Reminder

- Last class we discussed dynamic programming algorithms for:
  - global alignment
  - local alignment
- All of these assumed a pre-specified scoring rule (substitution matrix):
  \[ \sigma : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow \mathbb{R} \]

  that determines the quality of perfect matches, substitutions and indels, independently of neighboring positions.

A Probabilistic Model

- But how do we derive a “good” substitution matrix?
- It should “encourage” pairs, that are probable to change in close sequences, and “punish” others.
- Let’s examine a general probabilistic approach, guided by evolutionary intuitions.
- Assume that we consider only two options:
  - \( M \): the sequences are evolutionary related
  - \( R \): the sequences are unrelated

Unrelated Sequences

- Our model of 2 unrelated sequences \( s, t \) is simple:
  - For each position \( i \), both \( s[i], t[i] \) are sampled independently from some “background” distribution \( q(\cdot) \) over the alphabet \( \Sigma \).
  - Let \( q(a) \) be the probability of seeing letter \( a \) in any position.
- Then the likelihood of \( s, t \) (probability of seeing \( s, t \), given they are unrelated) is:
  \[ P(s[1..n], t[1..n] \mid R) = \prod_{i=1}^{n} q(s[i]) \prod_{j=1}^{n} q(t[j]) \]

Related Sequences

- Now let’s assume that each pair of aligned positions \( s[i], t[i] \) evolved from a common ancestor \( \Rightarrow s[i], t[i] \) are dependent!
  - We assume \( s[i], t[i] \) are sampled from some distribution \( p(\cdot, \cdot) \) of letters pairs.
  - Let \( p(a, b) \) be a probability that some ancestral letter evolved into this particular pair of letters.
- Then the likelihood of \( s, t \), given they are related is:
  \[ P(s[1..n], t[1..n] \mid M) = \prod_{i=1}^{n} p(s[i], t[i]) \]

Decision Problem

- Given two sequences \( s[1..n] \) and \( t[1..n] \) decide whether they were sampled from \( M \) or from \( R \).
- This is an instance of a decision problem that is quite frequent in statistics: hypothesis testing.
- We want to construct a procedure \( \text{Decide}(s, t) = D(s, t) \) that returns either \( M \) or \( R \).
- Intuitively, we want to compare the likelihoods of the data in both models…
Types of Error

- Our procedure can make two types of errors:
  I. \( s \) and \( t \) are sampled from \( R \) but \( D(s, t) = M \)
  II. \( s \) and \( t \) are sampled from \( M \) but \( D(s, t) = R \)

- Define the following error probabilities:
  \[ \alpha(D) = \Pr(D(s, t) = M \mid R) \]
  \[ \beta(D) = \Pr(D(s, t) = R \mid M) \]

- We want to find a procedure \( D(s, t) \) that minimizes both types of errors

Neyman-Pearson Lemma

- Suppose that \( D^* \) is such that for some \( k \)
  \[ D^*(s, t) = \begin{cases} 
  M & \frac{P(s, t \mid M)}{P(s, t \mid R)} > k \\
  R & \frac{P(s, t \mid M)}{P(s, t \mid R)} < k 
  \end{cases} \]

- If any other \( D \) is such that \( \alpha(D) \leq \alpha(D^*) \), then \( \beta(D) \geq \beta(D^*) \) \( \rightarrow \) \( D^* \) is optimal

- \( k \) might refer to the weights we wish to give to both types of errors, and on relative abundance of \( M \) comparing to \( R \)

Likelihood Ratio for Alignment

- The likelihood ratio is a quantitative measure of two sequences being derived from a common origin, compared to random.

- Let's see, that it is a natural score for their alignment!

- Plugging in the model, we have that:
  \[ \frac{P(s, t \mid M)}{P(s, t \mid R)} = \prod_i \frac{p(s[i], t[i])}{q(s[i])q(t[i])} = \prod_i \frac{p(s[i], t[i])}{q(s[i])q(t[i])} \]

Likelihood Ratio for Alignment

- Taking logarithm of both sides, we get
  \[ \log \frac{P(s, t \mid M)}{P(s, t \mid R)} = \log \prod_i \frac{p(s[i], t[i])}{q(s[i])q(t[i])} = \sum_i \log \frac{p(s[i], t[i])}{q(s[i])q(t[i])} \]

- We can see that the (log-)likelihood score decomposes to sum of single position scores, each dependent only on the two aligned letters!

Probabilistic Interpretation of Scoring Rule

- Therefore, if we take our substitution matrix be:
  \[ \sigma(a, b) = \log \frac{p(a, b)}{q(a)q(b)} \]

- then the score of an alignment is the log-ratio between the two models likelihoods, which is nice.

  - Score > 0 \( \Rightarrow \) \( M \) is more “probable” (k=1)
  - Score < 0 \( \Rightarrow \) \( R \) is more “probable”

Modeling Assumptions

- It is important to note that this interpretation depends on our modeling assumption of the two hypotheses!!

- For example, if we assume that the letter in each position depends on the letter in the preceding position, then the likelihood ratio will have a different form.
Constructing Scoring Rules

The formula
\[ \sigma(a, b) = \log \frac{p(a, b)}{q(a)q(b)} \]

suggests how to construct a scoring rule:
- Estimate \( p(\cdot, \cdot) \) and \( q(\cdot) \) from the data
- Compute \( \sigma(a, b) \) based on \( p(\cdot, \cdot) \) and \( q(\cdot) \)

Estimating Probabilities

- Suppose we are given a long string \( s[1..n] \) of letters from \( \Sigma \)
- We want to estimate the distribution \( q(\cdot) \) that "generated" the sequence
- How should we go about this?
- We build on the theory of parameter estimation in statistics

Statistical Parameter Fitting

- Consider instances \( x[1], x[2], \ldots, x[M] \) such that
  - The set of values that \( x \) can take is known
  - Each is sampled from the same (unknown) distribution of a known family (multinomial, Gaussian, Poisson, etc.)
  - Each is sampled independently of the rest
- The task is to find a parameters set \( \Theta \) defining the most likely distribution \( P(x|\Theta) \), from which the instances could be sampled.
- The parameters \( \Theta \) depend on the given family of probability distributions.

Example: Binomial Experiment

- When tossed, it can land in one of two positions: Head or Tail
- We denote by \( \theta \) the (unknown) probability \( P(H) \).
- Estimation task:
  - Given a sequence of toss samples \( x[1], x[2], \ldots, x[M] \) we want to estimate the probabilities \( P(H) = \theta \) and \( P(T) = 1 - \theta \)

Why Learning is Possible?

- Suppose we perform \( M \) independent flips of the thumbtack
- The number of head we see is a binomial distribution
  \[ P(\# Heads = k) = \binom{M}{k} \theta^k (1-\theta)^{M-k} \]
- and thus \( E[\# Heads] = M\theta \)

This suggests, that we can estimate \( \theta \) by
\[ \frac{\# Heads}{M} \]

Expected Behavior (\( \theta = 0.5 \))

- From most large datasets, we get a good approximation to \( \theta \)
- How do we derive such estimators in a principled way?
The Likelihood Function

- How good is a particular $\theta$?
  It depends on how likely it is to generate the observed data
  
  $$L(\theta : D) = P(D | \theta) = \prod_{m} P(x[m] | \theta)$$

- The likelihood for the sequence H,T, T, H, H is
  
  $$L(\theta : D) = \theta \cdot (1 - \theta) \cdot (1 - \theta) \cdot \theta \cdot \theta$$

Maximum Likelihood Estimation

- MLE Principle:
  Learn parameters that maximize the likelihood function

  - This is one of the most commonly used estimators in statistics
  - Intuitively appealing

Computing the Likelihood Functions

- To compute the likelihood in the thumbtack example we only require $N_H$ and $N_T$ (the number of heads and the number of tails)
  
  $$L(\theta : D) = \theta^{N_H} \cdot (1 - \theta)^{N_T}$$

- $N_H$ and $N_T$ are sufficient statistics for the binomial distribution

Example: MLE in Binomial Data

- Applying the MLE principle we get (after differentiating)
  
  $$\hat{\theta} = \frac{N_H}{N_H + N_T}$$

  (Which coincides with what we would expect)

  Example:
  
  $(N_H, N_T) = (3, 2)$
  
  MLE estimate is $3/5 = 0.6$

Sufficient Statistics

- A sufficient statistic is a function of the data that summarizes the relevant information for the likelihood

- Formally, $s(D)$ is a sufficient statistics if for any two datasets $D$ and $D'$
  
  - $s(D) = s(D')$
  
  $$\Rightarrow L(\theta | D) = L(\theta | D')$$

From Binomial to Multinomial

- Suppose $X$ can have the values $1, 2, ..., K$

- We want to learn the parameters $\theta_1, \theta_2, ..., \theta_K$

Sufficient statistics:

- $N_1, N_2, ..., N_K$ - the number of times each outcome is observed

Likelihood function:

$$L(\theta : D) = \prod_{k} \theta_k^{N_k}$$

MLE (differentiation with Lagrange multipliers):

$$\hat{\theta}_k = \frac{N_k}{\sum_i N_i}$$
At last: Estimating $q(\cdot)$

- Suppose we are given a long string $s[1..n]$ of letters from $\Sigma$
  - $s$ can be the concatenation of all sequences in our database
- We want to estimate the distribution $q(\cdot)$

Likelihood function:

$$L(q : s) = \prod_{i=1}^{n} q(s[i]) = \prod_{a} q(a)^{N_a}.$$ 

MLE parameters:

$$q(a) = \frac{N_a}{n}$$

Estimating $p(\cdot, \cdot)$

Intuition:
- Find pair of presumably related aligned sequences $s[1..n], t[1..n]$
- Estimate probability of pairs in the sequence:

$$p(a, b) = \frac{N_{a,b}}{n}$$

- Again, $s$ and $t$ can be the concatenation of many aligned pairs from the database

Problems:
- How do we find pairs of presumably related aligned sequences?
- Can we ensure that the two sequences are indeed based on a common ancestor?
- How far back should this ancestor be?
  - earlier divergence $\Rightarrow$ low sequence similarity
  - later divergence $\Rightarrow$ high sequence similarity
- The substitution score of each 2 letters should depend on the evolutionary distance of the compared sequences!

Let Evolution In

- Again, we need to make some assumptions:
  - Each position changes independently of the rest
  - The probability of mutations is the same in each positions
  - Evolution does not “remember”

Model of Evolution

- How do we model such a process?
- The process (for each position independently) is called a Markov Chain
- A chain is defined by the transition probability $P(X_{t+\Delta} = b | X_t = a)$ - the probability that the next state is $b$ given that the current state is $a$
- We often describe these probabilities by a matrix:

$$T[\Delta]_{ab} = P(X_{t+\Delta} = b | X_t = a)$$

Two-Step Changes

- Based on $T[\Delta]$, we can compute the probabilities of changes over two time periods

$$P(X_{t+2\Delta} = b | X_t = a) = \sum_c T_{ac} T_{cb}$$

- Thus $T[2\Delta] = T[\Delta] T[\Delta]$
- By induction: $T[k\Delta] = T[\Delta]^k$
Longer Term Changes

Idea:
- Estimate $T[\Delta]$ from some closely related sequences set $S$
- Use $T[\Delta]$ to compute $T[k\Delta]$
- Derive substitution probability after time $k\Delta$:
  
  
  $$p(a, b \mid k\Delta) = p(b \mid a, k\Delta)q(a) = T[k\Delta]_{ab}q(a)$$

  
  $T[k\Delta]_{ab}$ is the probability of a substitution from $a$ to $b$ after $k\Delta$.

  
  Note, that the score depends on evolutionary distance, as requested.

Estimating PAM1

- Collect counts $N_{ab}$ of aligned pairs $(a, b)$ in similar sequences in $S$
  
  
  - Sources include phylogenetic trees and closely related sequences (at least 85% positions have exact match)
  
  
  - Normalize counts to get transition matrix $T[\Delta]$, such that average number of changes is 1%
  
  
  - that is, $\sum_a p(a, a \mid \Delta) = 0.99$
  
  
  - this is called 1 point accepted mutation (PAM1) – an evolutionary time unit!

Using PAM

- The matrix PAM-$k$ is defined to be the score based on $T^k$
  
  
  - Historically researchers use PAM250
    
    
    - Longer than 100!
  
  
  - Original PAM matrices were based on small number of proteins
  
  
  - Later versions of PAM use more examples
  
  
  - Used to be the most popular scoring rule

Problems with PAM

- PAM extrapolates statistics collected from closely related sequences onto distant ones.
  
  
  - But "short-time" substitutions behave differently than "long-time" substitutions:
    
    
    - short-time substitutions are dominated by a single nucleotide changes that led to different translation (like L->I)
    
    
    - long-time substitutions do not exhibit such behavior, are much more random.
  
  
  - Therefore, statistics would be different for different stages in evolution.

BLOSUM (blocks substitution) matrix

- Source: aligned ungapped regions of protein families
  
  
  - These are assumed to have a common ancestor
  
  
  - Procedure:
    
    
    - Group together all sequences in a family with more than e.g. 62% identical residues
    
    
    - Count number of substitutions within the same family but across different groups
    
    
    - Estimate frequencies of each pair of letters
    
    
    - The resulting matrix is called BLOSUM62