Mathematical/Computational Techniques: A Brief Background

COMP 571 - Fall 2010
Luay Nakhleh, Rice University
Outline

(1) Averages
(2) Measures of variability
(3) Population values and sample values
(4) Binomial, Poisson, and normal distributions
(5) Markov chains and hidden Markov models
(1) Averages

* Making sense of a large number of observations is difficult, if not impossible
* We usually desire some measure of the central or typical value and some measure of the amount of variability
* We may also be interested in whether the values are symmetrically distributed about the central value, etc.
* Usually, we are interested in two quantities, the **mean** and the **variance**
The arithmetic mean of \( N \) measurements \( X_1, \ldots, X_N \) is defined by

\[
M_x = \bar{X} = \frac{1}{N} (X_1 + X_2 + \cdots + X_N) = \frac{1}{N} \sum_{i=1}^{N} X_i
\]
(1) Averages

* The geometric mean (useful for data that are not symmetrically distributed) of $N$ measurements $X_1, \ldots, X_N$ is defined by

$$G_x = \left( X_1 X_2 \cdots X_N \right)^{\frac{1}{N}} = \left( \prod_{i=1}^{N} X_i \right)^{\frac{1}{N}}$$

* For computation, it is more conveniently defined as the antilog of

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

where $x_i = \log X_i$
(1) Averages

* The **harmonic mean** of $N$ measurements $X_1, ..., X_N$ is the reciprocal of the arithmetic mean of the reciprocals

\[
\frac{1}{H_x} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{X_i}
\]
(2) Measures of Variability

* The variance, $V$, is defined as the mean of the squares of the deviations of the individual items from their mean

$$V_X = \frac{1}{N} \sum_{i=1}^{N} [(X_i - \bar{X})^2]$$

* For computation, it is more convenient to write the variance formula as

$$V_X = \bar{X}^2 - \bar{X}^2$$

* The standard deviation is defined by

$$\sigma = \sqrt{V}$$
(2) Measures of Variability

- Even though the standard deviation is measured in the same units of the original measurements and the variance is not, the latter is almost always the more useful quantity in population genetics, for two reasons:
  - The variance has properties of additivity and subdivisibility, whereas the standard deviation doesn’t
  - The rate of evolutionary change is more closely related to the variance than to other measures of population variability
(2) Measures of Variability

* The variance of the sum of two measurements is

\[ V_{X+Y} = V_X + V_Y + 2cov_{XY} \]

where \( cov_{XY} \) denotes the covariance of \( X \) and \( Y \)

\[ cov_{XY} = \frac{\sum (X_i - \overline{X})(Y_i - \overline{Y})}{N} \]

* For multiple quantities, we have

\[ V_{\sum X} = \sum V_X + 2 \sum cov_{XX'} \]

* For example, if we had measurements on hand length, forearm length, and upper arm length for each of \( N \) persons, the variance of the total arm length would be given by the sum of the three variances and six covariances.
(2) **Measures of Variability**

- If the quantities are independent, then their covariance is 0.
- We also have

\[ V_{KX} = K^2 V_X \]  \quad (K \text{ is a constant})

\[ V_X = \frac{1}{N} V_X \]
In the practical use of statistics, we are often interested in drawing inferences about some population on the basis of a sample of observations from the population.

If we wish to estimate the mean of a population, the mean of the sample is on average a correct (unbiased) estimate of the mean of the population.
On the other hand, the variance of a sample is a biased estimate of the variance of the population. The fix is easy: divide by $N-1$ instead of $N$.

$$V = \frac{\sum (X_i - \overline{X})^2}{N - 1}$$

The same correction applies to the covariance.

$$cov_{XY} = \frac{\sum (X_i - \overline{X})(Y_i - \overline{Y})}{N - 1}$$
(4) Binomial, Poisson, and Normal Distributions

- If the probability of an event is $p$, the probability that in $N$ independent trials the event will occur exactly $n$ times is given by the **binomial distribution**

$$prob(n) = \frac{N!}{n!(N-n)!} p^n (1-p)^{N-n}$$

- For multiple events, where $p_i$ is the probability of event $i$, the probability that in $N$ trials each event $i$ occurs $n_i$ times is given by the **multinomial distribution**

$$prob(n_1, n_2, n_3, \ldots) = \frac{N!}{n_1!n_2!n_3!\ldots} p_1^{n_1} p_2^{n_2} p_3^{n_3} \ldots$$

$$\sum n_i = N \quad \sum p_i = 1$$
For the binomial distribution, the variance of the number of occurrences is

$$V_N = Np(1 - p)$$

The variance of the proportion of occurrences in $N$ trials of an event with probability $p$ is

$$V_p = \frac{p(1 - p)}{N}$$

The expected number of occurrences of an event with probability $p$ is

$$Np$$
A case of special interest arises when $p$ is allowed to approach $0$ at the same time that $N$ becomes indefinitely large in such a way that the product $Np$ remains of moderate value.

The limiting form approached in this manner is the Poisson distribution.
The probability of exactly $n$ occurrences of the event when the mean number is $\mu (=Np)$ is given by

$$prob(n) = \frac{e^{-\mu} \mu^n}{n!}$$

The variance is the same as the mean for the Poisson distribution ($V_n = \mu$)
(4) Binomial, Poisson, and Normal Distributions

The distribution of measurements of many biological materials, and of great many other things, is often approximated by a symmetrical, bell-shaped curve that is the **normal distribution**

$$Y = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(X-\mu)^2}{2\sigma^2}}$$

The normal distribution is the limit of the binomial distribution as $N$ gets large while $p$ remains finite
(5) Markov Chains and Hidden Markov Models

- Modeling the statistical properties of biological sequences and distinguishing regions based on these models
- For the alignment problem, they provide a probabilistic framework for aligning sequences
Example: CpG Islands

- Regions that are rich in CG dinucleotide
- Promoter and “start” regions of many genes are characterized by high frequency of CG dinucleotides (in fact, more C and G nucleotides in general)
CpG Islands: Two Questions

* Q1: Given a short sequence, does it come from a CpG island?
* Q2: Given a long sequence, how would we find the CpG islands in it?
CpG Islands

Answer to Q1:

Given sequence x and probabilistic model M of CpG islands, compute $p = P(x|\text{M})$.

If $p$ is “significant”, then x comes from a CpG island; otherwise, x does not come from a CpG island.
CpG Islands

Answer to Q1:

Given sequence x, probabilistic model $M_1$ of CpG islands, and probabilistic model $M_2$ of non-CpG islands, compute $p_1 = P(x|M_1)$ and $p_2 = P(x|M_2)$

If $p_1 > p_2$, then x comes from a CpG island

If $p_1 < p_2$, then x does not come from a CpG island
Answer to Q2:

As before, use the models $M_1$ and $M_2$, calculate the scores for a window of, say, 100 nucleotides around every nucleotide in the sequence.

Not satisfactory.

A more satisfactory approach is to build a single model for the entire sequence that incorporates both Markov chains.
Difference Between the Two Solutions

* Solution to Q1:
  * One “state” for each nucleotide, since we have only one region
  * 1-1 correspondence between “state” and “nucleotide”

* Solution to Q2:
  * Two “states” for each nucleotide (one for the nucleotide in a CpG island, and another for the same nucleotide in a non-CpG island)
  * No 1-1 correspondence between “state” and “nucleotide”
Markov Chains vs. HMMs

* When we have a 1-1 correspondence between alphabet letters and states, we have a Markov chain.

* When such a correspondence does not hold, we only know the letters (observed data), and the states are "hidden"; hence, we have a hidden Markov model, or HMM.
Markov Chains

Associated with each edge is a transition probability
Markov Chains: The 1-1 Correspondence

Sequence: GAGCGCGTAC

States: $S_4S_1S_4S_2S_4S_2S_4S_3S_1S_2$
HMMs: No 1-1 Correspondence (2 States Per Nucleotide)
What's Hidden?

- We can “see” the nucleotide sequence.
- We cannot see the sequence of states, or path, that generated the nucleotide sequence.
- Hence, the state sequence (path) that generated the data is hidden.
Markov Chains and HMMs

* In Markov chains and hidden Markov models, the probability of being in a state depends solely on the previous state.

* Dependence on more than the previous state necessitates higher order Markov models.
Sequence Annotation Using Markov Chains

- The annotation is straightforward: given the input sequence, we have a unique annotation (mapping between sequence letters and model states)
- The outcome is the probability of the sequence given the model
Sequence Annotation Using HMMs

* For every input sequence, there are many possible annotations (paths in the HMM)

* Annotation corresponds to finding the best mapping between sequence letters and model states (i.e., the path of highest probability that corresponds to the input sequence)
Markov Chains: Formal Definition

* A set $Q$ of states

* Transition probabilities

  * $a_{st} = P(x_t = t | x_{t-1} = s)$: probability of state $t$ given that the previous state was $s$

* In this model, the probability of sequence $x = x_1 x_2 ... x_L$ is

\[
P(x) = P(x_L | x_{L-1}) P(x_{L-1} | x_{L-2}) \cdots P(x_2 | x_1) P(x_1) = P(x_1) \prod_{i=2}^{L} a_{x_{i-1} x_i}
\]
Markov Chains: Formal Definition

* Usually, two states “start” and “end” are added to the Markov chain to model the beginning and end of sequences, respectively.

* Adding these two states, the model defines a probability distribution on all possible sequences (of any length).
HMMs: Formal Definition

* A set $Q$ of states
* An alphabet $\Sigma$
* Transition probability $a_{st}$ for every two states $s$ and $t$
* Emission probability $e_k(b)$ for every letter $b$ and state $k$ (the probability of emitting letter $b$ in state $k$)
HMMs: Sequences and Paths

- Due to the lack of a 1-1 correspondence, we need to distinguish between the sequence of letters (e.g., DNA sequences) and the sequence of states (path).
- For every sequence (of letters) there are many paths for generating it, each occurring with its probability.
- We use $x$ to denote a (DNA) sequence, and $\pi$ to denote a (state) path.
HMMs: The Model Probabilities

- **Transition probability** \( a_{k\ell} = P(\pi_i = \ell | \pi_{i-1} = k) \)
- **Emission probability** \( e_k(b) = P(x_i = b | \pi_i = k) \)
HMMs: The Sequence Probabilities

- The joint probability of an observed sequence and a path is
  \[ P(x, \pi) = a_{0\pi_1} \prod_{i=1}^{L} e_{\pi_i}(x_i) a_{\pi_i \pi_{i+1}} \]

- The probability of a sequence is
  \[ P(x) = \sum_{\pi} P(x, \pi) \]
HMMs: The Parsing Problem

* Find the most probable state path that generates a given a sequence

\[ \pi^* = \operatorname{argmax}_\pi P(x, \pi) \]
HMMs: The Posterior Decoding Problem

* Compute "confidence" for the states on a path

\[ P(\pi_i = k | x) \]
HMMs: The Parameter Estimation Problem

* Compute the transition and emission probabilities of an HMM (from a given training data set)
A Toy Example: 5’ Splice Site Recognition

* From “What is a hidden Markov model?”, by Sean R. Eddy

* 5’ splice site indicates the “switch” from an exon to an intron
A Toy Example: 5' Splice Site Recognition

Assumptions

- Uniform base composition on average in exons
- Introns are A/T rich (40% A/T, 10% G/C)
- The 5' splice site consensus nucleotide is almost always a G (say, 95% G and 5% A)
A Toy Example: 5' Splice Site Recognition
HMMs: A DP Algorithm for the Parsing Problem

- Let $v_k(i)$ denote the probability of the most probable path ending in state $k$ with observation $x_i$

- The DP structure:

$$v_\ell(i + 1) = e_\ell(x_{i+1}) \max_k (v_k(i) a_{k,\ell})$$
## The Viterbi Algorithm

**Initialization**

\[ v_0(0) = 1, \quad v_k(0) = 0 \quad \forall k > 0 \]

**Recursion**

\[ v_\ell(i) = e_\ell(x_i) \max_k (v_k(i - 1) a_{k\ell}) \]
\[ \text{ptr}_i(\ell) = \arg\max_k (v_k(i - 1) a_{k\ell}) \quad i = 1 \ldots L \]

**Termination**

\[ P(x, \pi^*) = \max_k (v_k(L) a_{k0}) \]
\[ \pi_L^* = \arg\max_k (v_k(L) a_{k0}) \]

**Traceback**

\[ \pi_{i-1}^* = \text{ptr}_i(\pi_i^*) \quad i = 1 \ldots L \]
The Viterbi Algorithm

* Usually, the algorithm is implemented to work with logarithms of probabilities so that the multiplication turns into addition.

* The algorithm takes $O(Lq^2)$ time and $O(Lq)$ space, where $L$ is the sequence length and $q$ is the number of states.
A Toy Example: 5' Splice Site Recognition
A Toy Example: 5' Splice Site Recognition

Sequence: CTTCATGTGAAAGCAGACGTAAAGTCA

State path: EEEEEEEEEEEEEEEEEEEEEEEEE5

log P: -41.22

Parsing:

Posterior decoding:
Other Values of Interest

- The probability of a sequence, $P(x)$
- Posterior decoding: $P(\pi_i = k | x)$
- Efficient DP algorithms for both using the forward and backward algorithms
The Forward Algorithm

* $f_k(i)$: the probability of the observed sequence up to and including $x_i$, requiring that $\pi_i = k$

* In other words, $f_k(i) = P(x_1, \ldots, x_i, \pi_i = k)$

* The structure of the DP algorithm:

$$f_{\ell}(i + 1) = e_{\ell}(x_{i+1}) \sum_k f_k(i) a_{k,\ell}$$
The Forward Algorithm

* **Initialization:** \( f_0(0) = 1, \quad f_k(0) = 0 \quad \forall k > 0 \)

* **Recursion:** \( f_\ell(i) = e_\ell(x_i) \sum_k f_k(i - 1) a_{k\ell} \quad i = 1 \ldots L \)

* **Termination:** \( P(x) = \sum_k f_k(L) a_{k0} \)
The Backward Algorithm

- $b_k(i)$: the probability of the last observed $L-i$ letters, requiring that $\pi_i = k$
- In other words, $b_k(i) = P(x_L, ..., x_{i+1} | \pi_i = k)$
- The structure of the DP algorithm:

$$b_\ell(i) = \sum_k a_{\ell k} e_k(x_{i+1}) b_k(i + 1)$$
The Backward Algorithm

- **Initialization:** \( b_k(L) = a_{k0} \quad \forall k \)
- **Recursion:** \( b_\ell(i) = \sum_k a_{\ell k} e_\ell(x_{i+1}) b_k(i + 1) \quad i = L - 1, \ldots, 1 \)
- **Termination:** \( P(x) = \sum_k a_{0k} e_k(x_1) b_k(1) \)
The Posterior Probability

\[ f_k(i) b_k(i) = P(x, \pi_i = k) = P(\pi_i = k | x) P(x) \]

\[ \Rightarrow P(\pi_i = k | x) = \frac{f_k(i) b_k(i)}{P(x)} \]
The Probability of a Sequence

\[ P(x) = \sum_k f_k(L) a_{k0} \]

\[ P(x) = \sum_k a_{0k} e_k(x_1) b_k(1) \]
Computational Requirements of the Algorithms

Each of the algorithms takes $O(Lq^2)$ time and $O(Lq)$ space, where $L$ is the sequence length and $q$ is the number of states.
A Toy Example: 5' Splice Site Recognition
A Toy Example: 5' Splice Site Recognition
Applications of Posterior Decoding (1)

* Find the sequence $\pi'$ of states where $\pi'_i = \arg\max_k P(\pi_i = k | x)$

* This is a more appropriate path when we are interested in the state assignment at a particular point $i$ (however, this sequence of states may not be a legitimate path!)
Assume function $g(k)$ is defined on the set of states.

We can consider $G(i|x) = \sum_k P(\pi_i = k|x)g(k)$.

For example, for the CpG island problem, setting $g(k)=1$ for the “+” states, and $g(k)=0$ for the “-” states, $G(i|x)$ is precisely the posterior probability according to the model that base $i$ is in a CpG island.
Parameter Estimation for HMMs

- Two components:
  - the probabilities (emission and transition): there is a well-developed theory
  - the structure (states): more of an “art”
- We’ll focus on estimating the probabilities
Estimating HMM Emission and Transition Probabilities

* Given the structure of an HMM, and a set of training sequences, we’d want to estimate the probabilities from the training data set.

* There are two cases:
  * The training sequences are already annotated (i.e., the state sequences are known).
  * The training sequences are not annotated (i.e., the state sequences are not known).
Estimating HMM Probabilities: Known State Sequences

* Given a training data set, count the number of times each particular transition or emission is used; denote these by $A_{kl}$ and $E_k(b)$

* Then

\[
\begin{align*}
\alpha_{k \ell} &= \frac{A_{k \ell}}{\sum_{\ell'} A_{k \ell'}} \\
\epsilon_k(b) &= \frac{E_k(b)}{\sum_{b'} E_k(b')} 
\end{align*}
\]
Estimating HMM Probabilities: Unknown State Sequences

- The Baum-Welch algorithm, which is an expectation-maximization (EM) algorithm

- Informally, the algorithm first estimates the $A_{kl}$ and $E_k(b)$ by considering probable paths for the training sequences using the current values of $a_{kl}$ and $e_k(b)$

- Then, new values of the $a$s and $e$s are derived using the equations on the previous slide

- This process is iterated until some stopping criterion is reached
It is possible to show that the overall log likelihood of the model is increased by the iteration, and hence that the process will converge to a local maximum.

Unfortunately, there are usually many local maxima, and which one you end up with depends strongly on the starting values of the parameters.

The problem of local maxima is particularly severe when estimating large HMMs.
The Baum-Welch Algorithm

* More formally, the Baum-Welch algorithm calculates $A_{kl}$ and $E_k(b)$ as the expected number of times each transition or emission is used, given the training sequences.

* To do this, it uses the forward and backward values.
The Baum-Welch Algorithm

* The probability that $a_{kl}$ is used at position $i$ in sequence $x$ is

$$P(\pi_i = k, \pi_{i+1} = \ell | x, \theta) = \frac{f_k(i) a_{k\ell} e_\ell(x_{i+1}) b_\ell(i + 1)}{P(x)}$$
The Baum-Welch Algorithm

* From this we derive the expected number of times that $a_{kl}$ is used by summing over all positions and over all training data sets

$$A_{k\ell} = \sum_j \frac{1}{P(x^j)} \sum_i f^j_k(i) a_{k\ell} e_{\ell}(x^j_{i+1}) b^j_{\ell}(i + 1)$$

($f^j$ and $b^j$ are the forward and backward values for sequence $x^j$)
Similarly, we can find the expected number of times that letter \( b \) appears in state \( k \)

\[
E_k(b) = \sum_j \frac{1}{P(x^j)} \sum_{\{i \mid x^j_i = b\}} f^j_k(i) b^j_k(i)
\]
The Baum-Welch Algorithm

- Having calculated these expectations, the new model parameters are calculated just as before, using (1).
- We can iterate using the new values of the parameters to obtain new values of the As and Es as before, but in this case we are converging in a continuous-values space, and so will never in fact reach the maximum.
- It is therefore necessary to set a convergence criterion, typically stopping when the change in total log likelihood is sufficiently small.
The Baum-Welch Algorithm

1. **Initialization**: Pick arbitrary model parameters ($\theta$)

2. **Recurrence**:
   A. Set all the $A$ and $E$ variables to their pseudocount values $r$ (or to zero)
   B. For each sequence $j=1,...,n$
      I. Calculate $f_k(i)$ for sequence $j$ using the forward algorithm
      II. Calculate $b_k(i)$ for sequence $j$ using the backward algorithm
      III. Add the contribution of sequence $j$ to $A(2)$ and $E(3)$
   C. Calculate the new model parameters using (1)
   D. Calculate the new log likelihood of the model

3. **Termination**: Stop if the change in the log likelihood is less than some predefined threshold or the maximum number of iterations is exceeded
An alternative to the Baum-Welch algorithm is frequently used, which is called Viterbi training. In this approach, the most probable paths for the training sequences are derived using the Viterbi algorithm, and these are used in the re-estimation process. Again, the process is iterated when the new parameter values are obtained. In this case, the algorithm converges precisely, because the assignment of paths is a discrete process, and we can continue until none of the paths change. At this point the parameter estimates will not change either, because they are determined completely by the paths.
Viterbi Training

Unlike Baum-Welch, this procedure does not maximize the true likelihood (the probability of the sequences, given the parameters).

Instead, it finds the value of $\theta$ that maximizes the contribution to the likelihood $P(x_1, ..., x_n | \theta, \pi^*(x_1), ..., \pi^*(x_n))$ from the most probable paths for all the sequences.

This is a probable reason for why Viterbi training performs less well in general than Baum-Welch.

However, it is widely used, and it can be argued that when the primary use of the HMM is to produce decodings via Viterbi alignments, then it is good to train using them.