Genome Evolution, 2008 Version 0.9, May 25

Exam structure:

2 out of 3 math questions each consisting of two parts: the first requires repetition of one of the proofs/formula below. The second will require some small application of the first, possibly using one of the evolutionary insights from the course.

1 modeling question: Here you would be able to use any relevant method from the course to outline a solution for a given problem. The focus here is on choosing a good method and explaining why it suits your needs.

Mathematical theory you should memorize:

Continuous time Markov processes – understand the matrix exponential. Reversibility: define it, prove equivalence to detailed balance, understand examples.

Basics of probabilistic modeling: understand the notions: joint probability, posterior probability, likelihood, and marginalization. You should be able to compute these by hand for small examples of a tree, Bayes net or factor graph.

Inference on a tree: Up-Down algorithm for tree: state and prove correctness

EM: state the algorithm in general and for simple trees. Prove the algorithm is always improving. Show how to solve the maximization problem for simple trees.

Sampling: define the Metropolis MCMC algorithm, prove it defines a reversible Markov process with the appropriate stationary distribution. Do the same for Gibbs sampling.

Free Energy: Understand the energy~log(likelihood) equivalence. Define the variational free energy, show it bounds the –log(likelihood).

Mean field: Understand the MF free energy approximation. Define the algorithm for a tree. Derive the update formula. Derive their solution.

LBP: Define the algorithm in its basic (Bethe) form. Show the equivalence of the belief definitions and the message passing update rules.

Evo/Bio terms and notions:

No need to remember formulas, but make sure you understand the ideas and the connections between them.

* Neutral drift: basic population genetics model. How drift depends (or not depend) on population size.
* Recombination, linkage and genetic draft
* Effective population size
* Fitness and selection coefficients
* Ka/Ks and selection in protein coding genes
* Models for TFBS evolution: the sequence vs. binding energy vs. fitness question. Halpren-Burno framework and its assumption on binding energy = fitness. The "island" framework and its assumption on discrete fitness (1 for motifs, 0 for non motifs).

Reversibility: A Markov process can be defined by the transition probabilities $p_{i\to j}$.We call the process reversible if $\Pr(X_{t+dt}=i \mid X_t=j)=\Pr(X_t=i \mid X_{t+dt}=j)$ or in shorter notation if $p_{i\to j}=p_{j\leftarrow i}$. This means that reversing time does not change transition probabilities.
Claim: A Markov process is reversible iff there exists a stationary distribution π_i such that $p_{ij}\pi_i=p_{ji}\pi_j$ (this is called the detailed balance condition).

Proof: We are working with infinitesimal dt. Since a Markov process is defined by the infinitesimal rates, the proof will hold for any time interval. The proof is a direct application of the Bayes law: $p_{i\to j}=\frac{p_{i\leftarrow j}\pi_j}{\pi_i}$.

If detailed balance is holding: $p_{i\to j}=\frac{p_{i\leftarrow j}\pi_j}{\pi_i}=\frac{p_{j\leftarrow i}\pi_i}{\pi_i}=p_{j\leftarrow i}$

Conversely, if reversibility is holding: $p_{i\to j}=\frac{p_{i\leftarrow j}\pi_j}{\pi_i}=\frac{p_{j\to i}\pi_j}{\pi_i}\Rightarrow p_{ij}\pi_i=p_{ji}\pi_j$.

Probabilistic modeling: A Bayesian network is defined given a directed acyclic graph G on a set of random variables X using conditional probability distributions Pr(X|PaX). The *joint* distribution Pr(X) is defined by taking the product: $\Pr(x)=\prod\Pr(x_i \mid pax_i)$. We can work with a set of observations (Data), and in this case we denote the observed variables by S and the unobserved (hidden) variables by H. We define the likelihood of the data by $\Pr(s)=\sum_h\Pr(h,s)$ (taking the sum over an exponential number of hidden variables combinations. This process is sometime called marginalization). We would frequently be interested in the posterior probabilities $\Pr(h_i \mid s)=\sum_{h\setminus h_i}\Pr(h,s)/\Pr(s)$ (summing up over all hidden variable combination while fixing one or few random variables). A more general graphical model is called a factor graph which is defined based on a set of *potentials* which are not necessarily conditional distributions and include dependencies that can introduce cycles. The *joint* probability defined by a factor graph is $\Pr(x)=\frac{1}{Z}\prod_a\varphi_a(x_a)$ where φ_a is the potential for factor a, x_a is an assignment of values to the variables associated with the factor a, and Z is the partition function. The definition of the joint probability (for each type of model), along with the basic definition of conditional probabilities allow computation of any posterior probability for any group of variables. Such computation would however require summing over an exponential number of terms in the general case.

Inference on the evolutionary tree: for a node i we denote by l,r the left and right child nodes, by *pa*i the parent node and by sib the parent's other child node. The up-down algorithm for inference on tress is defined by two recursive update formulas (up and down):

$$up_i(x_i)=\sum_{x_l}up_l(x_l)\Pr(x_l \mid x_i)\sum_{x_r}up_r(x_r)\Pr(x_r \mid x_i)$$

$$down(x_i)=\sum_{x_{pai},x_{sib}}up_{sib}(x_{sib})down_{pai}(x_{pai})\Pr(x_{sib} \mid x_{pai})\Pr(x_i \mid x_{pai})$$

$$down(root)=\Pr(root), up(hiddenleaf)=1, up(obserevd)=observation$$

The recursive formula allows exact computation of the likelihood and posteriors:

$$(1)\ \Pr(s) = \sum_{x_i} up_i(x_i) down_i(x_i) \ \text{(for any variable)}$$

$$(2)\ \Pr(x_i \mid s) = up_i(x_i) down_i(x_i) / P(s)$$

$$(3)\ \Pr(x_i, x_{pai} \mid s) = \frac{1}{P(s)} up_i(x_i) \Pr(x_i \mid x_{pai}) down_{pai}(x_{pai}) \sum_{x_{sib}} up(x_{sib}) \Pr(x_{sib} \mid x_{pai})$$

We will prove directly the formula (1). Denote by T_i the subtree below node i, **excluding** i. Denote by T^j the subtree above node j **including** j. Remember that t_i represent assignment of values to nodes in T_i. We first argue that $up_i(x_i) = \sum_{t_i} \prod_{j \in T_i} \Pr(x_j \mid x_{paj}, D)$ for non leafs, or in words, the up probability is the likelihood of the data on the subtree below i, when fixing i's value to a x_i. Note that $\Pr(x_j \mid x_{paj}, D)$ is $\Pr(x_j \mid x_{paj})$ if j is hidden or is determined by the data completely if j is observed.

For a parent of two leafs the induction is true since:

$$up_i(x_i) = \sum_{x_l, x_r} \Pr(x_l \mid x_i) up(x_l) \Pr(x_r \mid x_i) up(x_r) = \sum_{t_i} \Pr(x_l \mid x_i, D) \Pr(x_r \mid x_i, D)$$

(the up of a leaf is defined as 1 if not observed and as 1 only on the observation when we have data). If the induction hypothesis is holding for the trees on less than n nodes, we combine two subtrees using a similar argument:

$$up_i(x_i) = \sum_{x_l, x_r} \Pr(x_l \mid x_i) up(x_l) \Pr(x_r \mid x_i) up(x_r) =$$

$$\sum_{x_l, x_r} \Pr(x_l \mid x_i) \Pr(x_r \mid x_i) \sum_{t_r} \prod_{T_r} \Pr(x_j \mid x_{paj}, D) \sum_{t_l} \prod_{T_l} \Pr(x_j \mid x_{paj}, D) =$$

$$\sum_{t_i} \prod_{T_i} \Pr(x_j \mid x_{paj}, D)$$

The general observation that makes the last passage possible is that the two subtrees are independent given the parent and we can therefore change the order of summation and multiplication using the distributive law.

We can show similarly that $down_i(x_i) = \sum_{t^i \setminus x_i} \prod_{j \in T^i} \Pr(x_j \mid x_{paj}, D)$. For the root this is holding by definition. Assuming it holds for trees smaller with fewer than n nodes, we can write:

$$down_i(x_i) = \sum_{x_{sib}, x_{pai}} \Pr(x_{sib} \mid x_{pai}) up(x_{sib}) \Pr(x_i \mid x_{pai}) down(x_{pai}) =$$

$$\sum_{x_{sib}, x_{pai}} \Pr(x_{sib} \mid x_{pai}) \Pr(x_i \mid x_{pai}) \sum_{t_{sib}} \prod_{T_{sib}} \Pr(x_j \mid x_{paj}, D) \sum_{t^{pai} \setminus x_{pai}} \prod_{T^{pai}} \Pr(x_j \mid x_{paj}, D) =$$

$$\sum_{t^i} \prod_{T^i} \Pr(x_j \mid x_{paj}, D)$$

It is now trivial to state:

$$\Pr(S) = \sum_{x_i} \sum_{t_i} \prod_{T_i} \Pr(x_j \mid x_{paj}, S) \sum_{t^i \setminus x_i} \prod_{T^i} \Pr(x_j \mid x_{paj}, S) = \sum_{h} \prod_{j} \Pr(x_j \mid x_{paj}, S)$$

EM: we often wish to maximize the likelihood of a set of model parameters given data:

$$\arg\max_{\theta} L(\theta \mid s) = \arg\max_{\theta} \Pr(s \mid \theta) = \arg\max_{\theta} \sum_{h} \Pr(s, h \mid \theta)$$

The EM algorithm is a generic iterative procedure for finding locally optimal parameters. The algorithm is based on the simple idea of maximizing the parameters given posterior probabilities for the hidden variables that were inferred using a current ad-hoc set of parameters:

$$\arg\max_{\theta} Q(\theta \mid \theta^k) = \arg\max_{\theta} \sum_{h} P(h \mid s, \theta^k) \log P(h, s \mid \theta)$$

In other words, we compute the maximum likelihood parameters, as if the hidden variables were actually observed, such that for each observation s, we observed h $P(h|s,\theta^k)$ times (or fractions of times). To show that optimizing Q improve the likelihood, observe that:

$$\log P(s \mid \theta) = \log P(h, s \mid \theta) - \log(h \mid s, \theta) = \sum_{h} P(h \mid s, \theta^k) \log P(h, s \mid \theta) - \sum_{h} P(h \mid s, \theta^k) \log P(h \mid s, \theta)$$

$$\log P(s \mid \theta) - \log P(s \mid \theta^k) = Q(\theta \mid \theta^k) - Q(\theta^k \mid \theta^k) + \sum_{h} P(h \mid s, \theta^k) \log \frac{P(h \mid s, \theta^k)}{P(h \mid s, \theta)}$$

$$= Q(\theta \mid \theta^k) - Q(\theta^k \mid \theta^k) + D(P(h \mid s, \theta^k) \,\|\, P(h \mid s, \theta)) \geq Q(\theta \mid \theta^k) - Q(\theta^k \mid \theta^k)$$

where we used the fact that the KL-divergence is non-negative.

When the joint (P(h,s)) is a BN (a tree in particular), the maximization of Q can be much simplified into a set of independent maximization problems:

$$\arg\max_{\theta_i} \sum_{x_i, pa(x_i)} \left[P(x_i, pa(x_i) \mid s, \theta^k) \log P(x_i \mid pa(x_i), \theta_i) \right]$$

(pa(xi) are all x_i's parents, and θ_i are the conditional probability parameters for variable i). Remember that we can compute the posterior of the **pair** x_i,pa(xi) efficiently in trees, or approximate it using MF/LBP/Sampling for more complex models. Remember that this formula is holding because log(P(h,s)) is breaking down into a sum of independent terms.

Sampling: MCMC inference can be used when it is difficult to compute the target distribution, but possible to compute simplified or restricted forms of it. Distributions that are difficult to compute are for example the posterior distribution P(h|s) in a BN or simply the joint P(x) in a factor graph (remember that these two are in fact equivalent: write P(h|s)=P(h,s)/P(s) and set Z=P(s) to see the analogy). In MCMC we use a Markov chain with a **provably** suitable stationary distribution and **tractable** transition probabilities. The trick is to use the detailed balance principle to show that a particular selection of transition probabilities give rise to the target stationary distribution.

The Metropolis algorithm uses an arbitrary symmetric proposal distribution S(y|x) and an acceptance criterion. The proposal distribution is required only to be ergodic, or in other words forming a (discrete time) Markov process that have non zero transition probabilities given sufficient time) from any state to any state. The overall Metropolis process is defined in two steps. Given a current state x, we first draw a sample from S(y|x). We then compute

P(y)/P(x). If it is larger than one, we change state to y. If it is smaller than one, we move to state y with probability P(y)/P(x) and keep x as the current state with probability 1-P(y)/P(x). To show this process is in detailed balance with stationary distribution P(x), we only have to write:

$$P(x)P(y \mid x) = P(x)\min(1, \frac{P(y)}{P(x)}) = \min(P(x), P(y)) = P(y)\min(\frac{P(x)}{P(y)}, 1)$$

Gibbs sampling uses a similar approach, but is based on sampling one random variable given all others: $P(x_i \mid x_1,..,x_{i-1},x_{i+1},..,x_n)$. This is also in detailed balance with stationary distribution P(x).

$$P(x_1,..,x_{i-1},x_i,x_{i+1},..,x_n)P(x'_i \mid x_1,..,x_{i-1},x_{i+1},..,x_n)$$
$$= P(x_1,..,x_{i-1},x_i,x_{i+1},..,x_n)\frac{P(x_1,..,x_{i-1},x'_i,x_{i+1},..,x_n)}{P(x_1,..,x_{i-1},x_{i+1},..,x_n)}$$
$$= P(x_1,..,x_{i-1},x'_i,x_{i+1},..,x_n)\frac{P(x_1,..,x_{i-1},x_i,x_{i+1},..,x_n)}{P(x_1,..,x_{i-1},x_{i+1},..,x_n)}$$
$$= P(x_1,..,x_{i-1},x'_i,x_{i+1},..,x_n)P(x_i \mid x_1,..,x_{i-1},x_{i+1},..,x_n)$$

To compute any statistics on P(x) (e.g., posterior probabilities of one or several variables), we simulate the (Gibbs/Metropolis) Markov process from an arbitrary initial condition, and start collecting samples after some "burn-in" period (which is difficult to define analytically).

Variational Free Energy: The variational transformation is trading one difficult problem (computing the likelihood) with a more difficult problem (minimizing the free energy of a trial distribution q). The basic idea is that the optimization problem can be more readily approximated. The free energy is defined as given a probabilistic model P(h,s) and a trial distribution q:

$$F(q) = -\sum_h q(h)\log p(h,s) + \sum_h q(h)\log q(h)$$

Where the first term is denoted as the "average energy" and the second term is the variational entropy. The association between energies and likelihood is rooted in the Boltzman theory from statistical mechanics which associates the probability of a certain ensemble state with its energy (p=(1/Z)exp(-kE)). The most useful property of the free energy is that it is a tight bound to the –log(likelihood). This is shown since:

$$F(q) = -\sum_h q(h)\log\big(p(h \mid s)p(s)\big) + \sum_h q\log q(h)$$
$$= \sum_h q(h)\log\frac{q(h)}{p(h \mid s)} - \log p(s) = D(q \parallel p(h \mid s)) - \log p(s) \geq -\log p(s)$$

It is a bound because the KL divergence is non negative (but note that for this we need p(h|s) to be a distribution!, summing p(h,s) over h is not enough!). The bound is tight because the KL divergence is zero for q=p(h|s).

Mean field: Using the variational transformation, a simple approximation to the optimization of the free energy is derived by considering only products of independent posteriors for variables or set of variables:

$$q = \prod_i q_i(h_i)$$

the mean field approximation of the free energy is computed by minimizing F(q) subject to the independence constraint. The approximation is effective when the joint is itself a product of (conditionally) independent terms, like a BN or a factor graph. In the case of a BN we can use the decomposition of q(h) and p(h,s) to rewrite the energy in a tractable form (we are using $p(h_i|pah_i)$ to denote both observed and hidden variables):

$$F_{MF} = -\sum_h (\Pi q_j(h_j)) \log \Pr(h,s \mid \theta) + \sum_h (\Pi q_j(h_j)) \log q(h)$$
$$= -\sum_h \sum_i \left(\Pi_j q_j(h_j)\right) \log \Pr(h_i \mid \text{pa}\, h_i) + \sum_i \sum_{h_i} q_i(h_i) \log q_i(h_i)$$
$$= -\sum_i \left(\sum_{h_i, \text{pa}\, h_i} \left(\prod_{pai} q_{\text{pa}\, i}(h_{\text{pa}\, i}) \right) q_i(h_i) \log \Pr(h_i \mid \text{pa}\, h_i) \right) + \sum_i \sum_{h_i} q_i(h_i) \log q_i(h_i)$$

The derived expression therefore represented the energy using a sum of terms that involve a product of posteriors of a variable and its parents with the conditional probability associating them. To optimize the mean field energy, we usually perform local optimization by selecting each time one variable and searching for its optimal q_i while fixing all the other q's. The key point in this optimization is that only a small number of terms in the above free energy expression are dependent on a particular q_i. These include terms of variables that are children or parents of i (as well as the variable itself). For a simple tree, optimizing F_{MF} by changing q_i involves working with the expression:

$$F_{MF} = \sum_{h_i, \text{pa}\, h_i} q_{\text{pa}\, i}(h_{\text{pa}\, i}) q_i(h_i) \log \Pr(h_i \mid \text{pa}\, h_i) + \sum_{h_i, h_l} q_l(h_l) q_i(h_i) \log \Pr(h_l \mid h_i)$$
$$+ \sum_{h_i, h_r} q_r(h_r) q_i(h_i) \log \Pr(h_r \mid h_i) + \sum_{h_i} q_i(h_i) \log q_i(h_i) + C$$
$$= \sum_{h_i} q_i(h_i) \left[\log q_i(h_i) - \sum_{pah_i} q_{\text{pa}\, i}(h_{\text{pa}\, i}) \log \Pr(h_i \mid \text{pa}\, h_i) - \sum_{h_l} q_l(h_l) \log \Pr(h_l \mid h_i) - \sum_{h_r} q_r(h_r) \log \Pr(h_r \mid h_i) \right] + C$$
$$= \sum_{h_i} q_i(h_i) \left[\log q_i(h_i) - c_i \right] + C$$

where r,l,pai are the children and parent of i and the c_i are constant for changes in q_i.
Optimizing the q_i given the constraint for it being a distribution is easily solved using LaGrange multipliers:

$$\arg\min_{q_i} F_{MF} = \arg\min_{q_i} \sum_{h_i} q_i(h_i) \left[\log q_i(h_i) - c_i \right] \quad s.t. \sum_{h_i} q_i(h_i) = 1$$
$$\Rightarrow q_i \propto \exp(c_i)$$

Note that the exact same formula is true for any BN, we only have to recompute the constants c_i using all of the terms involving parents and children of i. The formula is also correct when working with a factor graph instead of a BN, replacing the conditional probabilities with the factor potentials, and considering all factors that are dependent on the variable i to compute the c_i constants.

LBP: We are trying to infer posteriors and likelihood for a factor graph
$P(x) = 1/Z \prod_a f_a(x_a)$
The LBP algorithm is defined by the message update rules:

$$m_{i \to a}[x_i] = \prod_{b \in N(i) \setminus a} m_{b \to i}[x_i]$$
$$m_{a \to i}[x_i] = \sum_{x_a \setminus x_i} \left[f_a(x_a) \prod_{j \in N(a) \setminus i} m_{j \to a}[x_i] \right]$$

Where messages are initialized to arbitrary values and are updated iteratively until possible convergence to a fixed point (which is not guaranteed). The update rules give rise to *beliefs* on variables or factor's variables:

$$b_i[x_i] = \prod_{a \in N(i)} m_{a \to i}[x_i]$$

$$b_a[x_a] = f_a(x_a) \prod_{i \in N(a)} m_{i \to a}[x_i]$$

The message update rules can in fact be extracted from the beliefs formula and a requirement for marginalization of factor beliefs over variable beliefs:

$$b_i(x_i) = \sum_{x_a \setminus x_i} b_a(x_a)$$

To see the equivalence just equate the variable belief (left) to the marginalization of the factor beliefs (expressed in terms of variable beliefs):

$$\prod_{d \in N(i)} m_{d \to i}(x_i) = b_i(x_i) = \sum_{x_a - x_i} f_a(x_a) \prod_{j \in N(a)} \left(\prod_{c \in N(j) \setminus a} m_{c \to j}(x_j) \right)$$

$$m_{a \to i}(x_i) \left(\prod_{d \in N(i) \setminus a} m_{d \to i}(x_i) \right) = \sum_{x_a \setminus x_i} f_a(x_a) \left(\prod_{d \in N(i) \setminus a} m_{d \to i}(x_i) \right) \prod_{j \in N(a) \setminus i} \left(\prod_{c \in N(j) \setminus a} m_{c \to j}(x_j) \right)$$

$$m_{a \to i}(x_i) = \sum_{x_a \setminus x_i} f_a(x_a) \prod_{j \in N(a) \setminus i} \left(\prod_{c \in N(j) \setminus a} m_{c \to j}(x_j) \right) = \sum_{x_a \setminus x_i} f_a(x_a) \prod_{j \in N(a) \setminus i} m_{j \to a}$$

We note that as shown in class, the LBP fixed points (where the algorithm is converging) are equivalent to local optima of the Bethe free energy:

$$F_{BETHE} = -\sum_a \sum_{x_a} b_a(x_a) \log \frac{f_a(x_a)}{b_a(x_a)} + \sum_i (d_i - 1) \sum_{x_i} b_i(x_i) \log b_i(x_i)$$

We will not use Bethe theory and its generalizations in the exam.