Homework 6: Continuous HMM for profiling copy number alterations

GENOME 541: Cancer Genomics Module (Spring 2020)

Instructor: Gavin Ha

Due May 15, 11:59pm

In this assignment, you will implement a copy number alteration (CNA) caller. You will write functions for equations described in Lecture 3 for a standard, single-sample continuous HMM. For each function that you implement, you will use it to generate results for the first iteration of the EM algorithm. Then, you will implement the complete EM algorithm and run it until convergence to learn the parameters. Then, you will use the Viterbi algorithm to predict the copy number segments.

The input data is a text file `Homework6_log2ratios_chr1.txt` that contains a set of 411 genomic windows/bins in chr1 with normalized log2 ratios. I provided outputs for the first iteration of EM for each function so that you can use it to check your code.

Overall, your task is to implement the functions, run the EM algorithm to convergence, and then annotate the copy number status for each genomic bin. This assignment is divided into 4 parts:

0. Setup of the libraries, load the input data, and initialize parameter & hyperparameter settings
1. Implement the Gaussian emission likelihood
2. Implement the functions for use with the EM algorithm.
3. Run the EM algorithm for parameter inference and the Viterbi algorithm for copy number inference

There is a separate section for power analysis of mutation detection based on Lecture 4.

4. Power analysis for mutation detection when accounting for tumor heterogeneity.

Code template files `Homework6_SNVGenotyping_template.Rmd` and `Homework6_SNVGenotyping_template.ipynb` are provided for you to write your code and to generate outputs. This document contain the instructions for the assignment was generated using R Markdown, so the code syntax will be specific to R. However, the Python Jupyter notebook template contains the equivalent code in Python.

Please submit the following files:

1. The code as an R Markdown (`Homework6_code_FirstName-LastName.Rmd`) or Python Jupyter Notebook (`Homework6_code_FirstName-LastName.ipynb`) file.
2. The exported PDF (`Homework6_code_FirstName-LastName.pdf`) of the code, outputs, and plots.

0. Setup the libraries and input data

0.1 Install libraries

```r
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("HMMcopy")
```

0.2 Load libraries

```r
library(HMMcopy)
```

0.3 Load the input data

Load the input data from `Homework6_log2ratios_chr1.txt`. This file contains T bins (1Mb) where each bin t has a read count (`readCount`) and a corrected $\log_2$ ratio (`log2Ratio`). The column `log2Ratio` is the data, $x_{1:T}$, that we will use for this assignment.

```r
LogRatios <- read.delim("Homework6_log2ratios_chr1.txt", as.is = TRUE)
x <- as.matrix(LogRatios$log2Ratio) # input data
copyNumberStates <- c(1,2,3,4,5)
K <- length(copyNumberStates) # number of states
```

0.4. Initialize model parameters and hyperparameters.

Initialize parameters and hyperparameters for copy number states, $\{1, 2, 3, 4, 5\}$

```r
##### initial values for model parameters #####
pi.init <- c(0.1, 0.6, 0.1, 0.1, 0.1) # initial setting for pi (initial state distribution)
mu.init <- log2(c(copyNumberStates) / 2) # inital setting for Gaussian mean
var.init <- rep(var(x), times = K) # initial setting for Gaussian variance

A.init <- matrix(0, K, K) # initial setting for matrix of transition probabilities
for (i in 1:K) {
  A.init[i, ] <- (1 - 0.99999) / (K - 1) # transition to different state
  A.init[i, i] <- 0.99999  # self-transition probability
}

##### hyper-parameters for prior model #####
deltaPi.hyper <- c(2, 6, 2, 2, 2) # hyperparameter for Dirichlet prior on pi parameter

mMu.hyper <- mu.init # hyperparameter for Gaussian prior on mu parameter
sMu.hyper <- var.init # hyperparameter for Gaussian variance on mu parameter

# hyperparameters for Inverse Gamma prior on variance parameter
betaVar.hyper <- c(1,1,1,1,1)
alphaVar.hyper <- betaVar.hyper/ (apply(x, 2, var, na.rm = TRUE) / sqrt(K))

# hyperparameters for Dirichlet prior on the transition matrix
dirCounts <- 100000
deltaA.hyper <- A.init * dirCounts
```

1. Compute the Gaussian Emission Probabilities

1.1. Define a function to compute the likelihood probabilities.

This function, called `compute.gauss.lik`, will compute the Gaussian probability for each locus i and (conditional for) each copy number state $k \in \{1, 2, 3, 4, 5\}$.

$$p(x_t|Z_t = k, \mu_{1:K}, \sigma^2_{1:K}) = \mathcal{N}(x_t|\mu_k, \sigma^2)$$

You can use the built-in functions such as `dnorm` or implement the Gaussian probability density function ($\mathcal{N}(\mu, \sigma^2)$).

```
compute.gauss.lik <- function(){

}
```

1.2. Compute the Gaussian likelihood

Use the `compute.gauss.lik` function you wrote in Section 1.1 to compute the Gaussian probabilities for $x_{1:T}$ of the first EM iteration. Use the initial Gaussian parameters set in Section 0.4, `mu.init` and `var.init`.

Print the first 3 columns (i.e. for data points $x_{1,\ldots,3}$ and all states k) of probabiliities for the observed likelihood.

To check your code, here are the values you should see in your output.

```
##            [,1]       [,2]        [,3]
## [1,] 0.05614798 0.12623994 0.181602980
## [2,] 0.81848272 0.82875528 0.788168624
## [3,] 0.50497414 0.32064768 0.239377684
## [4,] 0.14316797 0.06528551 0.041046514
## [5,] 0.03186925 0.01124119 0.006186008
```

2. Implement functions for EM algorithm

2.1. Compute the responsibilities in the E-Step

Use the compiled C code from `HMMcopy` to compute the *forwards backwards* probabilities (responsibilities). This function computes the probability $\gamma(Z_t)$ of each locus t having every possible genotype $\forall k \in \{1,2,3,4,5\}$. The probabilities are computed for each locus t and each genotype k and returns a matrix with dimensions $K \times T$.

This function takes as arguments:

- `obs.lik`, the likelihood (binomial probabilities) computed from `compute.gauss.lik`
- `piZ`, the probability of the genotypes $\pi_{1:K}$
- `A`, the matrix of transition probabilities A

```
fwdback <- .Call("forward_backward", piZ, A, obs.lik, PACKAGE = "HMMcopy")
```

The call to this compiled C code will return a named list of elements, including:

- `rho`, responsibilities $\gamma(Z_t)$
- `xi`, 2 slice marginals $\xi(Z_{t-1}, Z_t)$
- `loglik`, log likelihood ℓ

To try out this function, compute the responsibilities using the initial settings of the parameters `pi.init`, `A.init` and Gaussian probabilities computed in Section 1.2, `obs.lik`. This is the basically the E-Step in the first iteration of EM.

Print out the log likelihood and the first 3 columns of the responsibility matrix ($\gamma(Z_{1:3})$).

Here is the output.

```
fwdback$loglik
```

```
## [1] -168.3382
```

```
fwdback$rho[, 1:3] #gamma
```

```
##              [,1]         [,2]         [,3]
## [1,] 3.409871e-08 5.516394e-09 1.163045e-09
## [2,] 9.999995e-01 9.999998e-01 9.999999e-01
## [3,] 4.060840e-07 1.490170e-07 4.955618e-08
## [4,] 7.895511e-08 6.072052e-09 3.304384e-10
## [5,] 1.644575e-08 2.219168e-10 1.811563e-12
```

2.2. Updating the initial state distribution parameter $\pi_{1:K}$ in the M-Step

2.2.1. Write a function to update the initial state distribution parameter

Write a function, called `update.pi`, that computes the MAP estimate update of π_k given the responsibilities $\gamma(Z_t)$ from the E-Step and the hyperparameter δ_k^π

$$\hat{\pi}_k = \frac{\gamma(Z_1 = k) + \delta^\pi(k) - 1}{\sum_{j=1}^{K} \{\gamma(Z_1 = j) + \delta^\pi(j) - 1\}}$$

The function should return a vector $\hat{\pi}_{1:K}$ with K elements, one value for each copy number state $k \in \{1,2,3,4,5\}$.

```r
update.pi <- function(){

}
```

2.2.2. Compute $\hat{\pi}$ for the first iteration of EM

Use `update.pi` to compute $\hat{\pi}$ for the first iteration of EM. Print out the values of $\hat{\pi}_{1:K}$.

Here is the output.

```
## [1] 0.1000000 0.5999999 0.1000000 0.1000000 0.1000000
```

2.3. Updating the Gaussian mean parameter $\mu_{1:K}$ in the M-Step

2.3.1. Write a function to update Gaussian mean parameter

Write a function, called `update.mu`, that computes the MAP estimate update of μ_k given the responsibilities $\gamma(Z_t)$ from the E-Step and the hyperparameters m_k and s_k

$$\hat{\mu}_k = \frac{s_k \sum_{t=1}^{T} \gamma(Z_t = k) x_t + m\sigma_k^2}{s_k \sum_{t=1}^{T} \gamma(Z_t = k) + \sigma_k^2}$$

The function should return a vector $\hat{\mu}_{1:K}$ with K elements, one value for each copy number state $k \in \{1, 2, 3, 4, 5\}$.

```r
update.mu <- function(){

}
```

2.3.2. Compute $\hat{\mu}$ for the first iteration of EM

Use `update.mu` to compute $\hat{\mu}_{1:K}$ for the first iteration of EM. Print out the values of $\hat{\mu}_{1:K}$.

Here is the output.

```
##             [,1]
## [1,] -0.74128990
## [2,] -0.06420233
## [3,]  0.41098632
## [4,]  0.99999410
## [5,]  1.32192731
```

2.4. Updating the Gaussian variance parameter $\sigma^2_{1:K}$ in the M-Step

2.4.1. Write a function to update Gaussian variance parameter

Write a function, called `update.var`, that computes the MAP estimate update of σ_k^2 given the responsibilities $\gamma(Z_t)$ from the E-Step and the hyperparameters m_k and s_k

$$\hat{\sigma}_k^2 = \frac{\sum_{t=1}^{T} \gamma(Z_t = k) \left(x_t - \bar{x}_k\right)^2 + 2\beta_k}{\sum_{t=1}^{T} \gamma(Z_t = k) + 2(\alpha_k + 1)}$$

where $\bar{x} = \frac{\sum_{t=1}^{T} \gamma(Z_t = k) x_t}{\sum_{t=1}^{T} \gamma(Z_t = k)}$

The function should return a vector $\hat{\sigma}^2_{1:K}$ with K elements, one value for each copy number state $k \in \{1, 2, 3, 4, 5\}$.

```
update.var <- function(){

}
```

2.4.2. Compute $\hat{\sigma}^2$ for the first iteration of EM

Use `update.mu` to compute $\hat{\sigma}^2_{1:K}$ for the first iteration of EM. Print out the values of $\hat{\sigma}^2_{1:K}$.

Here is the output.

```
## [1] 0.06542300 0.02591266 0.04337017 0.09182823 0.09182825
```

2.5. Updating the transition probabilities A in the M-Step

2.5.1. Write a function to update the transition probabilities

Write a function, called `update.A`, that computes the update of A given the 2-slice marginal probabilities $\xi(Z_{t-1}, Z_t)$ from the E-Step and the hyperparameter δ^A_k

$$\hat{A}_{jk} = \frac{\sum_{t=2}^{T} \xi(Z_{t-1} = j, Z_t = k) + \delta^A(k)}{\sum_{k\prime=1}^{K} \left\{ \sum_{t=2}^{T} \xi(Z_{t-1} = j, Z_t = k\prime) + \delta^A(k) \right\}}$$

The function should return a matrix $\hat{A}$ with $K \times K$ elements, each being the probability of transition from copy number state i at bin t (rows) to state j at bin $t + 1$ (columns) for each copy number state $k \in \{1, 2, 3, 4, 5\}$.

```
update.A <- function()){

}
```

2.5.2. Compute $\hat{\boldsymbol{A}}$ for the first iteration of EM

Use `update.A` to compute matrix $\hat{\boldsymbol{A}}$ for the first iteration of EM. Print out the values of $\hat{\sigma}^2_{1:K}$.

Here is the output.

```
##              [,1]         [,2]         [,3]         [,4]         [,5]
## [1,] 9.999800e-01 3.790994e-06 1.119623e-05 2.497885e-06 2.497873e-06
## [2,] 1.248252e-05 9.999787e-01 3.788882e-06 2.496488e-06 2.496484e-06
## [3,] 1.247694e-05 2.495523e-06 9.999800e-01 2.495432e-06 2.495414e-06
## [4,] 2.500022e-06 2.500001e-06 2.500018e-06 9.999900e-01 2.500000e-06
## [5,] 2.500003e-06 2.500000e-06 2.500003e-06 2.500000e-06 9.999900e-01
```

2.6. Compute the log posterior

2.6.1. Define the function, `compute.log.posterior`, to compute the log posterior distribution.

The log posterior distribution is used to monitor the convergence of EM at each iteration. This value is the sum of the log likelihood and the log of the priors in the model. The log likelihood is obtained from the Forwards-Backwards algorithm through the sum of the scaling factors for the forward recursion probabilities, $\sum_{t=1}^{T} \log \sum_{k=1}^{K} \alpha(Z_t = k)$.

$$\log \mathbb{P} = \ell + \log Dirichlet(\hat{\boldsymbol{\pi}}|\boldsymbol{\delta}) + \sum_{k=1}^{K} \left\{ \log \mathcal{N}(\hat{\mu}_k|m_k, s_k) + \log InvGamma(\hat{\sigma}_k^2|\alpha_k, \beta_k) + \log Dir(A_{k,1:K}^{(0)}|\hat{A}_{k,1:K}) \right\}$$

Note that the $Dir(A_{k,1:K}^{(0)}|\hat{A}_{k,1:K})$ corresponds to the k^{th} row and $A^{(0)}$ is the initial settings of the transition matrix, `A.init`.

```
compute.log.posterior <- function(){

}
```

2.6.2. Compute the log posterior for the input data

Use the `compute.log.posterior` function to compute the log posterior for the first iteration of EM. Use the log likelihood computed by the forwards-backwards algorithm (Section 2.1), `loglik`, and the updated parameters (from Sections 2.2.2, 2.3.2, 2.4.2, 2.5.2), $\hat{\pi}_{1:K}$, $\hat{\mu}_{1:K}$ and $\hat{\sigma}^2_{1:K}$, and $\hat{\boldsymbol{A}}$. Use the hyperparameters, `deltaPi.hyper`, `mMu.hyper`, `sMu.hyper`, `alphaVar.hyper`, `betaVar.hyper`, `A.init` for the priors that you set in Section 0.4.

Print out the log posterior for the first iteration of EM.

Here is the output.

```
## [1] -163.7345
```

This function should return a single number that will be used to monitor the EM algorithm for convergence.

3. Learn the HMM Parameters and Predict the Copy Number Segments.

3.1. Implement and run the EM algorithm to infer the copy number states and learn the parameters.

Implement the full EM algorithm for inferring the responsibilities and estimating the HMM parameters in a Bayesian framework.

Refer to Lecture 3 slide 22 for the EM algorithm. Run the EM algorithm until convergence, which is when the log posterior changes by less than $\epsilon = 10^{-2}$.

i. Print out converged parameters:
   - the Gaussian mean and variance parameters $\hat{\mu}_{1:K}$ and $\hat{\sigma}^2_{1:K}$
   - the initial state distribution $\hat{\pi}_{1:K}$
   - the transition matrix $\hat{\boldsymbol{A}}$
ii. Print out the log posterior for all iterations.
iii. Save the Gaussian likelihood probabilities `obs.lik` computed in the final iteration. You will need this for the Viterbi algorithm in Section 3.2.

```
pi.hat
```

```
##         1         2         3         4         5
## 0.1000000 0.5999999 0.1000001 0.1000000 0.1000000
```

```
mu.hat
```

```
##           1           2           3           4           5
## -0.78488116 -0.06697167  0.42999749  0.99992863  1.32192778
```

```
var.hat
```

```
##          1          2          3          4          5
## 0.04385671 0.02261849 0.03260682 0.09182814 0.09182825
```

```
A.hat
```

```
##              [,1]         [,2]         [,3]         [,4]         [,5]
## [1,] 9.999689e-01 1.270587e-05 1.343554e-05 2.498104e-06 2.497990e-06
## [2,] 2.362673e-05 9.999598e-01 1.153605e-05 2.496228e-06 2.496226e-06
## [3,] 1.247785e-05 1.247749e-05 9.999701e-01 2.495560e-06 2.495551e-06
## [4,] 2.500000e-06 2.500124e-06 2.500004e-06 9.999900e-01 2.500000e-06
## [5,] 2.500000e-06 2.500002e-06 2.500000e-06 2.500000e-06 9.999900e-01
```

```
logP
```

```
##  [1] -163.73450   99.84178  107.69031  108.82523  110.64413  115.49706
##  [7]  118.79024  120.75264  120.89195  120.92440  120.93219
```

3.2. Determine the copy number segments using the Viterbi algorithm

Use the compiled C code from the `HMMcopy` R package to compute the optimal sequence of copy number states. Note that since the Viterbi algorithm is also known as the max-sum algorithm, computation can be performed in log space.

This function takes as arguments:

- `log(pi.hat)`, the converged initial state distribution in log space, $\log \hat{\pi}_{1:K}$
- `log(A.hat)`, the converged matrix of transition probabilities $\log A$
- `log(obs.lik)`, the Gaussian likelihood probabilities in log space from the final iteration of EM, $\log \mathcal{N}(x_t|\mu_k, \sigma^2)$

```
states <- .Call("viterbi", log(pi.hat), log(A.hat), log(obs.lik), PACKAGE = "HMMcopy")$path
```

The call to this compiled C code will return a named list of elements, including the sequence of copy number states (`path`).

i. Run this function to obtain the sequence of copy number states using the converged parameters and Gaussian likelihood (from the final EM iteration) computed in Section 3.1 as input.

ii. Print out a table of the copy number segments by combining the output from the Viterbi algorithm with the original input file. A segment is defined as the consecutive set of bins with the same copy number state. Use the original input file `LogRatios` to determine the `start` and `end` coordinates and the median `log2Ratio` for the segment. The table should have the following columns:

    - `chr`, chromosome of the segment
    - `start`, start genomic coordinate of the segment
    - `end`, end genomic coordinate of the segment
    - `medianLog2Ratio`, median `log2Ratio` of the segment
    - `CopyNumber`, copy number state of the segment

Here are the final segment results for this assignment. Notice that these results correspond to chromosome 1 for Patient 288 (time point 1) from the lecture notes.

```
##     chr     start       end medianLog2Ratio CopyNumber
## 1:    1   1500001  76000000     -0.07011075          2
## 2:    1  76000001 108500000     -0.81098377          1
## 3:    1 109000001 112000000     -0.10203634          2
## 4:    1 112000001 114500000     -0.84934526          1
## 5:    1 114500001 117000000      0.46157140          3
## 6:    1 117000001 147000000     -0.01976506          2
## 7:    1 150000001 241000000      0.39820292          3
## 8:    1 241000001 248500000     -0.74545984          1
```

4. Estimate Power for Mutation Detection

4.1. Implement the function to calculate power.

Implement a function to calculate the theoretical mutation detection power (sensitivity) for observing ≥ 3 variant reads with read deptn N, copy number c, variant multiplicity M, and tumor fraction α.

$$power = p(X \geq 3) = 1 - [Bin(0|N, \mu) + Bin(1|N, \mu) + Bin(2|N, \mu)]$$

where

$$\mu = \frac{\alpha M}{\alpha c + 2(1 - \alpha)}$$

```r
computePower <- function(){

}
```

For all following questions, use copy number $cn = 2$ and multiplicity $M = 1$.

4.2. What is the power for depth N=40 and tumor fraction $\alpha = 0.2$.

4.3. Plot a graph of the power estimates (y-axis) for a range of tumor fractions (x-axis).

Compute the power for tumor fraction $\alpha = \{0.0, ..., 1.0\}$ at 0.1 increments and depth $N = 40$. Plot the curve.

Note that this type of plot informs the power for detecting a mutation based on a range of heterogeneity (i.e. tumor fraction) values.

4.4. Plot a graph of the power (y-axis) required for a range of read depths (x-axis).

Compute the power for read depths of $N = \{0, \ ..., \ 100\}$ at 1 increments when $\alpha = 0.2$ and for $\alpha = 0.5$; show one curve for each tumor fraction α value.

Note that this plot is intended to help decide the depth of coverage we should aim to sequence when accounting for the variable heterogeneity.

4.5. What is the minimum depth required for tumor fraction $\alpha = 0.1$ and desired power≥ 0.8?

For deciding the depth of sequencing, here, we assume 80% power is sufficient but one can choose to increase it for specific applications.