Lecture 1

1.1 Preliminaries on probabilistic models of biological sequences

1.2 Hidden Markov Models for sequence families

Probabilistic models

- Probabilistic model: abstract 'system' that produces different outcomes (objects) with different probabilities; the model assigns each object x an associated probability P(x)
- A model typically has (several) parameters (real numbers); we denote all parameters by Θ
- Probabilistic models $\leftrightarrow$ probability distributions of the object family

- Example: rolling a die
  - Parameters $\Theta = (p_1,\ldots,p_6)$
  - Probability of rolling i: $P(i) = p_i$
  - Unloaded (fair) die: $p_1 = \ldots = p_6 = 1/6$
  - Independence of consecutive rolls: $P(1,6,3) = p_1 p_6 p_3$

Random sequence model (Bernoulli model)

- Alphabet Σ of symbols
  - DNA alphabet (bases): A, C, G, T
  - RNA alphabet (bases): A, C, G, U
  - Protein alphabet (20 amino acids): A, ..., V

- q_a = the occurrence probability of $a \in \Sigma$ in a sequence, *independent* of the rest of the sequence (= Bernoulli model)

Random sequence model (Bernoulli model)

- Probability of sequence $x = x_1x_2...x_n$ is
  $P(x) = q_{x(1)}q_{x(2)}...q_{x(n)}$

- This is the base-level model to compare other models against

- NOTE on the notation used: Because of the limitations of PowerPoint, I must sometimes write x(i) instead of x_i

Maximum likelihood (ML) estimation

- Goal: estimate the parameters Θ of a probabilistic model from a training data D

- Example:
  - #a = total number of a's in all sequences of a sequence database DB
  - |DB| = total length of DB
  - ML estimate

$$q_a = \#a \,/\, |DB|$$

Overfitting

- D too small → danger of *overfitting* in ML estimation
- Example: rolling a die
  - 3 rolls gives, say, three times 6. Then D = 6, 6, 6
  - ML estimate for Θ:
    - $p_1 = ... = p_5 = 0$
    - $p_6 = \#6/3 = 3/3 = 1$
  - Any good? Obviously overfitting!
  - Solution: add pseudocounts to the observed counts

ML estimation in general

- Θ = parameters of the model
- Find Θ such that $P(D|\Theta)$ (= probability of the training data in model Θ) is largest possible
- ML model for D: $\boxed{\Theta_{ML} = \arg_\Theta \max P(D|\Theta)}$
- Overfitting
- Pseudocounts

Hidden Markov Models for sequence families

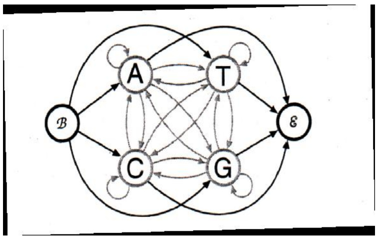
Durbin et al., Chapters 3,4,5

Markov chain

- **Definition:** A *Markov chain* for modeling sequences $x_1x_2 ...$ of symbols in *alphabet* Σ is a triplet (Q, $\{p(x_1=s) \mid s \in Q\}$, A), where:
  - Q is a finite set of *states*. Each state corresponds to a symbol in the alphabet Σ.
  - p gives the *initial state probabilities*.
  - A is the set of *state transition probabilities*, denoted by a_{st} for each $s, t \in Q$.
- For each $s, t \in Q$ the transition probability is:

  $$a_{st} \equiv P(x_i = t \mid x_{i-1} = s)$$

- $\sum_t a_{st} = 1$ for every s



Markov chain for modeling DNA sequences

Markov property

Assume that $X = (x_1, . . . , x_L)$ is a random process with a memory of length 1, i.e., the value of the random variable x_i depends only on its predecessor x_{i-1}. Then we can write:

$$\forall s_1, . . . , s_i \in \Sigma \qquad P(x_i = s_i \mid x_1 = s_1, . . . , x_{i-1} = s_{i-1}) = = P(x_i = s_i \mid x_{i-1} = s_{i-1}) = a_{s(i-1),s(i)}$$

The probability of the whole sequence X will therefore be:

$$P(X) = p(x_1) \cdot \prod_{i=2,...,L} a_{x(i-1),x(i)}$$

We can add fictitious *begin* and *end* states together with corresponding symbols x_0 and x_{L+1}. Then we can define $\forall s \in \Sigma$: $a_{0,s} \equiv p(s)$, where $p(s)$ is the initial probability of the symbol s. Hence:

$$P(X) = \prod_{i=1,...,L} a_{x(i-1),x(i)}$$

CpG islands

CpG island: DNA regions where dinucleotide CG occurs relatively often (normally the dinucleotide CG is quite rare because of frequent *methylation mutations* CG→TG that convert CG to TG)

Markov chain for CpG island:

+	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182

Markov chain for not-CpG island:

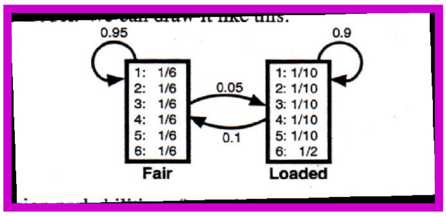
-	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

Hidden Markov Model

- **Definition** A *Hidden Markov Model (HMM)* is a triplet $M = (\Sigma, Q, \Theta)$, where:
  - Σ is an alphabet of symbols
  - Q is a finite set of states, capable of emitting symbols from the alphabet Σ
  - Θ is a set of probabilities, comprised of:
    - *State transition probabilities*, denoted by a_{kl} for each $k, l \in Q$, such that $\sum_t a_{kl} = 1$ for all k
    - *Emission probabilities*, denoted by $e_k(b)$ for each $k \in Q$ and $b \in \Sigma$, such that $\sum_b e_k(b) = 1$ for all k

Example: Dishonest casino

- The states are $Q = \{F,L\}$, where *F* stands for "fair" and *L* for "loaded"
- The alphabet is $\Sigma = \{1, 2, 3, 4, 5, 6\}$
- Fair die: $p_i = 1/6$ for all i
- Loaded die: $p_1 = \ldots = p_5 = 1/10$; $p_6 = ½$
- Probability of switching from fair to loaded is 0.05, and of switching back is 0.1



State transition probabilities

- A *path*

$$\Pi = (\pi_1, \ldots, \pi_L)$$

in the model *M* is a sequence of states. The path itself follows a simple Markov chain, so the probability of moving to a given state depends only on the previous state. As in the Markov chain model, we define the *state transition probabilities* on the path Π:

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

Emission probabilities

- In a hidden Markov model there isn't a one-to-one correspondence between the states and the symbols. Therefore, in a HMM we introduce a new set of parameters, $e_k(b)$, called the *emission probabilities*.
- Given an emission sequence $X = (x_1, \ldots, x_L) \in \Sigma^*$ for path Π, define:

$$e_k(b) = P(x_i = b \mid \pi_i = k)$$

- $e_k(b)$ is the probability that symbol b is seen when we are in state k.

Probability of emitting X from path Π

- The joint probability of the observed sequence X and the path of states Π is therefore:

$$\boxed{P(X, \Pi) = a_{\pi(0),\pi(1)} \cdot \prod_{i=1,\ldots,L} e_{\pi(i)}(x_i)\, a_{\pi(i),\pi(i+1)}}$$

where we denote

π_0 = *begin state,*

π_{L+1} = *end state*

The decoding problem

.

- **INPUT:** A hidden Markov model $M = (\Sigma, Q, \Theta)$ and a sequence $X \in \Sigma^*$, for which the generating path $\Pi = (\pi_1, \ldots, \pi_L)$ is unknown.

- **QUESTION:** Find the most probable generating path Π^* for X, i.e., a path such that $P(X, \Pi^*)$ is maximized:

$$\Pi^* = \text{argmax}_{\Pi}\{P(X, \Pi)\}$$

Viterbi algorithm

- Calculates the **most probable path** in a hidden Markov model using a dynamic programming algorithm (Viterbi 1967, Bellman 1957)

- Let X be a sequence of length L. For $k \in Q$ and $0 \leq i \leq L$, we consider a path Π ending at k, and the probability of Π generating the prefix $(x_1,\ldots,x_i)$ of X

- Denote by $v_k(i)$ the probability of the most probable path for the prefix $(x_1,\ldots,x_i)$ that ends in state k:

$$v_k(i) = \max_{\{\Pi|\Pi(i)=k\}} P(x_1,\ldots,x_i, \Pi)$$

Viterbi (cont.)

- 1. Initialize:
  $v_{begin}(0) := 1$
  $v_k(0) := 0$ if $k \neq begin$

- 2. For each $i = 0, \ldots, L - 1$ and for each $l \in Q$ calculate:
  $v_l(i + 1) := e_l(x_{i+1}) \cdot \max_{k \in Q} \{v_k(i) \cdot a_{kl}\}$

- 3. Finally, the value of $P(X,\Pi^*)$ is:
  $P(X,\Pi^*) := \max_{k \in Q} \{v_k(L) \cdot a_{k,end}\}$

- Reconstruct the path Π^* itself by keeping back pointers during the recursive stage and tracing them afterwards

Complexity of Viterbi

- **Complexity**: We calculate the values of $O(|Q| \cdot L)$ cells of the matrix V, spending $O(|Q|)$ operations per cell. Therefore the overall
  - time complexity is $O(L \cdot |Q|^2)$, and
  - the space complexity is $O(L \cdot |Q|)$

Viterbi example

```
Rolls   315116246446644245311321631164152133625144543631656626566666
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLL

Rolls   651166453132651245636664631636663162326455236266666625151631
Die     LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLFFFFFFFFF
Viterbi LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF

Rolls   222555441666566563564324364131513465146353411126414626253356
Die     FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls   366163666466232534413661661163252562462255265252266435353336
Die     LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls   233121625364414432335163243633665562466662632666612355245242
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
```

Running the Viterbi algorithm on the dishonest casino example: The numbers show 300 rolls of a die. Below is shown which die was actually used for that roll (F for fair and L for loaded). Under that, the prediction by the Viterbi algorithm is shown.

Exercise Problem (extra)

- Develop an algorithm that finds for a given HMM and length L the most probable emission sequence of length L.

Posterior Decoding

- **INPUT:** A hidden Markov model $M = (\Sigma, Q, \Theta)$ and a sequence $X \in \Sigma^*$, for which the generating path $\Pi = (\pi_1, \ldots, \pi_L)$ is unknown.
- **QUESTION:** For each $1 \leq i \leq L$ and $k \in Q$, compute the probability $P(\pi_i = k \mid X)$

- For this we shall need some extra definitions and algorithms

Forward algorithm

- Given a sequence $X = (x_1, \ldots, x_L)$, the problem is to compute the total probability of emitting X

  $$\mathbf{P(X) = \sum_{\Pi} P(X, \Pi)}$$

- Computation proceeds in forward direction, from time 0 to time L
- Denote by $f_k(i)$ the probability of emitting the prefix $(x_1, \ldots, x_i)$ and eventually reaching state $\pi_i = k$:

  $$f_k(i) = P(x_1, \ldots, x_i, \pi_i = k)$$

Forward (cont.)

- Use the same initial values for $f_k(0)$ as was done in the Viterbi algorithm:

  $f_{begin}(0) := 1$

  $f_k(0) := 0$, if k $\neq$ *begin*
- In analogy to Viterbi, for each $i = 0, \ldots, L - 1$ and for each $l \in Q$ calculate

  $f_l(i + 1) := e_l(x_{i+1}) \cdot \sum_{k \in Q} f_k(i) \cdot a_{kl}$
- Terminate the process by calculating

  $P(X) := \sum_{k \in Q} f_k(L) \cdot a_{k,end}$

Backward algorithm

- Given a sequence $X = (x_1, \ldots, x_L)$, the problem is (again) to compute

  $\mathbf{P(X) = \sum_{\Pi} P(X, \Pi)}$
- Computation proceeds backwards, from time L to time 0
- Denote by $b_k(i)$ the probability of emitting the suffix $(x_{i+1},\ldots,x_L)$, given $\pi_i = k$:

  $b_k(i) = P(x_{i+1}, \ldots, x_L, \pi_i = k)$

Backward (cont.)

- Initialization

  $b_k(L) := a_{k,end}$ for all $k \in Q$

- In the backward direction, for each i = L-1,...,0 and for each $l \in Q$ calculate

  $b_k(i) := \sum_{l \in Q} a_{kl} \cdot e_l(x_{i+1}) \cdot b_l(i+1)$

- Terminate the process by calculating

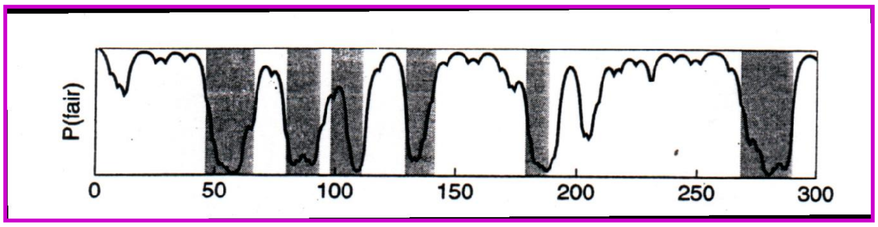
  $P(X) := \sum_{l \in Q} a_{begin,l} \cdot e_l(x_1) \cdot b_l(1)$

Complexity

- All the values of $f_k(i)$ and $b_k(i)$ can be calculated in $O(L \cdot |Q|^2)$ time and stored in $O(L \cdot |Q|)$ space, as it is the case with Viterbi algorithm

Posterior decoding (cont.)

- The forward and backward probabilities give $P(\pi_i = k \mid X)$:
- Since process X has memory of only length 1, we have
  $P(X,\ \pi_i = k) =$
  $= P(x_1,\ldots,x_i,\ \pi_i = k) \cdot P(x_{i+1},\ldots,x_L \mid x_1,\ldots,x_i,\ \pi_i = k) =$
  $=P(x_1,\ldots,x_i,\ \pi_i = k) \cdot P(x_{i+1},\ldots,x_L \mid \pi_i = k) =$
  $= f_k(i) \cdot b_k(i)$
- Using the definition of conditional probability, we obtain the solution to the posterior decoding problem:
  $$P(\pi_i = k \mid X) = \frac{P(X, \pi_i = k)}{P(X)} = \frac{f_k(i) \cdot b_k(i)}{P(X)}$$
  Here P(X) is obtained using the forward or backward algorithm



The posterior probability of being in the state corresponding to the fair die in the dishonest casino example. Shaded areas: the roll was generated by a loaded die.

Parameter estimation for HMMs

The learning problem for HMMs: Given training data $D = X^{(1)}, ..., X^{(n)}$ where each $X^{(i)}$ is a sequence in the emission alphabet, construct the HMM that will best characterize D

Solution: We need to assign values to Θ that will maximize the probabilities of the sequences $X^{(i)}$ (= ML estimate). Sequences are assumed independent, hence:

$$P(X^{(1)}, \ldots, X^{(n)} | \Theta) = \prod_{i=1}^{n} P(X^{(i)} | \Theta)$$

ML estimate

$$\Theta^* = \operatorname{argmax}_{\Theta}\{Score(X^{(1)}, \ldots, X^{(n)} | \Theta)\}$$

$$Score(X^{(1)}, \ldots, X^{(n)} | \Theta) = \log P(X^{(1)}, \ldots, X^{(n)} | \Theta) = \sum_{j=1}^{n} \log(P(X(i)|\Theta))$$

Estimation when the state sequence is known

Assume that the state sequences $\Pi^{(1)}, ..., \Pi^{(n)}$ through the HMM are known for $X^{(1)}, ..., X^{(n)}$ (for example, by an annotation of the $X^{(i)}$s that indicates the CpG islands (if we want to model CpG islands))

Count the total number of each event along these paths:

A_{kl} - the number of transitions from the state k to l

$E_k(b)$ - the number of times that an emission of the symbol b occurred in state k

ML estimators

$$a_{kl} = \frac{A_{kl}}{\sum_{q \in Q} A_{kq}} \qquad e_k(b) = \frac{E_k(b)}{\sum_{\sigma \in \Sigma} E_k(\sigma)}$$

Overfitting: use pseudocounts $A_{kl} := A_{kl} + r_{kl}$ . . . (Laplace rule: $r_{kl} = 1$)

Estimation when the state sequence is unknown: Baum-Welch training

- The *Baum-Welch algorithm*, which is a special case of the *EM technique (Expectation-Maximization)*, can be used for heuristically finding an approximate ML solution
- Big picture:
  - start with some Θ;
  - compute *expected values* for A_{kl} and $E_k(b)$ in model Θ for the training data $X^{(i)}$;
  - estimate new a and e (= new Θ) from these expected values;
  - continue iterating this way until the value of the objective function $\log P(X|\Theta)$ changes less than some predefined threshold.
- BW always monotonically converges to a *local* optimum

BW more precisely

- $f_k(i)$ and $b_k(i)$ as in Forward/Backward algorithms
- Probability of taking transition $k \rightarrow l$ and emitting x_{i+1} from state l when HMM emits a sequence $x = x_1 \ldots x_L$:

$$P(\pi_i=k,\pi_{i+1}=l \mid x,\Theta) = f_k(i)a_{kl}e_l(x_{i+1})b_l(i+1)/P(x|\Theta)$$

- $\rightarrow$ Expected number of times that $k \rightarrow l$ is used for training data D:

$$A_{kl} = \sum_j P(X^{(j)} |\Theta)^{-1} \sum_i f^{(j)}_k(i)\, a_{kl}\, e_l(x^{(j)}_{i+1})b^j_l(i+1) \qquad (*)$$

- $\rightarrow$ Expected number of times of emitting symbol b from state k for training data D:

$$E_k(b) = \sum_j P(x(j)|\Theta)^{-1} \sum_{\{i \mid x^j(i)=b\}} f^j_k(i)\, b^j_k(i) \qquad (**)$$

Baum-Welch Algorithm

- **Input**: training data D, threshold T, limit M
- **1. Initialization:** $\Theta := ((a_{kl})_{k\varepsilon V, l\varepsilon V}, (e_k(b))_{k\varepsilon V, b\varepsilon\Sigma})$ arbitrary initial values
- **2. Iterative search:**
  - Set all the A and E variables to their pseudocount values r (or 0)
  - *Expectation-step.* For each $X^{(j)}$ in D do:
    - Calculate $f_k(i)$ for all k, i using the Forward algorithm
    - Calculate $b_k(i)$ for all k, i using the Backward algorithm
    - Using the calculated values $f_k(i)$ and $b_k(i)$, evaluate and add the contribution of $x^{(j)}$ to values A_{kl} and $E_k(b)$ ((*) and (**) on the previous slide)
  - *Maximization step.* Calculate new Θ:

$$a_{kl} := \frac{A_{kl}}{\sum_{q\in Q} A_{kq}} \qquad e_k(b) := \frac{E_k(b)}{\sum_{\sigma\in\Sigma} E_k(\sigma)}$$

- **3. Stop?**
  - Repeat Step 2 until $\log P(D \mid \Theta_{new}) - \log P(D \mid \Theta_{old}) \le T$ or the number of iterations taken is = M

Viterbi training

- Similar to the BW-algorithm but parameters a and e are updted using the A and B counts obtained from the most probable paths $\Pi^*(x^{(1)})$, ..., $\Pi^*(x^{(n)})$ for $x^{(1)}$, ..., $x^{(n)}$. These paths can be found using the Viterbi algorithm.
- Converges always as the Viterbi paths can change only finitely many times (as they are finite structures)
- Does not maximize $\log P(D \mid \Theta)$ (see Durbin pp 64-65

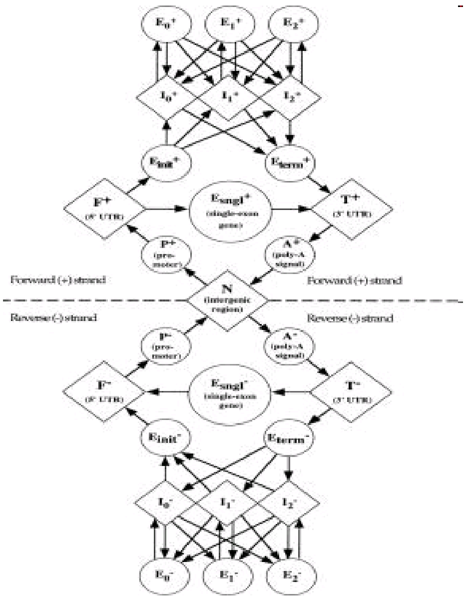
HMM model structure

- Choice of model topology: complete transition graph (i.e., E = V x V) is difficult to train as it has lots of local maxima
  - Prune E using *prior knowledge* of the problem.
  - **Elimination of transition k→l ↔ a_{kl} = 0**
  - The topological structure should be such that it has natural correspondence with the problem to be modeled

- Silent states:
  - no emissions
  - If there are no cycles consisting of only silent states, then the above algorithms work after small modifications (for example, the Forward algorithm should traverse the silent states in the so-called topological order; as there are no cycles, such an order exists)

Example: HMM architecture of GENESCAN

Prediction of exons (genes)



Numerical stability of HMM algorithms

- Long multiplications of probability values can lead to numerical problems: underflow of floating-point numbers
- Two main solution techniques
  - Log transformations: x → +
    - Does not work if both x and + are present in the algorithm (Viterbi ok, Forward/Backward not)
  - Scaling of probabilities
- Details: see Durbin pp 77-78