Optimización II - Proyecto - Maximum Likelihood Estimation of Mixture Proportions

Joel Chacón Castillo

Guanajuato, México

1. Introducción

Los procedimientos para la estimación de máxima verosimilitud de las proporciones de mezclas tienen una gran relevancia en la estadísticas, principalmente en los métodos basados en los enfoques no parámetricos empíricos de bayes (Empirical Bayes - EB). A pesar de que este problema ya se ha resuelto por medio de un algoritmo EM (Estimatión - Maximization), en varios trabajos como el desarrollado por Koenker and Mizera [1]se ha demostrado que los métodos modernos de optimización convexa pueden ser mas rápidos y eficientes. En particular, Koenker et al. utilizaron un método de punto interior para resolver la formulación dual del problema de optimización origina. Principalmente, en este proyecto se implementa el trabajo *A Fast Algorithm for Maximum Likelihood Estimation of Mixture Proportions Using Sequential Quadratic Programming* desarrollado por Kim et al. [2]. En este trabajo se desarrolla un nuevo enfoque de optimización, el cual es basado en Programación Secuencial Cuadrática (Sequential Quadratic Programming - SQP). Específicamente se combinan las siguientes ideas:

- Resuelve una reformulación del problema primal original en lugar de la formulación dual.
- Utiliza el un método SQP en lugar de un método de punto iterior (Interior Point - IP), con el propósito de aprovechar el cálculo del Hesiano y del gradiente.
- Utiliza un método de de los conjuntos activos para resolver el subproblema QP como parte del algoritmo SQP, esto con el propósito de explotar la forma ralas del problema.

- Utiliza aproximaciones rango bajo para acelerar el cálculo del Hesiano y el gradiente.

El código de este trabajo está en el lenguaje Julia (https://julialang.org/) y esta disponible en linea (https://github.com/stephenslab/mixsqp-paper). Sin embargo, en este proyecto se desarrolla en python este mismo enfoque, donde en lugar de utilizar el método de los conjuntos activos se implementa el procedimiento de punto interior (Mehrotra) visto en clase.

En general se considera la estimación de máxima verosimilitud de las proporciones de mezclas en un modelo de mezclas finitas donde los componentes de las densidades son conocidas. La formulación más sencilla es al considerar observaciones independientes e identicamente distribuidas (i.i.d.) $z_1, ..., z_n$ de una distribución de mezclas finitas con la siguiente densidad.

$$p(.|x) = \sum_{k=1}^{m} x_k g_k(.), \tag{1}$$

donde se conocen los componentes de las densidades $g_k(.)$ y no se conocen las proporciones de mezclas $x = (x_1, ..., x_m)^T$ (las cuales son no negativas y suman uno). Para encontrar el estimador de máxima verosimilitud (Maximum Likelihood Estimate - MLE) de x se describe a continuación.

$$\begin{aligned} &minimize \quad f(x) = -\frac{1}{n}\sum_{j=1}^{n} log(\sum_{k=1}^{m} L_{jk}x_k) \\ &subject \quad to \quad x \in S^m = \{x : \sum_{k=1}^{m} x_k = 1, x \preceq 0\} \end{aligned} \tag{2}$$

donde $L_{jk} = g_k(z_j) \geq 0$. La ecuación 2 indica un problema de optimización convexo y puede ser resuelto por medio del algoritmo *Expectation Maximization* (EM). Sin embargo, el algoritmo EM puede tener problemas de convergencia y además puede ser muy lento al considerar problemas compuestos por muchas mezclas de distribución. Por otra parte los autores Koenker and Mizera (2014) y Gu and Koenker (2017), resaltaron que algunos métodos de optimización convexa puede ser mas eficientes que el algoritmo EM. Ellos demostraron esto utilizando un método de punto interior para resolver la formulación dual del problema original.

2. Enfoque de basado en en Programación Cuadrática Secuencial (SQP - Sequential Quadratic Programming)

2.1. Reformulación

Para resolver el problema principal indicado en la ecuación 2 se han ideado varias reformulaciones, una reformulación popular, es la indicada por Gu and Koenker, donde se plante el siguiente problema primal.

$$minimize \quad -\frac{-1}{n}\sum_{j=1}^{n} log(y_j) \quad subject \quad to \quad Lx = y, \quad x \in S^m \tag{3}$$

entonces se procede a resolver la forma dual que se indica a continuación

$$minimize \quad -\frac{1}{n}\sum_{j=1}^{n} log(v_j) \quad subject \quad to \quad L^T v \preceq n\mathbf{1}_m, \quad v \in \Re^n \succeq 0 \tag{4}$$

donde $\mathbf{1}$ es un vector compuesto por uno de dimensión m. En esta última formulación dual los autores Koenker and Mizera reportaron una mayor rapidez que las formulaciones primales. Sin embargo, esta rapidez está afectada en el caso en que la cantidad de datos n es superior a la cantidad de funciones de densidad a considerar m ($n \gg m$). La formulación más reciente (se indica en la siguiente sección) [2] es diseñada en base a los siguientes dos principios:

- Hacer el mejor uso posible del Hesiano con el propódito de minimizar el número de evaluciones.
- Reducir el costo de calcular el Hesiano tanto como sea posible.

Para utilizar de forma adecuada el Heasiano se aplica Programación Cuadrática Secuencial (SQP). Esta reformulación relaja las restricciones del simplejo a una menos restrictiva en la no-negatividad, que también simplifica los cálculos. Para reducir el costo en el cálculo del Hesiano, implementa una factorización conocida como *rank-revealing* (RRQR) que consiste en una derivación de la factorización QR que también puede ser utilizado para determinar el rango de una matriz, aunque también se puede aplicar la descomposción SVD, esta última técnica no se considera como un método eficiente. Por lo tanto el cálculo del Hesiano es reducido de $O(nm^2)$ a $O(nr^2)$, donde n es el número de datos, m es el número de funciones de densidad a considerar y r es el rango.

2.2. Reformulación

La reformulación de la ecuación 2 es uno más simple con restricciones menos restrictivas a la no negatividad se basan en la siguiente definición y teorema.

Definition 1. *Se dice que una función* $\Phi : \Re_+^m \longrightarrow \Re$ *es invariantemente escalable si para cualquier* $c > 0$ *existe una* $C \in \Re$ *tal que para cualquier* $x \in \Re_+^m$ *tenemos un* $\Phi(cx) = \Phi(x) - C$.

Theorem 1. *Considerando el problema de optimización con restricciones en el simplejo*

$$minimize \quad \Phi(x) \quad subject \quad to \quad x \in S^m \tag{5}$$

donde $\Phi(x)$ *es invariantemente escalable, convexa, y no creciente con respectoa* x*, esto es* $x \preceq y$*, esto implica que* $\Phi(x) \leq \Phi(y)$ *para todos los* $x \in \Re_+^m$*. Además,* $x^*(\lambda)$ *denota la solución (parcial) del problema Lagrangiano relajado de la ecuación (5), como se indica a continuación.*

$$minimize \quad \Phi_\lambda(x) = \Phi(x) + \lambda \sum_{k=1}^{m} x_k \quad s.t. \quad x \in \Re_+^m = \{x \in \Re^m : x \succeq 0\} \tag{6}$$

Entonces $x^*)x^*(\lambda)/\sum_{k=1}^m x_k^*(\lambda)$ *es una solución al problema de la ecuación (5). Entonces* $x^* = \frac{x^*(\lambda)}{\sum_{k=1}^m x_k^*(\lambda)}$ *es una solución de de la ecuación (5).*

Corollary 1. *El problema de optimización inicial indicado por la ecuación 2 puese ser resuelto si en su lugar se resuelve el problema de la ecuación 6 con* $\Phi(x) = f(x)$.

$$minimize \quad f^*(x) = f(x) + \sum_{k=1}^{m} x_k \quad s.t. \quad x \succeq 0 \tag{7}$$

Además, en este caso asignando $\lambda = 1$ *proporcionauna solución que es normalizada, es decir,* $x^* = x^*(\lambda)$ *cuando* $\lambda = 1$.

2.3. Programación cuadrática secuencial

El problema de la ecuación 7 se resuelve por medio de un algoritmo SQP y además considerando busqueda lineal basada en *backtracking*. El procedimiento SQP es un algoritmo iterativo, donde en cada iteración se formula

una aproximación de segundo orden de la función objetivo dado un punto factible x^t, entonces se determina la dirección p^t en base a la aproximación de segundo orden. La dirección de búsqueda p^t es obtenida al resolver el siguiente program cuadrático con restricciones no negativas tambień llamado como "Subproblema-QP".

$$p^t = arg \quad min \quad \frac{1}{2} p^T H_t p + p^T g_t \quad s.t. \quad x^t + p \succeq 0 \tag{8}$$

donde $g_t = \nabla f^*(x^t)$ y $H_t = \nabla^2 f^*(x^t)$ don el gradiente y el Hesiano de $f^*(x)$ en x^t. Después de identificar la dirección de búsqueda se implementa una búsqueda lineal con backtracking para determinar el tamaño de paso suficiente $x^{t+1} = x^t + \alpha_t p^t$ para $\alpha_r \in [0, 1]$

2.4. Gradiente y Hesianos

Lemma 1. *Para todos $x \in \Re^m_+$, el gradiente y el Hesiano de la función objetivo indicada en la ecuación 7 en x son como si indica a continuación.*

$$g = -\frac{1}{n} L^T d + \mathbf{1}_m, \quad H = \frac{1}{n} L^T diag(d)^2 L \tag{9}$$

donde $\mathbf{1}_m$ es un vector compuesto por unos de dimención m y donde $d = (d_1, ..., d_n)^T$ es un vector columna con entradas $d_j = 1/(Lx)_j$. Además, para todo $x \in \Re^m_+$ se tiene que

$$x^T g = 0 \quad x^T H x = 1 \quad Hx + g = \mathbf{1}_m \tag{10}$$

En la práctica la matrix L es de rango menor $r < m$, por lo tanto se aproxima esta matriz por medio de la factorización *rank-revealing* QR - *RRQR*. Donde se tiene que la proximación de L es $\hat{L} = QRP^T$, con $Q \in \Re^{n \times r}$, $R \in \Re^{r \times m}$ y la matriz de permutación $P \in \Re^{m \times m}$, además el gradiente y el Hesiano pueden ser aproximados de la siguiente forma.

$$\hat{d}_j = 1/(QRP^T x)_j, \quad \hat{g} = \frac{-1}{n} PR^T Q^T d + \mathbf{1}_m, \quad \hat{H} = \frac{1}{n} PR^T Q^T diag(d)^2 QRP^T \tag{11}$$

2.5. Subproblema de programación cuadrática

Para resolver el problema indicado en la ecuación 8 se aplica el siguiente problema equivalente.

$$y^* = arg_y min \frac{1}{2} y^T H_t y + y^T a_t \quad s.t. \quad y \succeq 0 \tag{12}$$

donde $a_t = -H_y x^t + g_t$. Se deriva este problema sustituyendo $y = x^t + p$ en la ecuación 8. Además se puede observar que el problema indicado en la ecuación 8 se puede resolver asignando $p^* = y^* - x^t$. Por otra parte se considera que las restricciones de no-negatividad hacen que el problema se pueda resolver de una forma más sencilla. Dado que en la práctica la matriz L suele se ralas, podría ser conveniente resolver el subproblema por medio del método de conjuntos activos, sin embargo en la implementación de este trabajo sólo se considera el método de punto interior, no obstante se puede consultar el codigo del autor que esta en el lenguaje Julia.

2.6. El algoritmo MIX-SQP

Finalmente se unen todos los puntos mencionados en las secciones anteriores y como resultado se tiene el siguiente algoritmo.

Algorithm 1 MIX - SQP

Inputs: likelihood matrix $L \in \Re^{n \times m}$, initial iterate $x^{(0)} \in \Re^m_+$, sufficient decrease parameter $0 < \psi < 1$ (default is 0.5), step size reduction $0 < \rho < 1$ (default is 0.5), convergence tolerance $\epsilon > 0$ (default is 10^{-8}).
$Q, R, P = pqrfact(L)$
for t =0,1,2,... **do**
  $d_t = 1/(QRP^T x^t)$
  $g_t = \frac{1}{n} PR^T Q^T d_t + \mathbf{1}$
  $H_t = \frac{1}{n} PR^T Q^T diag(d_t)^2 QRP^T$
  $p_t = IP - Procedure(x^t, g_t, H_t)$
  if $min_k |(g_t)_k| < \epsilon$ and $||p_t|| < \epsilon$ **then**
    break
  $\alpha_t = 1$
  while $f(x^t + \alpha_t p_t) > f(x^t) + \alpha_t \psi p_t^T g_t$ **do**
    $\alpha_t = \rho \alpha_t$
  $x^{t+1} = x^t + \alpha_t p_t$
return x^t

3. Método de punto interior

Los métodos de punto interior pueden ser aplicados a problemas de programación cuadrática por medio de algoritmos lineales simples. En particular, el problema a tratar es de programación cuadrática convexa con desigualdades de restricción de la forma en que se indica a continuación:

$$min_x \quad q(x) = \frac{1}{2} x^T G x + x^T c \quad s.a. \quad Ax \geq b \tag{13}$$

donde G es una matriz simétrica positiva semidefinida ($x^T G x \geq 0$), además A $[m \times n]$ y b corresponden a las restricciones de desigualdad. Las condiciones

KKT son de la forma:

$$\begin{aligned} &Gx - A^T\lambda + c = 0 \\ &Ax - b \geq 0 \\ &(Ax - b_i)\lambda_i = 0 \quad \forall i \in 1, ..., m \\ &\lambda \geq 0 \end{aligned} \tag{14}$$

Agregando el vector slack s se tiene:

$$\begin{aligned} &Gx - A^T\lambda + c = 0 \\ &Ax - y - b \geq 0 \\ &y_i\lambda_i = 0 \quad \forall i \in 1, ..., m \\ &(y_i, \lambda) \geq 0 \end{aligned} \tag{15}$$

La medición de complementariedad μ es definido de la forma:

$$\mu = \frac{y^T\lambda}{m} \tag{16}$$

Además considerando las condiciones de KKT perturbadas:

$$F(x, y, \lambda; \sigma, \mu) = \begin{pmatrix} Gx - A^T\lambda + c \\ Ax - y - b \\ Y\Lambda e - \sigma\mu e \end{pmatrix} = \mathbf{0} \tag{17}$$

donde $Y = diag(y_1, .., y_m)$, $\Lambda = diag(\lambda_1, ..., \lambda_m)$, $e = (1, ..., 1)^T$, y $\sigma \in [0, 1]$, todos los valores positivos σ, μ definen la ruta central, que es la trayectoria en que la solución del problema cuadrático tiende a cero. Fijando el parámetro μ y aplicando el método de Newton se obtiene el sistema lineal siguiente:

$$\begin{bmatrix} G & 0 & -A^T \\ A & -I & 0 \\ 0 & \Lambda & Y \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_y \\ \Delta_\lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p \\ -\Lambda Y e + \sigma\mu e \end{bmatrix} \tag{18}$$

donde $rd = Gx - A^T\lambda + c$ y $rp = Ax - y - b$, de forma iterativa se obtiene que $(x^+, y^+, \lambda^+) = (x, y, \lambda) + \alpha(\Delta_x, \Delta_y, \Delta_\lambda)$, donde se escoge un α la cual mantenga la desigualdad $(y^+, \lambda^+) > 0$

Algorithm 2 Predicto-Corrector Mehrotra - SQP

1: Cacular (x_0, y_0, λ_0) con $(y_0, \lambda_0) > 0$
2: **for** k=1,2,.., **do**
3: Resolver

$$\begin{bmatrix} G & 0 & -A^T \\ A & -I & 0 \\ 0 & \Lambda & Y \end{bmatrix} \begin{bmatrix} \Delta_x^{aff} \\ \Delta_y^{aff} \\ \Delta_\lambda^{aff} \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p \\ -\Lambda Y e \end{bmatrix} \tag{19}$$

donde $r_d = Gx - A^T\lambda + c$, $r_p = Ax - y - b$
4: Calcular $\mu = \frac{y^T\lambda}{m}$
5: $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha(\Delta_x, \Delta_\lambda, \Delta_s)$
6: Calcular $\alpha_{aff}^{pri}, \alpha_{aff}^{dual}$ y μ_{aff} de la siguiente forma

$$\begin{aligned} \alpha_{aff}^{pri} &= min(1, min_{i:\Delta y_i^{aff}<0} - \frac{y_i}{\Delta y_i^{aff}}) \\ \alpha_{aff}^{dual} &= min(1, min_{i:\Delta \lambda_i^{aff}<0} - \frac{\lambda_i}{\Delta \lambda_i^{aff}}) \\ \alpha &= max(\alpha_{aff}^{pri}, \alpha_{aff}^{dual}) \\ \mu_{aff} &= (y + \alpha\Delta y^{aff})^T(\lambda + \alpha\Delta\lambda^{aff})/n \\ \sigma &= (\frac{\mu_{aff}}{\mu})^3 \end{aligned} \tag{20}$$

7: Resolver

$$\begin{bmatrix} G & 0 & -A^T \\ A & -I & 0 \\ 0 & \Lambda & Y \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_y \\ \Delta_\lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p \\ -\Lambda Y e - \Delta\Lambda^{aff}\Delta Y^{aff} e + \sigma\mu e \end{bmatrix} \tag{21}$$

donde $r_d = Gx - A^T\lambda + c$, $r_p = Ax - y - b$
8: Calcular $\alpha_{max}^{pri} = min_{i:\Delta y_i<0} - \frac{y_i - (1-\tau)y_i}{\Delta y_i}$ y $\alpha_{max}^{dual} = min_{i:\Delta\lambda_i<0} - \frac{\lambda_i - (1-\tau)\lambda_i}{\Delta\lambda_i}$
9: Asignar $\alpha^{pri} = min(1, \alpha_{max}^{pri})$ y $\alpha^{dual} = min(1, \alpha_{max}^{dual})$
10: Asignar $\alpha = min(\alpha^{pri}, \alpha^{dual})$
11: Actualizar $x = x + \alpha\Delta x$ y $(\lambda, s) = (\lambda, s) + \alpha(\Delta\lambda, \Delta s)$

Dado los puntos arbitratios (x_0, y_0, λ_0) el punto inicial es calculado como se indica a continuación:

$$y_0 = max(1, |\hat{(}y) + \Delta y^{aff}|, \quad \lambda_0 = max(1, |\hat{(}\lambda) + \Delta\lambda^{aff}|, \quad X_0 = \hat{(}x) \tag{22}$$

4. Validacion experimental

Se programó el algoritmo en python, para revisar su comportamiento se generaron datos de forma artificial, el procedimiento se realizó como se indica a continuación. Inicialmente se genera una nube de datos mediante varias distribuciones conocidas, posteriormente se genera un grid de distribuciones normales. Después, se ensambla la matriz de verosimilitud, la cual es optimizada con el método MIX-SQP, al final se generan datos de forma aleatoria

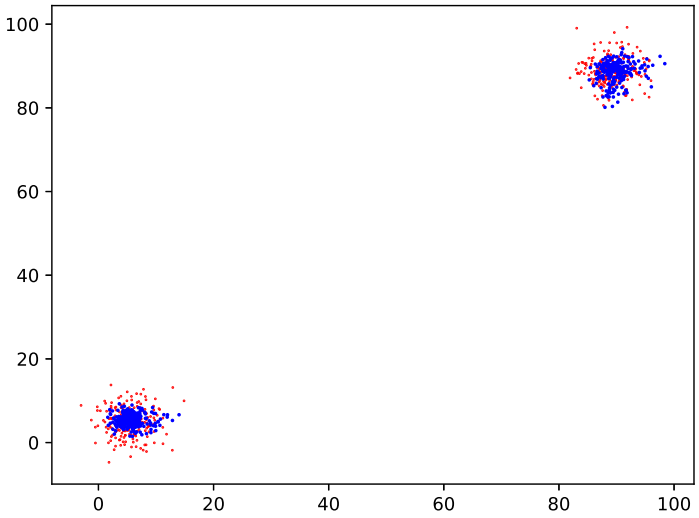

Figura 1: Estimating the data through a grid of normal bi-variate distributions, in blue are the computed points and with red the true points ($m = 800, n = 200$) .

en cada distribución del grid en base a su proporción en el vector de pesos, es decir para cada distribución en el grid se genera un númeo uniforme $[0, 1]$, si la probabilidad es menor a la indicada por el vector de pesos se generan datos de esa distribución. En la figura 1 se pueden observar que el procedimiento de optimización asigna pesos aceptables a las distribuciones normales que describen mejor los datos, en este caso se consideraron 800 funciones de densidad distribuidas de forma adecuada (igualmente espaciados) en las dos dimensiones.

En la figura 2 se puede observar la convergencia del algoritmo a lo largo de las iteraciones considerando cada vez mas funciones de densidad, principalmente se puede observar que la aptitud decrementa conforme se utiliza un mayor número de distribuciones gausianas, esto tiene sentido pued el espacio seŕa mejor cubierto, sobre todo en este caso en que la matriz de covarianza de cada distribución es de $\Sigma = [[0{,}2, 0{,}1], [0{,}1, 0{,}2]]$.

En la figura 3 se puede observar que el comportamiento del método conforme aumenta el número de distribuciones o diccionario de distribuciones.

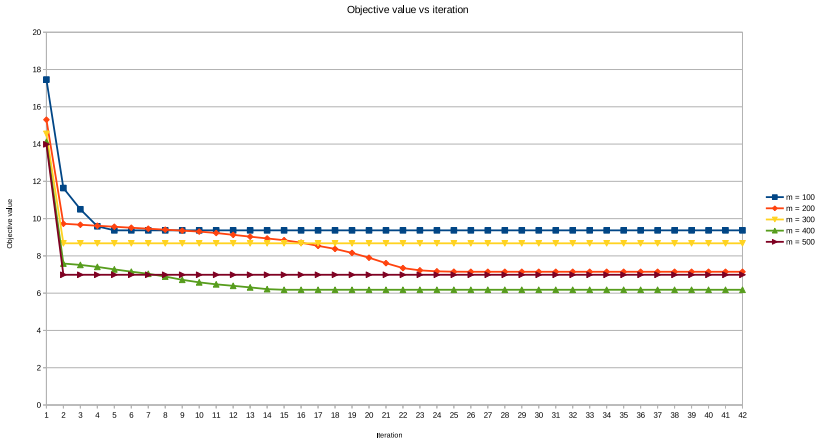

Figura 2: Iteraciones respecto al valor de la función objetivo.

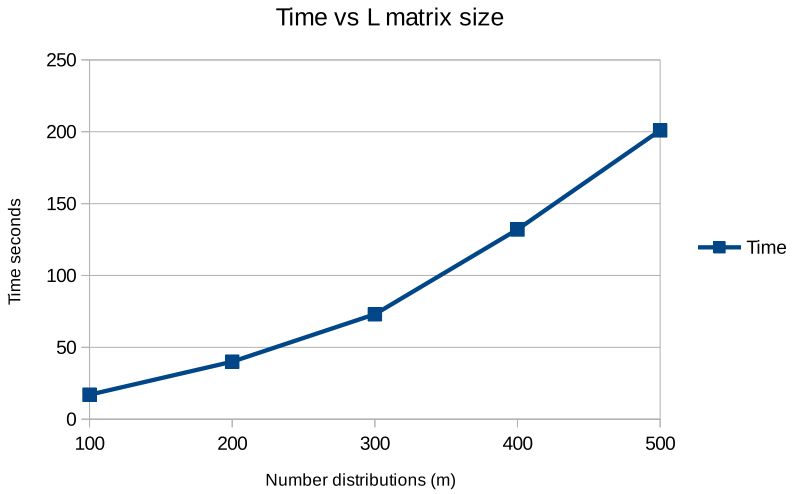

Figura 3: Tiempo respecto al número de distribuciones.

#Densities	F(x)	Time
100	9.3694418184	17
200	7.1473580682	40
300	8.673979796	73
400	6.1812852133	132
500	6.9866809625	201

Cuadro 1: Valor de la función objetivo y tiempo en segundo al incrementar el número de densidades, el número de datos considerados fue de 100.

[1] R. Koenker, I. Mizera, Convex optimization, shape constraints, compound decisions, and empirical bayes rules, Journal of the American Statistical Association 109 (2014) 674–685.

[2] Y. Kim, P. Carbonetto, M. Stephens, M. Anitescu, A fast algorithm for maximum likelihood estimation of mixture proportions using sequential quadratic programming, arXiv preprint arXiv:1806.01412 (2018).

[3] J. Gu, R. Koenker, Empirical bayesball remixed: Empirical bayes methods for longitudinal data, Journal of Applied Econometrics 32 (2017) 575–599.