STA410 Assignment #4

student number: 1003942326

Yulin WANG

28/11/2019

Question 1

(a)

$$\begin{aligned}\theta_k &= \theta_1 + (\theta_2 - \theta_1) + (\theta_3 - \theta_2) + \ \dots \ + (\theta_{k-1} - \theta_{k-2}) + (\theta_k - \theta_{k-1}) \\ &= \theta_1 + \phi_2 + \phi_3 \ + \dots + \ \phi_{k-1} + \phi_k \qquad \text{,since} \ \ \phi_i = \theta_i - \theta_{i-1} \ \ \text{for} \ \ i \geq 2 \\ &= \theta_1 + \sum_{i=2}^{k} \phi_i \qquad \text{for } k \geq 2\end{aligned}$$

(b)

The partial derivative of the objective function(2) with respect to θ_j, where $j = 1, ..., n$

$$\begin{aligned}\frac{\partial}{\partial \theta_j}\{\sum_{i=1}^{n}(y_i - \theta_i)^2 + \lambda \sum_{i=2}^{n}|\phi_i|\} &= 2\sum_{i=1}^{n}(y_i - \theta_i) \cdot (-1) \overset{set}{=} 0 \\ &\Rightarrow \sum_{i=1}^{n}(y_i - \hat{\theta}_i) = 0\end{aligned}$$

Thus, if $\hat{\theta}_1, ..., \hat{\theta}_n$ minimize the objective function, then $\sum_{i=1}^{n}(y_i - \hat{\theta}_i) = 0$

(c)

For each θ_j, where $j = 1, ..., n$, we can write the objective function as a sum of a differentiable and a non-differentiable function, both of which are convex. The subgradient is then the subgradient of the non-differentiable function (which may be an interval) translated by the derivative of the differentiable function. The subgradient of $\lambda \sum_{i=2}^{n} |\theta_i - \theta_{i-1}|$ is

$$\partial \lambda \sum_{i=2}^{n} |\theta_i - \theta_{i-1}| = \lambda \partial \sum_{i=2}^{n} |\theta_i - \theta_{i-1}| = \begin{cases} +2\lambda & \text{if } \ \theta_j > 0 \\ [-2\lambda, +2\lambda] & \text{if } \ \theta_j = 0 \\ -2\lambda & \text{if } \ \theta_j < 0 \end{cases}$$

and the derivative of the differentiable part is

$$-2\sum_{i=1}^{n}(y_i - \theta_i)$$

The convex function of $\theta_j$is minimized if and only if 0 lies in the subgradient at the minimizing point. Thus the function is minimized at $\theta_j = 0$ if

$$-2\sum_{i=1}^{n}(y_i - \hat{\theta}_i) \subset [-2\lambda, 2\lambda]^n \Rightarrow \sum_{i=1}^{n}(y_i - \hat{\theta}_i) \subset [-\lambda, \lambda]^n \Rightarrow (y_i - \hat{\theta}_i) \in [-\lambda, \lambda] \Rightarrow |y_i - \hat{\theta}_i| \leq \lambda$$

Thus, $|y_i - \hat{\theta}_i| \leq \lambda$ for all i.

(d)

```r
#  Adapted from Condat (2013) IEEE Signal Processing Letters
tvsmooth <- function(x,lambda) {
              lambda <- lambda/2
              n <- length(x)
              xhat <- rep(0,n)
              k <- 1
              k0 <- 1
              km <- 1
              kp <- 1
              vmin <- x[1] - lambda
              vmax <- x[1] + lambda
              umin <- lambda
              umax <- -lambda
              while (k < n) {
                  if (x[k+1]+umin<vmin-lambda) {
                     xhat[k0:km] <- vmin
                     k <- km+1
                     k0 <- k
                     km <- k
                     kp <- k
                     vmin <- x[k]
                     vmax <- x[k]+2*lambda
                     umin <- lambda
                     umax <- -lambda
                     }
                  else if (x[k+1]+umax>vmax+lambda) {
                        xhat[k0:kp] <- vmax
                        k <- kp+1
                        k0 <- k
                        km <- k
                        kp <- k
                        vmin <- x[k]-2*lambda
                        vmax <- x[k]
                        umin <- lambda
                        umax <- -lambda
                        }
                   else {
                        k <- k+1
                        umin <- umin + x[k] - vmin
                        umax <- umax + x[k] - vmax
                        if (umin>=lambda) {
                          vmin <- vmin + (umin-lambda)/(k-k0+1)
                          umin <- lambda
                          km <- k
                          }
                        if (umax<=-lambda) {
                          vmax <- vmax + (umax+lambda)/(k-k0+1)
                          umax <- -lambda
                          kp <- k
                          }
                        }
                if (k>=n) {
```

```r
                    if (umin<0) {
                      xhat[k0:km] <- vmin
                      k <- km+1
                      k0 <- k
                      km <- k
                      vmin <- x[k]
                      umin <- lambda
                      umax <- x[k] + lambda - vmax
                      if (k==n) xhat[n] <- vmin+umin
                      }
                    else if (umax>0) {
                      xhat[k0:kp] <- vmax
                      k <- kp+1
                      k0 <- k
                      kp <- k
                      vmax <- x[k]
                      umax <- -lambda
                      umin <- x[k] - lambda - vmin
                      if (k==n) xhat[n] <- vmin+umin
                      }
                    else {
                      xhat[k0:n] <- vmin + umin/(k-k0+1)
                      }
                  }
                  }
               xhat
}
x <- rep(10, 100)
#try different values for lambda
tvsmooth(x, 10000000000000)
```

```
##   [1] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##   [8] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##  [15] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##  [22] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##  [29] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##  [36] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##  [43] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##  [50] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##  [57] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##  [64] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##  [71] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##  [78] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##  [85] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##  [92] 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985 9.999985
##  [99] 9.999985 9.999985
```

```r
tvsmooth(x, 100000000000000000000000)
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```r
tvsmooth(x, 1000000000000000000000000000)
```

```
##   [1] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##   [6] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [11] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [16] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [21] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [26] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [31] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [36] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [41] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [46] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [51] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [56] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [61] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [66] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [71] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [76] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [81] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [86] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [91] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
##  [96] -3221225472 -3221225472 -3221225472 -3221225472 -3221225472
```

Question 2

(a)

$$
\begin{aligned}
E_\lambda(M) &= E(M|\lambda, n) \\
&= P(M = m|\lambda, n) \times (n + m) \\
&= (1 - K_\lambda(r)) \times (n + E_\lambda(M)) \\
&= n \cdot (1 - K_\lambda(r)) + E_\lambda(M) \cdot (1 - K_\lambda(r))
\end{aligned}
$$

Then we have:

$$
E_\lambda(M) \cdot [1 - (1 - K_\lambda(r))] = n(1 - K_\lambda(r)) \Rightarrow E_\lambda(M) = n(1 - K_\lambda(r))/K_\lambda(r)
$$

(b)

Using the law of total expectation (tower rule):

$$
\begin{aligned}
E_\lambda(\sum_{i=n+1}^{n+M} X_i|X_1 = x_1, ..., \ X_n = x_n) &= E_\lambda[E_\lambda(\sum_{i=n+1}^{n+M} X_i|M = m)] \\
&= E_\lambda[E_\lambda(X_{n+1} + X_{n+2} + ... + X_{n+m})] \\
&= E_\lambda[m \cdot E_\lambda(X_i|X_i \leq r)] \\
&= m \cdot E_\lambda(X_i|X_i \leq r) \\
&= E_\lambda(M)E_\lambda(X_i|X_i \leq r)
\end{aligned}
$$

(c)

The initial estimate of λ is:

$$
\hat{\lambda}^{(l)} = \frac{1*1317 + 2*239 + 3*42 + 4*14 + 5*4 + 6*4 + 7*1}{1317 + 239 + 42 + 14 + 4 + 4 + 1} = \frac{2028}{1621}
$$

Suppose that $\hat{\lambda}^{(l)}$ is the current estimate of λ.
Then the E-step is:

$$
\begin{aligned}
\hat{K}_\lambda(r)^{(l+1)} &= 1 - exp(-\hat{\lambda}^{(l)}) \\
\hat{E}_\lambda(M)^{(l+1)} &= 1621 \cdot (1 - \hat{K}_\lambda(r)^{(l+1)})/\hat{K}_\lambda(r)^{(l+1)}
\end{aligned}
$$

with the M-step:

$$
\hat{\lambda}^{(l+1)} = \frac{2028}{1621 + M}
$$

Here is the R implement with 1000 iterations:

```r
lambda <- (1*1317+2*239+3*42+4*14+5*4+6*4+7*1)/(1317+239+42+14+4+4+1) #initial value of lambda
n <- (1317+239+42+14+4+4+1) #value of n=1621
for (i in c(1:1000)) {
  k <- 1 - exp(-lambda)
  M <- n*(1-k)/k
  lambda <- 2028/(n+M)
}
M
```

```
## [1] 2730.148
```

```r
lambda
```

```
## [1] 0.4660839
```

Since the algorithm converges after 1000 iterarions, so this truncated Poisson model is useful for these data. $\hat{\lambda} = 0.4660839$, and the estimate of number M of policies with no claims is about 2730.148 which is reasonable.