So, you find a new piece of DNA...

What do you do?

...GTACTACCGGGTTACAGGATTATGGGTTACAGGTAACCGTT...

- Align it to things we know about
- Align it to things we don't know about
- Stare at it
  - Non-standard nucleotide composition?
  - Interesting k-mer frequencies?
  - Recurring patterns?
- Model it
  - Make some hypotheses about it
  - Build a 'generative model' to describe it
  - Find sequences of similar type

This week: Modeling biological sequences
(a.k.a. What to do with a huge chunk of DNA)

- Ability to emit DNA sequences of a certain type
  - Not exact alignment to previously known gene
  - Preserving 'properties' of type, not identical sequence
- Ability to recognize DNA sequences of a certain type (state)
  - What (hidden) state is most likely to have generated observations
  - Find set of states and transitions that generated a long sequence
- Ability to learn distinguishing characteristics of each state
  - Training our generative models on large datasets
  - Learn to classify unlabelled data
Why Probabilistic Sequence Modeling?

- Biological data is noisy
- Probability provides a calculus for manipulating models
- Not limited to yes/no answers – can provide "degrees of belief"
- Many common computational tools based on probabilistic models
  - Our tools:
    - Markov Chains and Hidden Markov Models (HMMs)

Definition: Markov Chain

A Markov chain is a triplet \((Q, p, A)\), where:

- \(Q\) is a finite set of states. Each state corresponds to a symbol in the alphabet \(\Sigma\)
- \(p\) is the initial state probabilities.
- \(A\) is the state transition probabilities, denoted by \(a_{st}\) for each \(s, t\) in \(Q\).

For each \(s, t\) in \(Q\) the transition probability is:

\[ a_{st} = P(x_t = t | x_{t-1} = s) \]

Property: The probability of each symbol \(x_t\) depends only on the value of the preceding symbol \(x_{t-1}\):

\[ P(x_t | x_{t-1}, \ldots, x_1) = P(x_t | x_{t-1}) \]

Formula: The probability of the sequence:

\[ P(x) = P(x_L, x_{L-1}, \ldots, x_1) = P(x_L | x_{L-1}) P(x_{L-1} | x_{L-2}) \ldots P(x_2 | x_1) P(x_1) \]

Output: The output of the model is the set of states at each instant time => the set of states are observable

Definitions: HMM (Hidden Markov Model)

A HMM is a 5-tuple \((Q, V, p, A, E)\), where:

- \(Q\) is a finite set of states, \(|Q| = N\)
- \(V\) is a finite set of observation symbols per state, \(|V| = M\)
- \(p\) is the initial state probabilities.
- \(A\) is the state transition probabilities, denoted by \(a_{st}\) for each \(s, t\) in \(Q\).

For each \(s, t\) in \(Q\) the transition probability is:

\[ a_{st} = P(x_t = t | x_{t-1} = s) \]

- \(E\) is a probability emission matrix, \(e_{sk} = P(v_t | q_t = s)\)

Property: Emissions and transitions are dependent on the current state only and not on the past.

The six algorithmic settings for HMMs

<table>
<thead>
<tr>
<th>One path</th>
<th>All paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Scoring (x), one path</td>
<td></td>
</tr>
<tr>
<td>(P(x, \pi))</td>
<td>(P(x) = \sum_{\pi} P(x, \pi))</td>
</tr>
<tr>
<td>Prob of a path, emissions</td>
<td></td>
</tr>
<tr>
<td>2. Scoring (x), all paths</td>
<td></td>
</tr>
<tr>
<td>3. Viterbi decoding</td>
<td></td>
</tr>
<tr>
<td>(x^* = \arg\max P(x, \pi))</td>
<td>Most likely path</td>
</tr>
<tr>
<td>4. Posterior decoding</td>
<td></td>
</tr>
<tr>
<td>(x^* = {\pi_i</td>
<td>\pi_i = \arg\max_n \sum_{i} P(x_k</td>
</tr>
<tr>
<td>5. Supervised learning, given (\Lambda)</td>
<td></td>
</tr>
<tr>
<td>(\Lambda^* = \arg\max \max \sum_{\pi} P(x, \pi</td>
<td>\Lambda))</td>
</tr>
<tr>
<td>6. Unsupervised learning</td>
<td></td>
</tr>
<tr>
<td>(\Lambda^* = \arg\max \sum_{\pi} P(x, \pi</td>
<td>\Lambda))</td>
</tr>
<tr>
<td>Viterbi training, best path</td>
<td></td>
</tr>
<tr>
<td>Baum-Welch training, over all paths</td>
<td></td>
</tr>
</tbody>
</table>

Example 1: Finding GC-rich regions

- Promoter regions frequently have higher counts of Gs and Cs
- Model genome as nucleotides drawn independently from two distributions: Background (B) and Promoters (P).
- Emission probabilities based on nucleotide composition in each.
- Transition probabilities based on relative abundance & avg. length

HMM as a Generative Model

\[ P(S | \pi) \]

\[ P(S | \pi) \]

\[ P(S | \pi) \]
Sequence Classification

PROBLEM: Given a sequence, is it a promoter region?
- We can calculate \( P(S | MP) \), but what is a sufficient \( P \) value?

SOLUTION: compare to a null model and calculate log-likelihood ratio
- e.g. background DNA distribution model, \( B \)

\[
\text{Score} = \log \frac{P(S | MP)}{P(S | B)}
\]

Finding GC-rich regions

- Could use the log-likelihood ratio on windows of fixed size
- Downside: have to evaluate all islands of all lengths repeatedly
- Need: a way to easily find transitions
3. DECODING: What was the sequence of hidden states?

Given: Model parameters $e_i$, $a_{ij}$
Given: Sequence of emissions $x$
Find: Sequence of hidden states $\pi$

Finding the optimal path

- We can now evaluate any path through hidden states, given the emitted sequences
- How do we find the best path?
- Optimal substructure! Best path through a given state is:
  - Best path to previous state
  - Best transition from previous state to this state
  - Best path to the end state

$\text{Viterbi algorithm}$
- Define $V_i(j) = \text{Probability of the most likely path through state } \pi_i = j$
- Compute $V_i(j+1) = e_i(x_{i+1}) \times \max_j a_{jk} V_j(i)$

$\text{Dynamic Programming}$

Finding the most likely path

- $\pi^*$ that maximizes total joint probability $P(x, \pi)$
  - $P(x, \pi) = a_0 \prod_i e_i(x_i) \times a_{\pi_i \pi_{i+1}}$

The Viterbi Algorithm

<table>
<thead>
<tr>
<th>State 1</th>
<th>2</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
</tr>
</tbody>
</table>

Input: $x = x_1 \ldots x_N$

Initialization:
$V_0(0) = 1$, $V_i(0) = 0$, for all $k > 0$

Iteration:
$V_i(j) = e_i(x_j) \times \max_k a_{kj} V_{i-1}(k)$

Termination:
$P(x, \pi^*) = \max_k V_N(k)$

Traceback:
Follow max pointers back
Similar to aligning states to seq
In practice:
Use log scores for computation

Running time and space:
Time: $O(KN)$
Space: $O(KN)$

The six algorithmic settings for HMMs

<table>
<thead>
<tr>
<th>One path</th>
<th>All paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Scoring $x$, one path $P(x, \pi)$</td>
<td>2. Scoring $x$, all paths $P(x) = \sum_\pi P(x, \pi)$</td>
</tr>
<tr>
<td>3. Viterbi decoding $\pi^* = \arg\max_\pi P(x, \pi)$</td>
<td>4. Posterior decoding $\pi^* = { \pi_i</td>
</tr>
</tbody>
</table>

Scoring

Prob of a path, emissions $P(x, \pi)$

Decoding

3. Viterbi decoding $\pi^* = \arg\max_\pi P(x, \pi)$

4. Posterior decoding $\pi^* = \{ \pi_i | \pi_1, \ldots, \pi_N, \arg\max_\pi \sum_\pi P(x | \pi_1) \}$

5. Supervised learning, given $\pi^*$ $\Lambda^* = \arg\max_\Lambda P(x, \pi | \Lambda)$

6. Unsupervised learning $\Lambda^* = \arg\max_\Lambda \sum_\pi P(x, \pi | \Lambda)$

Learning

5. Supervised learning, given $\pi^*$

6. Unsupervised learning

Viterbi training, best path

Baum-Welch training, over all paths
2. EVALUATION
(how well does our model capture the world)
Given: Model parameters $e_i, a_{ij}$
Given: Sequence of emissions $x$
Find: $P(x|M)$, summed over all possible paths $\pi$

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 $\pi_1$ according to prob $a_0$.
  2. Emit letter $x_1$ according to prob $e_{\pi_1}(x_1)$.
  3. Go to state $\pi_2$ 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 total probability 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)?
    - $P(x) = \sum_\pi P(x, \pi)$
  - Challenge: exponential number of paths

The Forward Algorithm – derivation
Define the forward probability:

\[
f(i) = P(x_1, \ldots, x_i, \pi = l) = \sum_{\pi_1, \ldots, \pi_{i-1}} P(x_1, \ldots, x_i, \pi_1, \ldots, \pi_{i-1}, \pi = l) e_{\pi_i}(x_i)
\]

\[
= \sum_{\pi_1, \ldots, \pi_{i-1}} P(x_1, \ldots, x_i, \pi_1, \ldots, \pi_{i-1}) a_{\pi_{i-1}\pi_i} f(j)
\]

\[
= \sum_{\pi_1, \ldots, \pi_{i-1}} \sum_j f(j) a_{\pi_{i-1}\pi_i} e_{\pi_i}(x_i)
\]

\[
= e_{\pi_i}(x_i) \sum_{\pi_{i-1}} f(j) a_{\pi_{i-1}\pi_i}
\]

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

- Assume we know $f$ for the previous time step $i-1$

- Calculate $f(i) = \sum_{\pi} e_{\pi}(x_i) \sum_{\pi_{i-1}} f(i-1) a_{\pi_{i-1}\pi}$

Calculate probability of emission over all paths

- Each path has associated probability
  - Some paths are likely, others unlikely: sum them all up
  - Return total probability that emissions are observed, summed over all paths
  - Viterbi path is the most likely one
    - How much ‘probability mass’ does it contain?
- (cheap) alternative:
  - Calculate probability over maximum (Viterbi) path $\pi^*$
  - Good approximation if Viterbi has highest density
  - BUT: incorrect
- (real) solution
  - Calculate the exact sum iteratively
    - $P(x) = \sum_\pi P(x, \pi)$
  - Can use dynamic programming
### The Forward Algorithm

**Input:** $x = x_1 \ldots x_N$

**Initialization:**
- $f_0(0) = 1$, $f_k(0) = 0$, for all $k > 0$

**Iteration:**
- $f_k(i) = e_k(x_i) \sum_j a_{jk} f_{k-1}(i-1)$

**Termination:**
- $P(x, \pi^*) = \sum_k f_k(N)$

*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: $O(KN)$

---

### The six algorithmic settings for HMMs

<table>
<thead>
<tr>
<th>One path</th>
<th>All paths</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scoring</strong></td>
<td><strong>Prob of a path, emissions</strong></td>
</tr>
<tr>
<td>$P(x, \pi)$</td>
<td>$P(x) = \sum_{\pi} P(x, \pi)$</td>
</tr>
<tr>
<td><strong>Viterbi decoding</strong></td>
<td><strong>Most likely path</strong></td>
</tr>
<tr>
<td>$x^* = \arg\max \ P(x, \pi)$</td>
<td>Path containing the most likely state at any time point</td>
</tr>
<tr>
<td><strong>Posterior decoding</strong></td>
<td><strong>Unsupervised learning</strong></td>
</tr>
<tr>
<td>$\pi^* = { \pi_i</td>
<td>\pi_i = \arg\max_k \sum_{\pi} P(\pi_i=k</td>
</tr>
</tbody>
</table>

### Introducing memory
- State, emissions, only depend on current state
- How do you count di-nucleotide frequencies?
  - CpG islands
  - Codon triplets
  - Di-codon frequencies
- Introducing memory to the system
  - Expanding the number of states

### Example 2: CpG islands: incorporating memory

- **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

### Counting nucleotide transitions: Markov/HMM
- **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

### What have we learned?
- Modeling sequential data
  - Recognize a type of sequence, genomic, oral, verbal, visual, etc…
- Definitions
  - Markov Chains
  - Hidden Markov Models (HMMs)
- Simple examples
  - Recognizing GC-rich regions.
  - Recognizing CpG dinucleotides
- Our first computations
  - Running the model: know model $\Rightarrow$ generate sequence of a 'type'
  - Evaluation: know model, emissions, states $\Rightarrow$ $p$?
  - Viterbi: know model, emissions $\Rightarrow$ find optimal path
- Next time:
  - Posterior decoding
  - Supervised learning
  - Unsupervised learning: Baum-Welch, Viterbi training