QCB 508 – Week 7

John D. Storey

Spring 2017

Contents

Numerical Methods for Likelihood

Challenges

Frequentist model:

$$X_1, X_2, \ldots, X_n \overset{\text{iid}}{\sim} F_{\boldsymbol{\theta}}$$

Bayesian model:

$$X_1, X_2, \ldots, X_n | \boldsymbol{\theta} \overset{\text{iid}}{\sim} F_{\boldsymbol{\theta}} \text{ and } \boldsymbol{\theta} \sim F_{\boldsymbol{\tau}}$$

Sometimes it's not possible to find formulas for $\hat{\boldsymbol{\theta}}_{\text{MLE}}$, $\hat{\boldsymbol{\theta}}_{\text{MAP}}$, $\text{E}[\boldsymbol{\theta}|\boldsymbol{x}]$, or $f(\boldsymbol{\theta}|\boldsymbol{x})$. We have to use numerical methods instead.

Approaches

We will discuss the following numerical approaches to likelihood based inference:

- Expectation-maximization (EM) algorithm
- Variational inference
- Markov chain Monte Carlo (MCMC)
  - Metropolis sampling
  - Metropolis-Hastings sampling
  - Gibbs sampling

Latent Variable Models

Definition

Latent variables (or hidden variables) are random variables that are present in the model, but unobserved.

We will denote latent variables by Z, and we will assume

$$(X_1, Z_1), (X_2, Z_2), \ldots, (X_n, Z_n) \overset{\text{iid}}{\sim} F_{\boldsymbol{\theta}}.$$

A realized value of Z is z, $\boldsymbol{Z} = (Z_1, Z_2, \ldots, Z_n)^T$, etc.

The EM algorithm and variational inference involve latent variables.

Bayesian models are a special case of latent variable models: the unobserved random parameters are latent variables.

Empirical Bayes Revisited

In the earlier EB example, we supposed that $X_i|\mu_i \sim \text{Normal}(\mu_i, 1)$ for $i = 1, 2, \ldots, n$ where these rv's are independent, and also that $\mu_i \overset{\text{iid}}{\sim} \text{Normal}(a, b^2)$.

The unobserved parameters $\mu_1, \mu_2, \ldots, \mu_n$ are latent variables. In this case, $\boldsymbol{\theta} = (a, b^2)$.

Normal Mixture Model

Suppose $X_1, X_2, \ldots, X_n \overset{\text{iid}}{\sim} F_{\boldsymbol{\theta}}$ where $\boldsymbol{\theta} = (\pi_1, \ldots, \pi_K, \mu_1, \ldots, \mu_K, \sigma_1^2, \ldots, \sigma_K^2)$ with pdf

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left\{ -\frac{(x_i - \mu_k)^2}{2\sigma_k^2} \right\}.$$

The MLEs of the unknown paramaters cannot be found analytically. This is a mixture common model to work with in applications, so we need to be able to estimate the parameters.

There is a latent variable model that produces the same maerginal distribution and likelihood function. Let $\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_n \overset{\text{iid}}{\sim} \text{Multinomial}_K(1, \boldsymbol{\pi})$ where $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$. Note that $Z_{ik} \in \{0, 1\}$ and $\sum_{k=1}^{K} Z_{ik} = 1$. Let $[X_i|Z_{ik} = 1] \sim \text{Normal}(\mu_k, \sigma_k^2)$, where $\{X_i|\boldsymbol{Z}_i\}_{i=1}^n$ are jointly independent.

The joint pdf is

$$f(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) = \prod_{i=1}^{n} \prod_{k=1}^{K} \left[\pi_k \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left\{ -\frac{(x_i - \mu_k)^2}{2\sigma_k^2} \right\} \right]^{z_{ik}}.$$

Note that

$$f(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) = \prod_{i=1}^{n} f(x_i, \boldsymbol{z}_i; \boldsymbol{\theta}).$$

It can be verified that $f(\boldsymbol{x}; \boldsymbol{\theta})$ is the marginal distribution of this latent variable model:

$$f(x_i; \boldsymbol{\theta}) = \sum_{\boldsymbol{z}_i} f(x_i, \boldsymbol{z}_i; \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left\{ -\frac{(x_i - \mu_k)^2}{2\sigma_k^2} \right\}.$$

Bernoulli Mixture Model

Suppose $X_1, X_2, \ldots, X_n \overset{\text{iid}}{\sim} F_{\boldsymbol{\theta}}$ where $\boldsymbol{\theta} = (\pi_1, \ldots, \pi_K, p_1, \ldots, p_K)$ with pdf

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k p_k^{x_i} (1 - p_k)^{1 - x_i}.$$

As in the Normal mixture model, the MLEs of the unknown paramaters cannot be found analytically.

As before, there is a latent variable model that produces the same maerginal distribution and likelihood function. Let $\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_n \overset{\text{iid}}{\sim} \text{Multinomial}_K(1, \boldsymbol{\pi})$ where $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$. Note that $Z_{ik} \in \{0, 1\}$ and $\sum_{k=1}^{K} Z_{ik} = 1$. Let $[X_i | Z_{ik} = 1] \sim \text{Bernoulli}(p_k)$, where $\{X_i | \boldsymbol{Z}_i\}_{i=1}^{n}$ are jointly independent.

The joint pdf is

$$f(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) = \prod_{i=1}^{n} \prod_{k=1}^{K} \left[p_k^{x_i} (1 - p_k)^{1 - x_i} \right]^{z_{ik}}.$$

EM Algorithm

Rationale

For any likelihood function, $L(\boldsymbol{\theta}; \boldsymbol{x}) = f(\boldsymbol{x}; \boldsymbol{\theta})$, there is an abundance of optimization methods that can be used to find the MLE or MAP. However:

- Optimization methods can be messy to implement
- There may be probabilistic structure that we can use to simplify the optimization process and also provide theoretical guarantees on its convergence
- Optimization isn't necessarily the only goal, but one may also be interested in point estimates of the latent variable values

Requirement

The expectation-maximization (EM) algorithm allows us to calculate MLEs and MAPs when certain geometric properties are satisfied in the probabilistic model.

In order for the EM algorithm to be a practical approach, then we should have a latent variable model $f(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta})$ that is used to do inference on $f(\boldsymbol{x}; \boldsymbol{\theta})$ or $f(\boldsymbol{\theta}|\boldsymbol{x})$.

Note: Sometimes $(\boldsymbol{x}, \boldsymbol{z})$ is called the **complete data** and $\boldsymbol{x}$ is called the **observed data** when we are using the EM as a method for dealing with missing data.

The Algorithm

1. Choose initial value $\boldsymbol{\theta}^{(0)}$
2. Calculate $f(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)})$
3. Calculate
$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) = \mathrm{E}_{\boldsymbol{Z}|\boldsymbol{X}=\boldsymbol{x}}\left[\log f(\boldsymbol{x}, \boldsymbol{Z}; \boldsymbol{\theta}); \boldsymbol{\theta}^{(t)}\right]$$
4. Set
$$\boldsymbol{\theta}^{(t+1)} = \mathrm{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$$
5. Iterate until convergence and set $\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}^{(\infty)}$

$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$

Continuous $\boldsymbol{Z}$:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) = \int \log f(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) f(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{(t)}) d\boldsymbol{z}$$

Discrete $\boldsymbol{Z}$:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) = \sum_{\boldsymbol{z}} \log f(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) f(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{(t)})$$

EM for MAP

If we wish to calculate the MAP we replace $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$ with

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) = \mathrm{E}_{\boldsymbol{Z}|\boldsymbol{X}=\boldsymbol{x}}\left[\log f(\boldsymbol{x}, \boldsymbol{Z}; \boldsymbol{\theta}); \boldsymbol{\theta}^{(t)}\right] + \log f(\boldsymbol{\theta})$$

where $f(\boldsymbol{\theta})$ is the prior distribution on $\boldsymbol{\theta}$.

EM Examples

Normal Mixture Model

Returning to the Normal mixture model introduced earlier, we first calculate

$$\log f(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) = \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik} \log \pi_k + z_{ik} \log \phi(x_i; \mu_k, \sigma_k^2)$$

where

$$\phi(x_i; \mu_k, \sigma_k^2) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left\{-\frac{(x_i - \mu_k)^2}{2\sigma_k^2}\right\}.$$

In caculating

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) = \mathrm{E}_{\boldsymbol{Z}|\boldsymbol{X}=\boldsymbol{x}}\left[\log f(\boldsymbol{x}, \boldsymbol{Z}; \boldsymbol{\theta}); \boldsymbol{\theta}^{(t)}\right]$$

we only need to know $\mathrm{E}_{\boldsymbol{Z}|\boldsymbol{X}=\boldsymbol{x}}[Z_{ik}|\boldsymbol{x}; \boldsymbol{\theta}]$, which turns out to be

$$\mathrm{E}_{\boldsymbol{Z}|\boldsymbol{X}=\boldsymbol{x}}[Z_{ik}|\boldsymbol{x}; \boldsymbol{\theta}] = \frac{\pi_k \phi(x_i; \mu_k, \sigma_k^2)}{\sum_{j=1}^{K} \pi_j \phi(x_i; \mu_j, \sigma_j^2)}.$$

Note that we take

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) = \mathrm{E}_{\boldsymbol{Z}|\boldsymbol{X}=\boldsymbol{x}}\left[\log f(\boldsymbol{x}, \boldsymbol{Z}; \boldsymbol{\theta}); \boldsymbol{\theta}^{(t)}\right]$$

so the parameter in $\log f(\boldsymbol{x}, \boldsymbol{Z}; \boldsymbol{\theta})$ is a free $\boldsymbol{\theta}$, but the paramaters used to take the conditional expectation of $\boldsymbol{Z}$ are fixed at $\boldsymbol{\theta}^{(t)}$. Let's define

$$\hat{z}_{ik}^{(t)} = \mathrm{E}\left[z_{ik}|\boldsymbol{x}; \boldsymbol{\theta}^{(t)}\right] = \frac{\pi_k^{(t)} \phi(x_i; \mu_k^{(t)}, \sigma_k^{2,(t)})}{\sum_{j=1}^{K} \pi_j^{(t)} \phi(x_i; \mu_j^{(t)}, \sigma_j^{2,(t)})}.$$

E-Step

We calculate

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) = \mathrm{E}_{\boldsymbol{Z}|\boldsymbol{X}=\boldsymbol{x}} \left[\log f(\boldsymbol{x}, \boldsymbol{Z}; \boldsymbol{\theta}); \boldsymbol{\theta}^{(t)} \right]$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} \hat{z}_{ik}^{(t)} \log \pi_k + \hat{z}_{ik}^{(t)} \log \phi(x_i; \mu_k, \sigma_k^2)$$

At this point the parameters making up $\hat{z}_{ik}^{(t)}$ are fixed at $\boldsymbol{\theta}^{(t)}$.

M-Step

We now caculate $\boldsymbol{\theta}^{(t+1)} = \mathrm{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}$, which yields:

$$\pi_k^{(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)}}{n}$$

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)} x_i}{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)}}$$

$$\sigma_k^{2,(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)} \left(x_i - \mu_k^{(t+1)} \right)^2}{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)}}$$

Note: You need to use a Lagrange multiplier to obtain $\{\pi_k^{(t+1)}\}_{k=1}^{K}$.

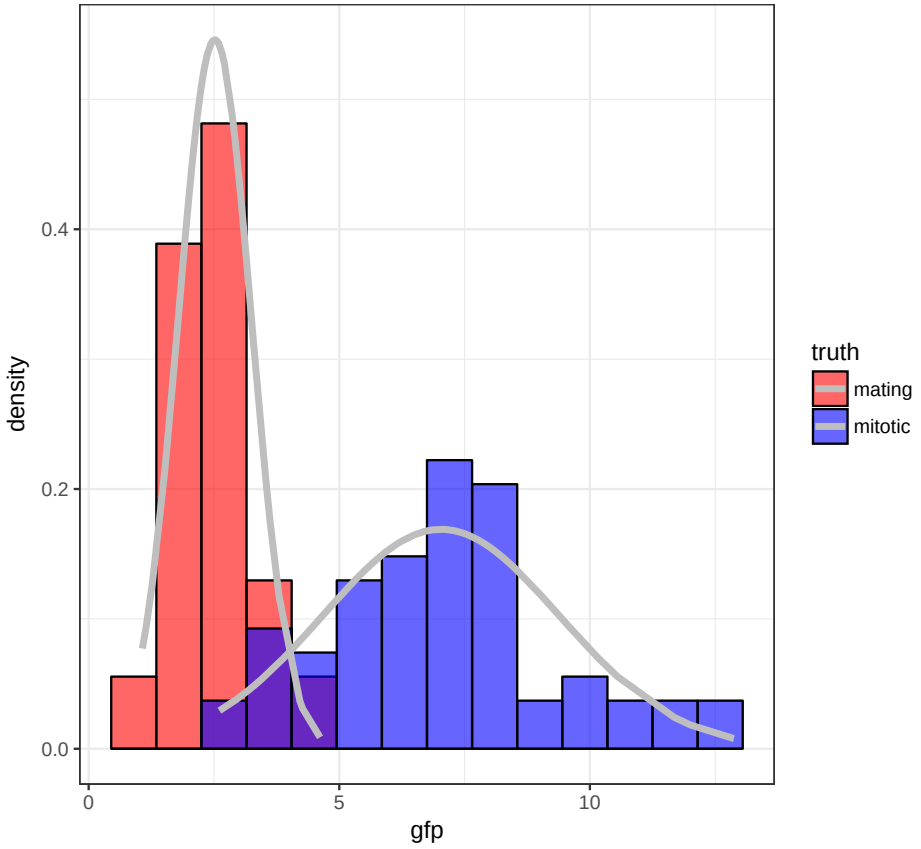
Caveat

If we assign one and only one data point to mixture component k, meaning $\mu_k^{(t)} = x_i$ and $\hat{z}_{ik}^{(t)} = 1$ for some k and i, then as $\sigma_k^{2,(t)} \to 0$, the likelihood goes to ∞.

Therefore, when implementing the EM algorithm for this particular Normal mixture model, we have to be careful to bound all $\sigma_k^{2,(t)}$ away from zero and avoid this scenario.

Yeast Gene Expression

Measured ratios of the nuclear to cytoplasmic fluorescence for a protein-GFP construct that is hypothesized as being nuclear in mitotic cells and largely cytoplasmic in mating cells.



Initialize Values

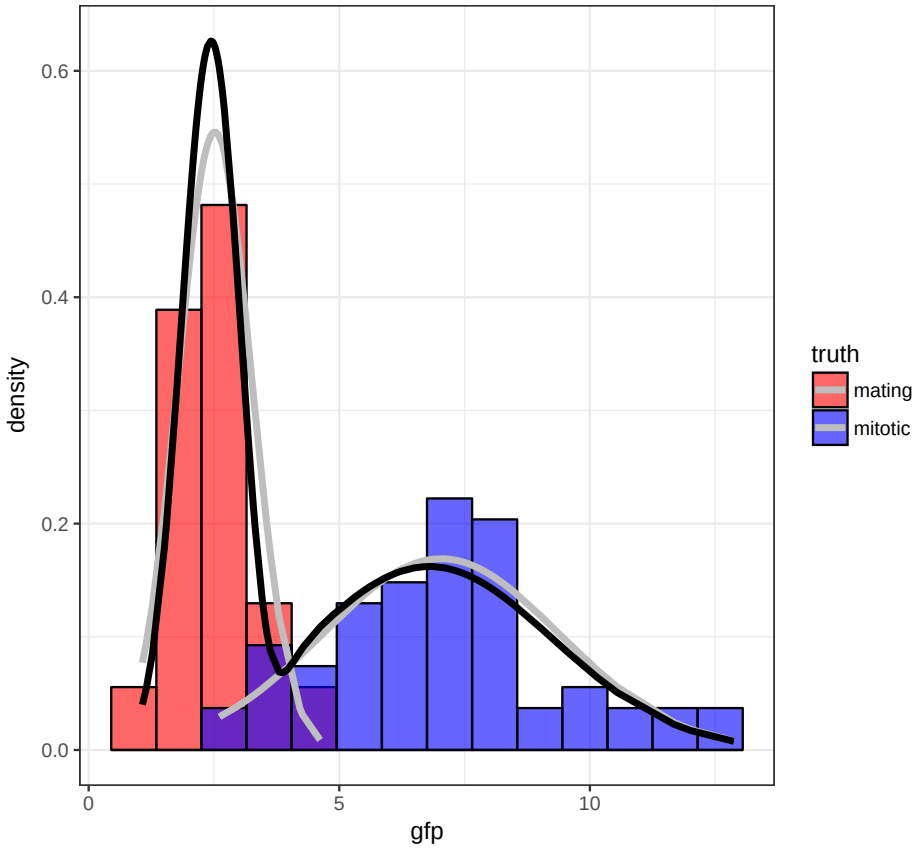
```
> set.seed(508)
> B <- 100
> p <- rep(0,B)
> mu1 <- rep(0,B)
> mu2 <- rep(0,B)
> s1 <- rep(0,B)
> s2 <- rep(0,B)
> p[1] <- runif(1, min=0.1, max=0.9)
> mu.start <- sample(x, size=2, replace=FALSE)
```

```
> mu1[1] <- min(mu.start)
> mu2[1] <- max(mu.start)
> s1[1] <- var(sort(x)[1:60])
> s2[1] <- var(sort(x)[61:120])
> z <- rep(0,120)
```

Run EM Algorithm

```
> for(i in 2:B) {
+   z <- (p[i-1]*dnorm(x, mean=mu2[i-1], sd=sqrt(s2[i-1])))/
+     (p[i-1]*dnorm(x, mean=mu2[i-1], sd=sqrt(s2[i-1])) +
+        (1-p[i-1])*dnorm(x, mean=mu1[i-1], sd=sqrt(s1[i-1])))
+   mu1[i] <- sum((1-z)*x)/sum(1-z)
+   mu2[i] <- sum(z*x)/sum(z)
+   s1[i] <- sum((1-z)*(x-mu1[i])^2)/sum(1-z)
+   s2[i] <- sum(z*(x-mu2[i])^2)/sum(z)
+   p[i] <- sum(z)/length(z)
+ }
>
> tail(cbind(mu1, s1, mu2, s2, p), n=3)
            mu1        s1    mu2       s2         p
 [98,] 2.455325 0.3637967 6.7952 6.058291 0.5340015
 [99,] 2.455325 0.3637967 6.7952 6.058291 0.5340015
[100,] 2.455325 0.3637967 6.7952 6.058291 0.5340015
```

Fitted Mixture Distribution



Bernoulli Mixture Model

As an exercise, derive the EM algorithm of the Bernoilli mixture model introduced earlier.

Hint: Replace $\phi(x_i; \mu_k, \sigma_k^2)$ with the appropriate Bernoilli pmf.

Other Applications of EM

- Dealing with missing data
- Multiple imputation of missing data
- Truncated observations
- Bayesian hyperparameter estimation
- Hidden Markov models

Theory of EM

Decomposition

Let $q(\boldsymbol{z})$ be a probability distribution on the latent variables, $\boldsymbol{z}$. Consider the following decomposition:

$$\log f(\boldsymbol{x};\boldsymbol{\theta}) = \mathcal{L}(q(\boldsymbol{z}),\boldsymbol{\theta}) + \mathrm{KL}(q(\boldsymbol{z})\|f(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}))$$

where

$$\mathcal{L}(q(\boldsymbol{z}),\boldsymbol{\theta}) = \int q(\boldsymbol{z}) \log\left(\frac{f(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})}\right) d\boldsymbol{z}$$

$$\mathrm{KL}(q(\boldsymbol{z})\|f(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})) = -\int q(\boldsymbol{z}) \log\left(\frac{f(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})}{q(\boldsymbol{z})}\right) d\boldsymbol{z}$$

Kullback-Leibler Divergence

The KL divergence provides an asymmetric measure of the difference between two probability distributions.

The KL divergence is such that $\mathrm{KL}(q\|f) \geq 0$ where $\mathrm{KL}(q\|f) = 0$ if and only if $q = f$. This property is known as **Gibbs inequality**.

Lower Bound

Note that $\mathcal{L}(q(\boldsymbol{z}),\boldsymbol{\theta})$ provides a lower bound on the likelihood function:

$$\log f(\boldsymbol{x};\boldsymbol{\theta}) \geq \mathcal{L}(q(\boldsymbol{z}),\boldsymbol{\theta})$$

If we set $q(\boldsymbol{z}) = f(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})$, then for a fixed $\boldsymbol{\theta}^{(t)}$ and as a function of $\boldsymbol{\theta}$,

$$\begin{aligned} \mathcal{L}(q(\boldsymbol{z}),\boldsymbol{\theta}) &\propto \int f(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}) \log f(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta}) d\boldsymbol{z} && (1) \\ &= Q(\boldsymbol{\theta},\boldsymbol{\theta}^{(t)}) && (2) \end{aligned}$$

EM Increases Likelihood

Since $\boldsymbol{\theta}^{(t+1)} = \text{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$, it follows that

$$Q(\boldsymbol{\theta}^{(t+1)}, \boldsymbol{\theta}^{(t)}) \geq Q(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^{(t)}).$$

Also, by the properties of KL divergence stated above, we have

$$\text{KL}(f(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{(t+1)}) \| f(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{(t)})) \geq \text{KL}(f(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{(t)}) \| f(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{(t)})).$$

Putting these together we have

$$\log f(\boldsymbol{x}; \boldsymbol{\theta}^{(t+1)}) \geq \log f(\boldsymbol{x}; \boldsymbol{\theta}^{(t)}).$$

Variational Inference

Rationale

Performing the EM algorithm required us to be able to compute $f(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta})$ and also optimize $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$. Sometimes this is not possible. Variational inference takes advantage of the decomposition

$$\log f(\boldsymbol{x}; \boldsymbol{\theta}) = \mathcal{L}(q(\boldsymbol{z}), \boldsymbol{\theta}) + \text{KL}(q(\boldsymbol{z}) \| f(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}))$$

and instead considers other forms of $q(\boldsymbol{z})$ to identify a more tractable optimization.

Optimization Goal

Since

$$\log f(\boldsymbol{x}; \boldsymbol{\theta}) = \mathcal{L}(q(\boldsymbol{z}), \boldsymbol{\theta}) + \text{KL}(q(\boldsymbol{z}) \| f(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}))$$

it follows that the closer $q(\boldsymbol{z})$ is to $f(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta})$, the term $\mathcal{L}(q(\boldsymbol{z}), \boldsymbol{\theta})$ grows larger while $\text{KL}(q(\boldsymbol{z}) \| f(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}))$ becomes smaller. The goal is typically to identify a restricted form of $q(\boldsymbol{z})$ that maximizes $\mathcal{L}(q(\boldsymbol{z}), \boldsymbol{\theta})$, which serves as an approximation to the posterior distribution $f(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta})$.

Mean Field Approximation

A mean field approximation implies we restrict $q(\boldsymbol{z})$ to be

$$q(\boldsymbol{z}) = \prod_{k=1}^{K} q_k(\boldsymbol{z}_k)$$

for some partition $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_K)$. This partition is very context specific and is usually driven by the original model and what is tractable.

Optimal $q_k(\boldsymbol{z}_k)$

Under the above restriction, it can be shown that the $\{q_k(\boldsymbol{z}_k)\}$ that maximize $\mathcal{L}(q(\boldsymbol{z}), \boldsymbol{\theta})$ have the form:

$$q_k(\boldsymbol{z}_k) \propto \exp\left\{ \int \log f(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) \prod_{j \neq k} q_j(\boldsymbol{z}_j) d\boldsymbol{z}_j \right\}.$$

These pdf's or pmf's can be calculated iteratively by cycling over $k = 1, 2, \ldots, K$ after intializing them appropriately. Note that convergence is guaranteed.

Remarks

- If $\boldsymbol{\theta}$ is also random, then it can be included in $\boldsymbol{z}$.
- The estimated $\hat{f}(\boldsymbol{z}|\boldsymbol{x})$ is typically concentrated around the high density region of the true $f(\boldsymbol{z}|\boldsymbol{x})$, so it is useful for calculations such as the MAP, but it is not guaranteed to be a good overall estimate of $f(\boldsymbol{z}|\boldsymbol{x})$.
- Variational inference is typically faster than MCMC (covered next).
- Given this is an optimization procedure, care can be taken to speed up convergence and avoid unintended local maxima.

Markov Chain Monte Carlo

Motivation

When performing Bayesian inferece, it is often (but not always) possible to calculate

$$f(\boldsymbol{\theta}|\boldsymbol{x}) \propto L(\boldsymbol{\theta}; \boldsymbol{x}) f(\boldsymbol{\theta})$$

but it is typically much more difficult to calculate

$$f(\boldsymbol{\theta}|\boldsymbol{x}) = \frac{L(\boldsymbol{\theta};\boldsymbol{x})f(\boldsymbol{\theta})}{f(\boldsymbol{x})}.$$

Markov chain Monte Carlo is a method for simulating data approximately from $f(\boldsymbol{\theta}|\boldsymbol{x})$ with knowledge of only $L(\boldsymbol{\theta};\boldsymbol{x})f(\boldsymbol{\theta})$.

Note

MCMC can be used to approximately simulate data from any distribution that is only proportionally characterized, but it is probably most well know for doing so in the context of Bayesian infererence.

We will explain MCMC in the context of Bayesian inference.

Big Picture

We draw a Markov chain of $\boldsymbol{\theta}$ values so that, in some asymptotic sense, these are equivalent to iid draws from $f(\boldsymbol{\theta}|\boldsymbol{x})$.

The draws are done competitively so that the next draw of a realization of $\boldsymbol{\theta}$ depends on the current value.

The Markov chain is set up so that it only depends on $L(\boldsymbol{\theta};\boldsymbol{x})f(\boldsymbol{\theta})$.

A lot of practical decisions need to be made by the user, so utilize MCMC carefully.

Metropolis-Hastings Algorithm

1. Initialize $\boldsymbol{\theta}^{(0)}$
2. Generate $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(b)})$ for some pdf or pmf $q(\cdot|\cdot)$
3. With probablity

$$A(\boldsymbol{\theta}^*, \boldsymbol{\theta}^{(b)}) = \min\left(1, \frac{L(\boldsymbol{\theta}^*;\boldsymbol{x})f(\boldsymbol{\theta}^*)q(\boldsymbol{\theta}^{(b)}|\boldsymbol{\theta}^*)}{L(\boldsymbol{\theta}^{(b)};\boldsymbol{x})f(\boldsymbol{\theta}^{(b)})q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(b)})}\right)$$

   set $\boldsymbol{\theta}^{(b+1)} = \boldsymbol{\theta}^*$. Otherise, set $\boldsymbol{\theta}^{(b+1)} = \boldsymbol{\theta}^{(b)}$
4. Continue for $b = 1, 2, \ldots, B$ iterations and *carefully* select which $\boldsymbol{\theta}^{(b)}$ are utilized to approximate iid observations from $f(\boldsymbol{\theta}|\boldsymbol{x})$

Metropolis Algorithm

The Metropolis algorithm restricts $q(\cdot,\cdot)$ to be symmetric so that $q(\boldsymbol{\theta}^{(b)}|\boldsymbol{\theta}^*) = q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(b)})$ and

$$A(\boldsymbol{\theta}^*, \boldsymbol{\theta}^{(b)}) = \min\left(1, \frac{L(\boldsymbol{\theta}^*;\boldsymbol{x})f(\boldsymbol{\theta}^*)}{L(\boldsymbol{\theta}^{(b)};\boldsymbol{x})f(\boldsymbol{\theta}^{(b)})}\right).$$

Utilizing MCMC Output

Two common uses of the output from MCMC are as follows:

1. $\mathrm{E}[f(\boldsymbol{\theta})|\boldsymbol{x}]$ is approximated by

$$\hat{\mathrm{E}}[f(\boldsymbol{\theta})|\boldsymbol{x}] = \frac{1}{B}\sum_{b=1}^{B} f\left(\boldsymbol{\theta}^{(b)}\right).$$

2. Some subsequence $\boldsymbol{\theta}^{(b_1)}, \boldsymbol{\theta}^{(b_2)}, \ldots, \boldsymbol{\theta}^{(b_m)}$ from $\left\{\boldsymbol{\theta}^{(b)}\right\}_{b=1}^{B}$ is utilized as an empirical approximation to iid draws from $f(\boldsymbol{\theta}|\boldsymbol{x})$.

Remarks

- The random draw $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(b)})$ perturbs the current value $\boldsymbol{\theta}^{(b)}$ to the next value $\boldsymbol{\theta}^{(b+1)}$. It is often a Normal distribution for continuous $\boldsymbol{\theta}$.
- Choosing the variance of $q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(b)})$ is important as it requires enough variance for the theory to be applicable within a reasonable number of computations, but it cannot be so large that new values of $\boldsymbol{\theta}^{(b+1)}$ are rarely generated.
- $A(\boldsymbol{\theta}^*, \boldsymbol{\theta}^{(b)})$ is called the acceptance probability.
- The algorithm must be run for a certain number of iterations ("burn in") before observed $\boldsymbol{\theta}^{(b)}$ can be utilized.
- The generated $\boldsymbol{\theta}^{(b)}$ are typically "thinned" (only sampled every so often) to reduce Markov dependence.

Full Conditionals

Suppose that $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_K)$. Define the subset vector as $\boldsymbol{\theta}_{a:b} = (\theta_a, \theta_{a+1}, \ldots, \theta_{b-1}, \theta_b)$ for any $1 \le a \le b \le K$.

The full conditional of θ_k is

$$\Pr(\theta_k|\boldsymbol{\theta}_{1:k-1}, \boldsymbol{\theta}_{k+1:K}, \boldsymbol{x})$$

Gibbs Sampling

Gibbs sampling a special type of Metropolis-Hasting MCMC. The algorithm samples one coordinate of $\boldsymbol{\theta}$ at a time.

1. Initialize $\boldsymbol{\theta}^{(0)}$.
2. Sample:
   $\theta_1^{(b+1)} \sim \Pr(\theta_1 | \boldsymbol{\theta}_{2:K}^{(b)}, \boldsymbol{x})$
   $\theta_2^{(b+1)} \sim \Pr(\theta_2 | \theta_1^{(b+1)}, \boldsymbol{\theta}_{3:K}^{(b)}, \boldsymbol{x})$
   $\theta_3^{(b+1)} \sim \Pr(\theta_3 | \boldsymbol{\theta}_{1:2}^{(b+1)}, \boldsymbol{\theta}_{3:K}^{(b)}, \boldsymbol{x})$
   $\vdots$
   $\theta_K^{(b+1)} \sim \Pr(\theta_K | \boldsymbol{\theta}_{1:K-1}^{(b+1)}, \boldsymbol{x})$
3. Continue for $b = 1, 2, \ldots, B$ iterations.

Gibbs and MH

As an exercise, show that Gibbs sampling is a special case of the Metropolis-Hastings algorithm where $A(\boldsymbol{\theta}^*, \boldsymbol{\theta}^{(b)}) = 1$.

Latent Variables

Note that MCMC is often used to calculate a posterior distribution on latent variables.

This makes sense because unobserved random paramaters are a special type of latent variable.

Theory

The goal of MCMC is to construct a Markov chain that converges to a stationary distribution that is equivalent to the target probability distribution.

Under reasonably general assumptions, one can show that the Metropolis-Hastings algorithm produces a Markov chain that is *homogeneous* and achieves *detailed balance*, which implies the Markov chain is *ergodic* so that $\boldsymbol{\theta}^{(B)}$ converges in distribution to $f(\boldsymbol{\theta}|\boldsymbol{x})$ as $B \to \infty$ and that

$$\hat{\mathrm{E}}[f(\boldsymbol{\theta})|\boldsymbol{x}] = \frac{1}{B}\sum_{b=1}^{B} f\left(\boldsymbol{\theta}^{(b)}\right) \overset{B\to\infty}{\longrightarrow} \mathrm{E}[f(\boldsymbol{\theta})|\boldsymbol{x}].$$

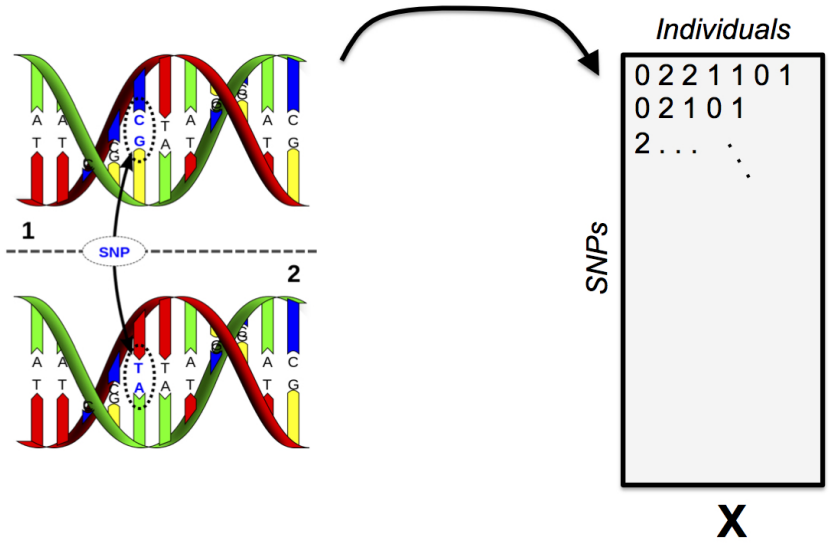
Software

Stan is probably the currently most popular software for doing Bayesian computation, including MCMC and variational inference.

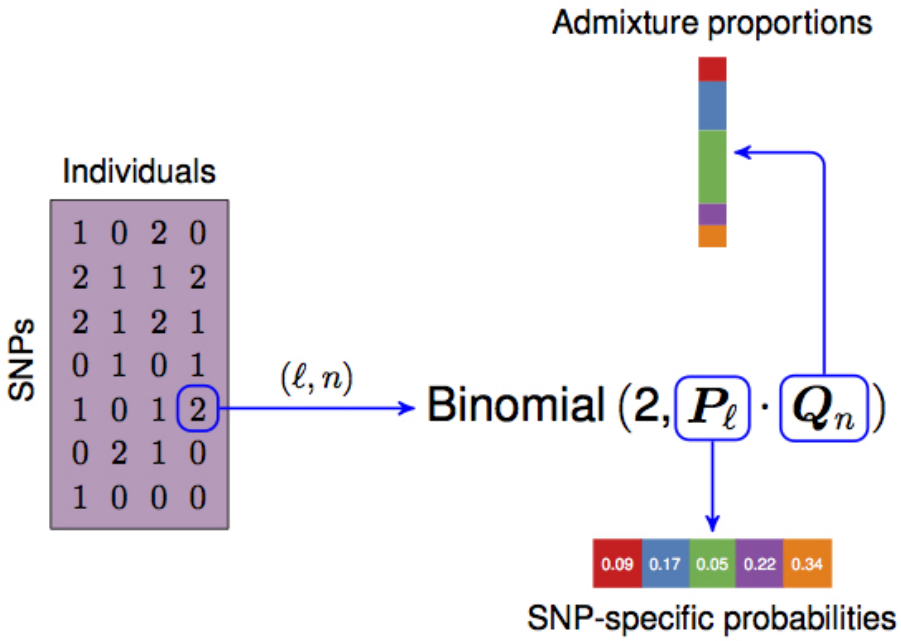
There are also popular R packages, such as `MCMCpack`.

MCMC Example

Single Nucleotide Polymorphisms



PSD Admixture Model



PSD model proposed in Pritchard, Stephens, Donnelly (2000) *Genetics*.

Gibbs Sampling Approach

The Bayesian Gibbs sampling approach to inferring the PSD model touches on many important ideas, such as conjugate priors and mixture models.

We will focus on a version of this model for diploid SNPs.

The Data

$\boldsymbol{X}$, a $L \times N$ matrix consisting of the genotypes, coded as $0, 1, 2$. Each row is a SNP, each column is an individual.

In order for this model to work, the data needs to be broken down into "phased" genotypes. For the 0 and 2 cases, it's obvious how to do this, and for the 1 case, it'll suffice for this model to randomly assign the alleles to chromosomes. We will explore phasing more on HW4.

Thus, we wind up with two $\{0, 1\}$ *binary* matrices $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$, both $L \times N$. We will refer to allele A and allele B. Note $\boldsymbol{X} = \boldsymbol{X}_A + \boldsymbol{X}_B$.

Model Components

- K, the number of populations that we model the genotypes as admixtures of. This is chosen before inference.
- $\boldsymbol{Q}$, a $N \times K$ matrix, the admixture proportions, values are in the interval $[0, 1]$ and rows are constrained to sum to 1.
- $\boldsymbol{P}$, a $L \times K$ matrix, the allele frequencies for each population, values are in the interval $[0, 1]$.
- $\boldsymbol{Z}_A$ and $\boldsymbol{Z}_B$, two $L \times N$ matrices that tell us which population the respective allele is from. Elements consist of the integers between 1 and K. This is a hidden variable.

The Model

- Each allele (elements of $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$) is a Bernoulli random variable, with success probability determined by which population that allele is assigned to (i.e., depends on $\boldsymbol{Z}_A$, $\boldsymbol{Z}_B$, and $\boldsymbol{P}$).
- We put a uniform Beta prior, i.e., Beta$(1, 1)$, on each element of $\boldsymbol{P}$.
- We put a uniform Dirichlet prior, i.e., Dirichlet$(1, \ldots, 1)$, on each *row* of $\boldsymbol{Q}$.
- $\boldsymbol{Z}_A$ and $\boldsymbol{Z}_B$ are K-class Multinomial draws where the probability of drawing each class is determined by each row of Q.

Conditional Independence

The key observation is to understand which parts of the model are dependent on each other in the data generating process.

- The data $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$ depends directly on $\boldsymbol{Z}_A$, $\boldsymbol{Z}_B$, and $\boldsymbol{P}$ (not $\boldsymbol{Q}$!).
- The latent variable $\boldsymbol{Z}_A$ and $\boldsymbol{Z}_B$ depend only on $\boldsymbol{Q}$ and they're conditionally independent given $\boldsymbol{Q}$.
- $\boldsymbol{Q}$ and $\boldsymbol{P}$ depend only on their priors.

$$\Pr(\boldsymbol{X}_A, \boldsymbol{X}_B, \boldsymbol{Z}_A, \boldsymbol{Z}_B, \boldsymbol{P}, \boldsymbol{Q}) =$$
$$\Pr(\boldsymbol{X}_A, \boldsymbol{X}_B | \boldsymbol{Z}_A, \boldsymbol{Z}_B, \boldsymbol{P}) \Pr(\boldsymbol{Z}_A | \boldsymbol{Q}) \Pr(\boldsymbol{Z}_B | \boldsymbol{Q}) \Pr(\boldsymbol{P}) \Pr(\boldsymbol{Q})$$

The Posterior

We desire to compute the posterior distribution $\Pr(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{Z}_A, \boldsymbol{Z}_B | \boldsymbol{X}_A, \boldsymbol{X}_B)$. Gibbs sampling tells us if we can construct conditional distributions for each random variable in our model, then iteratively sampling and updating our model parameters will result in a stationary distribution that is the same as the posterior distribution.

Gibbs sampling is an extremely powerful approach for this model because we can utilize conjugate priors as well as the independence of various parameters in the model to compute these conditional distributions.

Full Conditional for $\boldsymbol{Q}$

Note that $\boldsymbol{Z}_A$ and $\boldsymbol{Z}_B$ are the only parts of this model that directly depend on $\boldsymbol{Q}$.

$$\begin{aligned}
&\Pr(Q_n|\boldsymbol{Q}_{-n}, \boldsymbol{Z}_A, \boldsymbol{Z}_B, \boldsymbol{P}, \boldsymbol{X}_A, \boldsymbol{X}_B) \\
=&\Pr(Q_n|\boldsymbol{Z}_A, \boldsymbol{Z}_B) \\
\propto&\Pr(Z_{An}, Z_{Bn}|Q_n)\Pr(Q_n) \\
=&\Pr(Z_{An}|Q_n)\Pr(Z_{Bn}|Q_n)\Pr(Q_n) \\
\propto&\left(\prod_{\ell=1}^{L}\prod_{k=1}^{K} Q_{nk}^{\mathbb{1}(Z_{An\ell}=k)+\mathbb{1}(Z_{Bn\ell}=k)}\right)
\end{aligned}$$

$$= \prod_{k=1}^{K} Q_{nk}^{S_{nk}}$$

where S_{nk} is simply the count of the number of alleles for individual n that got assigned to population k.

Thus, $Q_n|\boldsymbol{Z}_A, \boldsymbol{Z}_B \sim \text{Dirichlet}(S_{j1}+1, \ldots, S_{jk}+1)$,.

We could have guessed that this distribution is Dirichlet given that $\boldsymbol{Z}_A$ and $\boldsymbol{Z}_B$ are multinomial! Let's use conjugacy to help us in the future.

Full Conditional for $\boldsymbol{P}$

$$\begin{aligned}
&\Pr(P_\ell|\boldsymbol{P}_{-\ell}, \boldsymbol{Z}_A, \boldsymbol{Z}_B, \boldsymbol{Q}, \boldsymbol{X}_A, \boldsymbol{X}_B) \\
\propto&\Pr(\boldsymbol{X}_{A\ell}, \boldsymbol{X}_{B\ell}|P_\ell, \boldsymbol{Z}_{A\ell}, \boldsymbol{Z}_{B\ell})\Pr(P_\ell)
\end{aligned}$$

We know $\Pr(\boldsymbol{X}_{A\ell}, \boldsymbol{X}_{B\ell}|P_\ell, \boldsymbol{Z}_{A\ell}, \boldsymbol{Z}_{B\ell})$ will be Bernoulli and $\Pr(P_\ell)$ will be beta, so the full conditional will be beta as well. In fact, the prior is uniform so it vanishes from the RHS.

Thus, all we have to worry about is the Bernoulli portion $\Pr(\boldsymbol{X}_{A\ell}, \boldsymbol{X}_{B\ell}|P_\ell, \boldsymbol{Z}_{A\ell}, \boldsymbol{Z}_{B\ell})$. Here, we observe that if the $\boldsymbol{Z}_A$ and $\boldsymbol{Z}_B$ are "known", then we known which value of P_ℓ to plug into our Bernoulli for $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$. Following the Week 6 lectures, we find that the full conditional for $\boldsymbol{P}$ is:

$$P_{\ell k}|\boldsymbol{Z}_A, \boldsymbol{Z}_B, \boldsymbol{X}_A, \boldsymbol{X}_B \sim \text{Beta}(1 + T_{\ell k0}, 1 + T_{\ell k1})$$

where $T_{\ell k0}$ is the total number of 0 alleles at SNP ℓ for population k, and $T_{\ell k1}$ is the analogous quantity for the 1 allele.

Full Conditional $\boldsymbol{Z}_A$ & $\boldsymbol{Z}_B$

We'll save some math by first noting that alleles A and B are independent of each other, so we can write this for only $\boldsymbol{Z}_A$ without losing any information. Also, all elements of $\boldsymbol{Z}_A$ are independent of each other. Further, note that each element of $\boldsymbol{Z}_A$ is a single multinomial draw, so we are working with a discrete random variable.

$$\begin{aligned}
&\Pr(Z_{A\ell n} = k|\boldsymbol{X}_A, \boldsymbol{Q}, \boldsymbol{P}) \\
=&\Pr(Z_{A\ell n} = k|X_{A\ell n}, Q_n, P_\ell) \\
\propto&\Pr(X_{A\ell n}|Z_{A\ell n} = k, Q_n, P_\ell)\Pr(Z_{A\ell n} = k|Q_n, P_\ell)
\end{aligned}$$

We can look at the two factors. First:

$$\Pr(Z_{A\ell n} = k|Q_n, P_\ell) = \Pr(Z_{A\ell n} = k|Q_n) = Q_{nk}$$

Then:

$$\Pr(X_{A\ell n}|Z_{A\ell n} = k, Q_n, P_\ell) = P_{\ell k}$$

Thus, we arrive at the formula:

$$\Pr(Z_{A\ell n} = k|\boldsymbol{X}_A, \boldsymbol{Q}, \boldsymbol{P}) \propto P_{\ell k}Q_{nk}$$

Gibbs Sampling Updates

It's neat that we wind up just iteratively counting the various discrete random variables along different dimensions.

$$\begin{aligned}
Q_n|\boldsymbol{Z}_A, \boldsymbol{Z}_B &\sim \text{Dirichlet}(S_{j1} + 1, \ldots, S_{jk} + 1) \\
P_{\ell k}|\boldsymbol{Z}_A, \boldsymbol{Z}_B, \boldsymbol{X}_A, \boldsymbol{X}_B &\sim \text{Beta}(1 + T_{\ell k0}, 1 + T_{\ell k1}) \\
Z_{A\ell n}|\boldsymbol{X}_A, \boldsymbol{Q}, \boldsymbol{P} &\sim \text{Multinomial}\left(\frac{P_\ell * Q_n}{P_\ell \cdot Q_n}\right)
\end{aligned}$$

where $*$ means element-wise vector multiplication.

Implementation

The Markov chain property means that we can't use vectorization forward in time, so R is not the best way to implement this algorithm.

That being said, we can vectorize the pieces that we can and demonstrate what happens.

Matrix-wise `rdirichlet` Function

Drawing from a Dirichlet is easy and vectorizable because it consists of normalizing independent gamma draws.

```
> rdirichlet <- function(alpha) {
+   m <- nrow(alpha)
+   n <- ncol(alpha)
+   x <- matrix(rgamma(m * n, alpha), ncol = n)
+   x/rowSums(x)
+ }
```

Inspect Data

```
> dim(Xa)
[1] 400  24
> X[1:3,1:3]
           NA18516 NA19138 NA19137
rs2051075        0       1       2
rs765546         2       2       0
rs10019399       2       2       2
> Xa[1:3,1:3]
           NA18516 NA19138 NA19137
rs2051075        0       0       1
rs765546         1       1       0
rs10019399       1       1       1
```

Model Parameters

```
> L <- nrow(Xa)
> N <- ncol(Xa)
>
> K <- 3
>
> Za <- matrix(sample(1:K, L*N, replace=TRUE), L, N)
```

```
> Zb <- matrix(sample(1:K, L*N, replace=TRUE), L, N)
> P <- matrix(0, L, K)
> Q <- matrix(0, N, K)
```

Update P

```
> update_P <- function() {
+   Na_0 <- Za * (Xa==0)
+   Na_1 <- Za * (Xa==1)
+   Nb_0 <- Zb * (Xb==0)
+   Nb_1 <- Zb * (Xb==1)
+   for(k in 1:K) {
+     N0 <- rowSums(Na_0==k)+rowSums(Nb_0==k)
+     N1 <- rowSums(Na_1==k)+rowSums(Nb_1==k)
+     P[,k] <- rdirichlet(1+cbind(N1, N0))[,1]
+   }
+   P
+ }
```

Update Q

```
> update_Q <- function() {
+   M_POP0 <- apply(Za, 2, function(x) {tabulate(x, nbins=K)} )
+   M_POP1 <- apply(Zb, 2, function(x) {tabulate(x, nbins=K)} )
+
+   rdirichlet(t(1+M_POP0+M_POP1))
+ }
```

Update (Each) Z

```
> update_Z <- function(X) {
+   Z <- matrix(0, nrow(X), ncol(X))
+   for(n in 1:N) {
+     PZ0 <- t(t((1-P)) * Q[n,])
+     PZ1 <- t(t(P) * Q[n,])
+     PZ <- X[,n]*PZ1 + (1-X[,n])*PZ0
+     Z[,n] <- apply(PZ, 1, function(p){sample(1:K, 1, prob=p)})
+   }
+   Z
+ }
```

Model Log-likelihood Function

```
> model_ll <- function() {
+   AFa <- t(sapply(1:L, function(i){P[i,][Za[i,]]}))
+   AFb <- t(sapply(1:L, function(i){P[i,][Zb[i,]]}))
+   # hint, hint, HW3
+   sum(dbinom(Xa, 1, AFa, log=TRUE)) +
+     sum(dbinom(Xb, 1, AFb, log=TRUE))
+ }
```

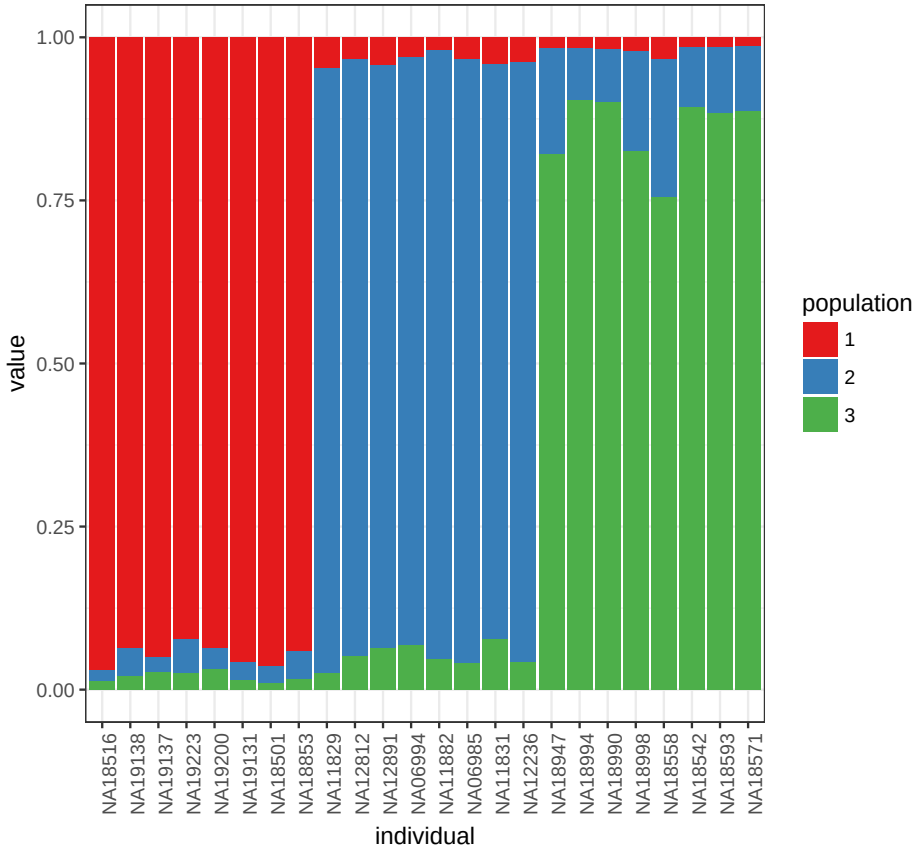
MCMC Configuration

```
> MAX_IT <- 20000
> BURNIN <- 5000
> THIN <- 20
>
> QSUM <- matrix(0, N, K)
>
> START <- 200
> TAIL <- 500
> LL_start <- rep(0, START)
> LL_end <- rep(0, TAIL)
```

Run Sampler

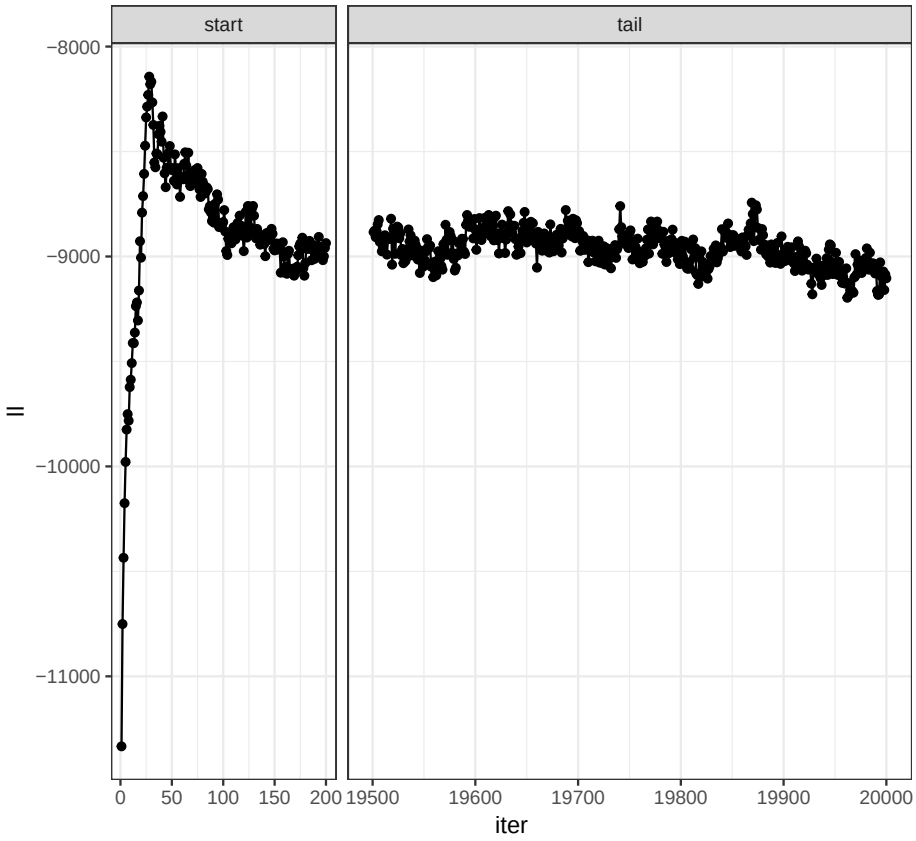
```
> set.seed(1234)
>
> for(it in 1:MAX_IT) {
+   P <- update_P()
+   Q <- update_Q()
+   Za <- update_Z(Xa)
+   Zb <- update_Z(Xb)
+
+   if(it > BURNIN && it %% THIN == 0) {QSUM <- QSUM+Q}
+   if(it <= START) {LL_start[it] <- model_ll()}
+   if(it > MAX_IT-TAIL) {LL_end[it-(MAX_IT-TAIL)] <- model_ll()}
+ }
>
> Q_MEAN <- QSUM/((MAX_IT-BURNIN)/THIN)
```

Posterior Mean of Q



Plot Log-likelihood Steps

Note both the needed burn-in and thinning.



What Happens for K=4?

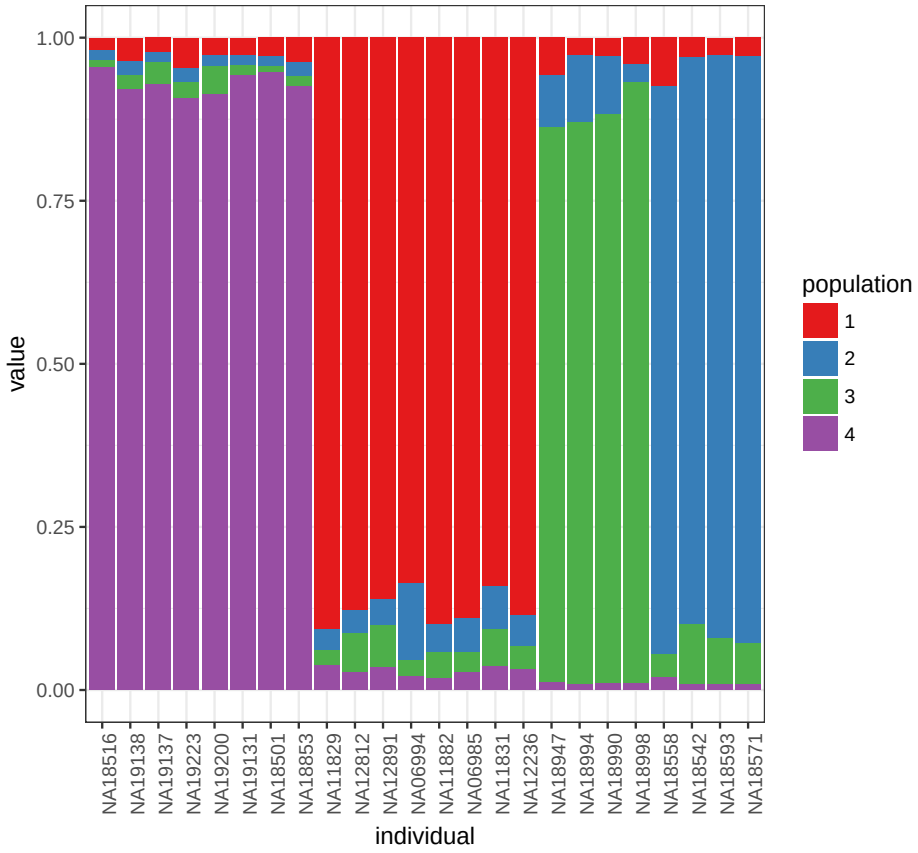
```
> K <- 4
> Za <- matrix(sample(1:K, L*N, replace=TRUE), L, N)
> Zb <- matrix(sample(1:K, L*N, replace=TRUE), L, N)
> P <- matrix(0, L, K)
> Q <- matrix(0, N, K)
> QSUM <- matrix(0, N, K)
```

Run Sampler Again

```
> for(it in 1:MAX_IT) {
+   P <- update_P()
+   Q <- update_Q()
+   Za <- update_Z(Xa)
```

```
+   Zb <- update_Z(Xb)
+
+   if(it > BURNIN && it %% THIN == 0) {
+     QSUM <- QSUM+Q
+   }
+ }
>
> Q_MEAN <- QSUM/((MAX_IT-BURNIN)/THIN)
```

Posterior Mean of Q



Further Reading

Bishop (2016)

One of the clearest treatments of the EM algorithm, variational inference, and MCMC can be found in Chapters 9-11 of *Pattern Recognition and Machine Learning*, by Christopher Bishop.

This is a great book in general!

EM Algorithm

Paper that popularized the method: Dempster, Laird, Rubin (1977)

Paper that got the theory correct: Wu (1983)

Variational Inference

Wainwright and Jordan (2008)

Ormerod and Wand (2010)

Blei et al. (2016)

MCMC

MCMC Without All the BS

Bayesian Data Analysis by Gelman et al.

Monte Carlo Strategies in Scientific Computing by Jun Liu

Extras

Source

License

Source Code

Session Information

```
> sessionInfo()
R version 3.3.2 (2016-10-31)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: macOS Sierra 10.12.3

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods
[7] base

other attached packages:
 [1] reshape2_1.4.2  dplyr_0.5.0     purrr_0.2.2
 [4] readr_1.0.0     tidyr_0.6.1     tibble_1.2
 [7] ggplot2_2.2.1   tidyverse_1.1.1 knitr_1.15.1
[10] magrittr_1.5    devtools_1.12.0

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.9         RColorBrewer_1.1-2 plyr_1.8.4
 [4] forcats_0.2.0       tools_3.3.2        digest_0.6.12
 [7] lubridate_1.6.0     jsonlite_1.2       evaluate_0.10
[10] memoise_1.0.0       nlme_3.1-131       gtable_0.2.0
[13] lattice_0.20-34     psych_1.6.12       DBI_0.5-1
[16] yaml_2.1.14         parallel_3.3.2     haven_1.0.0
[19] xml2_1.1.1          withr_1.0.2        stringr_1.1.0
[22] httr_1.2.1          hms_0.3            rprojroot_1.2
[25] grid_3.3.2          R6_2.2.0           readxl_0.1.1
[28] foreign_0.8-67      rmarkdown_1.3      modelr_0.1.0
[31] backports_1.0.5     scales_0.4.1       htmltools_0.3.5
[34] rvest_0.3.2         assertthat_0.1     mnormt_1.5-5
[37] colorspace_1.3-2    labeling_0.3       stringi_1.1.2
[40] lazyeval_0.2.0      munsell_0.4.3      broom_0.4.2
```