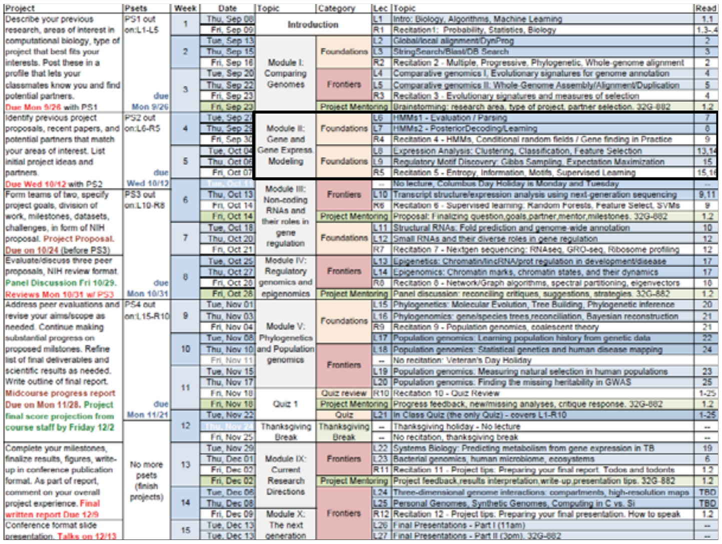
6.047/6.878/HST.507
Computational Biology: Genomes, Networks, Evolution

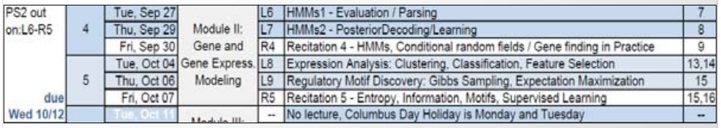
Lecture 07

Hidden Markov Models Part II



Project	Psets	Week	Date	Topic	Category	Lec	Topic	Read
Describe your previous research, areas of interest in computational biology, type of project that best fits your interests. Post these in a profile that lets your classmates know you and find potential partners. Due Mon 9/26 with PS1	PS1 out on:L1-L5 due Mon 9/26	1	Thu, Sep 08	Introduction		L1	Intro: Biology, Algorithms, Machine Learning	1.1
			Fri, Sep 09			R1	Recitation1: Probability, Statistics, Biology	1.3-.4
		2	Tue, Sep 13	Module I: Comparing Genomes	Foundations	L2	Global/local alignment/DynProg	2
			Thu, Sep 15			L3	StringSearch/Blast/DB Search	3
			Fri, Sep 16			R2	Recitation 2 - Multiple, Progressive, Phylogenetic, Whole-genome alignment	2
		3	Tue, Sep 20		Frontiers	L4	Comparative genomics I, Evolutionary signatures for genome annotation	4
			Thu, Sep 22			L5	Comparative genomics II: Whole-Genome Assembly/Alignment/Duplication	5
			Fri, Sep 23			R3	Recitation 3 - Evolutionary signatures and measures of selection	4
			Fri, Sep 23		Project Mentoring		Brainstorming: research area, type of project, partner selection. 32G-882	1,2
Identify previous project proposals, recent papers, and potential partners that match your areas of interest. List initial project ideas and partners. Due Wed 10/12 with PS2	PS2 out on:L6-R5 due Wed 10/12	4	Tue, Sep 27	Module II: Gene and Gene Express. Modeling	Foundations	L6	HMMs1 - Evaluation / Parsing	7
			Thu, Sep 29			L7	HMMs2 - PosteriorDecoding/Learning	8
			Fri, Sep 30			R4	Recitation 4 - HMMs, Conditional random fields / Gene finding in Practice	9
		5	Tue, Oct 04		Foundations	L8	Expression Analysis: Clustering, Classification, Feature Selection	13,14
			Thu, Oct 06			L9	Regulatory Motif Discovery: Gibbs Sampling, Expectation Maximization	15
			Fri, Oct 07			R5	Recitation 5 - Entropy, Information, Motifs, Supervised Learning	15,16
		6	Tue, Oct 11	Module III: Non-coding RNAs and their roles in gene regulation	Frontiers	--	No lecture, Columbus Day Holiday is Monday and Tuesday	--
Form teams of two, specify project goals, division of work, milestones, datasets, challenges, in form of NIH proposal. Project Proposal. Due on 10/24 (before PS3)	PS3 out on:L10-R8 due Mon 10/31		Thu, Oct 13			L10	Transcript structure/expression analysis using next-generation sequencing	9,11
			Fri, Oct 14			R6	Recitation 6 - Supervised learning: Random Forests, Feature Select, SVMs	9
			Fri, Oct 14		Project Mentoring		Proposal: Finalizing question,goals,partner,mentor,milestones. 32G-882	1,2
		7	Tue, Oct 18		Foundations	L11	Structural RNAs: Fold prediction and genome-wide annotation	10
			Thu, Oct 20			L12	Small RNAs and their diverse roles in gene regulation	12
			Fri, Oct 21			R7	Recitation 7 - Nextgen sequencing: RNAseq, GRO-seq, Ribosome profiling	12
Evaluate/discuss three peer proposals, NIH review format. Panel Discussion Fri 10/29. Reviews Mon 10/31 w/ PS3		8	Tue, Oct 25	Module IV: Regulatory genomics and epigenomics	Frontiers	L13	Epigenetics: Chromatin/lincRNA/prot regulation in development/disease	17
			Thu, Oct 27			L14	Epigenomics: Chromatin marks, chromatin states, and their dynamics	17
			Fri, Oct 28			R8	Recitation 8 - Network/Graph algorithms, spectral partitioning, eigenvectors	18
			Fri, Oct 28		Project Mentoring		Panel discussion: reconciling critiques, suggestions, strategies. 32G-882	1,2
Address peer evaluations and revise your aims/scope as needed. Continue making substantial progress on proposed milstones. Refine list of final deliverables and scientific results as needed. Write outline of final report. Midcourse progress report Due on Mon 11/28. Project final score projection from course staff by Friday 12/2	PS4 out on:L15-R10 due Mon 11/21	9	Tue, Nov 01	Module V: Phylogenetics and Population genomics	Foundations	L15	Phylogenetics: Molecular Evolution, Tree Building, Phylogenetic inference	20
			Thu, Nov 03			L16	Phylogenomics: gene/species trees,reconciliation, Bayesian reconstruction	21
			Fri, Nov 04			R9	Recitation 9 - Population genomics, coalescent theory	21
		10	Tue, Nov 08			L17	Population genomics: Learning population history from genetic data	22
			Thu, Nov 10		Frontiers	L18	Population genomics: Statistical genetics and human disease mapping	24
			Fri, Nov 11			--	No recitation: Veteran's Day Holiday	
		11	Tue, Nov 15			L19	Population genomics: Measuring natural selection in human populations	23
			Thu, Nov 17			L20	Population genomics: Finding the missing heritability in GWAS	25
			Fri, Nov 18	Quiz 1	Quiz review	R10	Recitation 10 - Quiz Review	1-25
			Fri, Nov 18		Project Mentoring		Progress feedback, new/missing analyses, critique response. 32G-882	1,2
		12	Tue, Nov 22		Quiz	L21	In Class Quiz (the only Quiz) - covers L1-R10	1-25
	No more psets (finish projects)		Thu, Nov 24	Thanksgiving Break	Thanksgiving Break	--	Thanksgiving holiday - No lecture	--
			Fri, Nov 25			--	No recitation, thanksgiving break	--
Complete your milestones, finalize results, figures, write-up in conference publication format. As part of report, comment on your overall project experience. Final written report Due 12/9		13	Tue, Nov 29	Module IX: Current Research Directions	Frontiers	L22	Systems Biology: Predicting metabolism from gene expression in TB	19
			Thu, Dec 01			L23	Bacterial genomics, human microbiome, ecosystems	6
			Fri, Dec 02			R11	Recitation 11 - Project tips: Preparing your final report. Todos and todonts	1,2
			Fri, Dec 02		Project Mentoring		Project feedback,results interpretation,write-up,presentation tips. 32G-882	1,2
		14	Tue, Dec 06		Frontiers	L24	Three-dimensional genome interactions: compartments, high-resolution maps	TBD
			Thu, Dec 08			L25	Personal Genomes, Synthetic Genomes, Computing in C vs. Si	TBD
			Fri, Dec 09	Module X: The next generation		R12	Recitation 12 - Project tips: Preparing your final presentation. How to speak	1,2
Conference format slide presentation. Talks on 12/13		15	Tue, Dec 13			L26	Final Presentations - Part I (11am)	--
			Tue, Dec 13			L27	Final Presentations - Part II (3pm). 32G-882	--

Module II: Modeling genes and gene expression



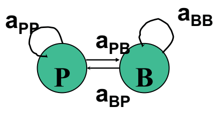
PS2 out on:L6-R5	4	Tue, Sep 27	Module II: Gene and Gene Express. Modeling	L6	HMMs1 - Evaluation / Parsing	7
		Thu, Sep 29		L7	HMMs2 - PosteriorDecoding/Learning	8
		Fri, Sep 30		R4	Recitation 4 - HMMs, Conditional random fields / Gene finding in Practice	9
	5	Tue, Oct 04		L8	Expression Analysis: Clustering, Classification, Feature Selection	13,14
		Thu, Oct 06		L9	Regulatory Motif Discovery: Gibbs Sampling, Expectation Maximization	15
due		Fri, Oct 07		R5	Recitation 5 - Entropy, Information, Motifs, Supervised Learning	15,16
Wed 10/12		Tue, Oct 11	Module III:	--	No lecture, Columbus Day Holiday is Monday and Tuesday	--

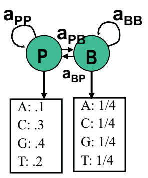
- Computational Foundations
  - Hidden Markov Models (HMMs): Central tool in CS
  - Decoding, evaluation, parsing, likelihood, scoring
  - Unsupervised Learning: Expectation Maximization
  - Supervised learning: generative/discriminative models
- Biological frontiers:
  - PS2: Modeling conservation, GC content, CpG islands
  - L6/L7: Genome annotation and parsing
  - L8: Gene expression analysis: cluster genes/conditions
  - L9: Regulatory motif discovery: EM, gibbs sampling, info

Goals for today: HMMs, part II

- Review: Three algorithms from last time
  - Markov Chains and Hidden Markov Models
  - Increasing the 'state' space / adding memory
  - Calculating likelihoods P(x,π)
  - Viterbi algorithm: Find π* = argmax_π P(x,π)
- Counting over all paths
  - Forward algorithm: Find P(x), over all paths
  - Model comparison: ex: "CpGs" vs. "Gs and Cs"
- Posterior decoding: Another way of 'parsing'
  - Find most likely state k_i, overall all possible paths
- Learning (ML training, Baum-Welch, Viterbi training)
  - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
  - Unsupervised: given only x → annotation + params

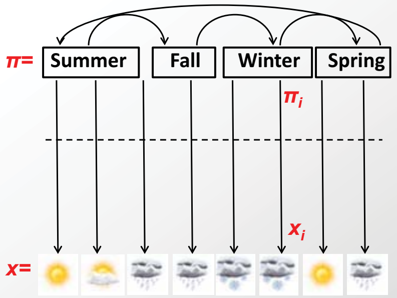
Markov Chains & Hidden Markov Models





- Markov Chain
  - Q: states
  - p: initial state probabilities
  - A: transition probabilities
- HMM
  - Q: states
  - V: observations
  - p: initial state probabilities
  - A: transition probabilities
  - E: emission probabilities

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*→*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 $P(\pi_i=k|x_i)$

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \Sigma_\pi P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_\pi P(x,\pi)$ Most likely path	4. Posterior decoding $\pi^\wedge = \{\pi_i \mid \pi_i=\text{argmax}_k \Sigma_\pi P(\pi_i=k \mid x)\}$ 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 $\Lambda^* = \text{argmax}_\Lambda \Sigma_\pi P(x,\pi \mid \Lambda)$ Baum-Welch training, over all paths

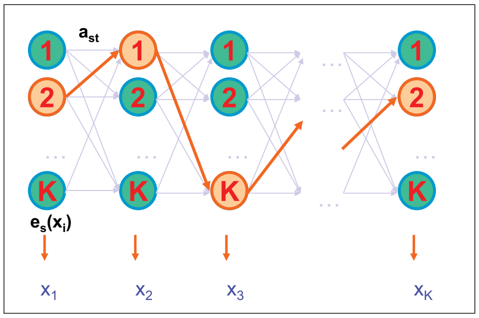
1. Scoring probability of a path + sequence

Multiply emissions, transitions

Probability of given path p, emissions x

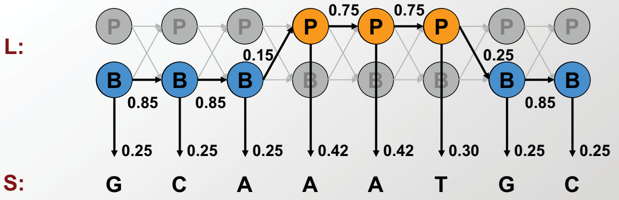
π is the (hidden) path

x is the (observed) sequence



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

Example: One particular P vs. B assignment



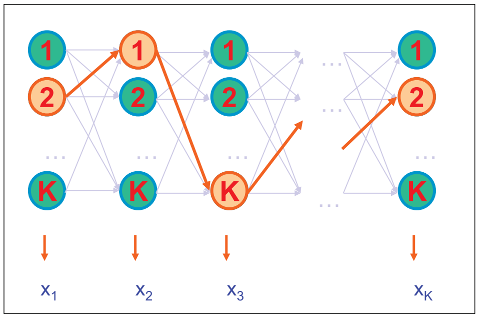
$$P = P(G \mid B)P(B_1 \mid B_0)P(C \mid B)P(B_2 \mid B_1)P(A \mid B)P(P_3 \mid B_2)...P(C \mid B_7)$$
$$= (0.85)^3 \times (0.25)^6 \times (0.75)^2 \times (0.42)^2 \times 0.30 \times 0.15$$
$$= 6.7 \times 10^{-7}$$

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \Sigma_\pi P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_\pi P(x,\pi)$ Most likely path	4. Posterior decoding $\pi^\wedge = \{\pi_i \mid \pi_i=\text{argmax}_k \Sigma_\pi P(\pi_i=k \mid x)\}$ 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 $\Lambda^* = \text{argmax}_\Lambda \Sigma_\pi P(x,\pi \mid \Lambda)$ Baum-Welch training, over all paths

3. Decoding: find the most likely path

Viterbi algorithm

Finding the most likely path



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

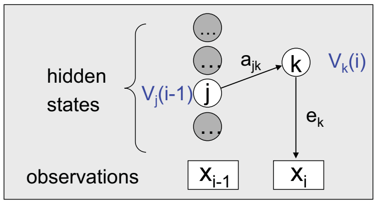
    start emission transition

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

current max

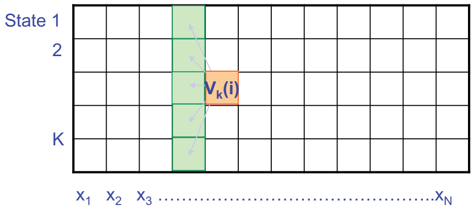
this emission

max ending in state j at step i

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: $O(K^2N)$

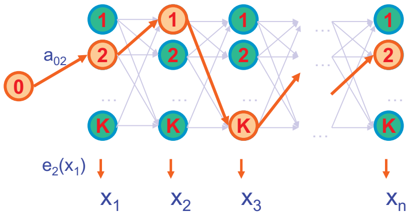
Space: $O(KN)$

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \Sigma_\pi P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_\pi P(x,\pi)$ Most likely path	4. Posterior decoding $\pi\hat{} = \{\pi_i \mid \pi_i=\text{argmax}_k \Sigma_\pi P(\pi_i=k \mid x)\}$ 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 $\Lambda^* = \text{argmax}_\Lambda \Sigma_\pi P(x,\pi \mid \Lambda)$ Baum-Welch training, over all paths

2. Model evaluation: Total P(x|M), summed over all paths

Forward algorithm

Simple: Given the model, generate some sequence x

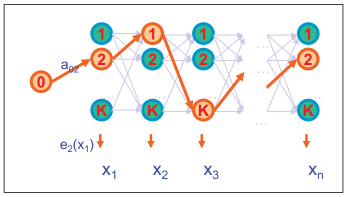


Given a HMM, we can generate a sequence of length n as follows:

1. Start at state π_1 according to prob $a_{0\pi 1}$
2. Emit letter x_1 according to prob $e_{\pi 1}(x_1)$
3. Go to state π_2 according to prob $a_{\pi 1 \pi 2}$
4. … until emitting x_n

We have some sequence x that can be emitted by p. Can calculate its likelihood.
However, in general, many different paths may emit this same sequence x.
How do we find the <u>total probability</u> of generating a given x, over any path?

Complex: Given x, was it generated by the model?



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

– $P(x) = \Sigma_\pi P(x,\pi)$

- Challenge: exponential number of paths
- (cheap) alternative:
  – Calculate probability over maximum (Viterbi) path π^*
- (real) solution
  – Calculate sum iteratively using <u>principles</u> of dynamic programming

The Forward Algorithm – derivation

Define the forward probability:

$$f_l(i) = P(x_1 \ldots x_i, \pi_i = l)$$

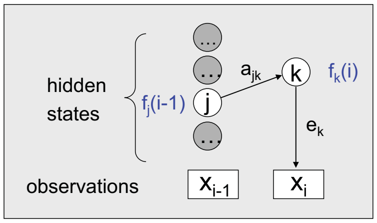
$$= \Sigma_{\pi 1 \ldots \pi i\text{-}1} P(x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-2}, \pi_{i-1}, \pi_i = l)\, e_l(x_i)$$

$$= \Sigma_k \boxed{\Sigma_{\pi 1 \ldots \pi i\text{-}2} P(x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-2}, \pi_{i-1} = k)}\, a_{kl}\, e_l(x_i)$$

$$= \Sigma_k \boxed{f_k(i-1)}\, a_{kl}\, e_l(x_i)$$

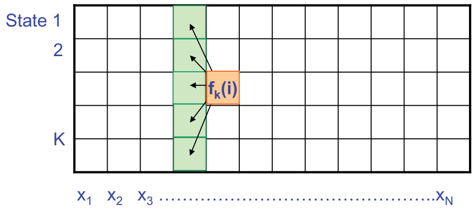
$$= e_l(x_i)\, \Sigma_k \boxed{f_k(i-1)}\, a_{kl}$$

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



- Assume we know f_j for the previous time step (i-1)
- Calculate $\underbrace{f_k(i)}_{\text{current max}} = \underbrace{e_k(x_i)}_{\text{this emission}} * \underbrace{\text{sum}_j \big(\underbrace{f_j(i\text{-}1)}_{\text{sum ending in state j at step i}} \times \underbrace{a_{jk}}_{\text{transition from state j}} \big)}_{\text{every possible previous state j}}$

The Forward Algorithm



Input: x = x1……xN

<u>Initialization:</u>
$f_0(0)=1$, $f_k(0) = 0$, for all k > 0

<u>Iteration:</u>
$f_k(i) = e_K(x_i) \times \text{sum}_j\, a_{jk}\, f_j(i\text{-}1)$

<u>Termination:</u>
$P(x, \pi^*) = \text{sum}_k\, f_k(N)$

<u>In practice:</u>
Sum of log scores is difficult
→ approximate exp(1+p+q)
→ scaling of probabilities

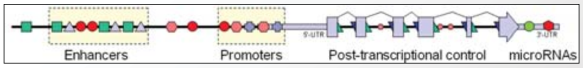
<u>Running time and space:</u>
Time: $O(K^2N)$
Space: $O(K)$

Application:
Distinguishing between two models

HMM1: Promoters = only Cs and Gs matter
HMM2: Promoters = it's actually CpGs that matter
("C"-phosphate-"G", i.e. on the same strand!)

(increasing the state space)

In the human genome, CpG islands matter!

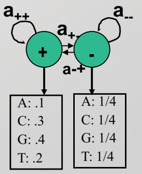


- Regions of regulatory importance in promoters of many genes
  – Defined by their methylation state (epigenetic information)
  – CpGs more important than simply the abundance of Cs and Gs
  – Provide evidence of methylation state!
- Methylation process in the human genome (form of silencing):
  – Methylation signature: high chance of methyl-C mutating to T in CpG
    ➔ CpG dinucleotides are rare, throughout the genome
  – BUT methylation is suppressed for active promoters
    ➔ CpG dinucleotides are much more frequent than elsewhere
    - Such regions are called **CpG islands**
    - A few hundred to a few thousand bases long
- Problems:
  – Given a short sequence, does it come from a CpG island or not?
  – How to find the CpG islands in a long sequence
- How do we encode this in an hidden Markov model?

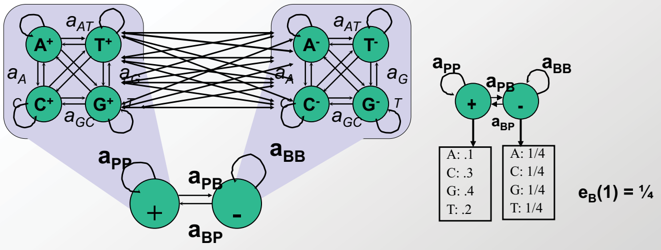
Increasing the state of the system (looking back)

- Markov Models are memory-less
  - In other words, all memory is encoded in the states
  - To remember additional information, augment state
- Our first HMM had minimal memory
  - State, emissions, only depend on **current** state
  - Current state only encoded **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



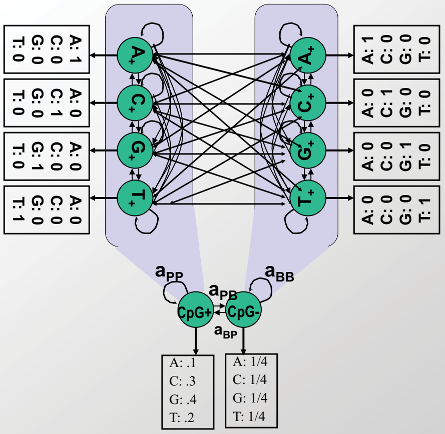
Modeling CpG islands: incorporating memory



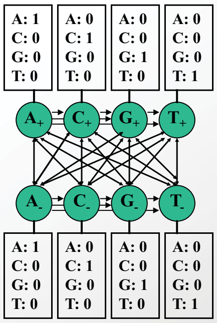
$e_B(1) = ¼$

- Markov Chain
  - Q: states
  - p: initial state probabilities
  - A: transition probabilities
- HMM
  - Q: states
  - V: observations
  - p: initial state probabilities
  - A: transition probabilities
  - E: emission probabilities

Example 2: CpG islands: incorporating memory



HMM for CpG islands



- Build a single model that combines two such Markov chains:
  - **'+' states**: A_+, C_+, G_+, T_+
    - Emit symbols: A, C, G, T in CpG islands
  - **'-' states**: A_-, C_-, G_-, T_-
    - Emit symbols: A, C, G, T in non-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

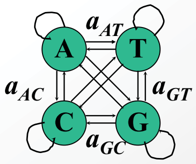
Why we need so many states…

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

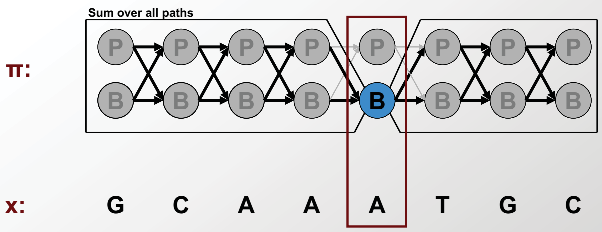
-	A	C	G	T
A	.300	.205	.285	.210
C	.322	.298	**.078**	.302
G	.248	.246	.298	.208
T	.177	.239	.292	.292

	One path	All paths
Scoring	1. Scoring x, one path P(x,π) ✓ Prob of a path, emissions	2. Scoring x, all paths ✓ $P(x) = \Sigma_{\pi} P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path ✓	4. Posterior decoding $\pi\hat{} = \{\pi_i \mid \pi_i=\text{argmax}_k \Sigma_{\pi} P(\pi_i=k\mid x)\}$ 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 $\Lambda^* = \text{argmax}_{\Lambda} \Sigma_{\pi} P(x,\pi\mid\Lambda)$ 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



P(Label$_i$=B|x)

- Calculate most probable label, L^*_i , at each position i
- Do this for all N positions gives us $\{L^*_1, L^*_2, L^*_3 \dots L^*_N\}$
- 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(π_7= CpG+ | x_7=G)

- With no knowledge (no characters)
  - Simply time spent in markov chain states
  - P(π_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(π_i=k | x_i=G) = P(π_i=κ) * P(x_i=G|π_i=k) / P(x_i=G)
- With knowledge of entire sequence (all characters)
  - P(π_i=k | x=AGCGCG…GATTATCGTCGTA)
  - Sum over all paths that emit ‘G’ at position 7
  - ➔ **Posterior** decoding

Motivation for the Backward Algorithm

We want to compute

P(π_i = k | x), the probability distribution on the ith position, given x

We start by computing

$$\begin{aligned} P(\pi_i = k, x) &= P(x_1\dots x_i, \pi_i = k, x_{i+1}\dots x_N) \\ &= P(x_1\dots x_i, \pi_i = k)\, P(x_{i+1}\dots x_N \mid x_1\dots x_i, \pi_i = k) \\ &= \boxed{P(x_1\dots x_i, \pi_i = k)}\, \boxed{P(x_{i+1}\dots x_N \mid \pi_i = k)} \end{aligned}$$

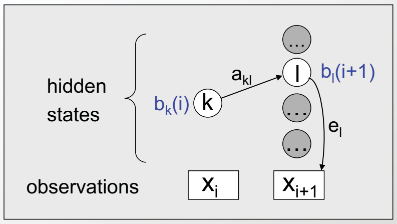
Forward, $f_k(i)$ Backward, $b_k(i)$

The Backward Algorithm – derivation

Define the backward probability:

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

Calculate total end probability recursively



- Assume we know b_l for the next time step (i+1)
- Calculate $b_k(i)$ = $\boxed{\text{sum}_l\,(\, e_l(x_{i+1}) \times a_{kl} \times b_l(i+1)\,)}$

current max

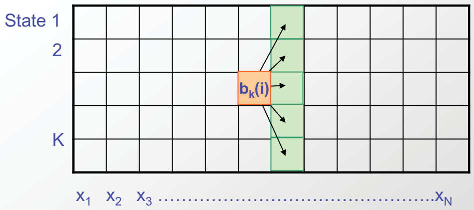
next emission

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)

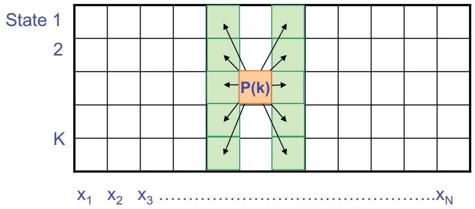
→ scaling of probabilities

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
  - Define most likely state for every of sequence x
  - $\pi\hat{}_i = \text{argmax}_k \; P(\pi_i = k \mid x)$
- Posterior decoding 'path' $\pi\hat{}_i$
  - For classification, more informative than Viterbi path $\pi*$
    - 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

Summary this far

- Generative model. Hidden states, observed emissions.
  - Generate a random sequence
    - Choose random transition, choose random emission (#0)
- Scoring the likelihood of a 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)
- Next: Learning (#5 and #6)

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions ✓	✓ 2. Scoring x, all paths $P(x) = \Sigma_\pi \; P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_\pi \; P(x,\pi)$ Most likely path ✓	✓ 4. Posterior decoding $\pi\hat{} = \{\pi_i \mid \pi_i=\text{argmax}_k \; \Sigma_\pi P(\pi_i=k\mid x)\}$ 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 $\Lambda^* = \text{argmax}_\Lambda \; \Sigma_\pi P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

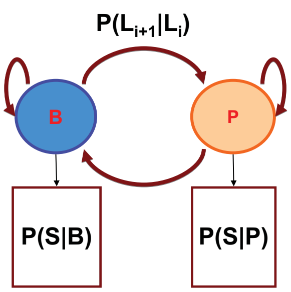
Learning: How to train an HMM

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:

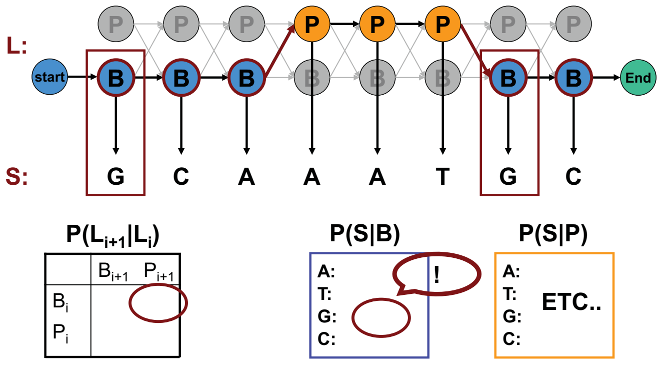
A_{kl} = # times k→l transition occurs in π
$E_k(b)$ = # times state k in π emits b in x

We can show that the maximum likelihood parameters θ are:

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

Learning From Labelled Data → Maximum Likelihood Estimation

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



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|θ) 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

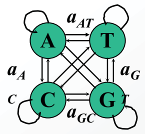
A_{kl} = # times k→l transition occurs in π + r_{kl}
$E_k(b)$ = # times state k in π emits b in x + $r_k(b)$

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

Larger pseudocounts ⇒ Strong priof belief

Small pseudocounts (ε < 1): just to avoid 0 probabilities

Example: Training Markov Chains for CpG islands



+	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

- Training Set:
  - set of DNA sequences w/ known CpG islands
- Derive two Markov chain models:
  - **‘+’ model**: from the CpG islands
  - **‘-’ model**: from the remainder of sequence
- Transition probabilities for each model:

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

c^+_{st} **is the number of times letter *t* followed letter *s* <u>inside</u> the CpG islands**

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

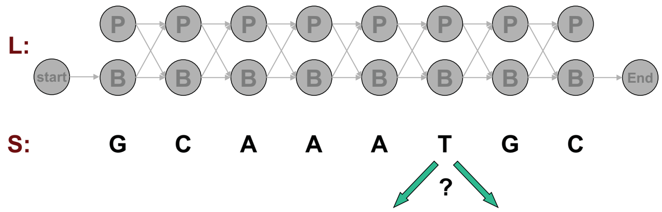
c^-_{st} **is the number of times letter *t* followed letter *s* <u>outside</u> the CpG islands**

6: Unsupervised learning

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

Unlabelled Data

How do we know how to count?



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

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

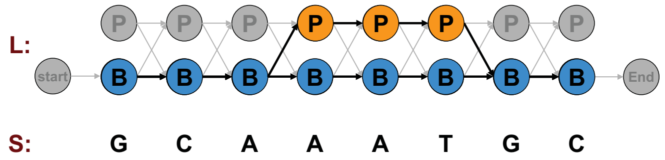
P(S|B)

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

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

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

$$params^t = \underset{params}{\arg\max}\, E_Q\left[\log P(S, labels \mid params^{t-1})\right]$$

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 $x = x_1 \dots x_N$
for which the true $\pi = \pi_1 \dots \pi_N$ is unknown,

We can get to a provably more likely parameter set θ

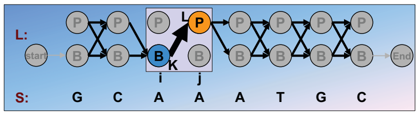
Principle: EXPECTATION MAXIMIZATION

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:

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

$$\begin{aligned} \text{where } Q &= P(x_1 \dots x_i, \pi_i = k, \pi_{i+1} = l, x_{i+1} \dots x_N) = \\ &= P(\pi_{i+1} = l, x_{i+1} \dots x_N \mid \pi_i = k)\, P(x_1 \dots x_i, \pi_i = k) = \\ &= P(\pi_{i+1} = l, x_{i+1} x_{i+2} \dots x_N \mid \pi_i = k)\, f_k(i) = \\ &= P(x_{i+2} \dots x_N \mid \pi_{i+1} = l)\, P(x_{i+1} \mid \pi_{i+1} = l)\, P(\pi_{i+1} = l \mid \pi_i = k)\, f_k(i) = \\ &= b_l(i+1)\, e_l(x_{i+1})\, a_{kl}\, f_k(i) \end{aligned}$$

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,

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

Similarly,

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

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,

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

Similarly,

$$E_k(\boxed{b}) = \Sigma_x (1/P(x))\Sigma_{\{i \mid x^i = \boxed{b}\}} \boxed{f_k(i)}\,\boxed{b_k(i)}$$

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 × $O(K^2N)$

- 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 $P(x) = \Sigma_\pi P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_\pi P(x,\pi)$ Most likely path	4. Posterior decoding $\pi^\wedge = \{\pi_i \mid \pi_i=\text{argmax}_k \Sigma_\pi P(\pi_i=k \mid x)\}$ 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 $\Lambda^* = \text{argmax}_\Lambda \Sigma_\pi P(x,\pi \mid \Lambda)$ Baum-Welch training, over all paths

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)

The main questions on HMMs

SCORING

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)

PARSING

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

LEARNING

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