An SQP Method for QN2 Quasi Newton Acceleration of EM and Comparison With EM

Final Project
CSCI-GA.2945 Numerical Optimization
Justin Mao-Jones
New York University

December 14, 2014

Abstract

The EM algorithm is a popular method for computing maximum likelihood estimates from incomplete data. EM's tendency towards slow convergence prompted researches to develop EM "accelerators". I build upon the QN2 accelerator by Jamshidian and Jennrich [1997], which showed promising results. I show how QN2 construction of the mixture of two Poissons problem can be generalized to the well-known n-mixture problem using Sequential Quadratic Programming. I show that special initial conditions can maintain QN2 SQP equality constraints for mixture problems. Finally, I apply EM and QN2 SQP to the generalized n mixtures of Poissons problem and compare results. In these comparisons, I find that, despite the lure of superlinear convergence, QN2 SQP shows worse overall performance than EM due to line search failures. Here I use Strong Wolfe conditions with backtracking. I recommend that further work be performed in choosing a better line search strategy for QN2 SQP.

1 Introduction

In 1977, Dempster, Laird, and Rubin proposed the EM algorithm, which has since become, and continues to be, a popular method for computing maximum likelihood estimates from incomplete data.

A drawback of the EM algorithm is its tendency towards slow convergence. In certain practical applications, the EM algorithm has been shown to exhibit sublinear convergence. For examples see Lange [1995], Jamshidian and Jennrich [1993, 1997].

Modifications and extensions of the EM algorithm have been proposed to speed up convergence, and are often referred to as "accelerators". In this project, I focus on one particular accelerator, dubbed QN2, introduced by Jamshidian and Jennrich [1997]. In that paper, QN2 was applied to the problem of finding the maximum likelihood estimate

of a sample set derived from a mixture of 2 Poisson-distributed populations. That particular example, as well as other examples in Jamshidian and Jennrich [1997], illustrated significant improvements in calculation runtime by QN2 over EM.

Motivated by those findings, I formulated this project around building my own understanding of EM and QN2 through derivations, review of concepts, and computation. The QN2 has some very interesting properties that I will discuss.

My own contributions are as follows. I build upon the work by Jamshidian and Jennrich [1997] and show how QN2 construction of the mixture of two Poissons problem can be generalized to the well-known n-mixture problem using Sequential Quadratic Programming. I also investigate special properties of QN2 and show how selection of certain initial conditions can help maintain equality constraints within QN2. Finally, I apply EM and QN2 to the generalized n mixtures of Poissons problem and compare results, which turn out to be contrary to what I expected.

Before I continue, I would like to say that, while I claim some of the above as my own contributions, I have not performed an exhaustive literature search, and so some of these specific contributions may have already been done elsewhere.

The structure of this report begins by explaining the derivation of EM and QN2. For the sake of the interested reader, I include the major steps of the derivation of both the EM and QN2 algorithms, the outline of which mirrors the development of my own understanding of EM and QN2. Note that the derivation of EM is critical to understanding the formulation of QN2. I briefly cover relevant convergence concepts as well as some of the finer details of my implementation of QN2, such as line search methodology and termination criteria. Following that is a report on the computational experiments.

2 EM Algorithm

The EM algorithm, otherwise known as expectation-maximization, is a method for computing the maximum likelihood parameter estimates from incomplete data. Incomplete data is defined as a set of data in which some or all observations contain variables with missing data.

To use an illustrative example, consider a set of m randomly sampled observations $\{x_1, ..., x_m\}$. Suppose that we would like to explore the possibility that each of these observations comes from one of n poisson-distributed populations with unknown parameters $\theta = (\gamma_1, ..., \gamma_n, \lambda_1, ..., \lambda_n)^T, \theta \in \Re^{2n}$, where λ_r parameterizes the poisson probability density of population r and γ_r is the prior probability that x_i was chosen from population r. This construction is known as a mixture of poissons. Formulas are provided below.

We seek to estimate θ using maximum likelihood. However, we do not know which of the n populations the observation x_i corresponds to. Let $z_i \in \{1, ..., n\}$ denote the identity of the population that x_i belongs to. In this example, $\{x_1, ..., x_m\}$ is incomplete data, $\{z_1, ..., z_m\}$ is missing data, and $\{(x_1, z_1), ..., (x_m, z_m)\}$ is the complete (unknown) data.

The probability distribution of an incomplete observation x_i conditional on the knowl-

edge that $z_i = r$ is:

$$p(x_i = j|z_i = r, \theta) = \frac{e^{-\lambda_r}\lambda_r^j}{j!} \qquad \lambda_r \geq 0$$

The probability distribution of the missing data is:

$$p(z_i = r|\theta) = \gamma_r \qquad \text{where} \qquad \sum_{r=1}^{n} \gamma_r = 1 \qquad \text{and} \qquad 0 \leq \gamma_r \leq 1$$

The probability distribution of the complete data is:

$$p(x_i = j, z_i = r|\theta) = p(x_i = j|z_i = r, \theta)p(z_i = r|\theta) = \gamma_r \frac{e^{-\lambda_r}\lambda_r^j}{j!} \tag{1}$$

The probability distribution of the incomplete data is:

$$p(x_i = j|\theta) = \sum_{r=1}^{n} p(x_i = j, z_i = r|\theta) = \sum_{r=1}^{n} \gamma_r \frac{e^{-\lambda_r}\lambda_r^j}{j!}$$

We wish to compute the maximum likelihood estimates, and so we begin with the log likelihood function, which is defined as:

$$l(\theta) = \log(\prod_{i=1}^{m} p(x_i|\theta)) = \sum_{i=1}^{m} \log(p(x_i|\theta))$$

This is the point at which a method such as the EM algorithm becomes useful. The following is a derivation of the EM algorithm that can be found in McLachlan and Krishnan. Educational tutorials can also be found at:

- Andrew Ng Lecture Notes http://cs229.stanford.edu/notes/cs229-notes8.pdf
- Bilmes [1998]

Since z_i are unobserved, solving for the roots of the derivative of $l(\theta)$ can be an intractable problem:

$$\sum_{i=1}^{m} \log(p(x_i|\theta)) = \sum_{i=1}^{m} \log\left(\sum_{r=1}^{n} p(x_i, z_i|\theta))\right)$$

Fortunately, $l(\theta)$ can be converted into a more manageable form. Consider the probability distribution $p(\vec{z}|\vec{x}, \phi)$. Note that this function is parameterized by ϕ and not θ. The likelihood function can be deconstructed as follows:

$$l(\theta) = \log(p(\vec{x}|\theta)) = \int_{\vec{z}} p(\vec{z}|\vec{x}, \phi) \log(p(\vec{x}|\theta)) d\vec{z}$$

$$= E[\log(p(\vec{x}, \vec{z}|\theta))|\vec{x}, \phi] - E[\log(p(\vec{z}|\vec{x}, \theta))|\vec{x}, \phi]$$

$$= Q(\theta|\phi) - H(\theta|\phi) \tag{2}$$

where $Q(\theta|\phi)$ and $H(\theta|\phi)$ are defined as the left and right conditional expectations, respectively. In the words of Lange [1995], this derivation can seem "slightly mysterious". In order to shed some light on this result, I will show how it follows from our specific mixture of poissons problem. We will use the following identity, which follows from the assumption that our observations are independent:

$$1 = \int_{\vec{z}} p(\vec{z}|\vec{x}, \phi) d\vec{z} = \prod_{j=1}^{m} \int_{z_j} p(z_j|x_j, \phi) dz_j$$

The previous identity is true due to the identity $\int_{z_j} p(z_j|x_j, \phi) dz_j = 1$. Now, multiply $l(\theta)$ by $\int_{\vec{z}} p(\vec{z}|\vec{x}, \vec{\phi}) d\vec{z} = 1$:

$$l(\theta) = \left(\sum_{i=1}^{m} \log(p(x_i|\theta) \right) \prod_{j=1}^{m} \int_{z_j} p(z_j|x_j, \phi) dz_j$$

$$= \sum_{i=1}^{m} \left(\int_{z_i} p(z_i|x_i, \phi) \log(p(x_i|\theta) dz_i \prod_{j \neq i}^{m} \int_{z_j} p(z_j|x_j, \phi) dz_j \right) = \sum_{i=1}^{m} \int_{z_i} p(z_i|x_i, \phi) \log(p(x_i|\theta) dz_i$$

The previous result was derived by putting the i^{th} integral inside of the summation and the log inside of the integral. The remaining product term is just 1 and so we can remove it. Now we use Bayes' theorem to derive a useful result:

$$\sum_{i=1}^{m} \int p(z_i|x_i, \phi) \log(p(x_i|\theta)) dz_i = \sum_{i=1}^{m} \int p(z_i|x_i, \phi) \log \left(p(x_i, z_i|\theta) \frac{p(x_i|\theta)}{p(x_i, z_i|\theta)} \right) dz_i$$

$$= \sum_{i=1}^{m} \int p(z_i|x_i, \phi) \log \left(\frac{p(x_i, z_i|\theta)}{p(z_i|x_i, \theta)} \right) dz_i$$

$$= \sum_{i=1}^{m} \int p(z_i|x_i, \phi) \log \left(p(x_i, z_i|\theta) \right) dz_i - \sum_{i=1}^{m} \int p(z_i|x_i, \phi) \log \left(p(z_i|x_i, \theta) \right) dz_i$$

$$= E[\log(p(\vec{x}, \vec{z}|\theta))|\vec{x}, \phi] - E[\log(p(\vec{z}|\vec{x}, \theta))|\vec{x}, \phi]$$

$$= Q(\theta|\phi) - H(\theta|\phi)$$

This result is useful, because $\nabla_\theta l(\theta)$ evaluated at $\theta = \phi$ makes $H(\theta|\phi)$ go to 0:

$$\nabla_\theta H(\theta|\phi)|_{\theta=\phi} = \nabla_\theta|_{\theta=\phi} \sum_{i=1}^{m} \int p(z_i|x_i, \phi) \log \left(p(z_i|x_i, \theta) \right) dz_i$$

$$= \sum_{i=1}^{m} \int p(z_i|x_i, \phi) \nabla_\theta|_{\theta=\phi} \log \left(p(z_i|x_i, \theta) \right) dz_i = \sum_{i=1}^{m} \int \frac{p(z_i|x_i, \phi)}{p(z_i|x_i, \phi)} \nabla_\theta|_{\theta=\phi} p(z_i|x_i, \theta) dz_i$$

$$= \sum_{i=1}^{m} \nabla_\theta|_{\theta=\phi} \int p(z_i|x_i, \theta) dz_i = \sum_{i=1}^{m} \nabla_\theta|_{\theta=\phi} 1 = 0$$

Thus, we have the following EM identity:

$$\nabla_\theta l(\theta)|_{\theta=\phi} = \nabla_\theta Q(\theta|\phi)|_{\theta=\phi} \tag{3}$$

One way to think about this identity is that it shows where $\nabla_\theta l(\theta)|_{\theta=\phi}$ intersects with $\nabla_\theta Q(\theta|\phi)|_{\theta=\phi}$ in an extended parameter space. The EM algorithm moves along this intersection in the hopes of finding a zero. Also, it just so happens that $\nabla_\theta Q(\theta|\phi)|_{\theta=\phi}$ tends to be a much more tractable function than $\nabla_\theta l(\theta)$.

This brings us to the specific steps of the EM algorithm, which works as follows. Given an iterate θ_k, the next iterate θ_{k+1} is defined as the θ that maximizes $Q(\theta|\phi)$ where ϕ is set to θ_k.

$$\theta_{k+1} = \arg\max_\theta Q(\theta|\theta_k) = M(\theta_k)$$

The function $M(\theta_k)$ is known as the EM operator. The algorithm begins with an initial parameter set θ_0 and terminates when either $\|M(\theta_k) - \theta_k\|$ is small enough or a maximum number of iterations has been reached.

Algorithm 2.1 Expectation-Maximization

1: Initialize θ_0; Define **maxit**, **ftol**
2: $k = 0$
3: **while** $\|M(\theta_k) - \theta_k\| <$ **ftol** and $k <$ **maxit do**
4: $\quad k = k + 1$
5: $\quad \theta_{k+1} = M(\theta_k)$
6: **end while**

2.1 Convergence

Wu [1983] identified several key convergence properties of the EM algorithm. I list the relevant those relevant to the aims of this project:

- "If the unobserved complete-data specification can by defined by a curved exponential family with compact parameter space, all the limit points of any EM sequence are stationary points of the likelihood function."

- It cannot be guaranteed that EM converges to the maximum likelihood estimator.

- It can be shown that if $Q(\theta|\phi)$ is continuous on both θ and ϕ, then the limit points of the sequence of an EM algorithm are stationary points of the likelihood function. Thus, EM converges to a local maximum. For exponential family distributions, this continuity condition is satisfied.

Mixtures of Poissons are within the exponential family, and thus we can assume that the EM algorithm will converge to a stationary point. Furthermore, since we are working with probabilities, the maximum likelihood function is bounded above by 1.

Note we do not have enough information about the likelihood function of the mixture of Poissons to determine whether or not it converges to the MLE. Thus, in the Comparison section below, I discuss results in terms of convergence to stationary points.

3 QN2 Algorithm

The QN2 Algorithm by Jamshidian and Jennrich [1997] is described as a Quasi Newton method using Broyden-Fletcher-Goldfarb-Shanno (BFGS) symmetric rank 2 updating. QN2 is constructed as follows.

To begin, we define the following:

$$g(\theta_k) = \nabla_\theta Q(\theta|\phi)|_{\theta=\theta_k,\phi=\theta_k} = \nabla_\theta Q(\theta_k|\theta_k) \tag{4}$$

$$\tilde{g}(\theta_k) = M(\theta_k) - \theta_k \tag{5}$$

The first function $g(\theta_k)$ is the gradient of Q with respect to θ evaluated at $\theta = \theta_k$ and $\phi = \theta_k$. To simplify notation, I re-write this as $\nabla_\theta Q(\theta_k|\theta_k)$. Note that nowhere in this paper will I ever take a derivative with respect to ϕ The second function is the would-be step size of the EM operator M.

Jamshidian and Jennrich [1993] showed the following property of the EM step:

$$\tilde{g}(\theta_k) \approx -(\nabla^2_\theta Q(\hat{\theta}|\hat{\theta}))^{-1} g(\theta_k) \tag{6}$$

where $\nabla^2_\theta Q$ is the Hessian of Q and $\hat{\theta}$ is a local maximum of $l(\theta)$.

We are using a Quasi Newton method, and so would like to approximate $(\nabla^2_\theta l(\theta))^{-1}$, the inverse Hessian of the log likelihood function,

$$(\nabla^2_\theta l(\theta))^{-1} = (\nabla^2_\theta Q(\theta|\phi) - \nabla^2_\theta H(\theta|\phi))^{-1}$$

which, using equations (2) and (6), can be approximated as follows:

$$(\nabla^2_\theta l(\theta))^{-1} g(\theta) \approx -\tilde{g}(\theta) + Sg(\theta)$$

where we define S as a Quasi-Newton approximation to

$$(\nabla^2_\theta l(\theta))^{-1} - (\nabla^2_\theta Q(\hat{\theta}|\hat{\theta}))^{-1}$$

To summarize, let A be an approximation of $(\nabla^2_\theta l(\theta))^{-1}$ and B_0 be $(\nabla^2_\theta Q(\hat{\theta}|\hat{\theta}))^{-1}$. Then

$$A = B_0 + S$$

And

$$Ag(\theta) \approx -\tilde{g}(\theta) + Sg(\theta) \tag{7}$$

This approximation will be used to define the Quasi-Newton update step. Notice that this is a modification to the conventional Quasi-Newton method, because it contains a floating inverse Hessian approximation of $Q(\hat{\theta}|\hat{\theta})$. The presumption here is that this floating value should aid the overall approximation of the inverse Hessian of the log likelihood, and thus speed up convergence. This is rather interesting.

The resulting update step to θ_k is defined below and assumes a line search methodology is used to determine α_k, which is described in the next section. For ease of reading, I make abbreviations such as $g_k = g(\theta_k)$.

$$\theta_{k+1} = \theta_k - \alpha_k d_k \quad \text{where} \quad d_k = \tilde{g}_k - S_k g_k$$

NOTE: Jamshidian and Jennrich [1997] p.575 contains an error in step a, which incorrectly defines $d_k = -\tilde{g}_k(\theta_k) + S_k g(\theta_k)$.

Jamshidian and Jennrich [1997] point out that d_k can be viewed as a modification to the EM step, which is why they label QN2 as an EM accelerator.

The BFGS update is then derived as follows. Using the notation used by Jamshidian and Jennrich [1997], define:

$$\Delta\tilde{g}_k = \tilde{g}_{k+1} - \tilde{g}_k$$
$$\Delta g_k = g_{k+1} - g_k$$

The classic BFGS update of the inverse Hessian is (see Nocedal and Wright, p. 136-140) as follows. Let A_k be the inverse Hessian approximation of $(\nabla^2_{\theta_k} l(\theta_k))^{-1}$. The update to A_k is:

$$A_{k+1} = (I - \frac{\Delta\theta_k \Delta g_k^T}{\Delta g_k^T \Delta\theta_k}) A_k (I - \frac{\Delta g_k \Delta\theta_k^T}{\Delta g_k^T \Delta\theta_k}) + \frac{\Delta\theta_k \Delta\theta_k^T}{\Delta g_k^T \Delta\theta_k}$$

$$= A_k - A_k \frac{\Delta g_k \Delta\theta_k^T}{\Delta g_k^T \Delta\theta_k} - \frac{\Delta\theta_k \Delta g_k^T}{\Delta g_k^T \Delta\theta_k} A_k + \frac{\Delta\theta_k \Delta g_k^T A_k \Delta g_k \Delta\theta_k^T}{(\Delta g_k^T \Delta\theta_k)^2} + \frac{\Delta\theta_k \Delta\theta_k^T}{\Delta g_k^T \Delta\theta_k}$$

Thus,

$$\Delta A_k = A_{k+1} - A_k = (1 + \frac{\Delta g_k^T A_k \Delta g_k}{\Delta g_k^T \Delta\theta_k}) \frac{\Delta\theta_k \Delta\theta_k^T}{\Delta g_k^T \Delta\theta_k} - \frac{\Delta\theta_k \Delta g_k^T A_k + (\Delta\theta_k \Delta g_k^T A_k)^T}{\Delta g_k^T \Delta\theta_k}$$

Using Eqn (7), it follows that

$$A_k g_k \approx -\tilde{g}_k + S_k g_k$$
$$A_k g_{k+1} \approx -\tilde{g}_{k+1} + S_k g_{k+1}$$
$$A_k \Delta g_k \approx -\Delta\tilde{g}_k + S_k \Delta g_k$$

The size of the update step is

$$A_{k+1} - A_k = B_0 - B_0 + S_{k+1} - S_k = \Delta S_k$$

The QN2 BFGS update step is:

$$\Delta S_k = \left(1 + \frac{\Delta g_k^T \Delta\theta_k^*}{\Delta g_k^T \Delta\theta}\right) \frac{\Delta\theta_k \Delta\theta_k^T}{\Delta g_k^T \Delta\theta_k} - \frac{\Delta\theta_k^* \Delta\theta_k^T + (\Delta\theta_k^* \Delta\theta_k^T)^T}{\Delta g_k^T \Delta\theta_k} \tag{8}$$

where

$$\Delta\theta_k^* = -\Delta\tilde{g}_k + S_k \Delta g_k$$

As will be seen in the next section, it is important to initialize S_0 to $\mathbf{0}_{2n\times 2n}$.

3.1 Constraints

The problem of finding the maximum likelihood estimate is often a constrained optimization problem. For example, in the mixture of poissons, we have the following constraints:

$$(\lambda_k)_r \geq 0 \qquad 0 \leq (\gamma_k)_r \leq 1 \qquad r \in \{1, ..., n\}$$

$$\sum_{r=1}^{n}(\gamma_k)_r = 1 \tag{9}$$

Here $(\lambda_k)_r$ and $(\gamma_k)_r$ are defined as the λ and γ parameters, respectively, of population r at iteration k. While I have not performed an exhaustive literature search, it seems there is a tendency in the literature to ignore all but the equality constraint. The presumption then must be that in practical problems, the maximum likelihood estimate exists strictly inside the feasible region.

In the mixture of poissons example, this would make practical sense. $(\lambda_k)_r = 0$ would require that all observations have value 0, and thus maximum likelihood estimation would not be useful. $(\gamma_k)_r$ cannot become zero, because, as will be shown later, the derivative would go to infinity. Similarly, $(\gamma_k)_r$ cannot become 1 either, since a value of 1 in any $(\gamma_k)_r$ would imply a zero in the other γ's.

In order to enforce the inequality parameter constraints, we use the following pseudo line search method described in Jamshidian and Jennrich [1997]:

Algorithm 3.1 Constraint Enforcement

1: Given θ_k, d_k
2: $\alpha_k = 1$
3: **while** $\theta_k + \alpha_k d_k$ violates constraints **do**
4: $\quad \alpha_k = \alpha_k/2$
5: **end while**
6: return α_k

Now I show how the equality parameter constraint can be handled by appropriate initialization of θ_0 and S_0. To explain why, consider that at iteration k, we assume the relationship $\sum_{r=1}^{n}[\gamma_k]_r = 1$ holds. Then, in order for Eqn (9) to be satisfied at each successive iteration $k+1$, the step along the γ parameters $\Delta\gamma_k = \gamma_{k+1} - \gamma_k = \alpha_k(d_k)_\gamma$ must satisfy the relationship

$$\sum_{r=1}^{n}[\Delta\gamma_k]_r = \sum_{r=1}^{n}[\gamma_{k+1}]_r - [\gamma_k]_r = 0 \tag{10}$$

Thus, $\sum_{r=1}^{n}[\gamma_{k+1}]_r = 1$ if and only if $\sum_{r=1}^{n}[\Delta\gamma_k]_r = 0$. In order for this to be true, it must be true that:

$$\sum_{r=1}^{n}[(d_k)_\gamma]_r = \sum_{r=1}^{n}[(\tilde{g}_k)_\gamma]_r - [(S_k g_k)_\gamma]_r = 0 \tag{11}$$

Here terms such as $(\circ)_\gamma$ refer to the γ components (i.e. the first n components) of $\circ$. First, assume that the following is true for S_k and any vector $v \in \Re^{2n}$:

$$\sum_{r=1}^{n}[(S_k v)_\gamma]_r = 0 \tag{12}$$

Second, consider:

$$\sum_{r=1}^{n}[(\tilde{g}_k)_\gamma]_r = \sum_{r=1}^{n}[(M(\theta_k) - \theta_k)_\gamma]_r = \sum_{r=1}^{n}[(M(\theta_k))_\gamma]_r - \sum_{r=1}^{n}[\gamma_k]_r = 1 - 1 = 0 \tag{13}$$

which holds true for all k. By definition, $M(\theta_k)$ is another estimate of θ that satisfies all parameter constraints. An explicit example of $M(\theta_k)$ is provided in Section 4

Then by assumption (12) and Eqn (13), Eqn (11) holds true, and thus Eqn (10) holds for k. However, this does not yet prove that Eqn (10) will hold for iteration $k+1$ or any other subsequent iterations.

Thus, now we need to show that:

$$\sum_{r=1}^{n}[(d_{k+1})_\gamma]_r = \sum_{r=1}^{n}[(\tilde{g}_{k+1})_\gamma]_r - [(S_{k+1}g_{k+1})_\gamma]_r = 0 \tag{14}$$

Eqn (13) implies that:

$$\sum_{r=1}^{n}[(\tilde{g}_{k+1})_\gamma]_r = 0$$

And so all that is left is to show that:

$$0 = \sum_{r=1}^{n}[(S_{k+1}g_{k+1})_\gamma]_r = \sum_{r=1}^{n}[((S_k + \Delta S_k)g_{k+1})_\gamma]_r = \sum_{r=1}^{n}[(S_k g_{k+1})_\gamma]_r + \sum_{r=1}^{n}[(\Delta S_k g_{k+1})_\gamma]_r$$

By the assumption in Eqn (12), we already have that $\sum_{r=1}^{n}[(S_k g_{k+1})_\gamma]_r = 0$. Thus, to show that $\sum_{r=1}^{n}[(\Delta S_k g_{k+1})_\gamma]_r = 0$, we need to evaluate the BFGS update, which has three matrix terms: $\Delta\theta_k\Delta\theta_k^T, \Delta\theta_k\Delta\theta_k^{*T}$, and $\Delta\theta_k^*\Delta\theta_k^T$. Consider any vector $u \in \Re^{2n}$

$$\Delta\theta_k\Delta\theta_k^T u = \begin{bmatrix} \Delta\gamma_k \\ \Delta\lambda_k \end{bmatrix}\Delta\theta_k^T u \quad \text{which implies} \quad \sum_{r=1}^{n}[(\Delta\theta_k\Delta\theta_k^T u)_\gamma]_r = \Delta\theta_k^T u\sum_{r=1}^{n}[\Delta\gamma_k]_r = 0$$

$$\Delta\theta_k\Delta\theta_k^{*T} u = \begin{bmatrix} \Delta\gamma_k \\ \Delta\lambda_k \end{bmatrix}\Delta\theta_k^{*T} u \quad \text{which implies} \quad \sum_{r=1}^{n}[(\Delta\theta_k\Delta\theta_k^{*T} u)_\gamma]_r = \Delta\theta_k^{*T} u\sum_{r=1}^{n}[\Delta\gamma_k]_r = 0$$

As for the third matrix term:

$$\Delta\theta_k^*\Delta\theta_k^T u = (-\Delta\tilde{g}_k + S_k\Delta g_k)\Delta\theta_k^T u$$

Eqn (13) implies:

$$\sum_{r=1}^{n}[(\Delta\tilde{g}_k)_\gamma]_r = 0$$

And assumption (12) allows that:

$$\sum_{r=1}^{n}[(S_k\Delta g_k)_\gamma]_r = 0$$

Therefore,

$$\sum_{r=1}^{n}[(\Delta S_k u)_\gamma]_r = \sum_{r=1}^{n}[(a\Delta\theta_k\Delta\theta_k^T u + b\Delta\theta_k^*\Delta\theta_k^T u + c\Delta\theta_k\Delta\theta_k^{*T} u)_\gamma]_r = 0 \tag{15}$$

where a, b, c are constants (which need not be derived). It follows that

$$\sum_{r=1}^{n}[(S_{k+1}u)_\gamma]_r = 0$$

Ergo, Eqn (10) must also be true for $k+1$ and, by induction, must hold for all k, as long as S_k is chosen appropriately. It follows that we should choose S_0 such that:

$$\sum_{r=1}^{n}[(S_0u)_\gamma]_r = 0$$

Hence, we should choose $S_0 = \mathbf{0}_{2n\times 2n}$. This is an interesting result of the BFGS update applied to QN2 on mixture problems that assume the construction $p(z_i = r|\theta) = \gamma_r$. It allows us to maintain feasibility in the equality constraint in all iterations.

3.2 Sequential Quadratic Programming Method of QN2

In this section I show how to construct QN2 as an SQP in order to appropriately handle the equality constraint. Another nice property of $p(z_i = r|\theta) = \gamma_r$ for QN2 is that the Lagrange multiplier λ^* at the optimal solution is easy to compute. The Lagrangian function of Eqn (2) is:

$$\mathcal{L}(\theta, \lambda) = l(\theta) - \lambda(\sum_{r=1}^{n}\gamma_r - 1)$$

Using Eqn (3), the gradient of the Lagrangian is:

$$\nabla_\theta\mathcal{L}(\theta,\lambda)|_{\theta=\phi} = \nabla_\theta Q(\theta|\phi) - \lambda J^T \quad \text{where} \quad J^T = \begin{bmatrix} \mathbf{1}_n \\ \mathbf{0}_n \end{bmatrix} \begin{matrix} \leftarrow \gamma \\ \leftarrow \lambda \end{matrix}$$

Using our concrete mixture of poissons example:

$$Q(\theta|\phi) = E[\log(p(\vec{x},\vec{z}|\theta)|\vec{x},\phi] = \sum_{r=1}^{n}\sum_{i=1}^{m} p(z_i = r|x_i,\phi)\log(p(x_i, z_i = r|\theta))$$

$$= \sum_{r=1}^{n}\sum_{i=1}^{m} p(z_i = r|x_i,\phi)\log(p(z_i|\theta)p(x_i|z_i = r,\theta))$$

$$= \sum_{r=1}^{n}\sum_{i=1}^{m} p(z_i = r|x_i,\phi)\left(\log(\gamma) + \log(p(x_i|z_i = r,\theta))\right)$$

The gradient of the Lagrangian is:

$$\frac{\partial}{\partial\gamma_r}\mathcal{L}(\theta,\lambda)|_{\theta=\phi} = \frac{1}{\gamma_r}\left[\sum_{i=1}^{m} p(z_i = r|x_i,\phi)\right] - \lambda$$

Solving for the zero of the gradient yields:

$$\gamma_r = \frac{1}{\lambda}\sum_{i=1}^{m} p(z_i = r|x_i,\phi)$$

Summing over all γ_r and setting to one yields:

$$1 = \sum_{r=1}^{n} \gamma_r = \sum_{r=1}^{n} \frac{1}{\lambda} \sum_{i=1}^{m} p(z_i = r|x_i, \phi) = \frac{1}{\lambda} \sum_{i=1}^{m} \sum_{r=1}^{n} p(z_i = r|x_i, \phi) = \frac{m}{\lambda}$$

Thus,

$$\lambda^* = \frac{1}{m}$$

This is a well-known result of mixtures. With it, we can show how to formulate QN2 within the SQP construct. The SQP can be formulated as follows. See Nocedal and Wright p. 530-532 for more details.

$$\begin{bmatrix} \nabla^2_{\theta_k} \mathcal{L}(\theta_k, \lambda_k) & -J^T \\ J^T & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \lambda_{k+1} \end{bmatrix} = - \begin{bmatrix} g_k \\ c(\theta_k) \end{bmatrix} \quad \text{where} \quad c(\theta_k) = \sum_{r=1}^{n} [\gamma_k]_r - 1$$

If we initialize the multiplier $\lambda_0 = \lambda^* = 1/m$, as well as initialize $S_0 = \mathbf{0}_{2n \times 2n}$ and $\sum_{r=1}^{n} [\gamma_0]_r = 1$, then $\lambda_k = \lambda^* = 1/m$ for all iterations. Thus, the SQP formulation becomes:

$$\begin{bmatrix} \nabla^2_{\theta_k} l(\theta_k) & -J^T \\ J^T & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \lambda^* \end{bmatrix} = - \begin{bmatrix} g_k \\ 0 \end{bmatrix}$$

Which reduces to:

$$\nabla^2_{\theta_k} l(\theta_k) d_k = -(g_k - \lambda^* J^T)$$

Which is approximated by:

$$d_k = \tilde{g}_k - S_k \bar{g} \quad \text{where} \quad \bar{g}_k = g_k - \lambda^* J^T$$

Note that this has not altered our approximation of S_k, since at each BFGS update we are concerned with Δg_k, in which the $\lambda^* J^T$ terms cancel out. Furthermore, it was shown in Eqn (15) that the equality constraints will still hold in all iterations with this new formulation.

3.3 Line Search

Jamshidian and Jennrich [1997] use the line search described by Jamshidian and Jennrich [1993], with initial step length equal to 2. I will not refer to that here and instead use the Armijo and Wolfe sufficient increase conditions (i.e. the Strong Wolfe conditions) with backtracking to perform the line search. These steps are outlined below. See Gill and Wright [2014] Lecture Notes 5, p. 155 for more details. The step size α_k is initialized using the output from the constraint enforcement step (3.1) to ensure that all line search function evaluations occur within the feasible region.

Algorithm 3.2 Strong Wolfe (i.e. Armijo/Wolfe) Line Search

Given θ_k, d_k
$\alpha_k = \text{ConstraintEnforcement}(\theta, d)$
$\eta_s = 10^{-4}$
$\eta_w = 0.99$
$T = 10$
$t = 0$
while $l(\theta_k + \alpha_k d_k) - l(\theta_k) \geq \eta_s \alpha_k \bar{g}_k^T d_k$ and $|\bar{g}(\theta_k + \alpha_k d_k)^T d_k| \leq \eta_w |\bar{g}_k^T d_k|$ and $t < T$ **do**
    $\alpha_k = \alpha_k / 2$
    $t = t + 1$
end while
if $t < T$ **then**
    foundalpha = True
else
    foundalpha = False
end if
return α_k, foundalpha

This line search runs until it either finds an α_k that satisfies the Strong Wolfe Conditions or else terminates if the number of line search attempts exceeds the maximum 10 attempts, in which case we consider the line search a failure. If the line search fails, we "restart" QN2 by re-initializing S_k to S_0.

3.4 Convergence

A nice property of the BFGS update is that, assuming the objective function is twice continuously differentiable with a compact level set, and the initial Hessian approximation is symmetric negative definite, then the sequence of iterates will converge to a maximizer (see Nocedal and Wright).

Given the special structure of QN2, however, I hesitate to translate that assumption to QN2. While Jamshidian and Jennrich [1997] showed that $\nabla^2_\theta Q(\hat{\theta}|\hat{\theta})$ is negative definite, the structure of QN2 is quite special, and more work would need to be done to explore its convergence properties. Given more time, I would have pursued this. For now, this remains an unknown for the project.

3.5 Termination

Jamshidian and Jennrich [1997] use a "relative gradient" merit function proposed by Khalfan et al. [1993] shown below.

$$rg = \max_i \left[|\bar{g}(\theta_k)|_i \frac{\max\{|\theta_k + \Delta\theta_k|_i, 1\}}{max\{|l(\theta_k + \Delta\theta_k|), 1\}} \right] < f_{tol} \tag{16}$$

Note that where Jamshidian and Jennrich [1997] use g, I use $\bar{g}$. An alternative merit function is the 2-norm of $\bar{g}_k$.

3.6 Initialization

Given an initial set of parameters θ_0, Jamshidian and Jennrich [1993] recommend running a few iterations of EM before beginning QN2. We do the same here.

As discussed above, set:

$$S_0 = \mathbf{0}_{2n \times 2n} \qquad \theta_0 \text{ feasible} \qquad \lambda^* = \frac{1}{m}$$

3.7 Algorithm Summary

Below I summarize the major steps of my implementation of the QN2 algorithm.

Algorithm 3.3 QN2 Implementation

Given θ_0 feasible
Define $\lambda^* = \frac{1}{m}$
Define maxit
Define f_{tol}
$\theta = \text{EM}(\theta_0; \text{ 6 iterations})$ *Algorithm (2.1)*
$\bar{g} = \bar{g}(\theta) = g(\theta) - \lambda^* J^T$ *Eqn (4)*
$\tilde{g} = \tilde{g}(\theta)$ *Eqn (5)*
$S = \mathbf{0}_{2n \times 2n}$
$rg = rg(\theta)$ *Eqn (16)*
$k = 0$
while $rg \geq f_{tol}$ and $k <$ maxit **do**
    $k = k + 1$
    $d = \tilde{g} - S\bar{g}$
    $\alpha = \text{ConstraintEnforcement}(\theta, d)$ *Algorithm (3.1)*
    α, foundalpha = StrongWolfeLineSearch(θ, d, α) *Algorithm (3.2)*
    if foundalpha == True **then**
        $\Delta\theta = \alpha d$
        $\Delta g = g(\theta + \Delta\theta) - g$
        $\Delta\tilde{g} = \tilde{g}(\theta + \Delta\theta) - \tilde{g}$
        $\Delta\theta^* = \tilde{g} - S\bar{g}$
        $\Delta S = \text{BFGS Update}$ *Eqn (8)*
        $\theta = \theta + \Delta\theta$
        $rg = rg(\theta)$
        $\bar{g} = \bar{g} + \Delta g$
        $\tilde{g} = \tilde{g} + \Delta\tilde{g}$
        $S = S + \Delta S$
    else
        $S = \mathbf{0}_{2n \times 2n}$
    end if
end while

4 Mixture of Poissons

Jamshidian and Jennrich [1997] provide the gradient and EM operator solutions for a mixture two poissons, while using $\gamma, 1-\gamma$ to model γ_1, γ_2. This approach is fine in 2 dimensions, but becomes significantly more complicated in higher dimensions. Here I list the computations for the generalized n-mixture Poisson problem.

Given a mixture of n poissons parameterized by $\gamma_1, ..., \gamma_n, \lambda_1, ..., \lambda_n$, the gradient of the Lagrange function is as follows:

$$\frac{\partial \mathcal{L}(\theta)}{\partial \gamma_r} = \frac{\partial Q(\theta|\phi)}{\partial \gamma_r}|_{\phi=\theta} - \lambda^* = \frac{1}{\gamma_r}\sum_{j=0}^{J} w_{rj}c_j - \lambda^*$$

$$\frac{\partial \mathcal{L}(\theta)}{\partial \lambda_r} = \frac{\partial Q(\theta|\phi)}{\partial \lambda_r} = \frac{1}{\lambda_r}\sum_{j=0}^{J} j w_{rj}c_j - \sum_{j=0}^{J} w_{rj}c_j$$

$$w_{rj} = p(z=r|x=j,\phi) = \frac{[\gamma_\phi]_r e^{-[\lambda_\phi]_r}[\lambda_\phi]_r^j}{\sum_{s=1}^{n}[\gamma_\phi]_s e^{-[\lambda_\phi]_s}[\lambda_\phi]_s^j}$$

$$c_j = \sum_{i=1}^{m} 1_{x_i=j} \qquad J = \max(x_i)$$

Here w_{rj} is parameterized by ϕ and $[\gamma_\phi]$ and $[\lambda_\phi]$ are the γ and λ elements of ϕ.

The EM operator $M(\theta_k)$ for the mixture of poissons is determined by solving for the zero of the derivative of the Lagrange function with respect to θ conditional on $\phi = \theta_k$. At the maximum likelihood estimate, $M(\theta_k) = \theta_k$.

$$[\gamma_{k+1}]_r = \frac{\sum_{j=0}^{J} w_{rj}c_j}{\sum_{j=0}^{J} c_j} \qquad [\lambda_{k+1}]_r = \frac{\sum_{j=0}^{J} j w_{rj}c_j}{\sum_{j=0}^{J} w_{rj}c_j}$$

The log likelihood function is:

$$l(\theta) = \sum_{j=1}^{J} c_j \log\left(\sum_{r=1}^{n} \gamma_r \frac{e^{-\lambda_r}\lambda_r^j}{j!}\right)$$

4.1 Computing Poisson Densities

The natural structure of the Poisson density function runs into issues when computed numerically due to the terms λ_r^j and $j!$. Therefore, I convert these densities to a different form that is more numerically stable:

$$\frac{e^{-\lambda_r}\lambda_r^j}{j!} \qquad \rightarrow \qquad e^{-\lambda_r + j\log(\lambda_r) - \sum_{i=0}^{j}\log(j)}$$

5 Comparison of EM and QN2 SQP on Mixtures of Poissons

My implementation of EM and QN2 SQP is written in the Python programming language version 2.7.6. All simulations were run in iPython 1.2.1 using a T420 Lenovo Thinkpad, with an Intel Core i5-2520M CPU 2.50GHz x 4A processor with 3.7GB of RAM on a Ubuntu 14.04 operating system. A copy of the code is provided as a supplement to this paper.

My comparison of EM and QN2 SQP encompasses "practical problems", rather than edge-case problems. I build a random problem generator, which I use to generate a large sample of problems on which I run EM and QN2 SQP.

From a practical perspective, two high level criteria for comparing algorithms are speed and reliability. From the user's perspective, speed is execution time, and reliability is whether or not the algorithm converges to the solution. Therefore, I compare convergence rates, execution times, convergence failures, and solutions.

I run comparisons on mixtures of 2 poissons (sample size 3000), 5 poissons (sample size 3000), and 10 poissons (sample size 3000). For each mixture size, EM and QN2 were each tested on a total of 100 randomly generated problems. EM and QN2 were each run from 3 different starting points, A,B,and C (described in the next section). That is a total of 1800 runs.

5.1 Random Problem Generator

I generate hidden samples $z_1, ..., z_m$ from m multinomial distribution of size 1 parameterized by random γ values. Next, I generate λ values for each of the n Poisson distributions using an exponential distribution with mean 1/10. Then I generate sample observations $x_1, ..., x_m$ where observation x_i is generated from poisson distribution z_i, which is parameterized by λ_{z_i}. Finally, only the sample observations $x_1, ..., x_m$ are passed to each algorithm.

5.2 Initialization

For each randomly generated problem, I test three initial starting conditions:

Initialization A:

$$\gamma_0 = [\frac{1}{n(n+1)/2}, ..., \frac{n}{n(n+1)/2}]^T$$

$$\lambda_0 = [1, ..., n]^T$$

Initialization B:

$$\gamma_0 = \{\text{parameters used for random problem generator}\}$$

$$\lambda_0 = \{\text{parameters used for random problem generator}\}$$

Initialization C:

$$\gamma_0 = [\frac{1}{n}, ..., \frac{1}{n}]^T$$

$$\lambda_0 = [1, ..., n]^T$$

As an aside, any point in which $\lambda_1 = \cdots = \lambda_n$ is a stationary point, though very likely not the maximum likelihood estimator.

5.3 Results: Size 2 Mixtures

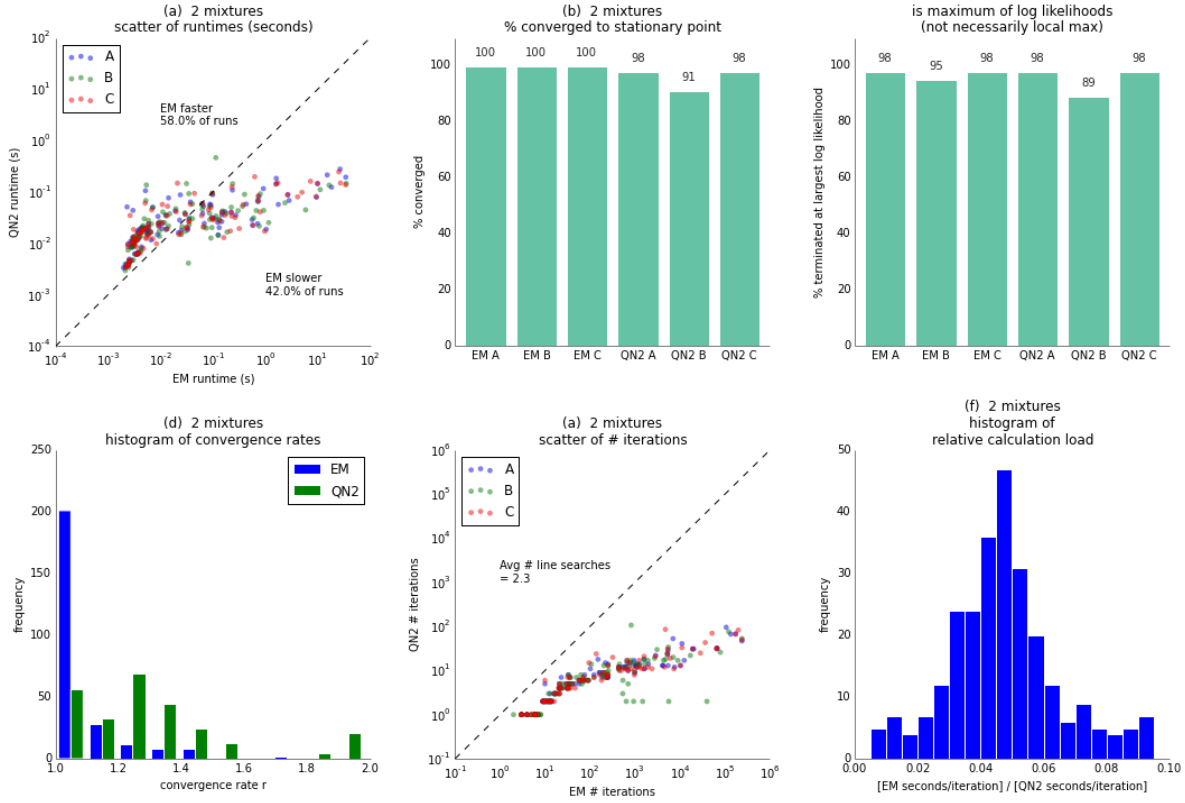

Figure 1: Results from running EM and QN2 SQP on randomly generated data from poisson mixtures of size 2. EM and QN2 were tested on a total of 100 randomly generated problems. EM and QN2 SQP were each run from 3 different starting points, A,B,and C (described in the previous section).

Figure 1a shows a scatter plot of the runtimes for each problem, with EM runtimes on the x-axis and QN2 SQP runtimes on the y-axis. The color scale refers to the initialization points A, B, and C. Points below the dotted line correspond to problems in which EM took longer than QN2 SQP to terminate. Note that the runtimes are on a log scale.

Given EM's reputation for slow convergence and Quasi-Newton's reputation for superlinear convergence, it is surprising to find that, overall, EM tends to finish before QN2 SQP. However, we can see that EM runtimes have a tendency to "explode", corresponding to sublinear convergence, lending credence to EM's reputation. QN2 SQP runtimes, on the other hand, tend to lie in a much narrower band (roughly between $10^{-2.5}$ and 10^{-1} seconds). From this speed perspective, QN2 SQP seems more dependable.

Figures 1b measures an aspect of reliability: how often does the algorithm terminate at a stationary point? Convergence to a stationary point is defined as $rg(\theta_k) \leq f_{tol}$, where $rg(\theta_k)$ is the merit function defined Eqn (16). All EM runs successfully terminated at a stationary point. Not all of the QN2 SQP runs converged to a stationary point. QN2 SQP runs that failed to converge did so because the line search failed. Recall that as part of the QN2 SQP algorithm, if line search fails, then S_k is reset to $\mathbf{0}_{2n\times 2n}$. If after this reset

line search fails again, then QN2 SQP has no more options and consequently breaks and returns an error. QN2 SQP with initialization B shows the worst performance. Further work could be done to determine more successful line search strategies for QN2 SQP.

Figure 1c measures another aspect of reliability: did the algorithm converge to the largest likelihood? Here I loosely defined a tolerance range of 10^{-3} for comparing one log likelihood function value to another, meaning that values within 10^{-3} of each other are considered the same. This threshold seemed to sufficiently capture nearly all of the of the "close" values. Initialization B shows a lower reliability in both EM and QN2 SQP runs.

Figure 1d plots a histogram of the observed convergence rates of EM and QN2 SQP. Since optimal points are unknown, the convergence rate r at iteration k is loosely defined as:

$$\frac{|rg(\theta_k + 1)|}{|rg(\theta_k)|^r} = 1 \qquad \rightarrow \qquad r = \frac{\log(rg(\theta_k + 1))}{\log(rg(\theta_k))}$$

The reported convergence rates are the average of the convergence rate in the final 2 iterations for each run. Convergence rates were not included for runs that terminated without converging.

Figure 1d shows unsurprising results. EM shows linear convergence and QN2 SQP shows superlinear convergence.

Figure 1e is another scatter plot showing the number of iterations required to terminate. Each dot represents the EM iterations vs the QN2 SQP iterations for a given problem and given initialization. This plot shows the same behavior as Figure 1a, although it is clear that more iterations are required for EM on all runs. The average number of QN2 SQP line searches is 2.3.

The larger number of iterations does not imply longer runtimes. Figure 1f shows the implied relative calculation load, which is calculated as:

$$\frac{\text{EM seconds per iteration}}{\text{QN2 SQP seconds per iteration}}$$

The frequency plot is centered around 0.05, which indicates that, on average, a QN2 SQP iteration takes roughly 20 times longer than an EM iteration.

5.4 Results: Size 5 Mixtures

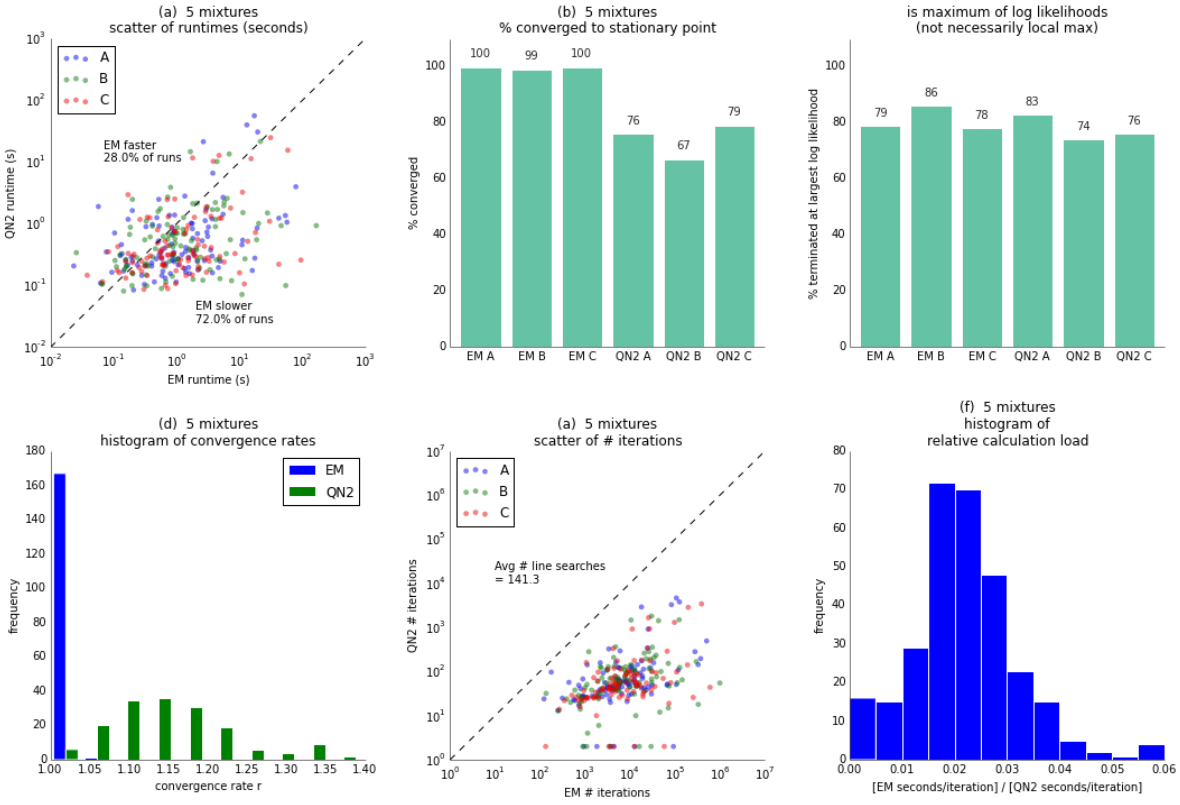

Figure 2: Results from running EM and QN2 SQP on randomly generated data from poisson mixtures of size 5. EM and QN2 SQP were tested on a total of 100 randomly generated problems. EM and QN2 SQP were each run from 3 different starting points, A,B,and C (described in the previous section).

EM shows much slower relative performance in size 5 mixture runs. Except for one run, all EM runs converged to a stationary point, whereas roughly only 73% of QN2 SQP terminated at a stationary point. Again, QN2 SQP initialized on B performed the worst. The relative calculation load averages at roughly 0.02, meaning approximately 50 EM iterations per 1 QN2 SQP iteration. This is an increase from the 20:1 ratio seen for size 2 mixtures, indicating that the calculation load for QN2 is growing with mixture size. This makes sense given that the number of QN2 SQP line searches has increased dramatically from 2.3 to 141.

5.5 Results: Size 10 Mixtures

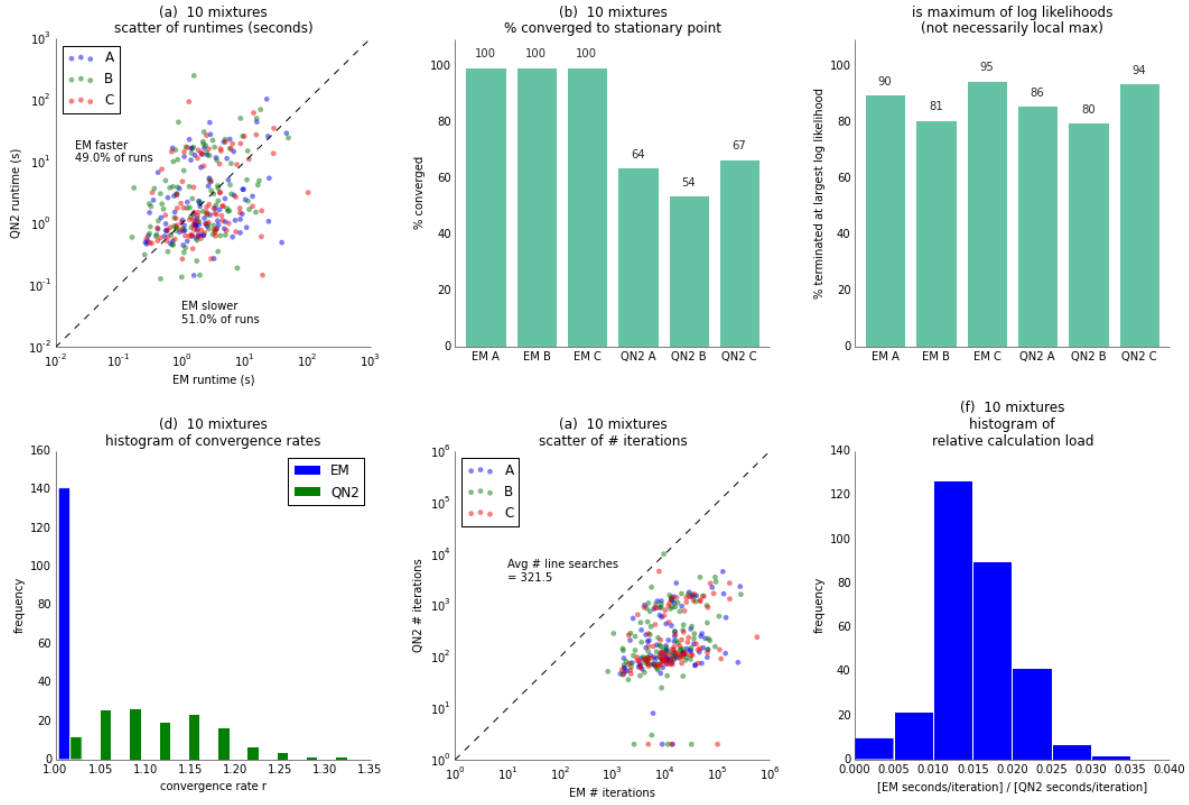

Figure 3: Results from running EM and QN2 SQP on randomly generated data from poisson mixtures of size 10. EM and QN2 SQP were tested on a total of 100 randomly generated problems. EM and QN2 SQP were each run from 3 different starting points, A,B,and C (described in the previous section).

Here we see that the division in runtimes between EM and QN2 SQP is roughly even. The spread in the scattor plot also look much more even than in mixtures of size 2 and 5. Interestingly, the spread of QN2 SQP runtimes is greater than that of EM. Thus, from a speed perspective, EM looks better.

The ability of EM to converge to a stationary point is again perfect, but that of QN2 SQP has decreased further to approximately 59%. The distribution of QN2 SQP convergence rates has also decreased somewhat from that for size 2 mixtures. The average number of QN2 SQP line searches has increased further to 321.5, which is more than double that seen for size 5 mixtures. The relative calculation load hovers around 0.015, which implies a 66:1 EM iterations to QN2 SQP iterations ratio.

5.6 Discussion

Two themes arise from these results. EM shows little difference in performance across mixture sizes. QN2 SQP shows worsening performance. The promising results in Jamshidian and Jennrich [1997] are tempered by these findings.

The reason for QN2 SQP's worsening performance lies in the number of line searches required. This is seen in a couple of ways. First is the increase in convergence failures. Second is the relative calculation load. Further work could be done to find better line search strategies for QN2 SQP.

Even though EM showed linear convergence while QN2 SQP showed superlinear convergence, the overall calculation load of QN2 SQP grew substantially, rendering QN2 SQP's relative advantage in convergence rate irrelevant.

As discussed in sections 2.1 and 3.4, given my current understanding and knowledge of the mixture of Poissons likelihood function, not much can be said about convergence to the MLE, which is a disadvantage of both EM and QN2 SQP. Furthermore, Figures 1c, 2c, and 3c show that convergence to the global maximizer is not guaranteed, and is dependent on algorithm selection and initialization.

In summary, the expected speed improvements of QN2 SQP are seen in smaller sized mixtures, but as the mixture size grows, EM tends to be the better option. From a practical perspective, EM looks to be the better choice across all mixture sizes. The runtime for mixture size two for EM is rather low. For larger mixture sizes, EM and QN2 SQP are not that much different. EM is much more reliable in terms of convergence than QN2 SQP. Furthermore, the amount of thinking and work required to implement a QN2 SQP algorithm is much higher than that for EM. Therefore, I concede that for mixtures of poissons and for this specific implementation of QN2 SQP, EM is the better algorithm.

6 Supplementary Files

```
Code:
 - readme.txt (see this for descriptions of code and data files)
 - PoissonQN2_MultiMixture.py
 - PoissonQN2_Multi_TestRuns.py
 - Analysis.py
 - utils.py

Results Data:
 - RunData2_K2_init3.pk1
 - RunData2_K5_init3.pk1
 - RunData2_K10_init3.pk1
```

References

Jeff A. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute, Berkeley, CA*, 1998.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. R. Statist. Soc.*, B(39):1–38, 1997.

P.E. Gill and M. H. Wright. Advanced topics in numerical analysis: Numerical optimization lecture notes (csci-ga.2495-001). *New York University*, 2014.

Mori Jamshidian and Robert I. Jennrich. Conjugate gradient acceleration of the em algorithm. *J. Am. Statist. Ass.*, 88:221–228, 1993.

Mortaza Jamshidian and Robert I. Jennrich. Acceleration of the em algorithm by using quasi newton methods. *Journal of the Royal Statistical Society*, 1997.

H. F. Khalfan, R. H. Byrd, and R. B. Schnabel. A theoretical and experimental study of the symmetric rank-one update. *SIAM Journal of Optimization*, (3):1–24, 1993.

Kenneth Lange. A quasi newton acceleration of the em algorithm. *Statist. Sin.*, 5:1–18, 1995.

Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. 1st edition.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. 2nd edition.

C. F. Jeff Wu. On the convergence properties of the em algorithm. *Ann. Statist.*, 11: 95–103, 1983.