Notes on Simulating Mutants: Efficient MSM-based Sampling Strategies

Robert T. McGibbon

I. INTRODUCTION

Mutational analysis is one of the central tools in experimental protein science, but remains a challenge for simulation. Using standard simulation approaches, the study of a protein and a single mutant requires 2x the computational effort of studying just the original protein. Can we use MSMs to do better?

Assume that we've run extensive, converged simulation of a protein, A, and we build an MSM. We now want to run a mutant, A'. We assume that A, and A' share a common state space – the interest is in how the mutation affects the transition probabilities.

If we assume that the mutation affects the transition matrix in a "small" way – that the mutation is appropriately classified as a perturbation, then it should be possible to "win". First, short trajectories (of length equal to a single lag-time) are sufficient, because (by assumption), we don't have to discover any new states. We just have to estimate the perturbed transition probabilities. Second, because the mutation is small, there's a lot of mutual information between transition probabilities in A and those in A'. We're not starting from scratch here.

II. PROBLEM STATEMENT

What is the mathematical model for the cross-talk between A and A'? Our observation of A must basically be the prior for A'.

Consider a row of the transition matrix $T^{A'}$, the transition probabilities leaving state i, $\vec{p}_i^{A'}$. In the absence of any simulation data on A', what is our prior distribution on $P(\mathbf{p_i}^{A'})$?

The simplest idea is that the $\vec{p}_i^{A'}$ are $\mathrm{Dir}(\alpha)$, where the α parameters (effectively the psuedocounts) are determined by the number of observed counts in protein A, $\vec{c}_i^{A}$. If there is a lot of mutual information between A and A', then the α should be equal to $\vec{c}_i^{A}$. This formalizes the idea that our predictions about protein A' made from data collected on A are equally confident as our predictions about protein A using the same data collected on A. But if there's not a lot of mutual information between the two proteins, then our prior on $\vec{p}_i^{A'}$ should be non-informative.

$$\vec{p}_i^{A'} \sim \mathrm{Dir}(q_i \cdot \vec{c}_i^{A} + 1/2)$$

Here, the parameter $q_i \in [0, 1]$ gives something like the expected strength of the information transfer between state i in the the two models. When $q_i = 1$, a count measured in A is "worth it's weight" in the A' model. But as $q_i \to 0$, those counts are worthless in A' and we get a noninformative Jeffreys prior.

This statement of the problems is nice because it takes into account some uncertainty in the gold-standard model for A. It doesn't just look the MLE estimate of the transition matrix in A, but uses the counts directly to parameterize the distribution over A'. So for regions of the state space where $\vec{c}_i^{A'}$ is low, we're going to get a mostly uninformative prior naturally.

One question: should q be a single parameter for the whole model, or should every state have its own q_i? The later case could encode the idea that some of the states behave very similarly in A vs. A', whereas other might be very different.

Next: clearly q_i should not be a fixed parameter. It should also be a random variable, estimated from the data. Perhaps the appropriate prior on q_i is $q_i \sim \mathrm{Beta}(\alpha, \beta)$.

Now, if observe some outbound counts, $\vec{K}$, from state i in simulations of the mutant protein A' (note for consistency, we could notate $\vec{K}$ as $\vec{c}_i^{A'}$ instead), they are distributed as a multinomial with parameters $\vec{p}_i^{A'}$. The model specified is then:

$$\begin{aligned}
\vec{c}_i^{A} &= \text{prior observed counts in protein } A \\
\alpha, \beta &= q\text{'s hyperparameters} \\
q_i &\sim \mathrm{Beta}(\alpha, \beta) \\
\vec{p}_i^{A'} &\sim \mathrm{Dirichlet}(q \cdot \vec{c}_i^{A} + 1/2) \\
K &\sim \mathrm{Multinomial}(\vec{p}_i^{A'})
\end{aligned}$$

Can we do inference here? I think so. Let's start with the joint distribution of of $\vec{p}_i^{A'}$ and q_i. By bayes rule,

$$P(q_i, \vec{p}_i^{A'} \mid \vec{K}) \propto P(\vec{K} \mid q, \vec{p}_i^{A'})\ P(q_i, \vec{p}_i^{A'})$$

K is conditionally independent of q_i given $\vec{p}_i^{A'}$, and is multinomial. $P(q_i, \vec{p}_i^{A'})$ factors as $P(\vec{p}_i^{A'} \mid q_i) \cdot P(q_i)$, a Dirichlet times a Beta. Because of the conjugacy, $P(\vec{K} \mid q_i, \vec{p}_i^{A'})$ and $P(\vec{p}_i^{A'} \mid q_i)$ group together into an updated Dirichlet. Putting it all together, we get

$$P(q_i, \vec{p}_i^{A'} \mid \vec{K}) \propto \mathrm{Dir}(q_i \cdot \vec{c}_i^{A} + \vec{K} + 1/2) \cdot P_{\alpha,\beta}(q)$$

And then marginalizing out q_i, we have

$$P(\vec{p}_i^{A'} \mid \vec{K}) \propto \int_0^1 dq\ \mathrm{Dir}(q \cdot \vec{c}_i^{A} + \vec{K} + 1/2) \cdot P_{\alpha,\beta}(q_i)$$

With MCMC, I think sampling from this posterior is pretty straightforward. But what about an analytic solution? When I write out the integral explicitly, it looks pretty bad.

$$P(\vec{p}_i^{A'} = \vec{x} \,|\, \vec{K}) \propto \int_0^1 dq \, \frac{1}{Z(q_i)} \prod_{j=1} x_j^{q_i \cdot \vec{c}_{ij}^{A'} + \vec{K}_j + 1/2} q_i^{\alpha-1} (1-q_i)^{\beta-1}$$

$$Z(q_i) = \frac{1}{B(q \cdot \vec{c}_i^A + \vec{K} + 1/2) \cdot B(\alpha, \beta)}$$

But if instead we knew q directly – i.e. by just fixing at a certain value, then the distribution for $P(\vec{p}_i^{A'})$ would be simple. Maybe we do MCMC to sample over q, and then use analytic formulas given q...

III. ACTIVE LEARNING

We've done a little bit of simulation in A', and now we want to start some new simulations. From which state i should we start our next round of sampling?

We should choose a state i to simulate from that maximizes the expected K-L divergence between the revised posterior (after observing a new count) and the current posterior distribution based on the data we already have. This is basically choosing the state that maximizes the expected information gain.

The K-L divergence from one Dirichlet distribution, p, to another, q, is given `www.fil.ion.ucl.ac.uk/~wpenny/publications/densities.ps` as

$$D_{KL}(\lambda_q || \lambda_p) = \log \frac{\Gamma(\lambda_{qt})}{\Gamma(\lambda_{pt})} + \sum_{s=1}^{m} \log \frac{\Gamma(\lambda_p(s))}{\Gamma(\lambda_q(s))} + \sum_{s=1}^{m} [\lambda_q(s) - \lambda_p(s)] \, [\Psi(\lambda_q(s)) - \Psi(\lambda_{qt})]$$

Where λ_p and λ_q are the parameters of the two Dirichlet distributions, $\lambda_{qt} = \sum_{s=1}^{m} \lambda_q(s)$, $\lambda_{pt} = \sum_{s=1}^{m} \lambda_p(s)$, and m is the number of states.

But how about the expected K-L divergence after collecting an additional sample, L?

$$E[D_{KL}(\lambda_q || \lambda_p + \vec{L})] = ?$$

We going to have to sample over $\vec{L}$. Also, this analytic form for the K-L divergence is only right conditional on q_i, because it's only when conditioned on q_i that we have a Dirichlet distribution for $\vec{p}_i^{A'}$.

So here's the procedure: first, we run an MCMC sampler over this Beta-Dirichlet-Multinomial model, keeping track of the posterior distribution samples over $\vec{p}_i^{A'}$ and $\alpha = q_i \cdot \vec{c}_i^A + \vec{K} + 1/2$. For each of these samples, we simulate draws from the multinomial, make "virtual" updates to the model at fixed q, and check the K-L divergence.

Algorithm 1 MCMC Expected Information Gain

```
Sampled-KLs ← List()
for (p_i^{A'}, α) in MCMC posterior samples do
    for i in 1..M do
        Sample L from Mult(1, p_i^{A'})
        Append D_KL(α || α + L) to Sampled-KLs
    end for
end for
return MEAN(Sampled-KLs)
```

Note: there is a simplifying assumption built in here. It's that the additional sample L doesn't change q_i. I think this is okay, and it's *key* for the tractability because it lets us use the Dirichlet/multinomial conjugacy to update the α with the virtual count.

Then choose the state that maximized this expected information gain.

As an idea for further thought, there might other approaches that would avoid the need for any MCMC. If q_i was instead a fixed parameter, and it was estimated from the data by cross validation for instance. But I'm not sure how to get around the need to sample over L...