GRADUATE STUDENT STAT 840 A4

Vsevolod Ladtchenko 20895137

Problem 1

a)

The observed components of the data are $x_1, ..., x_n$ which are the draws from the mixture density. The missing data are latent variables $z_1, ..., z_n$ where z_i denotes from which distribution x_i came from: if $z_i = 1$ then x_i came from the first distribution, if $z_i = 0$ then x_i came from the second distribution. Thus z_i is a Bernoulli random variable with probability p.

The steps of the EM algorithm are:

0. initialize values for parameters p, λ_1, λ_2
1. E step: we want to maximize our log-likelihood, which contains latent (unknown) data. So take the expectation with respect to the distribution of this unknown data, which is conditioned on the observed data and current parameters.
2. M step: maximize the likelihood now that we took the expectation over the missing data.
3. check for convergence, go back to step 1. run from different starting points since it converges to local maximum.

$$
\begin{aligned}
f_{mis}(z_i) &= p^{z_i}(1-p)^{1-z_i} \\
f_{obs|mis}(x_i \mid z_i) &= Pois(x_i, \lambda_1)^{z_i} Pois(x_i, \lambda_2)^{1-z_i} \\
Lik_{com} &= \prod_{i=1}^{n} f_{com}(x_i, z_i) \\
&= \prod_{i=1}^{n} f_{obs|mis}(x_i \mid z_i) f_{mis}(z_i) \\
lik_{com} &= \sum_{i=1}^{n} z_i \log Pois(x_i, \lambda_1) + (1 - z_i) \log Pois(x_i, \lambda_2) \\
&\quad + \sum_{i=1}^{n} z_i \log p + (1 - z_i) \log(1 - p)
\end{aligned}
$$

E-step:

$$
\begin{aligned}
\mathbb{E}_{mis|obs,\theta^{(k)}}[z_i \mid x_i] &= w_i^{(k)} \\
w_i^{(k)} &= P(z_i = 1 \mid x_i) \\
&= \frac{P(x_i \mid z_i = 1)P(z_i = 1)}{P(x_i)} \\
&= \frac{pPois(x_i, \lambda_1)}{pPois(x_i, \lambda_1) + (1-p)Pois(x_i, \lambda_2)} \\
Q(\theta, \theta^{(k)}) &= \mathbb{E}_{mis|obs,\theta^{(k)}}[lik_{com} \mid x_i] \\
&= \sum_{i=1}^{n} w_i^{(k)} \log Pois(x_i, \lambda_1) + (1 - w_i^{(k)}) \log Pois(x_i, \lambda_2) \\
&\quad + \sum_{i=1}^{n} w_i^{(k)} \log p + (1 - w_i^{(k)}) \log(1-p)
\end{aligned}
$$

M-step:

$$
\begin{aligned}
\log Pois(x, \lambda) &= x \log \lambda - \lambda - \log(x!) \\
Q(\theta, \theta^{(k)}) &= \sum_{i=1}^{n} w_i^{(k)} \Big(x_i \log \lambda_1 - \lambda_1 - \log(x_i!) \Big) + (1 - w_i^{(k)}) \Big(x_i \log \lambda_2 - \lambda_2 - \log(x_i!) \Big) \\
&\quad + \sum_{i=1}^{n} w_i^{(k)} \log p + (1 - w_i^{(k)}) \log(1-p)
\end{aligned}
$$

Partial p:

$$
\begin{aligned}
\partial_p Q &= \sum_{i=1}^{n} \frac{w_i^{(k)}}{p} - \frac{(1 - w_i^{(k)})}{1-p} \\
0 &= \frac{s}{p} - \frac{n-s}{1-p} \\
\frac{n-s}{1-p} &= \frac{s}{p} \\
p(n-s) &= (1-p)s \\
pn - ps &= s - ps \\
pn &= s \\
\hat{p}^{(k+1)} &= \frac{1}{n} \sum_{i=1}^{n} w_i^{(k)}
\end{aligned}
$$

Partial lambda 1:

$$
\begin{aligned}
\partial_{\lambda_1} Q &= \sum_{i=1}^n w_i^{(k)} \left(\frac{x_i}{\lambda_1} - 1 \right) \\
0 &= \sum_{i=1}^n w_i^{(k)} \frac{x_i}{\lambda_1} - \sum_{i=1}^n w_i^{(k)} \\
\sum_{i=1}^n w_i^{(k)} &= \frac{1}{\lambda_1} \sum_{i=1}^n w_i^{(k)} x_i \\
\lambda_1 \sum_{i=1}^n w_i^{(k)} &= \sum_{i=1}^n w_i^{(k)} x_i \\
\hat{\lambda}_1^{(k+1)} &= \frac{\sum_{i=1}^n w_i^{(k)} x_i}{\sum_{i=1}^n w_i^{(k)}}
\end{aligned}
$$

Partial lambda 2:

$$
\begin{aligned}
\partial_{\lambda_2} Q &= \sum_{i=1}^n (1 - w_i^{(k)}) \left(\frac{x_i}{\lambda_2} - 1 \right) \\
0 &= \sum_{i=1}^n \frac{(1 - w_i^{(k)}) x_i}{\lambda_2} - \sum_{i=1}^n (1 - w_i^{(k)}) \\
\sum_{i=1}^n (1 - w_i^{(k)}) &= \sum_{i=1}^n \frac{(1 - w_i^{(k)}) x_i}{\lambda_2} \\
\lambda_2 \sum_{i=1}^n (1 - w_i^{(k)}) &= \sum_{i=1}^n (1 - w_i^{(k)}) x_i \\
\hat{\lambda}_2^{(k+1)} &= \frac{\sum_{i=1}^n (1 - w_i^{(k)}) x_i}{\sum_{i=1}^n (1 - w_i^{(k)})}
\end{aligned}
$$

b)

```r
em_algo = function(x, p, lam1, lam2)
{
  lik = -Inf

  while (TRUE)
  {
    # E step
    pois1 =     p * dpois(x, lam1)
    pois2 = (1-p) * dpois(x, lam2)
    w     = pois1 / (pois1 + pois2)

    # M step
    p    = mean(w)
    lam1 = sum(w     * x) / sum(w)
    lam2 = sum((1-w) * x) / sum(1-w)

    # check convergence
    lik_new = sum(log(pois1 + pois2))
    if (abs(lik_new - lik) < 1e-8)
```

```
      break

    lik = lik_new
  }
  return(c(p, lam1, lam2, lik_new))
}
```

c)

The estimates are somewhat close to the true parameters, but also far. Due to the fairly large gap, we implemented a strategy to generate random data from a mixture model in order to test the EM implementation. It appears that with a sample size of 20, as per the given data, the results are fairly unreliable. This variability can be easily attributed to the small sample size. As we increase the sample, the estimates get closer to the true values. For n = 1000, they are very close. Also note that we can get an identical mirror result, where the lambda values are flipped, and the Bernoulli probability is $1 - p$.

```
grid_search = function(x) # search over a range of parameters
{
  df = matrix(nrow=20*20*9,ncol=4)
  colnames(df) = c('p', 'lam1', 'lam2', 'lik')
  i = 1
  for (p in seq(0.1, 0.9, 0.1))
  {
    for (lam1 in seq(1,20))
    {
      for (lam2 in seq(1,20))
      {
        df[i,] = em_algo(x, p, lam1, lam2)
        i = i + 1
      }
    }
  }
  max_idx = which.max(df[,4])
  return(df[max_idx,])
}

generate_data = function(n = 20, p = 0.3, lam1 = 5, lam2 = 15)
{
  x = rep(NA,n)
  for (i in 1:n)
  {
    lam = if (runif(1) < p) lam1 else lam2
    x[i] = rpois(1, lam)
  }
  return(x)
}

# problem statement
x = c(24, 18, 21, 5, 5, 11, 11, 17, 6, 7, 20, 13, 4, 16, 19, 21, 4, 22, 8, 17)
grid_search(x)
```

```
##           p        lam1        lam2         lik
##   0.3934055   6.2334344  18.1302874 -62.2106189
```

```
# check implementation
#for (n in c(20,100,1000))
#  print(grid_search(generate_data(n)))
```

d)

Compute the Hessian. Note the mixed partials are all zero. This makes the Hessian diagonal, so to invert it we simply take the inverse of the diagonal elements.

$$
\begin{aligned}
\partial_p Q &= \sum_{i=1}^{n} \frac{w_i^{(k)}}{p} - \frac{(1-w_i^{(k)})}{1-p} \\
&= \frac{s}{p} - \frac{n-s}{1-p} \\
\partial_{pp} Q &= \frac{-s}{p^2} - \frac{n-s}{(1-p)^2} \\
\partial_{\lambda_1\lambda_1} Q &= \frac{-1}{\lambda_1^2} \sum_{i=1}^{n} w_i^{(k)} x_i \\
\partial_{\lambda_2\lambda_2} Q &= \frac{-1}{\lambda_2^2} \sum_{i=1}^{n} (1-w_i^{(k)}) x_i
\end{aligned}
$$

$$
l_{obs} = \sum \log(pPois(x_i, \lambda_1) + (1-p)Pois(x_i, \lambda_2))
$$

Due to the difficulty in computing the observed likelihood, we will try using numerical methods. Note that this may be a good use case for automatic differentiation. Nevertheless, we will compute the Hessian of the observed likelihood function numerically, using parameter estimates from our previous EM algorithm run. The results look believable, and make sense as to the error we got from the true values of the parameters.

```
library(numDeriv)

p = 0.5
lam1 = 1
lam2 = 10

x = c(24, 18, 21, 5, 5, 11, 11, 17, 6, 7, 20, 13, 4, 16, 19, 21, 4, 22, 8, 17)

lik = -Inf

while (TRUE)
{
  # E step
  pois1 =     p * dpois(x, lam1)
  pois2 = (1-p) * dpois(x, lam2)
  w     = pois1 / (pois1 + pois2)

  # M step
  p    = mean(w)
  lam1 = sum(w     * x) / sum(w)
  lam2 = sum((1-w) * x) / sum(1-w)

  # check convergence
```

```r
  lik_new = sum(log(p * dpois(x, lam1) + (1-p) * dpois(x, lam2)))
  if (abs(lik_new - lik) < 1e-8)
    break

  lik = lik_new
}

observed_lik = function(params)
{
  p = params[1]
  lam1 = params[2]
  lam2 = params[3]

  pois1 = dpois(x, lam1)
  pois2 = dpois(x, lam2)

  lik = sum(log(p*pois1 + (1-p)*pois2))

  return(lik)
}

# https://www.rdocumentation.org/packages/numDeriv/versions/2016.8-1.1/topics/hessian
H = hessian(observed_lik, c(p,lam1,lam2))

var_cov_matrix = solve(-H)

err = sqrt(diag(var_cov_matrix))

c(p=p,lam1=lam1,lam2=lam2)
```

```
##          p       lam1       lam2
##  0.3933956  6.2333136 18.1301716
```

```r
c(std_p=err[1], std_lam1=err[2], std_lam2=err[3])
```

```
##     std_p  std_lam1  std_lam2
## 0.1257246 1.1737013 1.4310834
```