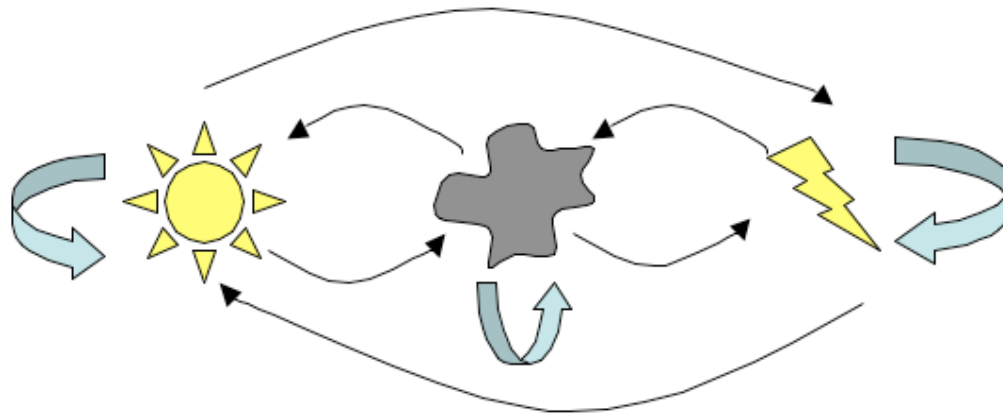


Baum-Welch and HMM applications

Analysis of Biological Sequences 140.638

Markov chains

- 3 states of weather: sunny, cloudy, rainy
 - Observed once a day at the same time



- All transitions are possible, with some probability
- Each state depends only on the previous state

Hidden Markov Models

Weather today

		Sunny	Cloudy	Rainy
Weather yesterday	Sunny	0.50	0.20	0.30
	Cloudy	0.10	0.60	0.30
	Rainy	0.20	0.40	0.40

Dog

in out porch

All we observe is the dog:

I O O O I P I I I I O O O O O P P I I I I I P I

Sun	0.2	0.7	0.1
Cloud	0.4	0.4	0.2
Rain	0.7	0.1	0.2

Hidden Markov Models: the three questions

Evaluation

Given: a HMM, M , and a sequence x

Find: $P(x|M)$

Forward, Backward and Forward-Backward algorithms

Decoding

Given: a HMM, M , and a sequence x

Find: the sequence Q of states that maximizes $P(x,Q|M)$

Viterbi algorithm

Learning

Given: an unknown HMM, M , and a sequence x

Find: parameters θ that maximize $P(x|\theta, M)$

Baum-Welch expectation maximization (EM) algorithm

Baum-Welch expectation maximization algorithm

You have: observed data

You want: parameters of the HMM that generated that data

Problem: the calculation space is too big for exact calculation -> use heuristic method (even though it's a partial solution it's very useful!)

We are finding locally optimal parameters.

Baum-Welch expectation maximization algorithm

Assume: data come from some random process that we can fit to a HMM

Assumption #1: alphabet A and the number of states, N , are fixed. Transition, emission and initial distribution probabilities are all unknown.

Baum-Welch expectation maximization algorithm

Assumption #2: data are a set of observed sequences $\{x^{(d)}\}$ each of which has a hidden state sequence Q^d

Assumption #3: we can set all parameters/probabilities to some initial values

- Can choose from some uniform distribution
- Can choose to incorporate some prior knowledge
- Can just be random

Baum-Welch expectation maximization algorithm

Remember:

$$\alpha(t, i) = P(x_1, x_2, \dots, x_t \mid q_t = S_i)$$

$$\beta(t, i) = P(x_T, x_{T-1}, \dots, x_t \mid q_t = S_i)$$

$$P(x|M) = \sum_{i=0}^N (\alpha(T, i)) = \sum_{i=0}^N (\beta(1, i))$$

Baum-Welch expectation maximization algorithm

$$\alpha(t, i) = P(x_1, x_2, \dots, x_t \mid q_t = S_i)$$

$$\beta(t, i) = P(x_T, x_{T-1}, \dots, x_t \mid q_t = S_i)$$

$$P(x|M) = \sum_{i=0}^N (\alpha(T, i)) = \sum_{i=0}^N (\beta(1, i))$$

$$P(q_t = S_i, q_{t+1} = S_\ell \mid x, \theta) = [\alpha(t, i) p_{i\ell} b_\ell(x_{t+1}) \beta(t+1, \ell)] / P(x)$$

$$p'_{i\ell} = \sum_d 1/P(x^d) \sum_t \alpha^d(t, i) p_{i\ell} b_\ell(x^d_{t+1}) \beta^d(t+1, \ell)$$

$$b'_\ell(c) = \sum_d 1/P(x^d) \sum_{t \mid x^d_t = c} \alpha^d(t, \ell) \beta^d(t, \ell)$$

Baum-Welch expectation maximization algorithm

$$P(q_t=S_i, q_{t+1}=S_\ell \mid \mathbf{x}, \theta) = [\alpha(t, i) p_{i\ell} b_\ell(x_{t+1}) \beta(t+1, \ell)] / P(\mathbf{x})$$

$x_1 x_2 x_3 \dots x_t x_{t+1} \dots x_{T-1} x_T$

$q_1 q_2 q_3 \dots q_t q_{t+1} \dots q_{T-1} q_T$

$\dots S_i S_\ell \dots$

Baum-Welch expectation maximization algorithm

$$P(q_t=S_i, q_{t+1}=S_\ell | \mathbf{x}, \theta) = [\alpha(t, i)p_{i\ell}b_\ell(x_{t+1})\beta(t+1, \ell)]/P(\mathbf{x})$$

$$p'_{i\ell} = \sum_d 1/P(\mathbf{x}^d) \sum_t \alpha^d(t, i)p_{i\ell}b_\ell(x_{t+1}^d)\beta^d(t+1, \ell)$$

Figure out the probability of a hidden state $i \rightarrow \ell$ transition

- 1) postulate that transition at every single spot in every single observed sequence (separately)
- 2) see how those probabilities compare to the best probabilities for those observed sequences
- 3) use that ratio for the updated $p_{i\ell}$ transition probability

Baum-Welch expectation maximization algorithm

$$P(q_t=S_i, q_{t+1}=S_\ell | x, \theta) = [\alpha(t, i)p_{i\ell}b_\ell(x_{t+1})\beta(t+1, \ell)]/P(x)$$

$$b'_\ell(c) = \sum_d 1/P(x^d) \sum_{t|x_t^d=c} \alpha^d(t, \ell)\beta^d(t, \ell)$$

Figure out the probability of an emission of symbol c from hidden state ℓ

- 1) postulate that hidden state under every symbol c in every single observed sequence (separately)
- 2) see how those probabilities compare to the best probabilities for those observed sequences
- 3) use that ratio for the updated $b'_\ell(c)$ transition probability

Baum-Welch expectation maximization algorithm

Then recalculate $P(x^d|M, \theta)$ for all observed data in the learning set (use Forward, Backward, or Forward/Backward to do this)

Rinse & repeat . . .

Successive iterations increase $P(\text{data})$ and we stop when the probability stops increasing significantly (usually measured as log-likelihood ratios).

Baum-Welch example

- I observe dog #2 at noon every day. Sometimes he's inside, sometimes he's outside.
- I guess that since he can't open the door by himself (yet) that there is another factor, hidden from me, that determines his behavior
- Since I am lazy I will guess that there are only two hidden states

Baum-Welch example

- guessing two hidden states. I need to invent a transition matrix and an emission matrix.

today

	S1	S2	
yesterday	S1	0.5	0.5
	S2	0.4	0.6

	in	out
S1	0.2	0.8
S2	0.9	0.1

initial: $p(S1) = 0.3$, $p(S2) = 0.7$

Baum-Welch example

observations: II, II, II, II, IO, OO, OI, II,
II

today

yesterday

	S1	S2
S1	0.5	0.5
S2	0.4	0.6

	in	out
S1	0.2	0.8
S2	0.9	0.1

initial: $p(S1) = 0.3$, $p(S2) = 0.7$

Baum-Welch example

guess: if II came from S1•S2 the probability is
 $0.3 * 0.2 * 0.5 * 0.9 = 0.027$

today

		S1	S2
yesterday	S1	0.5	0.5
S2	0.4	0.6	

	in	out
S1	0.2	0.8
S2	0.9	0.1

initial: $p(S1) = 0.3$, $p(S2) = 0.7$

Baum-Welch example

estimating the transition matrix:

Seq	P(Seq) if S1·S2	Best P(seq)
II	0.027	0.3403 S2·S2
II	0.027	0.3403 S2·S2
II	0.027	0.3403 S2·S2
II	0.027	0.3403 S2·S2
IO	0.003	0.2016 S2·S1
OO	0.012	0.096 S1·S1
OI	0.108	0.108 S1·S2
II	0.027	0.3403 S2·S2
II	0.027	0.3403 S2·S2
Total	0.285	2.4474

Our estimate for the S1->S2 transition probability is now $0.285/2.4474 = 0.116$. Calculate the S2->S1, S2->S2, S1->S1 as well and normalize so they add up to 1 as needed, to update the transition matrix.

Baum-Welch example

estimating the emission matrix:

Seq	Best P(Seq) if O came from S1	Best P(seq)
IO	0.2016 (S2·S1)	0.2016 (S2·S1)
OO	0.096 (S1·S1)	0.096 (S1·S1)
OI	0.108 (S1·S2)	0.108 (S1·S2)

adding both columns, it looks like $p(\text{S1 emits O})$ is 1?

Baum-Welch example

estimating initial probabilities:

- 1) assume all sequences start with hidden state S1, calculate best probability
- 2) assume all sequences start with hidden state S2, calculate best probability
- 3) normalize to 1

Baum-Welch example

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

If you have guessed the wrong number of hidden states, it will be clear, though it's a very bad strategy to go through a huge range of possible hidden states to find the best model – you will over-optimize.

Applications of HMMs

- Exon finding through orthology (Haussler)
- ECG signal analysis (beat segmentation and classification)
- Analysis of microarray data especially tiling arrays
- Sequence feature prediction using homology information
- Sequence alignments, pairwise and multiple
- Analyzing ChIP-chip on tiling arrays

Finding genes

- The first gene finders were for prokaryotes
 - No introns
 - Distinct and known signals
- GLIMMER (1998, Salzberg et al.) was an early gene-finding program and was very successful
 - Only for prokaryotes (first version)
 - Tested on relatively short sequences

GLIMMER

- Uses Interpolated Markov Models (IMMs)
 - Instead of confining the algorithm to predetermined oligomer lengths, many lengths are sampled
 - First-order HMM: look at the closest previous emission
 - Second-order HMM: look at the two most recent emissions
 - etc.
- Uses 6 submodels for each reading frame plus one for noncoding sequence

GLIMMER

- Higher-order Markov models do better than lower-order models
 - For example: if 3rd codon position depends on 1st and 2nd codon positions a 2nd-order Markov Model will do much better than a 1st-order model
- BUT, for higher-order models there may not be enough data
 - For example: 6th, 7th, 8th order models require larger oligomers and some of those oligomers might not even be present in the learning set!

GLIMMER

- Solution: use an IMM -> uses a combination of all probabilities based on 0, 1, 2, . . . k bases (k is a parameter, k=8 in GLIMMER)
- Use predictions from lower-order models to adjust higher-order models (since more data is available for lower-order models)

GLIMMER system

- 2 programs
 - build_imm — takes input sequence and creates IMMs
 - Glimmer — uses IMM to identify putative genes
 - Identifies all ORFs over a certain threshold
 - Scores all 6 reading frames and compares to higher threshold
 - Looks for overlaps; eliminate lower-scoring ORF of the two that overlap
- Found >97% of the genes in *H. influenza*, de novo.

And along came eukaryotes . . .

- With so much eukaryotic sequence suddenly available, questions arose about gene-finding
 - By orthology
 - de novo
 - Experimentally
- What's different about eukaryotic genes? What changes about how we need to go looking for genes?

GENSCAN (1997)

- GENSCAN (Burge and Karlin) was a huge breakthrough in eukaryotic gene-finding, and is still used by some groups
- How is it different?
 - Assumes that the input sequence can have no genes, one gene, multiple genes, or parts of genes
 - Models all known aspects of a eukaryotic gene
 - Uses general 3-periodic inhomogeneous fifth-order Markov model of coding regions
 - Does not use specific models of protein structure or database homology

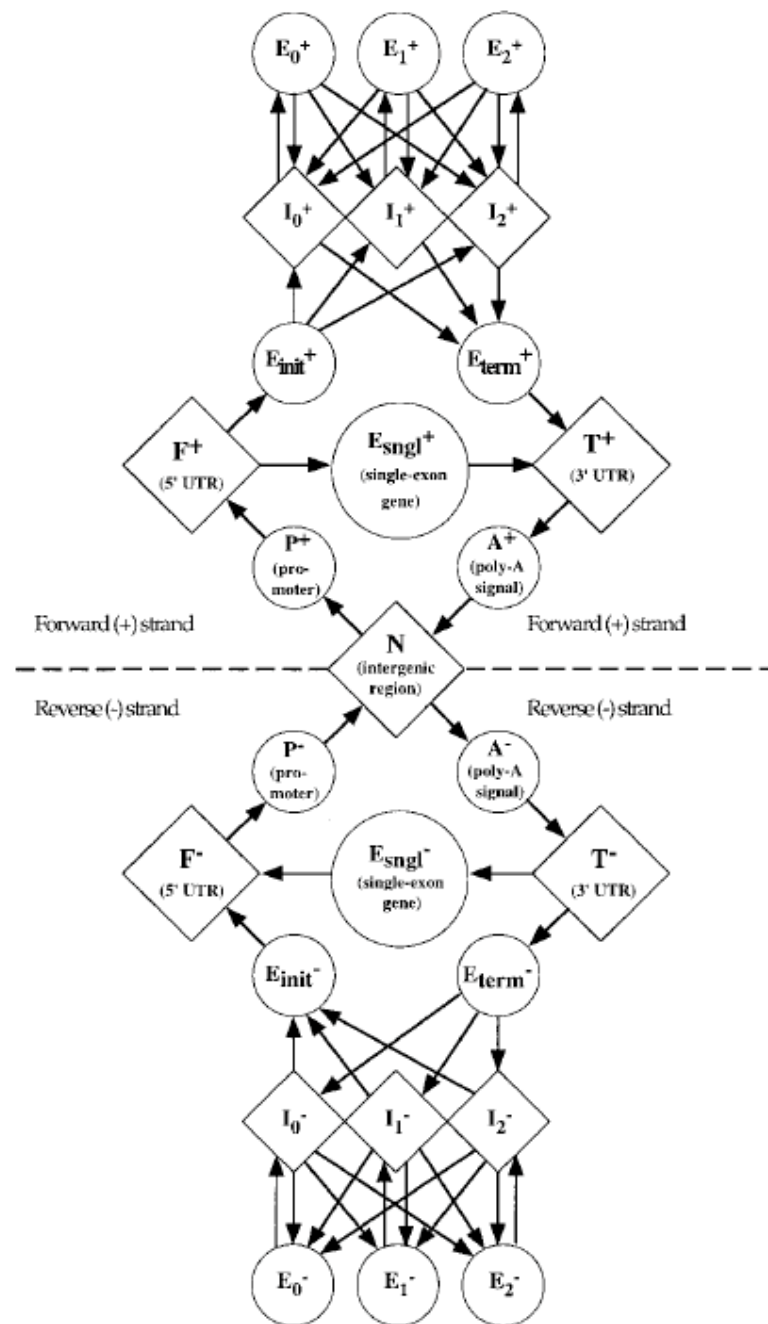
GENSCAN

- Uses double-stranded genomic sequence
- Partial genes, single gene, no genes all modeled
- Maximal Dependence Decomposition: models functional signals in DNA or protein to allow for statistical dependencies between signal positions
 - e.g. splice sites

exon | GT . . intron . . AG | . . . exon

GENSCAN

- Explicit state duration HMM
 - Uses forward-backward algorithm
 - Choose initial state depending on the distribution of all possible states



GENSCAN — MDD

- Maximal Dependence Decomposition
 - Need aligned set of several hundred signal sequences
 - Use conditional probabilities to capture the most significant dependencies between positions
 - Calculate χ^2 for each pair of positions to detect dependencies

Next generation

- Three types of de novo predictors
 - Single genome sequence (mostly HMMs)
 - Two aligned genomes
 - Multiple aligned genomes } infer local rates & patterns of mutation
- With good programs can expect 50-70% of the genes correctly predicted, in a compact genome

Next generation

- Single-genome predictors
 - Easier to train, faster to run
 - First step in annotation
 - Use information from intrinsic sequence signals

Next generation

- Dual-genome predictors
 - Rely on functional regions being more conserved
 - SLAM (HMM) - uses joint probability for sequence alignment and gene structure to define types of alignments seen in coding vs noncoding sequence
 - More powerful approaches use HMM and dynamic programming
 - Problem: in closely related species most of the conserved sequences are noncoding

Next generation

- Multi-genome predictors
 - More genomes -> stronger evidence
 - Hard to get enough species for a good alignment (translocations, deletions, inversions etc destroy alignments)
 - Some use phylogenetic trees (phylo-HMMs)

Complete Composition Vector

- extremely closely related organisms can be defined by slight changes in Markov-type transitions:

AAACCT vs AAACCA

AAACC -> AACCT

vs

AAACC -> AACCA

Complete Composition Vector

Expected probability of string $\alpha_1\alpha_2\dots\alpha_k$ through a Markov Model:

$$p^e(\alpha_1\alpha_2\dots\alpha_k) = \frac{p(\alpha_1\alpha_2\dots\alpha_{k-1}) \times p(\alpha_2\alpha_3\dots\alpha_k)}{p(\alpha_2\alpha_3\dots\alpha_{k-1})}$$

if $p(\alpha_2\alpha_3\dots\alpha_{k-1}) \neq 0$

$p(\alpha_1\alpha_2 \text{ etc})$ comes from nucleotide
frequency calculations

Complete Composition Vector

$$d(\alpha_1\alpha_2\dots\alpha_k) = \frac{p(\alpha_1\alpha_2\dots\alpha_k) - p^e(\alpha_1\alpha_2\dots\alpha_k)}{p^e(\alpha_1\alpha_2\dots\alpha_k)}$$

if $p^e(\alpha_2\alpha_3\dots\alpha_{k-1}) \neq 0$

do this for every genome/genome fragment to get a distance d for each (d = “evolutionary information,” more accurately)

Distance between two genomes is then $\sqrt{(d_1 - d_2)^2}$ and that can be used to build a tree.

Complete Composition Vector

- CCV method has been used successfully to classify avian flu and HIV swarms
- recent paper on fungal phylogeny