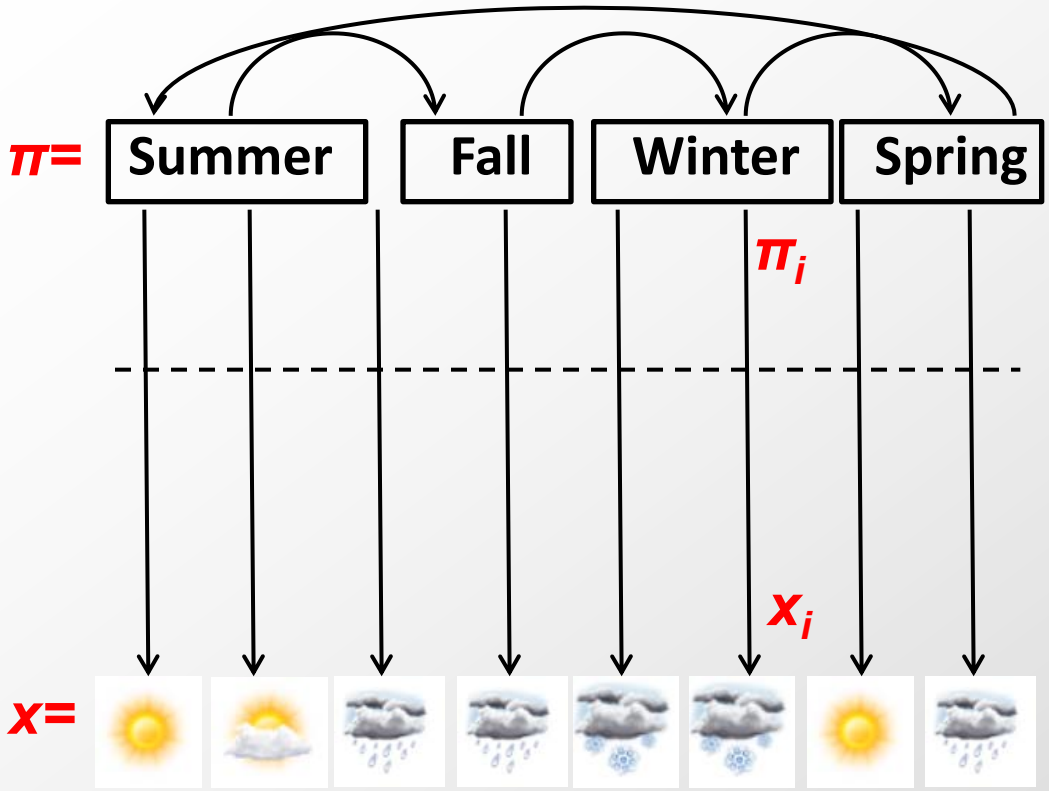


All observed

- Markov Chain
  - Q: states
  - p: initial state probabilities
  - A: transition probabilities
- What you see is what you get: next state only depends on current state (no memory)

- HMM
  - Q: states, p: initial, A: transitions
  - V: observations
  - E: emission probabilities
- Hidden state of the world determines emission probabilities
- State transitions are a Markov chain

HMM nomenclature for this course



Transitions: $a_{kl}=P(\pi_i=l|\pi_{i-1}=k)$

Transition probability from state *k* to state *l*

Emissions: $e_k(x_i)=P(x_i|\pi_i=k)$

Emission probability of symbol x_i from state *k*

- Vector x = Sequence of observations
- Vector π = Hidden path (sequence of hidden states)
- Transition matrix A=a_{kl}=probability of $k \rightarrow l$ state transition
- Emission vector $E=e_k(x_i)$ = prob. of observing x_i from state k
- Bayes's rule: Use $P(x_i|\pi_i=k)$ to estimate $P(\pi_i=k|x_i)$

Example: The Dishonest Casino

A casino has two dice:

- Fair die

  P(1) = P(2) = P(3) = P(5) = P(6) = 1/6

- Loaded die

  P(1) = P(2) = P(3) = P(4) = P(5) = 1/10

  P(6) = 1/2

Casino player switches between fair and loaded die on average once every 20 turns

Game:

1. You bet $1
2. You roll (always with a fair die)
3. Casino player rolls (maybe with fair die, maybe with loaded die)
4. Highest number wins $2

Slide credit: Serafim Batzoglou

Examples of HMMs for genome annotation

Application	Detection of GC-rich regions	Detection of conserved regions	Detection of protein-coding exons	Detection of protein-coding conservation	Detection of protein-coding gene structures	Detection of chromatin states
Topology / Transitions	2 states, different nucleotide composition	2 states, different conservation levels	2 states, different tri-nucleotide composition	2 states, different evolutionary signatures	~20 states, different composition/ conservation, specific structure	40 states, different chromatin mark combinations
Hidden States / Annotation	GC-rich / AT-rich	Conserved / non-conserved	Coding exon / non-coding (intron or intergenic)	Coding exon / non-coding (intron or intergenic)	First/last/middle coding exon,UTRs, intron1/2/3, intergenic, *(+/- strand)	Enhancer / promoter / transcribed / repressed / repetitive
Emissions / Observations	Nucleotides	Level of conservation	Triplets of nucleotides	Nucleotide triplets, conservation levels	Codons, nucleotides, splice sites, start/stop codons	Vector of chromatin mark frequencies

The main questions on HMMs

1. Scoring x, one path = Joint probability of a sequence and a path, given the model

- GIVEN a HMM M, a path π, and a sequence x,
- FIND Prob[x, π | M]

➔ "Running the model", simply multiply emission and transition probabilities

➔ Application: "all promoter" vs. "all backgorund" comparisons

2. Scoring x, all paths = total probability of a sequence, summed across all paths

- GIVEN a HMM M, a sequence x
- FIND the total probability P[x | M] summed across all paths

➔ Forward algorithm, sum score over all paths (same result as backward)

SCORING

3. Viterbi decoding = parsing a sequence into the optimal series of hidden states

- GIVEN a HMM M, and a sequence x,
- FIND the sequence π^* of states that maximizes P[x, π | M]

➔ Viterbi algorithm, dynamic programming, max score over all paths, trace pointers find path

4. Posterior decoding = total prob that emission x_i came from state k, across all paths

- GIVEN a HMM M, a sequence x
- FIND the total probability P[π_i = k | x, M)

➔ Posterior decoding: run forward & backward algorithms to & from state π_l =k

PARSING

5. Supervised learning = optimize parameters of a model given training data

- GIVEN a HMM M, with unspecified transition/emission probs., labeled sequence x,
- FIND parameters $\theta = (e_i, a_{ij})$ that maximize P[x | θ]

➔ Simply count frequency of each emission and transition observed in the training data

6. Unsupervised learning = optimize parameters of a model given training data

- GIVEN a HMM M, with unspecified transition/emission probs., unlabeled sequence x,
- FIND parameters $\theta = (e_i, a_{ij})$ that maximize P[x | θ]

➔ Viterbi training: guess parameters, find optimal Viterbi path (#2), update parameters (#5), iterate

➔ Baum-Welch training: guess, sum over all emissions/transitions (#4), update (#5), iterate

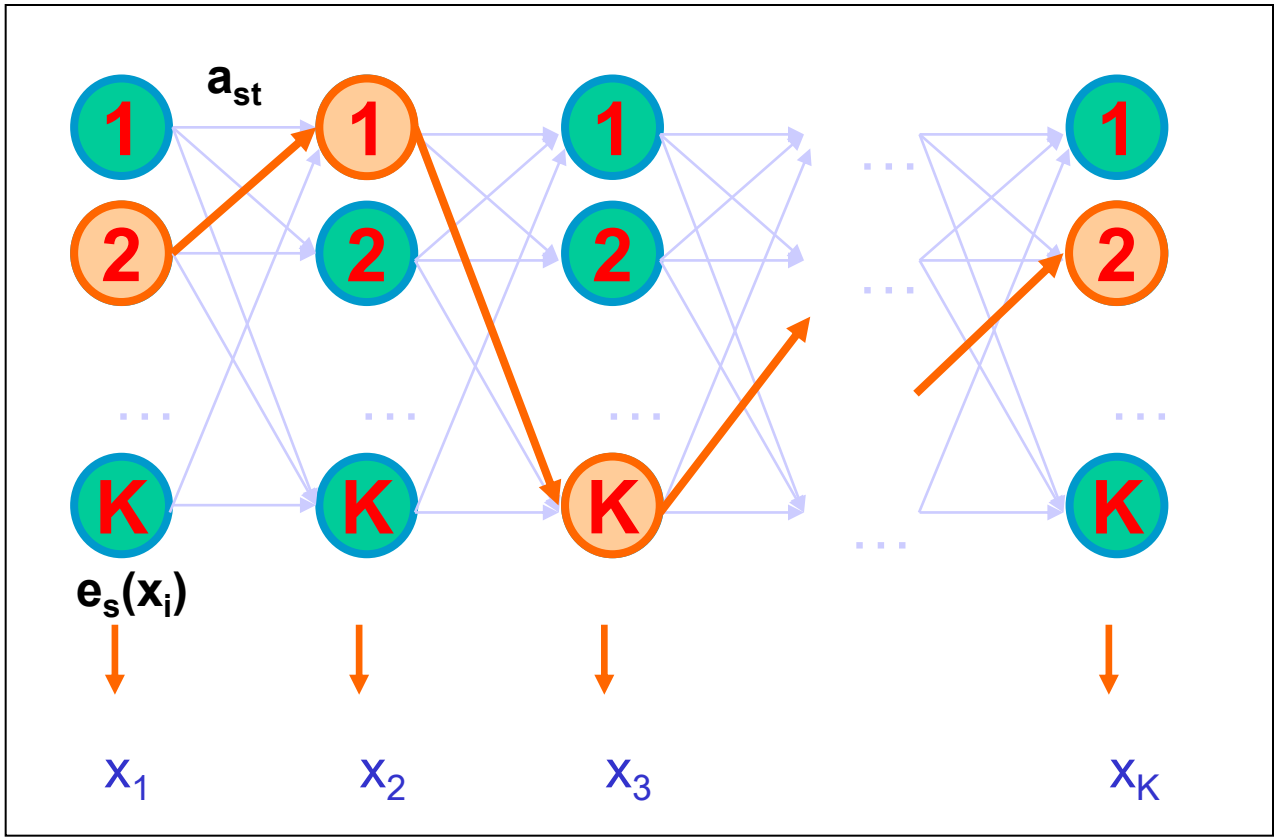
LEARNING

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path	4. Posterior decoding $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \sum_{\pi} P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

Probability of given path p, emissions x

π is the (hidden) path

x is the (observed) sequence

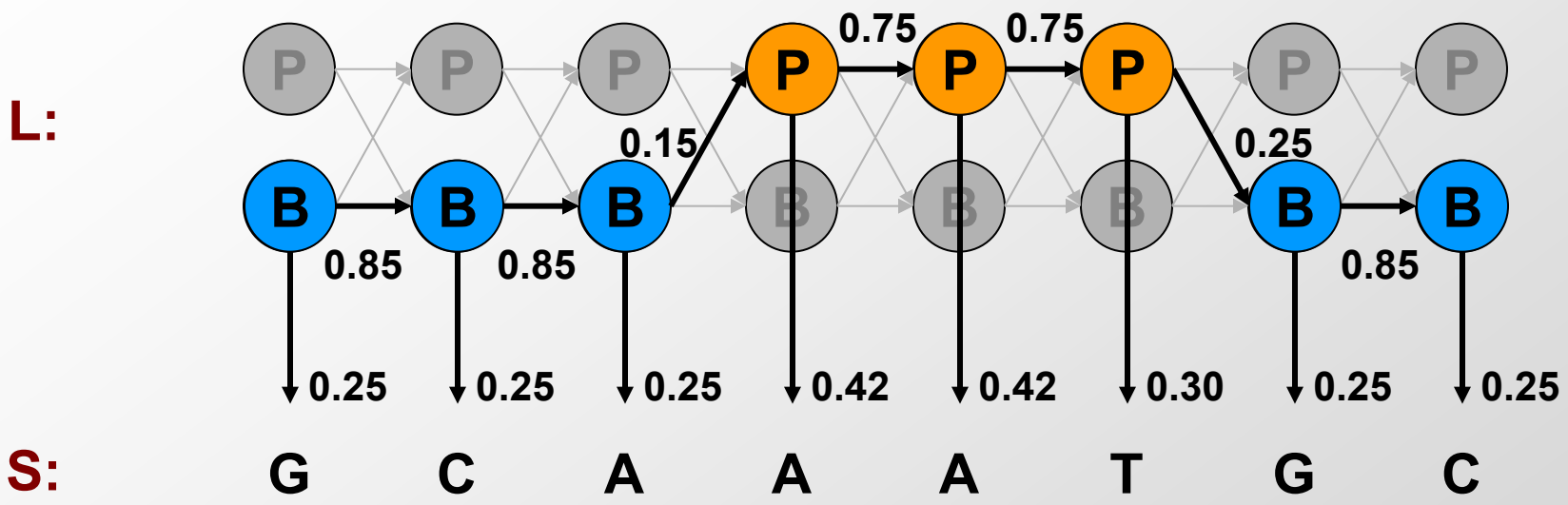


Courtesy of Serafim Batzoglou. Used with permission.

- $P(x,\pi) = a_{0\pi_1} * \prod_i e_{\pi_i}(x_i) \times a_{\pi_i\pi_{i+1}}$

start emission transition

Example: One particular P vs. B assignment



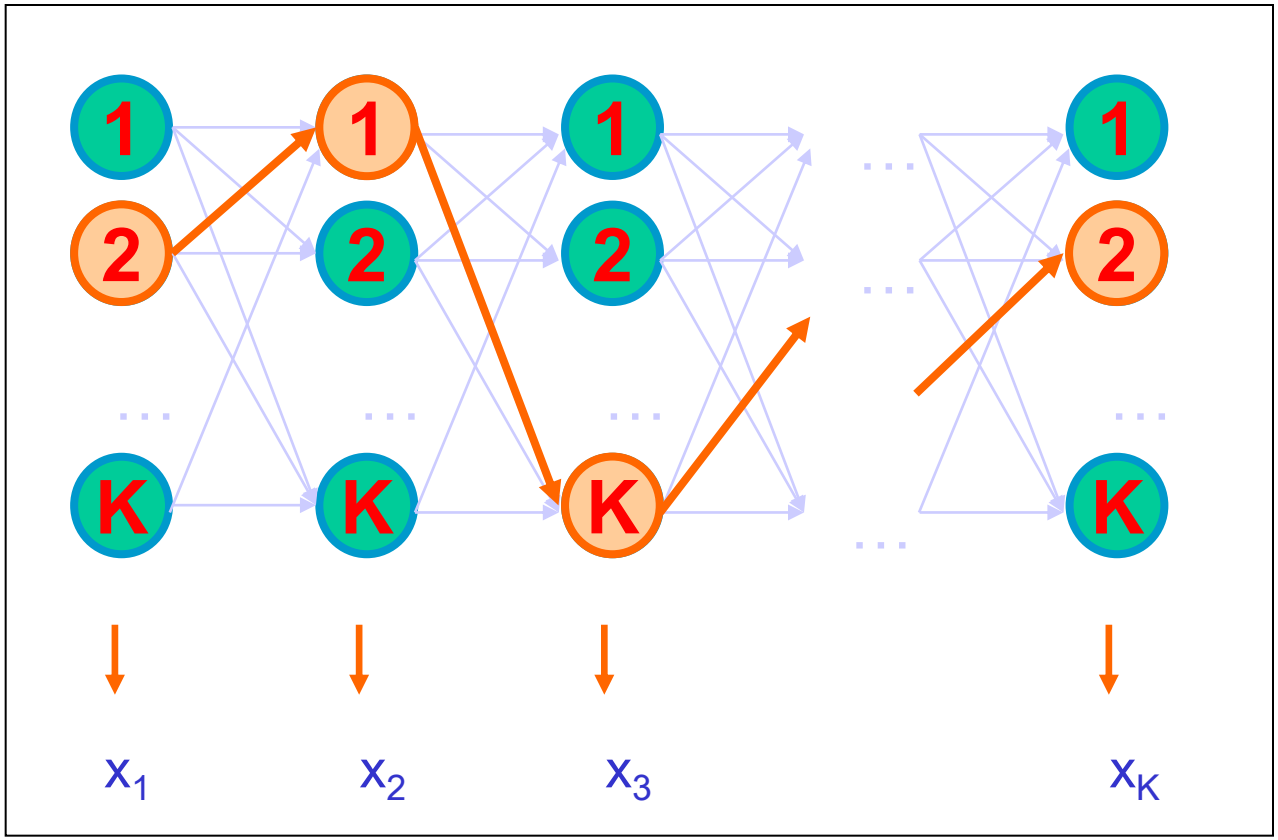
$$P = P(G|B)P(B_1|B_0)P(C|B)P(B_2|B_1)P(A|B)P(P_3|B_2)...P(C|B_7)$$

$$= (0.85)^3 \times (0.25)^6 \times (0.75)^2 \times (0.42)^2 \times 0.30 \times 0.15$$

$$= 6.7 \times 10^{-7}$$

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path	4. Posterior decoding $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \sum_{\pi} P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

Finding the most likely path



- Find path π^* that maximizes total joint probability $P[x, \pi]$
- $\text{argmax}_\pi P(x,\pi) = \text{argmax}_\pi \; a_{0\pi_1} * \prod_i e_{\pi_i}(x_i) \times a_{\pi_i \pi_{i+1}}$

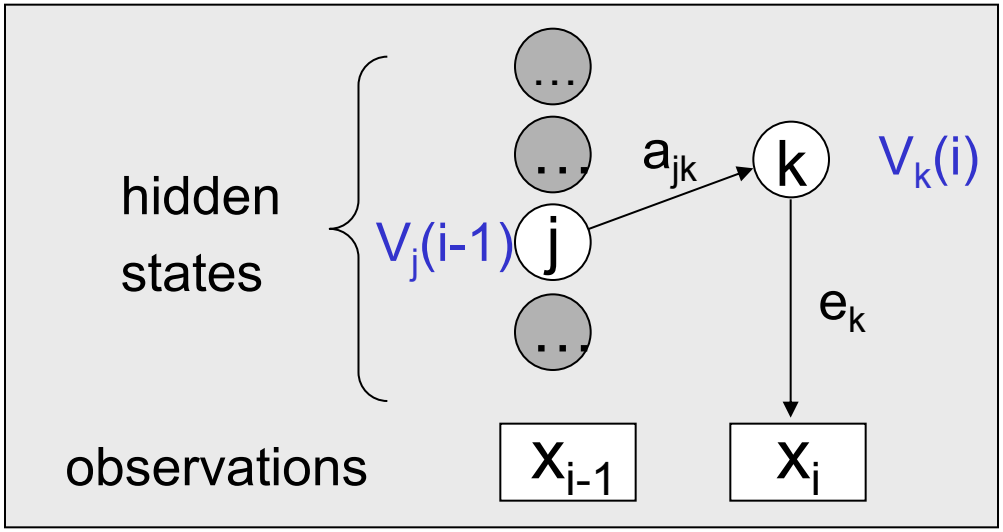
start emission transition

Calculate maximum P(x,π) recursively

Viterbi algortithm

Define $V_k(i)$ = Probability of the most likely path through state π_i=k

Compute $V_k(i+1)$ recursively, as a function of $\max_{k'} \{ V_{k'}(i) \}$



- Assume we know V_j for the previous time step (i-1)
- Calculate $V_k(i) = e_k(x_i) * \max_j (V_j(i-1) \times a_{jk})$

$V_k(i)$: current max

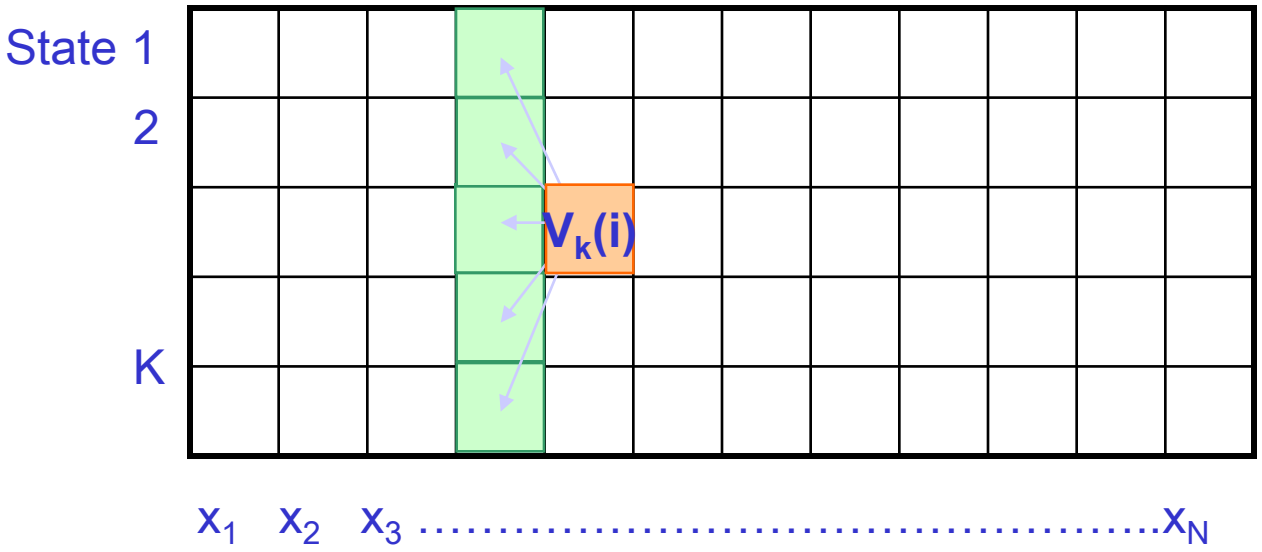
$e_k(x_i)$: this emission

$V_j(i-1)$: max ending in state j at step i

a_{jk}: Transition from state j

all possible previous states j

The Viterbi Algorithm



Input: x = x1……xN

Initialization:

$V_0(0)=1$, $V_k(0) = 0$, for all $k > 0$

Iteration:

$V_k(i) = e_K(x_i) \times \max_j a_{jk} V_j(i-1)$

Termination:

$P(x, \pi^*) = \max_k V_k(N)$

Traceback:

Follow max pointers back

In practice:

Use log scores for computation

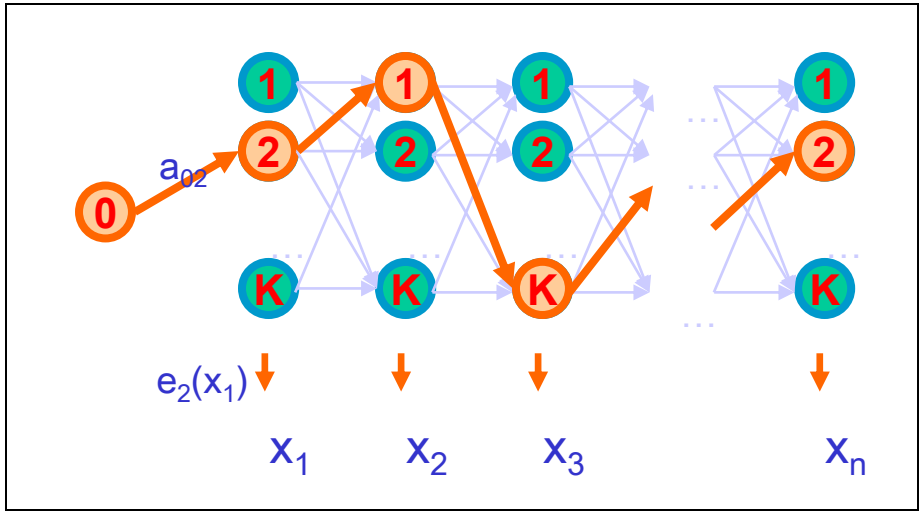
Running time and space:

Time: $O(K^2N)$

Space: $O(KN)$

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path	4. Posterior decoding $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi \mid \Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi \mid \Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \sum_{\pi} P(x,\pi \mid \Lambda)$ Baum-Welch training, over all paths

P(x) ⇔ Prob that model emits x, sum over all paths



Given a sequence x,
What is the probability that x was generated by the model (using any path)?

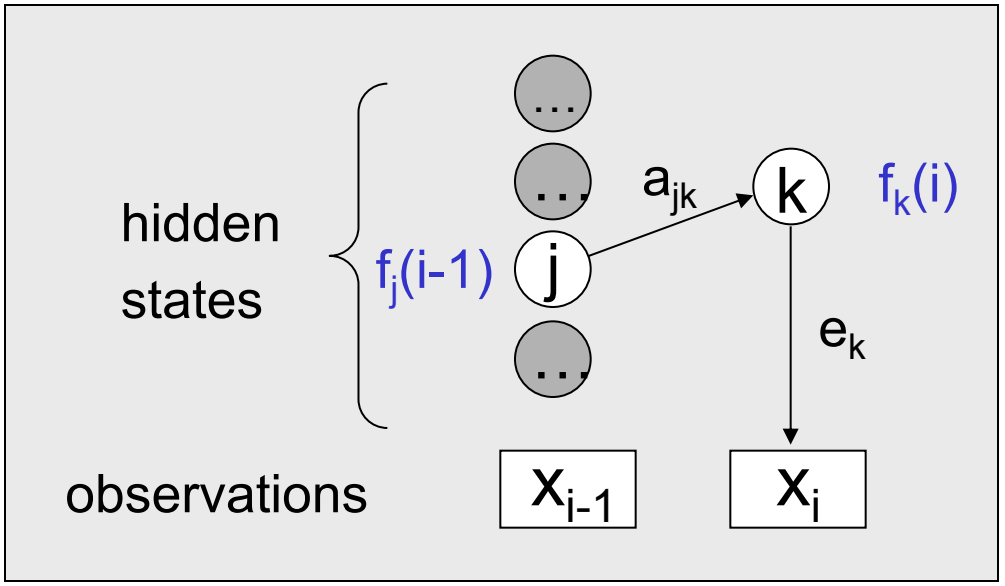
- $P(x) = \sum_{\pi} P(x,\pi)$

- Challenge: exponential number of paths
  - Sum over all paths, weighing the path probability, and the emission probs
  - Prob of emitting sequence: use individual emission probs from each state
  - Prob of path: use both emission and transition prob, based on previous path

- $P(x) = \sum_{\pi} a_{0\pi_1} * \prod_i e_{\pi_i}(x_i) \times a_{\pi_i\pi_{i+1}}$

start emission transition

Calculate total probability $\Sigma_\pi P(x,\pi)$ recursively



- Assume we know f_j for the previous time step (i-1)
- Calculate $f_k(i) = e_k(x_i) * sum_j (f_j(i-1) \times a_{jk})$

current sum

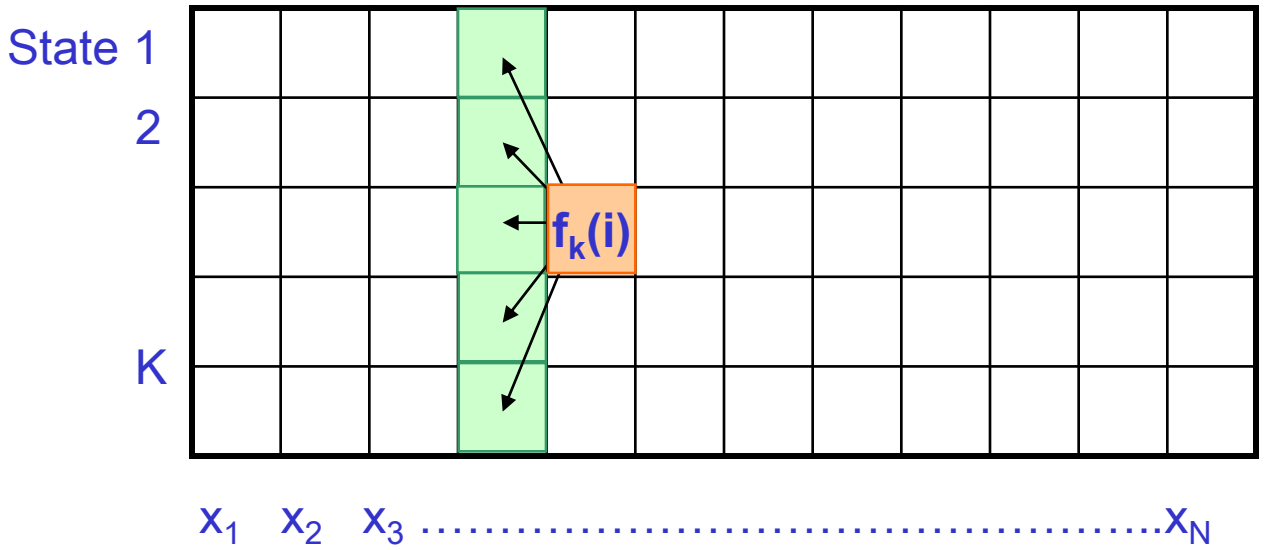
this emission

sum ending in state j at step i

transition from state j

Sum over all previous states j

The Forward Algorithm



Input: x = x1……xN

Initialization:

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

Iteration:

$f_k(i) = e_K(x_i) \times \text{sum}_j\ a_{jk}\ f_j(i\text{-}1)$

Termination:

$P(x, \pi^*) = \text{sum}_k\ f_k(N)$

In practice:

Sum of log scores is difficult

→ approximate exp(1+p+q)

→ scaling of probabilities

Running time and space:

Time: $O(K^2N)$

Space: $O(K)$

Goals for today: HMMs, part II

1. Review: Basics and three algorithms from last time
   - Markov Chains and Hidden Markov Models
   - Calculating likelihoods $P(x,\pi)$ (algorithm 1)
   - Viterbi algorithm: Find $\pi^* = \text{argmax}_\pi P(x,\pi)$ (alg 3)
   - Forward algorithm: Find $P(x)$, over all paths (alg 2)
2. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM
3. Posterior decoding: Another way of 'parsing'
   - Find most likely state π_i, sum over all possible paths
4. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params

Increasing the state space (remembering more)

HMM1: Promoters = **only Cs and Gs matter**

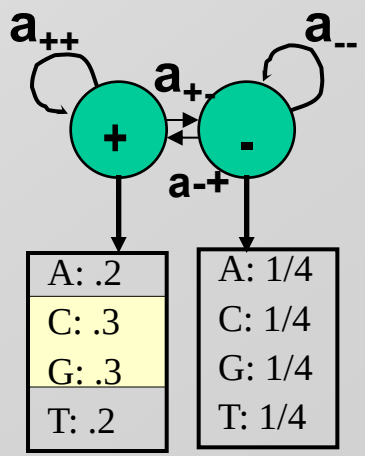
HMM2: Promoters = **it's actually CpGs that matter**

(di-nucleotides, remember previous nucleotide)

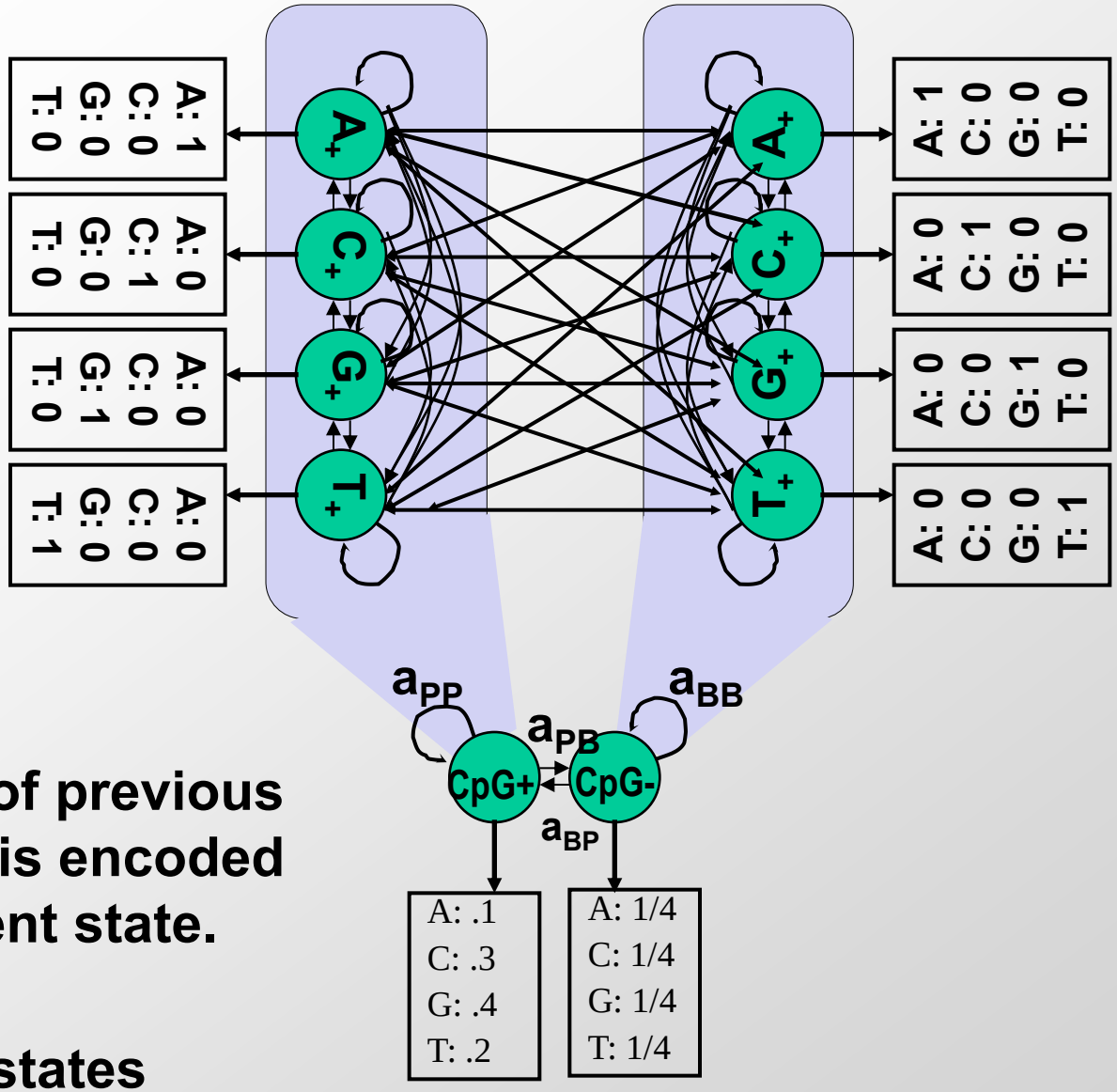
Increasing the state of the system (looking back)

- Markov Models are memory-less
  - In other words, all memory is encoded in the states
  - To remember additional information, augment state
- A two-state HMM has minimal memory
  - Two states: GC-rich vs. equal probability
  - State, emissions, only depend on **current** state
  - Current state only encodes **one** previous nucleotide
- How do you count **di**-nucleotide frequencies?
  - CpG islands: di-nucleotides
  - Codon triplets: tri-nucleotides
  - Di-codon frequencies: six nucleotides

➔ Expanding the number of states



Remember previous nucleotide: expand both states

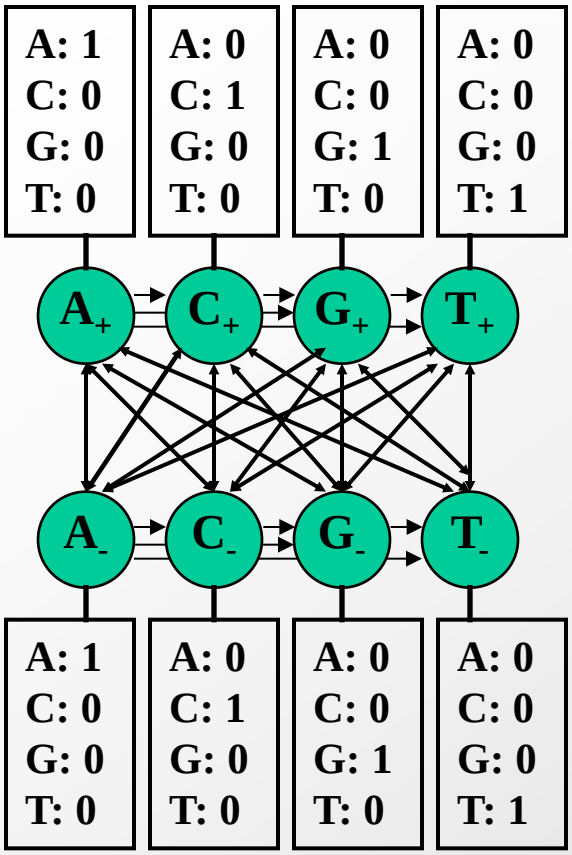


"Memory" of previous nucleotide is encoded in the current state.

GC-rich: 4 states

Background: 4 states

HMM for CpG islands



- A single model combines two Markov chains, each of four nucleotides:
  - **'+' 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-islands
- Emission probabilities distinct for the '+' and the '-' states
  - Infer most likely set of states, giving rise to observed emissions

➔ 'Paint' the sequence with + and - states

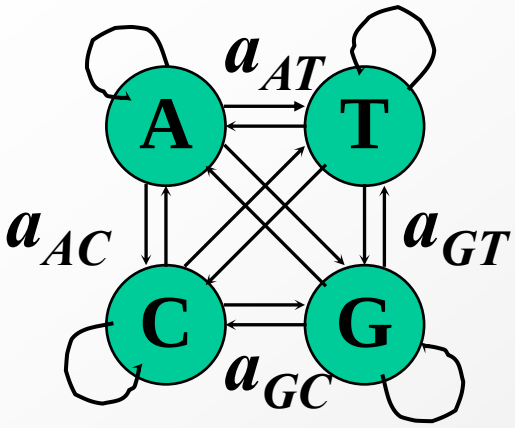
Why we need so many states…

In our simple GC-content example, we only had 2 states (+|-)

Why do we need 8 states here: 4 CpG+ / 4 CpG- ?

➔ Encode 'memory' of previous state: nucleotide transitions

Training emission parameters for CpG+/CpG- states



- Count di-nucleotide frequencies:
  - 16 possible di-nucleotides. 16 transition parameters.
  - Alternative: 16 states, each emitting di-nucleotide
- Derive two Markov chain models:
  - **'+' model**: from the CpG islands
  - **'-' model**: from the remainder of sequence
- Transition probabilities for each model:
  - Encode differences in di-nucleotide frequencies

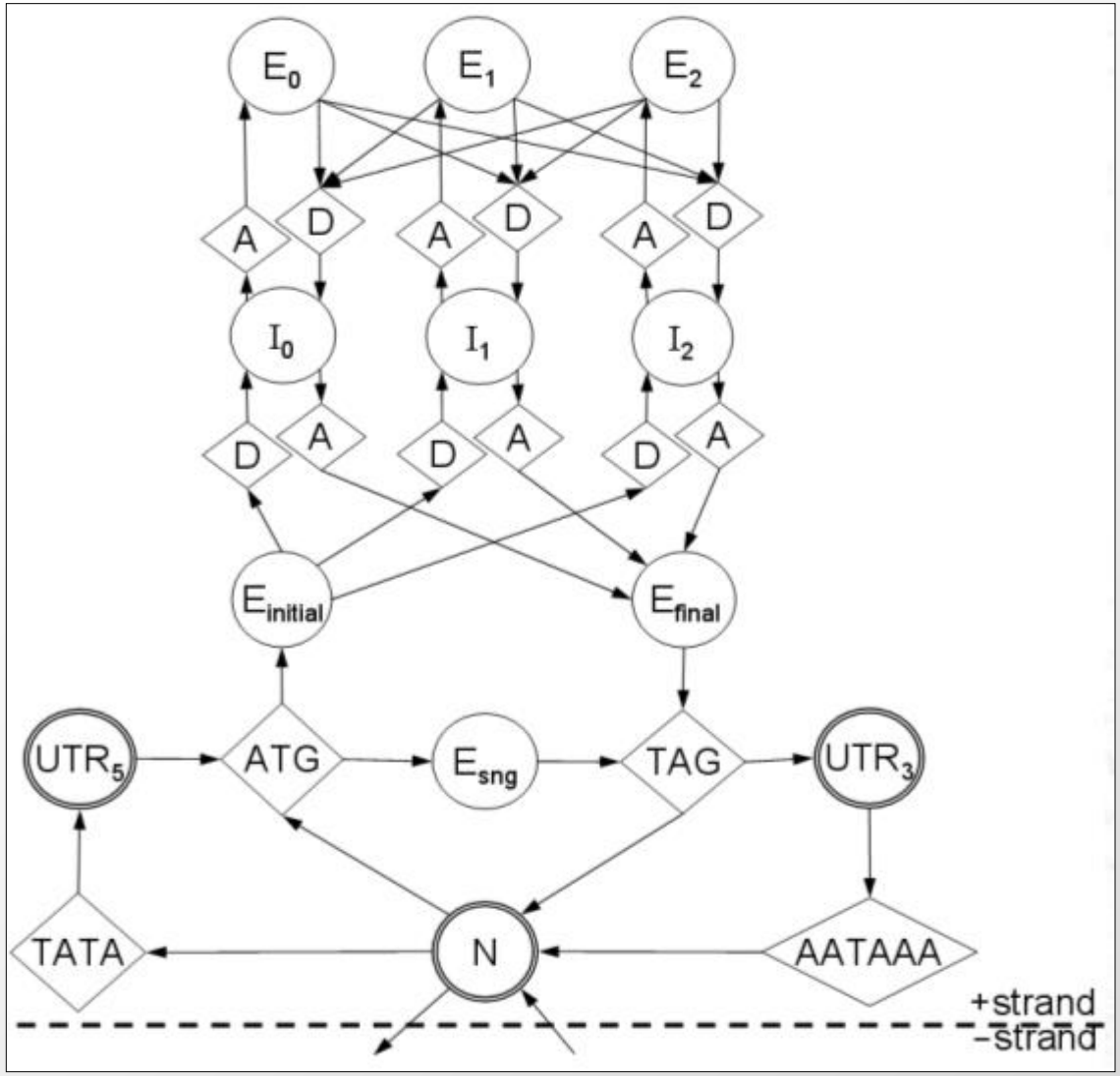
+	A	C	G	T
A	.180	.274	.426	.120
C	.171	.368	**.274**	.188
G	.161	.339	.375	.125
T	.079	.355	.384	.182

-	A	C	G	T
A	.300	.205	.285	.210
C	.322	.298	**.078**	.302
G	.248	.246	.298	.208
T	.177	.239	.292	.292

Examples of HMMs for genome annotation

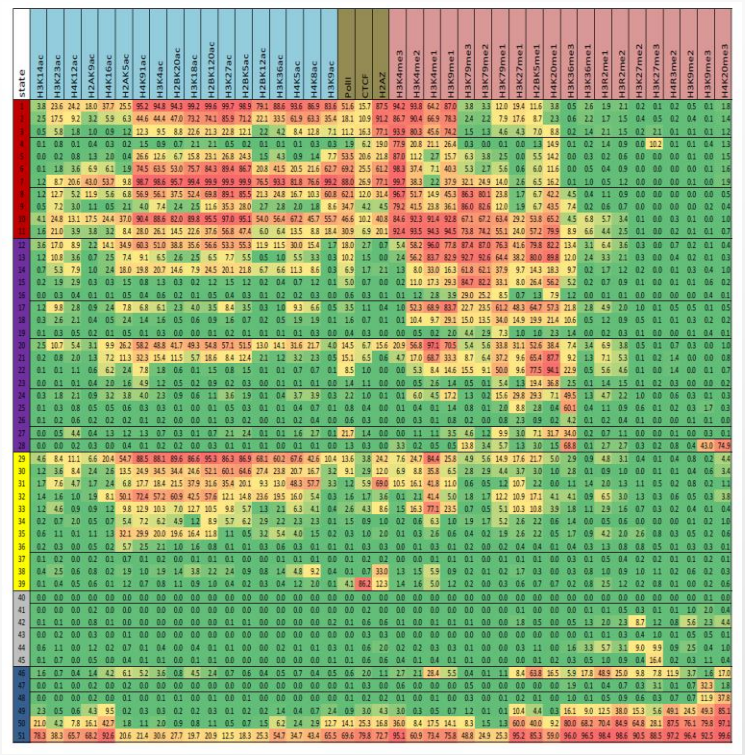
Detection of GC-rich regions	**Detection of CpG-rich regions**	**Detection of conserved regions**	**Detection of protein-coding exons**	**Detection of protein-coding conservation**	**Detection of protein-coding gene structures**	**Detection of chromatin states**
2 states, different nucleotide composition	8 states, 4 each +/-, different transition probabilities	2 states, different conservation levels	2 states, different tri-nucleotide composition	2 states, different evolutionary signatures	~20 states, different composition/ conservation, specific structure	40 states, different chromatin mark combinations
GC-rich / AT-rich	CpG-rich / CpG-poor	Conserved / non-conserved	Coding exon / non-coding (intron or intergenic)	Coding exon / non-coding (intron or intergenic)	First/last/middle coding exon,UTRs, intron1/2/3, intergenic, *(+/- strand)	Enhancer / promoter / transcribed / repressed / repetitive
Nucleotides	Di-Nucleotides	Level of conservation	Triplets of nucleotides	64x64 matrix of codon substitution frequencies	Codons, nucleotides, splice sites, start/stop codons	Vector of chromatin mark frequencies

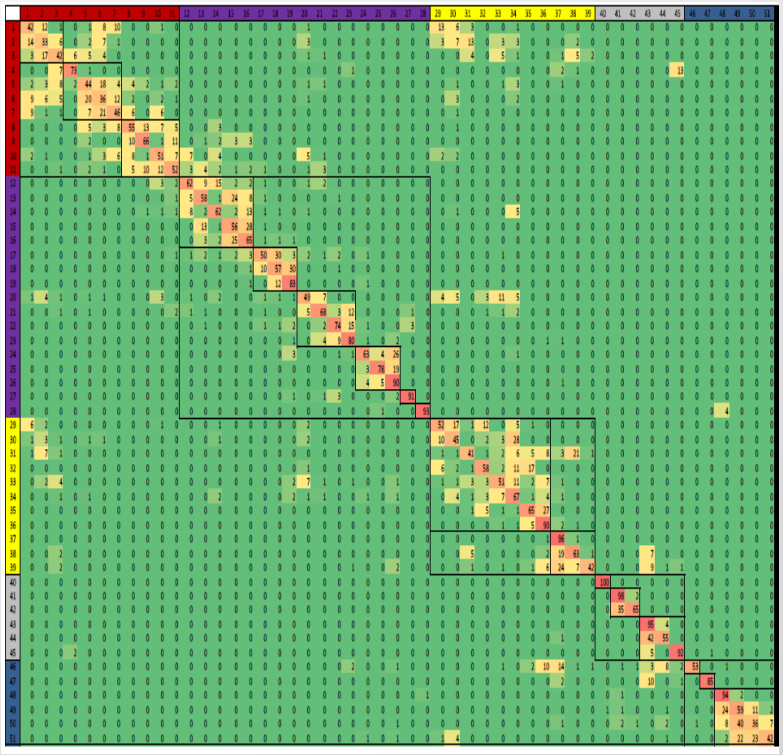
HMM architecture matters: Protein-coding genes



- Gene vs. Intergenic
- Start & Stop in/out
- UTR: 5' and 3' end
- Exons, Introns
- Remembering frame
  - E0,E1,E2
  - I0,I1,I2
- Sequence patterns to transition between states:
  - ATG, TAG, Acceptor/Donor, TATA, AATAA

Chromatin State: Emission & Transition Matrices





Source: Ernst, Jason and Manolis Kellis. "Discovery and characterization of chromatin states for systematic annotation of the human genome." Nature Biotechnology 28, no. 8 (2010): 817-825.

- **Emission matrix:**
  - Multi-variate HMM
  - Emits vector of values
- **Transition matrix:**
  - Learn spatial relationships
  - No a-priori ‘gene’ structure

Ernst and Kellis, Nature Biotech 2010, Nature 2011, Nature Methods 2012

Goals for today: HMMs, part II

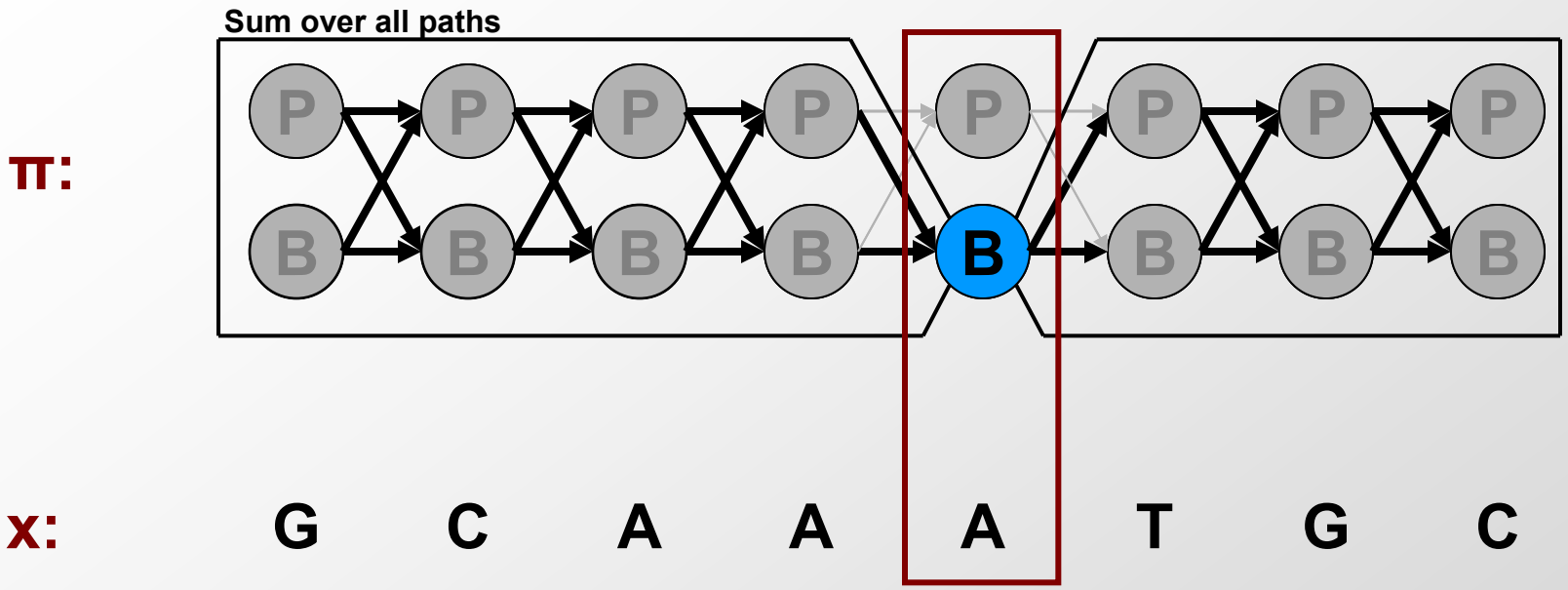
1. Review: Basics and three algorithms from last time
   - Markov Chains and Hidden Markov Models
   - Calculating likelihoods $P(x,\pi)$ (algorithm 1)
   - Viterbi algorithm: Find $\pi^* = \text{argmax}_\pi P(x,\pi)$ (alg 3)
   - Forward algorithm: Find $P(x)$, over all paths (alg 2)
2. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM
3. Posterior decoding: Another way of 'parsing'
   - Find most likely state π_i, sum over all possible paths
4. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions ✓	2. Scoring x, all paths $P(x) = \sum_\pi P(x,\pi)$ ✓ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_\pi P(x,\pi)$ Most likely path ✓	4. Posterior decoding $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_\pi P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_\Lambda P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_\Lambda \max_\pi P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_\Lambda \sum_\pi P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

4. Decoding, all paths

Find the likelihood an emission x_i is generated by a state

Calculate most probable label at a single position



- Calculate most probable label, L^*_i , at each position i
- Do this for all N positions gives us $\{L^*_1, L^*_2, L^*_3 \ldots L^*_N\}$
- How much information have we observed? Three settings:
  - **Observed nothing: Use prior information**
  - **Observed only character at position i: Prior + emission probability**
  - **Observed entire sequence: Posterior decoding**

Calculate $P(\pi_7 = \texttt{CpG+} \mid x_7 = G)$

- With no knowledge (no characters)
  - Simply time spent in markov chain states
  - $P(\pi_i = k)$ = most likely state (**prior**)

- With very little knowledge (just that character)
  - Time spent, adjusted for different emission probs.
  - Use Bayes rule to change inference directionality
  - $P(\pi_i = k \mid x_i = G) = P(\pi_i = \kappa) * P(x_i = G \mid \pi_i = k) / P(x_i = G)$

- With knowledge of entire sequence (all characters)
  - $P(\pi_i = k \mid x = \text{AGCGCG…GATTATCGTCGTA})$
  - Sum over all paths that emit 'G' at position 7

  ➔ **Posterior** decoding

Motivation for the Backward Algorithm

We want to compute

$P(\pi_i = k \mid x)$, the probability distribution on the i^{th} position, given x

We start by computing

$$P(\pi_i = k, x) = P(x_1 \ldots x_i, \pi_i = k, x_{i+1} \ldots x_N)$$
$$= P(x_1 \ldots x_i, \pi_i = k)\, P(x_{i+1} \ldots x_N \mid x_1 \ldots x_i, \pi_i = k)$$
$$= P(x_1 \ldots x_i, \pi_i = k)\, P(x_{i+1} \ldots x_N \mid \pi_i = k)$$

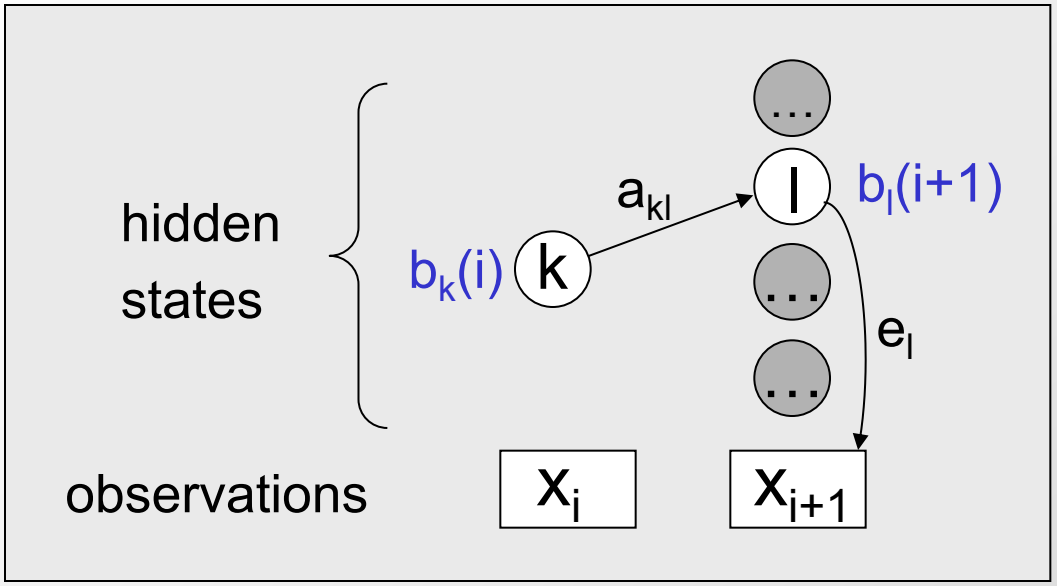
Forward, $f_k(i)$ Backward, $b_k(i)$

The Backward Algorithm – derivation

Define the backward probability:

$$
\begin{aligned}
b_k(i) &= P(x_{i+1} \ldots x_N \mid \pi_i = k) \\
&= \Sigma_{\pi i+1 \ldots \pi N} P(x_{i+1}, x_{i+2}, \ldots, x_N, \pi_{i+1}, \ldots, \pi_N \mid \pi_i = k) \\
&= \Sigma_l \Sigma_{\pi i+1 \ldots \pi N} P(x_{i+1}, x_{i+2}, \ldots, x_N, \pi_{i+1} = l, \pi_{i+2}, \ldots, \pi_N \mid \pi_i = k) \\
&= \Sigma_l e_l(x_{i+1}) a_{kl} \boxed{\Sigma_{\pi i+1 \ldots \pi N} P(x_{i+2}, \ldots, x_N, \pi_{i+2}, \ldots, \pi_N \mid \pi_{i+1} = l)} \\
&= \Sigma_l e_l(x_{i+1}) a_{kl} \boxed{b_l(i+1)}
\end{aligned}
$$

Calculate total end probability recursively



- Assume we know b_l for the next time step (i+1)
- Calculate $b_k(i)$ = sum_l ($e_l(x_{i+1})$ × a_{kl} × $b_l(i+1)$)

current max

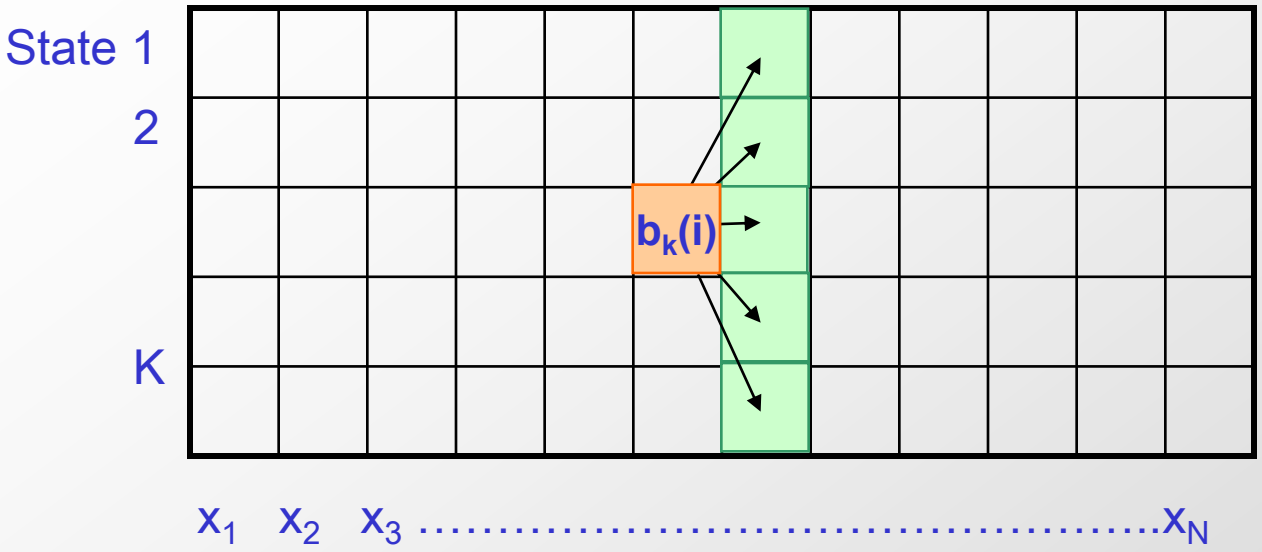
next emission

transition to next state

prob sum from state l to end

sum over all possible next states

The Backward Algorithm



Input: x = x1……xN

Initialization:

$b_k(N) = a_{k0}$, for all k

Iteration:

$b_k(i) = \Sigma_l\, e_l(x_{i+1})\, a_{kl}\, b_l(i+1)$

Termination:

$P(x) = \Sigma_l\, a_{0l}\, e_l(x_1)\, b_l(1)$

In practice:

Sum of log scores is difficult

→ approximate exp(1+p+q)

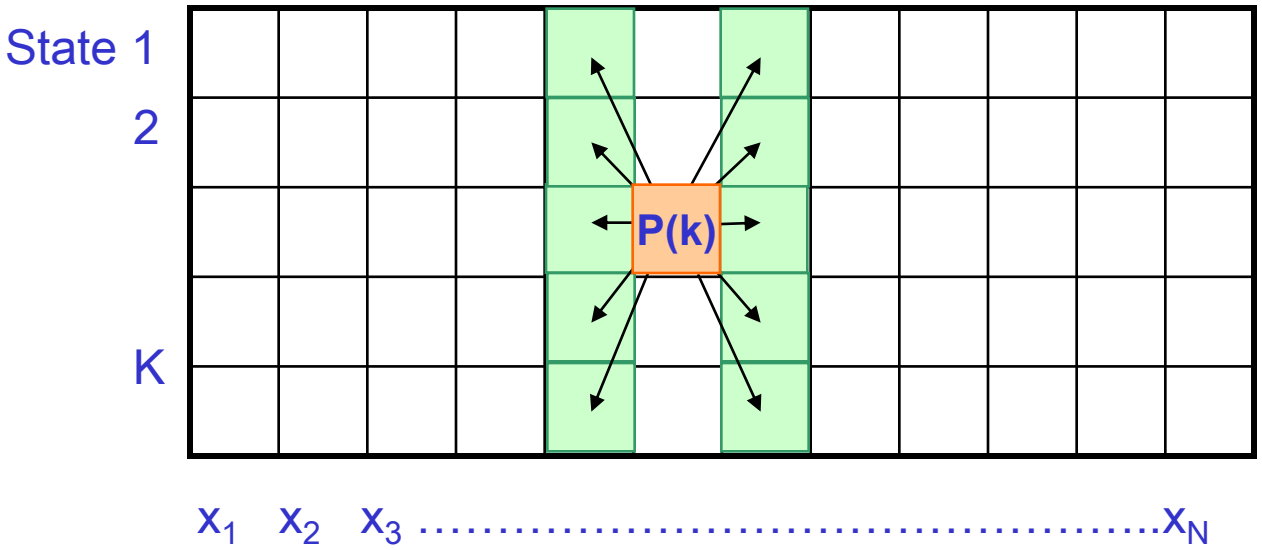
→ scaling of probabilities

Running time and space:

Time: $O(K^2N)$

Space: $O(K)$

Putting it all together: Posterior decoding



- $P(k) = P(\pi_i=k \mid x) = f_k(i)*b_k(i) / P(x)$
  - Probability that i^{th} state is k, given all emissions x
- Posterior decoding
  - Find the most likely state at position i over all possible hidden paths given the observed sequence x
  - $\hat{\pi}_i = \text{argmax}_k P(\pi_i = k \mid x)$
- Posterior decoding 'path' $\hat{\pi}_i$
  - For classification, more informative than Viterbi path π^*
    - More refined measure of "which hidden states" generated x
  - However, it may give an invalid sequence of states
    - Not all j→k transitions may be possible

Goals for today: HMMs, part II

1. Review: Basics and three algorithms from last time
   - Markov Chains and Hidden Markov Models
   - Calculating likelihoods $P(x,\pi)$ (algorithm 1)
   - Viterbi algorithm: Find $\pi^* = \text{argmax}_\pi P(x,\pi)$ (alg 3)
   - Forward algorithm: Find $P(x)$, over all paths (alg 2)
2. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM
3. Posterior decoding: Another way of 'parsing'
   - Find most likely state π_i, sum over all possible paths
4. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_\pi P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_\pi P(x,\pi)$ Most likely path	4. Posterior decoding $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_\pi P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_\Lambda P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_\Lambda \max_\pi P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_\Lambda \sum_\pi P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

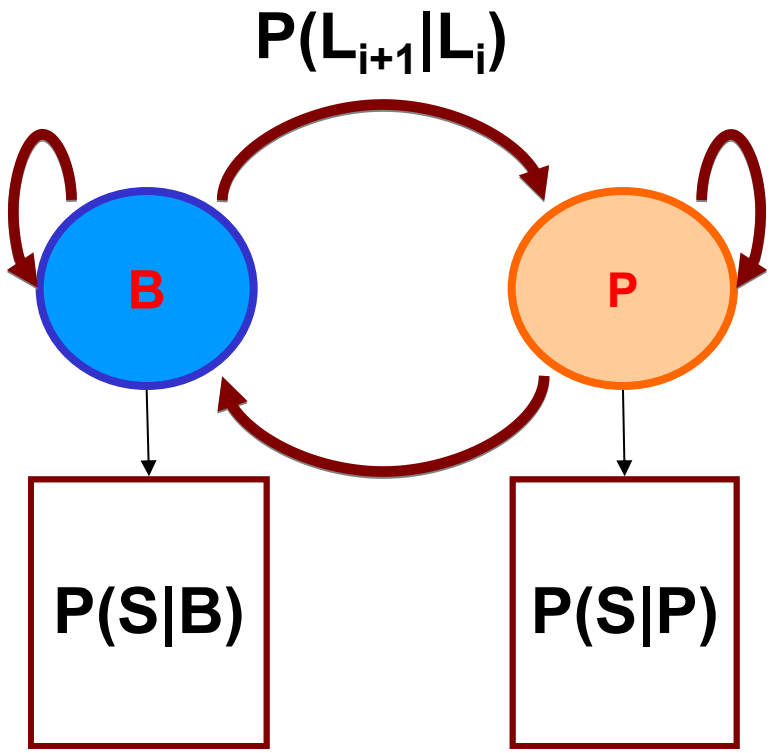
Learning: How to train an HMM

Transition probabilities

e.g. $P(P_{i+1}|B_i)$ – the probability of entering a pathogenicity island from background DNA

Emission probabilities

i.e. the nucleotide frequencies for background DNA and pathogenicity islands



Two learning scenarios

Case 1. Estimation when the "right answer" is known

Examples:

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

Case 2. Estimation when the "right answer" is unknown

Examples:

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

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

Two types of learning: Supervised / Unsupervised

5. Supervised learning

infer model parameters given **labeled** training data

- GIVEN:
  - a HMM M, with unspecified transition/emission probs.
  - labeled sequence x,
- FIND:
  - parameters θ = (Ei, Aij) that maximize P[x | θ]

➔ Simply count frequency of each emission and transition, as observed in the training data

6. Unsupervised learning

infer model parameters given **unlabelled** training data

- GIVEN:
  - a HMM M, with unspecified transition/emission probs.
  - unlabeled sequence x,
- FIND:
  - parameters θ = (Ei, Aij) that maximize P[x | θ]

➔ Viterbi training:
guess parameters, find optimal Viterbi path (#2), update parameters (#5), iterate

➔ Baum-Welch training:
guess parameters, sum over all paths (#4), update parameters (#5), iterate

5: Supervised learning

Estimate model parameters
based on **labeled** training data

Case 1. When the right answer is known

Given $x = x_1 \ldots x_N$
for which the true $\pi = \pi_1 \ldots \pi_N$ is known,

Define:

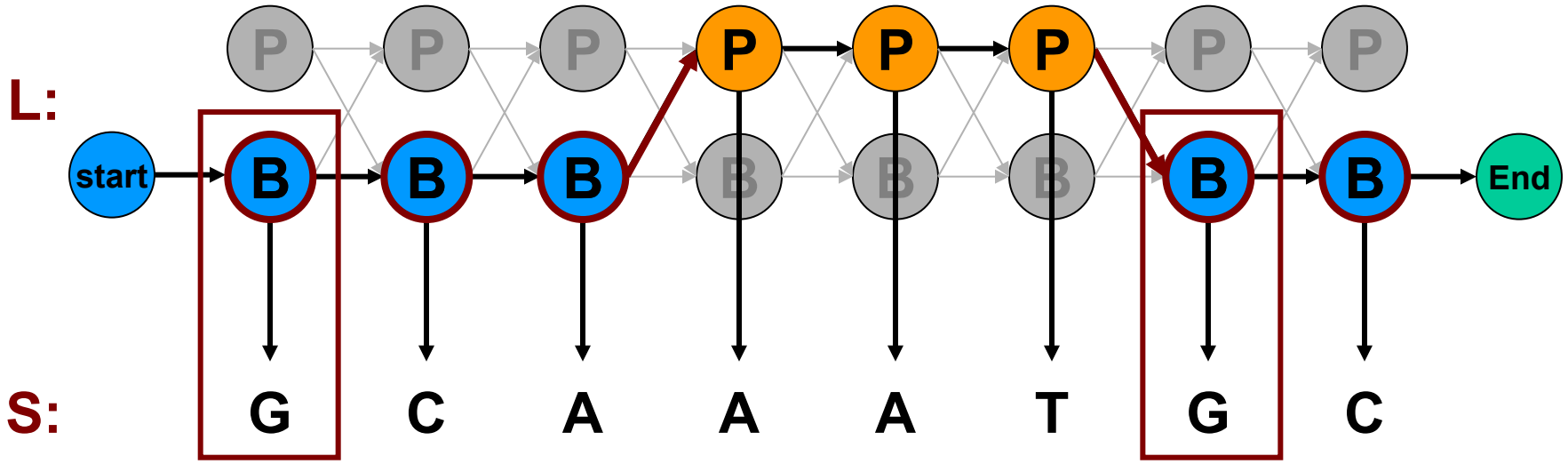
A_{kl} = # times $k \rightarrow l$ transition occurs in π
$E_k(b)$ = # times state k in π emits b in x

We can show that the maximum likelihood parameters θ are:

$$a_{kl} = \frac{A_{kl}}{\sum_i A_{ki}} \qquad e_k(b) = \frac{E_k(b)}{\sum_c E_k(c)}$$

Learning From Labelled Data
→ Maximum Likelihood Estimation

If we have a sequence that has islands marked, we can simply count



$P(L_{i+1}|L_i)$

	B_{i+1}	P_{i+1}
B_i		
P_i		

$P(S|B)$

A:
T:
G:
C:
!

$P(S|P)$

A:
T:
G:
C:
ETC..

Case 1. When the right answer is known

Intuition: When we know the underlying states,
Best estimate is the average frequency of
transitions & emissions that occur in the training data

Drawback:

Given little data, there may be **overfitting**:
$P(x|\theta)$ is maximized, but θ is unreasonable
0 probabilities – VERY BAD

Example:

Given 10 nucleotides, we observe

x = C, A, G, G, T, C, C, A, T, C
π = P, P, P, p, p, P, P, P, P, P

Then:

$a_{PP} = 1$; $a_{PB} = 0$
$e_P(A) = .2$;
$e_P(C) = .4$;
$e_P(G) = .2$;
$e_P(T) = .2$

Pseudocounts

Solution for small training sets:

Add pseudocounts

A_{kl} = # times k$\rightarrow$l transition occurs in π + r_{kl}

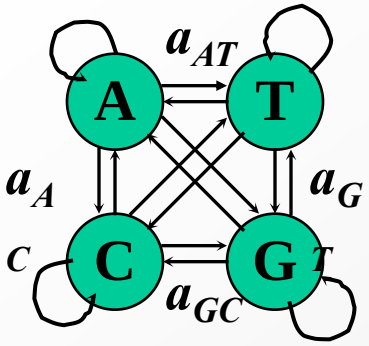
$E_k(b)$ = # times state k in π emits b in x + $r_k(b)$

r_{kl}, $r_k(b)$ are pseudocounts representing our prior belief

Larger pseudocounts $\Rightarrow$ Strong priof belief

Small pseudocounts ($\varepsilon < 1$): just to avoid 0 probabilities

Example: Training Markov Chains for CpG islands



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

-	A	C	G	T
A	.300	.205	.285	.210
C	.322	.298	**.078**	.302
G	.248	.246	.298	.208
T	.177	.239	.292	.292

$$a_{st}^{+} = \frac{c_{st}^{+}}{\sum_{t'} c_{st'}^{+}}$$

c_{st}^{+} **is the number of times letter *t* followed letter *s* inside the CpG islands**

$$a_{st}^{-} = \frac{c_{st}^{-}}{\sum_{t'} c_{st'}^{-}}$$

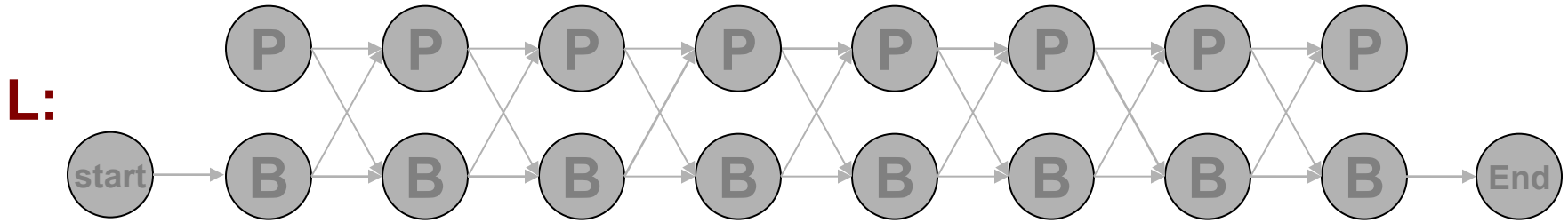
c_{st}^{-} **is the number of times letter *t* followed letter *s* outside the CpG islands**

6: Unsupervised learning

Estimate model parameters
based on **unlabeled** training data

Unlabelled Data

How do we know how to count?



S: G C A A A T G C

?

$P(L_{i+1}|L_i)$

	B_{i+1}	P_{i+1}	End
B_i			
P_i			
Start			

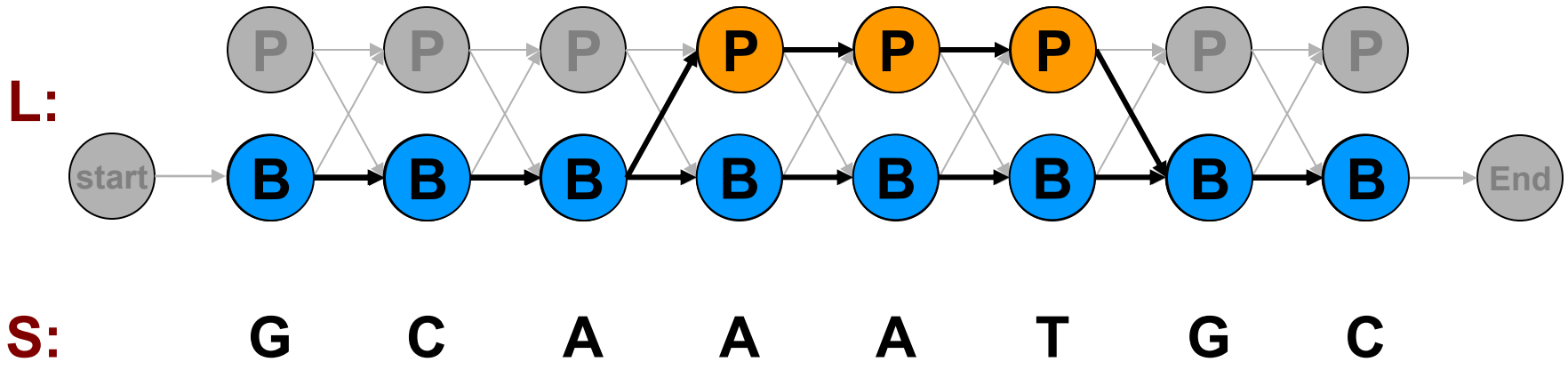
P(S|B)

A:
T:
G:
C:

P(S|P)

A:
T:
G:
C:

Unlabeled Data



An idea:

1. Imagine we start with some parameters
2. We *could* calculate the most likely path, P*, given those parameters and S
3. We *could* then use P* to update our parameters by maximum likelihood
4. And iterate (to convergence)

~~$P(L_{i+1}|L_i)^0$ $P(S|B)^0$ $P(S|P)^0$~~

~~$P(L_{i+1}|L_i)^1$ $P(S|B)^1$ $P(S|P)^1$~~

$P(L_{i+1}|L_i)^2$ $P(S|B)^2$ $P(S|P)^2$

…

$P(L_{i+1}|L_i)^K$ $P(S|B)^K$ $P(S|P)^K$

Learning case 2. When the right answer is unknown

We don't know the true A_{kl}, $E_k(b)$

Idea:

- We estimate our "best guess" on what A_{kl}, $E_k(b)$ are (M step, maximum-likelihood estimation)
- We update the probabilistic parse of our sequence, based on these parameters (E step, expected probability of being in each state given parameters)
- We repeat

Two settings:

- Simple: Viterbi training (best guest = best path)
- Correct: Expectation maximization (all paths, weighted)

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions ✓	2. Scoring x, all paths $P(x) = \sum_\pi P(x,\pi)$ Prob of emissions, over all paths ✓
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_\pi P(x,\pi)$ Most likely path ✓	4. Posterior decoding $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_\pi P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point. ✓
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_\Lambda P(x,\pi\mid\Lambda)$ ✓ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_\Lambda \max_\pi P(x,\pi\mid\Lambda)$ Viterbi training, best path	7. Unsupervised learning $\Lambda^* = \text{argmax}_\Lambda \sum_\pi P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

Simple casae: Viterbi Training

Initialization:

Pick the best-guess for model parameters

(or arbitrary)

Iteration:

1. Perform Viterbi, to find π^*
2. Calculate A_{kl}, $E_k(b)$ according to π^* + pseudocounts
3. Calculate the new parameters a_{kl}, $e_k(b)$

Until convergence

Notes:

- Convergence to local maximum guaranteed. Why?
- Does not maximize $P(x \mid \theta)$
- In general, worse performance than Baum-Welch

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions ✓	2. Scoring x, all paths $P(x) = \sum_\pi P(x,\pi)$ Prob of emissions, over all paths ✓
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_\pi P(x,\pi)$ Most likely path ✓	4. Posterior decoding $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_\pi P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point. ✓
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_\Lambda P(x,\pi\mid\Lambda)$ ✓ 6. Unsupervised learning. ✓ $\Lambda^* = \text{argmax}_\Lambda \max_\pi P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_\Lambda \sum_\pi P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

Expectation Maximization (EM)

The basic idea is the same:

1.Use model to estimate missing data (E step)
2.Use estimate to update model (M step)
3.Repeat until convergence

EM is a general approach for learning models (ML estimation) when there is "missing data"
Widely used in computational biology

EM pervasive in computational biology

➡ Rec 3 (SiPhy), Lec 8 (Kmeans), Lec 9 (motifs)

Expectation Maximization (EM)

1. **Initialize parameters randomly**

2. **E Step Estimate expected probability of hidden labels, Q, given current (latest) parameters and observed (unchanging) sequence**

$$Q = P(Labels | S, params^{t-1})$$

3. **M Step Choose new maximum likelihood parameters over probability distribution Q, given current probabilistic label assignments**

$$params^{t} = \arg\max_{params} E_{Q}\left[\log P(S, labels \mid params^{t-1})\right]$$

4. **Iterate**

P(S|Model) *guaranteed* to increase each iteration

Case 2. When the right answer is unknown

Starting with our best guess of a model M, parameters θ:

Given $x = x_1 \ldots x_N$

for which the true $\pi = \pi_1 \ldots \pi_N$ is unknown,

We can get to a provably more likely parameter set θ

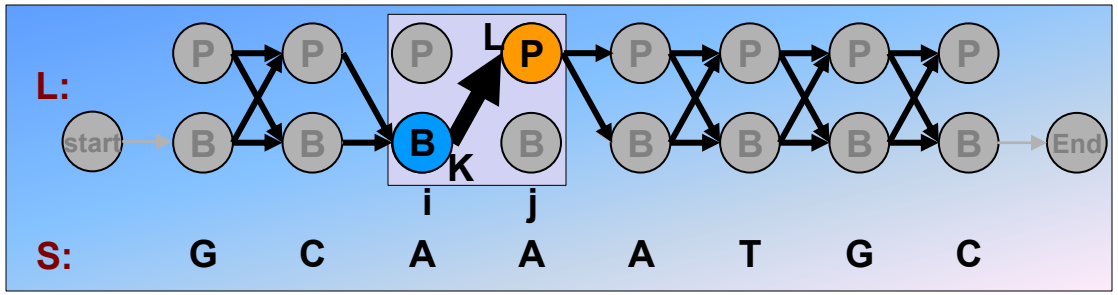
Principle: EXPECTATION MAXIMIZATION

1. Estimate probabilistic parse based on parameters (E step)
2. Update parameters A_{kl}, E_k based on probabilistic parse (M step)
3. Repeat 1 & 2, until convergence

Estimating probabilistic parse given params (E step)

To estimate $\mathbf{A_{kl}}$:

At each position i:



Find probability transition k→l is used:

$$P(\pi_i = k, \pi_{i+1} = l \mid x) = [1/P(x)] \times P(\pi_i = k, \pi_{i+1} = l, x_1 \ldots x_N) = Q/P(x)$$

where $Q = P(x_1 \ldots x_i, \pi_i = k, \pi_{i+1} = l, x_{i+1} \ldots x_N) =$

$= P(\pi_{i+1} = l, x_{i+1} \ldots x_N \mid \pi_i = k) P(x_1 \ldots x_i, \pi_i = k) =$

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

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

$= b_l(i+1) e_l(x_{i+1}) a_{kl} f_k(i)$

So: $$\mathbf{P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \frac{f_k(i)\, a_{kl}\, e_l(x_{i+1})\, b_l(i+1)}{P(x \mid \theta)}}$$

(For one such transition, at time step i→i+1)

New parameters given probabilistic parse (M step)

(Sum over all k→l transitions, at any time step i)

So,

$$A_{kl} = \sum_i P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \sum_i \frac{f_k(i)\, a_{kl}\, e_l(x_{i+1})\, b_l(i+1)}{P(x \mid \theta)}$$

Similarly,

$$E_k(b) = [1/P(x)] \sum_{\{i \mid x_i = b\}} f_k(i)\, b_k(i)$$

Dealing with multiple training sequences

(Sum over all training seqs, all k→l transitions, all time steps i)

If we have several training sequences, $x^1, \ldots, x^M$, each of length N,

$$A_{kl} = \sum_x \sum_i P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \sum_x \sum_i \frac{f_k(i)\, a_{kl}\, e_l(x_{i+1})\, b_l(i+1)}{P(x \mid \theta)}$$

Similarly,

$$E_k(b) = \sum_x (1/P(x)) \sum_{\{i \mid x^i = b\}} f_k(i)\, b_k(i)$$

The Baum-Welch Algorithm

Initialization:

Pick the best-guess for model parameters

(or arbitrary)

Iteration:

1. Forward
2. Backward
3. ➔ Calculate new log-likelihood $P(x \mid \theta)$ (E step)
4. Calculate A_{kl}, $E_k(b)$
5. ➔ Calculate new model parameters a_{kl}, $e_k(b)$ (M step)

GUARANTEED TO BE HIGHER BY EXPECTATION-MAXIMIZATION

Until $P(x \mid \theta)$ does not change much

The Baum-Welch Algorithm – comments

Time Complexity:

iterations $\times$ $O(K^2N)$

- Guaranteed to increase the log likelihood of the model

$P(\theta \mid x) = P(x, \theta) / P(x) = P(x \mid \theta) / (P(x) P(\theta))$

- Not guaranteed to find globally best parameters

Converges to local optimum, depending on initial conditions

- Too many parameters / too large model: Overtraining

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path	4. Posterior decoding $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \sum_{\pi} P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

Examples of HMMs for genome annotation

Detection of GC-rich regions	**Detection of CpG-rich regions**	**Detection of conserved regions**	**Detection of protein-coding exons**	**Detection of protein-coding conservation**	**Detection of protein-coding gene structures**	**Detection of chromatin states**
2 states, different nucleotide composition	8 states, 4 each +/-, different transition probabilities	2 states, different conservation levels	2 states, different tri-nucleotide composition	2 states, different evolutionary signatures	~20 states, different composition/ conservation, specific structure	40 states, different chromatin mark combinations
GC-rich / AT-rich	CpG-rich / CpG-poor	Conserved / non-conserved	Coding exon / non-coding (intron or intergenic)	Coding exon / non-coding (intron or intergenic)	First/last/middle coding exon,UTRs, intron1/2/3, intergenic, *(+/- strand)	Enhancer / promoter / transcribed / repressed / repetitive
Nucleotides	Di-Nucleotides	Level of conservation	Triplets of nucleotides	64x64 matrix of codon substitution frequencies	Codons, nucleotides, splice sites, start/stop codons	Vector of chromatin mark frequencies

What have we learned ?

- Generative model. Hidden states, observed emissions.
  - Generate a random sequence
    - Choose random transition, choose random emission (#0)
- Scoring: Finding the likelihood of a given sequence
  - Calculate likelihood of annotated path and sequence
    - Multiply emission and transition probabilities (#1)
  - Without specifying a path, total probability of generating x
    - Sum probabilities over all paths
    - Forward algorithm (#3)
- Decoding: Finding the most likely path, given a sequence
  - What is the most likely path generating entire sequence?
    - Viterbi algorithm (#2)
  - What is the most probable state at each time step?
    - Forward + backward algorithms, posterior decoding (#4)
- Learning: Estimating HMM parameters from training data
  - When state sequence is known
    - Simply compute maximum likelihood A and E (#5a)
  - When state sequence is not known
    - Viterbi training: Iterative estimation of best path / frequencies (#5b)
    - Baum-Welch: Iterative estimation over all paths / frequencies (#6)

Goals for today: HMMs, part II

1. Review: Basics and three algorithms from last time
   - Markov Chains and Hidden Markov Models
   - Calculating likelihoods $P(x,\pi)$ (algorithm 1)
   - Viterbi algorithm: Find $\pi^* = \text{argmax}_\pi P(x,\pi)$ (alg 3)
   - Forward algorithm: Find $P(x)$, over all paths (alg 2)
2. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM
3. Posterior decoding: Another way of 'parsing'
   - Find most likely state π_i, sum over all possible paths
4. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params

6.047 / 6.878 / HST.507 Computational Biology
Fall 2015