STAT-6494 Advanced Statistical Computing with R

Homework 6

Wenjie Wang

06 April 2016

1 Exercise: A Poisson-HMM for Earthquake Data

Consider a m-state HMM where the observed data follows a Poisson distribution with mean λ_i for state i, $i = 1, \ldots, m$, respectively. This model have three sets of parameter: initial distribution δ, transition probability matrix Γ, and Poisson means $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$. Suppose the observed data is a vector x. Write a function `poisson.hmm.em` that implements the EM algorithm for this model with these argument:

- `x`: the observed data;
- `m`: the number of states;
- `lambda`: initial value of λ;
- `Gamma`: initial value of Γ;
- `delta`: initial value of δ;
- `control`: a named list similar to `glm.control` that specifies the tolerance, maximum number of iteration, and whether or not trace the iteration.

Apply the function to the count of major earthquakes (magnitude 7 or above) in the world from 1900 to 2015 with a 2-state HMM and a 3-state HMM.

```
## frequency of major earthquake in the world from 1900 to 2015
## raw data accessed at http://earthquake.usgs.gov/earthquakes/search/
qk7freq <- c(3, 2, 4, 1, 2, 5, 8, 3, 2, 5, 5, 7, 3, 4, 6, 4, 8, 5, 12, 8, 7, 9,
             7, 12, 9, 12, 13, 11, 16, 15, 9, 19, 9, 8, 12, 14, 11, 9, 21, 14,
             7, 13, 11, 18, 13, 5, 10, 13, 11, 9, 13, 11, 7, 9, 6, 10, 8, 21, 8,
             8, 13, 12, 10, 17, 12, 18, 9, 11, 22, 14, 17, 20, 16, 9, 11, 13,
             14, 10, 12, 8, 6, 10, 7, 14, 14, 15, 11, 13, 11, 9, 18, 17, 13, 12,
             13, 20, 15, 16, 12, 18, 15, 16, 13, 15, 16, 11, 11, 18, 12, 17, 24,
             20, 16, 19, 12, 19)

## forward/backward probability for Poisson-HMM from
## Zucchini and MacDonald (2009):
## Hidden Markov Models for Time Series: An Introduction with R.
pois.HMM.lalphabeta <- function(x, m, lambda, gamma, delta = NULL) {
    if (is.null(delta))
        delta <- solve(t(diag(m) - gamma + 1), rep(1, m))
    n <- length(x)
    lalpha <- lbeta <- matrix(NA, m, n)
    allprobs <- outer(x, lambda, dpois)
    foo <- delta * allprobs[1, ]
    sumfoo <- sum(foo)
    lscale <- log(sumfoo)
    foo <- foo/sumfoo
    lalpha[, 1] <- log(foo) + lscale
    for (i in 2:n) {
```

```
        foo <- foo %*% gamma * allprobs[i, ]
        sumfoo <- sum(foo)
        lscale <- lscale + log(sumfoo)
        foo <- foo / sumfoo
        lalpha[, i] <- log(foo) + lscale
    }
    lbeta[, n] <- rep(0, m)
    foo <- rep(1/m, m)
    lscale <- log(m)
    for (i in (n - 1):1) {
        foo <- gamma %*% (allprobs[i + 1, ] * foo)
        lbeta[, i] <- log(foo) + lscale
        sumfoo <- sum(foo)
        foo <- foo/sumfoo
        lscale <- lscale + log(sumfoo)
    }
    list(la = lalpha, lb = lbeta)
}
```

2 Implementation

With the notation in the course note and Zucchini and MacDonald (2009), the complete-data-log-likelihood (CDLL) is

$$\log\left(\Pr(\boldsymbol{x}^{(T)}, \boldsymbol{c}^{(T)})\right) = \sum_{j=1}^{m} \mu_j(1) \log \delta_j + \sum_{j=1}^{m}\sum_{k=1}^{m}\left(\sum_{t=2}^{T} v_{jk}(t)\right) \log \gamma_{jk} + \sum_{j=1}^{m}\sum_{t=1}^{T} \mu_j(t) \log p_j(x_t).$$

2.1 E-step

$$\hat{\mu}_j(t) = \Pr(C_t = j|\boldsymbol{x}^{(T)}) = \alpha_t(j)\beta_t(j)/L_T,$$

and

$$\hat{v}_{jk}(t) = \Pr(C_{t-1} = j, C_t = k|\boldsymbol{x}^{(T)}) = \alpha_{t-1}(j)\gamma_{jk}p_k(x_t)\beta_t(k)/L_T,$$

where $L_T = \boldsymbol{\alpha}_t\boldsymbol{\beta}_t^{'}$.

2.2 M-step

We may find that the CDLL can be splited into three seperate maximizations, which lead to the closed form solutions as follows:

1. For $\boldsymbol{\delta}$:
$$\hat{\delta}_j = \arg\max \sum_{j=1}^{m} \hat{\mu}_j(1) \log \delta_j = \hat{\mu}_j(1) / \sum_{j=1}^{m} \hat{\mu}_j(1) = \hat{\mu}_j(1).$$
2. For $\boldsymbol{\Gamma}$:
$$\hat{\gamma}_{jk} = \arg\max \sum_{j=1}^{m}\sum_{k=1}^{m}\left(\sum_{t=2}^{T} \hat{v}_{jk}(t)\right) \log \gamma_{jk} = f_{jk} / \sum_{k=1}^{m} f_{jk},$$
where $f_{jk} = \sum_{t=2}^{T} \hat{v}_{jk}(t)$.

3. For $\boldsymbol{\lambda}$:

$$\hat{\lambda}_j = \arg\max \sum_{j=1}^{m} \sum_{t=1}^{T} \hat{\mu}_j(t) \log p_j(x_t) = \sum_{t=1}^{T} \hat{\mu}_j(t) x_t / \sum_{t=1}^{T} \hat{\mu}_j(t).$$

2.3 Function `poisson.hmm.em`

```
poisson.hmm.em <- function (x, m, lambda, Gamma, delta,
                            control = list(), ...) {
    control <- do.call(stats::glm.control, control)
    if (missing(lambda)) lambda <- seq(mean(x), by = 1, length.out = m)
    if (missing(Gamma)) Gamma <- matrix(1 / m, m, m)
    if (missing(delta)) delta <- rep(1 / m, m)
    n <- length(x)

    if (control$trace) {
        cat("Initial values:\n", sep = "")
        print(list(lambda = lambda, Gamma = Gamma, delta = delta))
        cat(rep("-", 70), "\n", sep = "")
    }

    for (iter in seq_len(control$maxit)) {
        est <- do1step(lambda, Gamma, delta, m, n, x)
        lambda_new <- est$lambda
        Gamma_new <- est$Gamma
        delta_new <- est$delta
        tol <- sum((lambda - lambda_new) ^ 2) / sum(lambda ^ 2) +
            sum((Gamma - Gamma_new) ^ 2) / sum(Gamma ^ 2) +
            sum((delta - delta_new) ^ 2) / sum(delta ^ 2)
        if (control$trace) {
            cat("\nIteration ", iter, ":\n", sep = "")
            print(est)
            cat(rep("-", 70), "\n", sep = "")
        }
        if (tol < control$epsilon) return(est)
        lambda <- lambda_new
        Gamma <- Gamma_new
        delta <- delta_new
    }
    cat("Not converge yet after", iter, "iterations.\n")
    invisible(est)
}

### internal function
do1step <- function (lambda, Gamma, delta, m, n, x) {
    ## E-step
    logPmat <- outer(x, lambda, dpois, log = TRUE)
    fooList <- pois.HMM.lalphabeta(x, m, lambda, Gamma, delta)
    la <- fooList$la
    lb <- fooList$lb
    const <- max(la[, n])
    logL_T <- const + log(sum(exp(la[, n] - const)))
    lmu <- la + lb - logL_T
```

```
    lv1 <- array(NA, c(n - 1, m, m))
    lb <- t(lb)
    la <- t(la)
    ind <- 2 : n
    for (j in seq_len(m)) {
        lv1[, j, ] <- la[ind - 1, j] + logPmat[ind, ] + lb[ind, ]
    }
    ## M-step
    muMat <- exp(lmu)
    delta <- muMat[, 1]
    delta <- delta / sum(delta)
    f_jk <- Gamma / exp(logL_T) * colSums(exp(lv1), dims = 1)
    Gamma <- f_jk / rowSums(f_jk)
    lambda <- rowSums(muMat * rep(x, each = m)) / rowSums(muMat)
    list(lambda = lambda, Gamma = Gamma, delta = delta)
}
```

3 Application to Earthquake Data

3.1 Two-state HMM model

We first consider a two-state HMM model for the earthquake data. The sample mean of the data is about 11. Thus, we set the initial parameter of the Poisson distribution to be 10, 30, respectively. For the transition probability matrix $\mathbf{\Gamma}$, the initial value is a two-by-two matrix with off-diagonal elements 0.1. The initial probability of each state is set to be equal, which is also the default value of the implementation here. The estimates are returned if the algorithm converges. The results are shown as follows:

```
(lambda2 <- round(mean(qk7freq)) + c(- 1, 19))
```

```
> [1] 10 30
```

```
Gamma2 <- matrix(c(0.9, 0.1, 0.1, 0.9), 2)
poisson.hmm.em(qk7freq, m = 2, lambda2, Gamma2)
```

```
> $lambda
> [1]  4.291757 12.788765
>
> $Gamma
>              [,1]       [,2]
> [1,] 9.442807e-01 0.05571926
> [2,] 2.396863e-10 1.00000000
>
> $delta
> [1] 1.000000e+00 3.846656e-86
```

3.2 Three-state HMM model

We further consider a three-state HMM model. The initial parameter of the Poisson distribution to be 10, 20, 30, respectively. The initial value of the transition probability matrix is a three-by-three matrix with off-diagonal elements 0.1. The initial probability of each state is set to be equal as well.

```
(lambda3 <- round(mean(qk7freq)) + c(- 1, 9, 19))
```

```
> [1] 10 20 30
```

```
Gamma3 <- diag(0.8, 3)
Gamma3[upper.tri(Gamma3)] <- Gamma3[lower.tri(Gamma3)] <- 0.1
poisson.hmm.em(qk7freq, m = 3, lambda3, Gamma3)
```

```
> Not converge yet after 25 iterations.
```

By default, the maximum iteration times is 25. Usually, we need more iterations for the EM algorithm to converge. We specify `maxit = 1000` in argument `control` to allow at most 1000 iterations before convergence. The algorithm converges this time and the estiamtes are returned as follows:

```
poisson.hmm.em(qk7freq, m = 3, lambda3, Gamma3, control = list(maxit = 1e3))
```

```
> $lambda
> [1]  4.090391 13.034579  8.426146
>
> $Gamma
>              [,1]         [,2]         [,3]
> [1,] 9.384028e-01 6.402882e-17 6.159716e-02
> [2,] 5.341616e-63 1.000000e+00 7.537227e-15
> [3,] 1.770390e-10 1.287034e-01 8.712966e-01
>
> $delta
> [1]  1.000000e+00  0.000000e+00 3.985406e-145
```

3.3 Trace the iterations

In addition, we may specify `trace = TRUE` in arugment `control` to trace each iteration towards convergence. For example, the iterations of the two-state are shown as follows:

```
est <- poisson.hmm.em(qk7freq, m = 2, lambda2, Gamma2,
                      control = list(maxit = 5, epsilon = 1e-3, trace = TRUE))
```

```
> Initial values:
> $lambda
> [1] 10 30
>
> $Gamma
>      [,1] [,2]
> [1,]  0.9  0.1
> [2,]  0.1  0.9
>
> $delta
> [1] 0.5 0.5
>
> ------------------------------------------------------------------------
>
```

```
> Iteration 1:
> $lambda
> [1] 11.14993 20.75852
>
> $Gamma
>           [,1]       [,2]
> [1,] 0.9785590 0.02144096
> [2,] 0.5872629 0.41273706
>
> $delta
> [1] 1.000000e+00 6.183461e-09
>
> ------------------------------------------------------------------------
>
> Iteration 2:
> $lambda
> [1] 10.87018 18.77050
>
> $Gamma
>           [,1]      [,2]
> [1,] 0.9642147 0.0357853
> [2,] 0.4114161 0.5885839
>
> $delta
> [1] 1.000000e+00 1.608176e-12
>
> ------------------------------------------------------------------------
>
> Iteration 3:
> $lambda
> [1] 10.39264 17.41285
>
> $Gamma
>           [,1]       [,2]
> [1,] 0.9450406 0.05495937
> [2,] 0.2723278 0.72767221
>
> $delta
> [1] 1.000000e+00 1.310326e-15
>
> ------------------------------------------------------------------------
>
> Iteration 4:
> $lambda
> [1]  9.733346 16.294775
>
> $Gamma
>           [,1]       [,2]
> [1,] 0.9287445 0.07125555
> [2,] 0.1725130 0.82748695
>
> $delta
> [1] 1.000000e+00 1.590155e-18
>
```

```
> ------------------------------------------------------------------------
>
> Iteration 5:
> $lambda
> [1]  9.087862 15.462311
>
> $Gamma
>           [,1]       [,2]
> [1,] 0.9153554 0.08464465
> [2,] 0.1216303 0.87836974
>
> $delta
> [1] 1.000000e+00 1.966311e-21
>
> ------------------------------------------------------------------------
> Not converge yet after 5 iterations.
```

Notice that the algorithm does not converge yet after only five iterations.

Reference

Zucchini, Walter, and Iain L MacDonald. 2009. *Hidden Markov Models for Time Series: An Introduction Using R*. CRC press.