Maximum Likelihood Estimation of Multinomial Mixture Models using High Performance Computing

Andrew M. Raim
Advisor: Nagaraj K. Neerchal

Department of Mathematics and Statistics
University of Maryland, Baltimore County
Baltimore, MD, USA

39th Annual Meeting of the Statistical Society of Canada
June 12–15 at Acadia University, Wolfville, NS

Overview

- Parallel computing is starting to become mainstream in statistical applications
- Many important applications are considered *embarrassingly parallel*
  - Can be easily decomposed into smaller problems that have little dependence on each other
  - For example: Monte Carlo simulations, Bootstrap, parameter studies
- One popular tool is the SNOW package for R ("Simple Network of Workstations")
  - Easy to handle embarrassingly parallel problems with functions like `parApply` and `clusterCall`
- What about computations which aren't embarrassingly parallel?
  - We consider the MLE problem for mixtures of multinomials
  - We'll show that Fisher Scoring iterations can be effectively parallelized on an HPC cluster, for *large enough* problems

Mixture of multinomials

- Suppose we have s multinomial populations

$$f(\mathbf{x} \mid \mathbf{p}_1, m), \ldots, f(\mathbf{x} \mid \mathbf{p}_s, m)$$

  where the ℓth population has density

$$f(\mathbf{x} \mid \mathbf{p}_\ell, m) = \frac{m!}{x_1! \ldots x_k!} p_{\ell 1}^{x_1} \ldots p_{\ell k}^{x_k}, \qquad \mathbf{x} \in \mathcal{X}$$

  and $\mathbf{p}_\ell = (p_{\ell 1}, \ldots, p_{\ell k})$ represents a discrete probability distribution on the k categories

- If ℓth population occurs wp π_ℓ, and we draw $\mathbf{Y}$ from the mixed population,

$$\mathbf{Y} \sim f_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{\ell=1}^{s} \pi_\ell f(\mathbf{x} \mid \mathbf{p}_\ell, m), \qquad \mathbf{x} \in \mathcal{X}, \qquad \boldsymbol{\theta} = (\mathbf{p}_1, \ldots, \mathbf{p}_s, \boldsymbol{\pi})$$

  This is a natural way to model mixed populations

- We'll write $\mathbf{Y} \sim \text{MultMix}(\mathbf{P}, \boldsymbol{\pi}, m)$, where $\mathbf{P} = (\mathbf{p}_1 \ \ldots \ \mathbf{p}_s) : k \times s$

Mixture of multinomials

Example: Housing satisfaction survey from J. R. Wilson (1989)

Non-metropolitan area				Metropolitan area			
Neighborhood	US	S	VS	Neighborhood	US	S	VS
1	3	2	0	1	0	4	1
2	3	2	0	2	0	5	1
3	0	5	0	3	0	3	2
⋮				⋮			
17	4	1	0	17	4	1	0
18	5	0	0				

With labels, a reasonable likelihood is product of two multinomials

$$L(\boldsymbol{\theta}) = \left[\prod_{i=1}^{18} f(\mathbf{x}_i \mid \mathbf{p}_1, m)\right] \left[\prod_{i=1}^{17} f(\mathbf{x}_i \mid \mathbf{p}_2, m)\right], \qquad m = 5$$

Mixture of multinomials

Example: Housing satisfaction survey from J. R. Wilson (1989)

???				???			
Neighborhood	US	S	VS	Neighborhood	US	S	VS
1	3	2	0	19	0	4	1
2	3	2	0	20	0	5	1
3	0	5	0	21	0	3	2
⋮				⋮			
17	4	1	0	35	4	1	0
18	5	0	0				

Without labels, a reasonable likelihood is mixture of two multinomials

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{35} \left\{ \pi f(\mathbf{x}_i \mid \mathbf{p}_1, m) + (1-\pi) f(\mathbf{x}_i \mid \mathbf{p}_2, m) \right\}, \qquad m = 5$$

Maximum Likelihood Estimation for MultMix

- Consider computation of the MLE for the mixture of multinomials

- Given a sample $\mathbf{Y}_1, \ldots, \mathbf{Y}_n \overset{iid}{\sim} \text{MultMix}(\mathbf{P}, \boldsymbol{\pi}, m)$, to find $\widehat{\mathbf{P}}$ and $\widehat{\boldsymbol{\pi}}$ which maximize the likelihood

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} \sum_{\ell=1}^{s} \pi_\ell \left\{ \frac{m!}{y_{i1}! \ldots y_{ik}!} p_{\ell 1}^{y_{i1}} \ldots p_{\ell k}^{y_{ik}} \right\}$$

- Can't maximize in closed form, so we turn to numerical methods. Some options:
  - Direct numerical maximization (e.g. `optim` in R)
  - Expectation Maximization (EM)
  - Newton-Raphson type iterations, e.g. **Fisher Scoring**

Fisher Scoring

Fisher Scoring Algorithm (FSA)

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \mathcal{I}^{-1}(\boldsymbol{\theta}^{(t)})\, u(\boldsymbol{\theta}^{(t)}; \mathbf{Y}), \qquad t = 0, 1, \ldots$$

where

1. $u(\boldsymbol{\theta}; \mathbf{Y}) = \frac{\partial}{\partial \boldsymbol{\theta}} \log L(\boldsymbol{\theta}; \mathbf{Y})$ is the score vector
2. $\mathcal{I}^{-1}(\boldsymbol{\theta})$ is the inverse Fisher Information Matrix (IFIM) for the sample
3. $\boldsymbol{\theta}^{(t)}, t = 0, 1, \ldots$ is the estimate after iteration t

Problem: The MultMix FIM does not have a simple form

- We can compute it naively

$$\begin{aligned}\mathcal{I}(\boldsymbol{\theta}) &:= \mathsf{E}\left[\left\{\frac{\partial}{\partial \boldsymbol{\theta}} \log f_{\boldsymbol{\theta}}(\mathbf{x})\right\}\left\{\frac{\partial}{\partial \boldsymbol{\theta}} \log f_{\boldsymbol{\theta}}(\mathbf{x})\right\}^T\right] \\ &= \sum_{\mathbf{x} \in \mathcal{X}} \left\{\frac{\partial}{\partial \boldsymbol{\theta}} \log f_{\boldsymbol{\theta}}(\mathbf{x})\right\}\left\{\frac{\partial}{\partial \boldsymbol{\theta}} \log f_{\boldsymbol{\theta}}(\mathbf{x})\right\}^T f_{\boldsymbol{\theta}}(\mathbf{x})\end{aligned}$$

- Then take the inverse
- But this quickly becomes impractical as m and k are increased

Approximate FIM

- Morel & Nagaraj (1993) and Liu's (2005, PhD thesis) approximate FIM

$$\widetilde{\mathcal{I}}(\boldsymbol{\theta}) := \begin{pmatrix} \pi_1 \mathbf{F}_1 & & 0 \\ & \ddots & \\ & & \pi_s \mathbf{F}_s \\ 0 & & \mathbf{F}_\pi \end{pmatrix} \qquad (sk-1) \times (sk-1)$$

$$\mathbf{F}_\ell = m \left[\text{diag}(p_{\ell 1}^{-1}, \ldots, p_{\ell,k-1}^{-1}) - p_{\ell k}^{-1} \mathbf{1}\mathbf{1}^T \right] \qquad (k-1) \times (k-1)$$

$$\mathbf{F}_\pi = \text{diag}(\pi_1^{-1}, \ldots, \pi_{s-1}^{-1}) - \pi_s^{-1} \mathbf{1}\mathbf{1}^T \qquad (s-1) \times (s-1)$$

- Simple block-diagonal form, requires little computation to construct
  - Simple forms for the inverse and determinant
  - Note that $\mathbf{F}_\ell$ is FIM of standard multinomial

- **Theorem:** $\widetilde{\mathcal{I}}(\boldsymbol{\theta}) - \mathcal{I}(\boldsymbol{\theta}) \to 0$ as $m \to \infty$, for multinomial mixtures
  - Convergence rate is exponential for each matrix element

Approximate Fisher Scoring

- Liu (2005) suggests using $\widetilde{\mathcal{I}}(\boldsymbol{\theta})$ in place of $\mathcal{I}(\boldsymbol{\theta})$ for FSA
  - Resulting algorithm is called *Approximate Fisher Scoring* (AFSA)
  - AFSA works well when FSA becomes intractable

- AFSA iteration

$$\begin{aligned}\boldsymbol{\theta}^{(\text{new})} &= \boldsymbol{\theta} + \widetilde{\mathcal{I}}^{-1}(\boldsymbol{\theta})\, u(\boldsymbol{\theta}; \mathbf{Y}) \\ &= \begin{pmatrix} \boldsymbol{\theta}_1 \\ \vdots \\ \boldsymbol{\theta}_s \\ \boldsymbol{\theta}_\pi \end{pmatrix} + \begin{pmatrix} \pi_1^{-1}\mathbf{F}_1^{-1} & & & 0 \\ & \ddots & & \\ & & \pi_s^{-1}\mathbf{F}_s^{-1} & \\ 0 & & & \mathbf{F}_\pi^{-1} \end{pmatrix} \begin{pmatrix} u_1(\boldsymbol{\theta}) \\ \vdots \\ u_s(\boldsymbol{\theta}) \\ u_\pi(\boldsymbol{\theta}) \end{pmatrix}\end{aligned}$$

- Repeat until convergence

Approximate FSA and Parallel Computing

- The block-diagonal structure suggests a way to do parallel computing

$$\boldsymbol{\theta}^{(\text{new})} = \begin{pmatrix} \boldsymbol{\theta}_1 \\ \vdots \\ \boldsymbol{\theta}_s \\ \boldsymbol{\theta}_\pi \end{pmatrix} + \begin{pmatrix} \pi_1^{-1}\mathbf{F}_1^{-1} & & & 0 \\ & \ddots & & \\ & & \pi_s^{-1}\mathbf{F}_s^{-1} & \\ 0 & & & \mathbf{F}_\pi^{-1} \end{pmatrix} \begin{pmatrix} u_1(\boldsymbol{\theta}) \\ \vdots \\ u_s(\boldsymbol{\theta}) \\ u_\pi(\boldsymbol{\theta}) \end{pmatrix}$$

$$= \begin{pmatrix} \boldsymbol{\theta}_1 + \pi_1^{-1}\mathbf{F}_1^{-1}u_1(\boldsymbol{\theta}) & \leftarrow \text{process } 1 \\ \vdots & \vdots \\ \boldsymbol{\theta}_s + \pi_s^{-1}\mathbf{F}_s^{-1}u_s(\boldsymbol{\theta}) & \leftarrow \text{process } s \\ \boldsymbol{\theta}_\pi + \mathbf{F}_\pi^{-1}u_\pi(\boldsymbol{\theta}) & \leftarrow \text{process } s+1 \end{pmatrix}$$

- **Idea:** Split the work this way every iteration, obtain $\boldsymbol{\theta}^{(\text{new})}$, repeat...
- What information needs to be shared?
  - The data $\mathbf{Y}$, which stays constant
  - The current guess $\boldsymbol{\theta}$, at each iteration
- Can parallelize if not block diagonal, but don't get this decomposition

Parallel Computing

- By relaxing the requirement for exact FIM, we have an opportunity for parallel computing
  - Can use # processes $p \in \{1, \ldots, s+1\}$ for a mixture of s populations
- **Question:** Can we use this to benefit our computations of MLEs?
  - To solve larger problems (s, k, n), in less time
  - How large, for parallel computing / HPC to be worth the effort?
- tara @ High Performance Computing Facility, UMBC
  - 86-node distributed-memory cluster set up in Fall 2009
  - two quad-core Intel Nehalem processors (2.66 GHz, 8192 kB cache per core) and 24 GB memory per node
  - high performance InfiniBand interconnect
- We'll consider plain R code and C++ w/ MPI
  - Verification run
  - Performance study

Verification Run

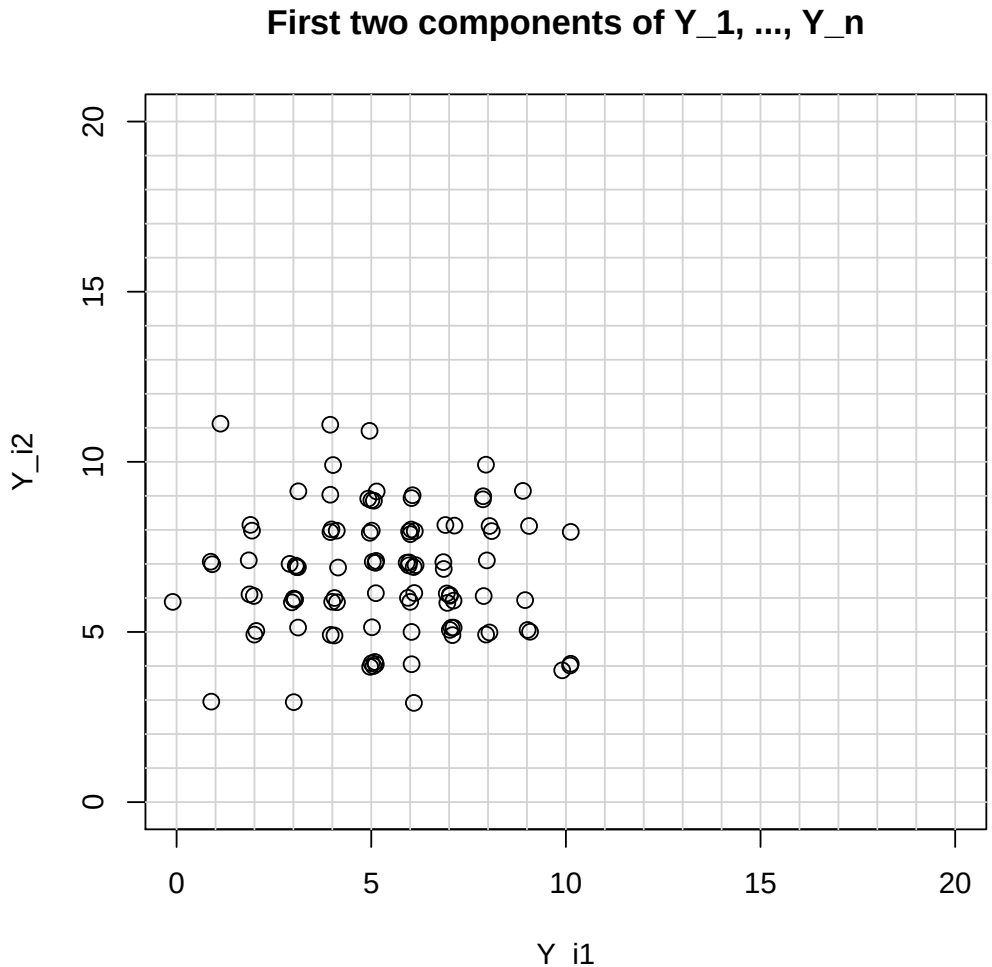
$$\mathbf{Y}_1, \ldots, \mathbf{Y}_n \overset{iid}{\sim} \text{MultMix}(\mathbf{P}, \boldsymbol{\pi}, m),$$

$$\mathbf{P} = \begin{pmatrix} 1/3 & 1/6 \\ 1/3 & 2/6 \\ 1/3 & 3/6 \end{pmatrix}$$

$$\boldsymbol{\pi} = \begin{pmatrix} 2/3 \\ 1/3 \end{pmatrix}$$

$$m = 20$$

$$n = 100$$



Verification Run

run	FSA-R	AFSA-R	EM-R	Optim-R (L-BFGS-B)	AFSA-C++
iterations	23	90	89	18	90
elapsed (sec)	5.276	5.848	4.557	6.913	0.099

- All methods give the same result

$$\widehat{\mathbf{P}} = \begin{pmatrix} 0.3199 & 0.1683 \\ 0.3506 & 0.3017 \\ 0.3295 & 0.5300 \end{pmatrix}, \quad \widehat{\boldsymbol{\pi}} = \begin{pmatrix} 0.6511 \\ 0.3489 \end{pmatrix}$$

- Initial guess $\widehat{\boldsymbol{\theta}}^{(0)}$ same for all methods, chosen at random
- Convergence criteria: $\left| L(\widehat{\boldsymbol{\theta}}^{(t+1)}) - L(\widehat{\boldsymbol{\theta}}^{(t)}) \right| < 1e\text{-}8$ (except for Optim-R)
- FSA requires exact IFIM, only tractable for smaller sample spaces
- `optim` doesn't perform well for larger (general) mixture problems

Performance Runs

- To observe performance characteristics as problem sizes vary

$$\mathbf{P} : k \times s, \qquad \boldsymbol{\pi} : s \times 1$$

$$\mathbf{Y}_1, \ldots, \mathbf{Y}_n \overset{iid}{\sim} \text{MultMix}(\mathbf{P}, \boldsymbol{\pi}, m)$$

  - Cluster size $m = 20$
  - Number of components $s \in \{3, 7, 15, 31\}$
  - Number of categories $k \in \{15, 31\}$
  - Sample size $n \in \{10^2, 10^3, 10^4, 10^5\}$

- Consider the following MLE codes
  - AFSA, serial R
  - EM, serial R
  - AFSA, C++ MPI

- Initial guess is random for each scenario, but same used for all codes

- Stop after 10 iterations
  - To see computation performance, regardless of convergence

Performance Results: Smaller scenarios

Scenario			Serial R		C++ MPI			
s	k	n	AFSA	EM	$p = 1$	$p = 2$	$p = 4$	$p = 8$
3	15	100	0.89	0.66	0.05	0.05	0.03	—
		1,000	8.84	6.50	0.45	0.29	0.19	—
		10,000	88.85	65.61	4.32	2.63	1.78	—
		100,000	971.84	754.89	43.47	26.67	19.56	—
3	31	100	1.02	0.73	0.08	0.08	0.04	—
		1,000	9.89	7.21	0.71	0.44	0.30	—
		10,000	101.73	73.95	7.07	4.25	2.94	—
		100,000	1025.87	745.39	71.13	42.33	29.44	—
7	15	100	1.69	1.07	0.17	0.11	0.08	0.04
		1,000	16.59	10.75	1.56	0.90	0.59	0.41
		10,000	166.27	117.57	15.34	8.98	5.74	4.23
		100,000	1666.05	1089.84	158.31	88.70	56.98	42.08
7	31	100	1.94	1.25	0.26	0.17	0.11	0.07
		1,000	19.70	12.72	2.57	1.48	0.92	0.69
		10,000	194.03	124.91	25.36	14.62	9.23	6.86
		100,000	1979.33	1465.97	247.14	145.47	92.35	67.85

Elapsed walltime in seconds

Performance Results: Larger scenarios

Scenario			C++ MPI					
s	k	n	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$
15	15	100	0.57	0.34	0.20	0.12	0.22	—
		1,000	5.59	3.12	1.81	1.21	0.88	—
		10,000	55.86	31.28	18.01	12.23	8.81	—
		100,000	555.56	310.55	180.11	123.31	88.52	—
15	31	100	0.93	0.52	0.30	0.20	0.15	—
		1,000	9.32	5.03	2.97	2.00	1.46	—
		10,000	91.70	50.53	29.62	19.90	14.45	—
		100,000	907.90	500.16	292.55	197.78	144.30	—
31	15	100	2.13	1.16	0.64	0.38	0.29	0.24
		1,000	21.29	11.12	6.12	3.79	2.50	1.84
		10,000	218.79	116.83	61.13	37.37	24.84	18.46
		100,000	2241.50	1107.10	609.17	375.77	248.35	184.05
31	31	100	2.47	1.30	0.73	0.44	0.29	0.25
		1,000	35.13	18.30	10.15	6.14	4.04	2.99
		10,000	349.44	181.82	108.47	61.19	40.31	29.93
		100,000	3473.05	1806.06	1003.13	613.12	403.84	299.76

Elapsed walltime in seconds

Conclusions I

The parallelization scheme for AFSA can be effective

- But problem sizes need to be fairly large
  1. Many mixing components
  2. Lots of categories
  3. Large sample size
- Then splitting the work within iterations is productive
- But larger problems also become more difficult in general
- Many iterations ($>$1,000) may be needed for convergence. Good initial guesses can help
- Applications of these magnitudes do exist

A few computing options beyond plain R

- R w/ SNOW: speedup without too much effort, but maybe still slow overall
- R w/ serial C++: a lot of effort, much faster, can use from within R
- C++ w/ MPI: most effort, best performance, use for biggest problems

Conclusions II

This parallelization scheme is naturally geared toward large s and k

- The best scheme will depend on the application
- For repeated estimation, most efficient scheme will be "embarrassingly parallel"

For special multinomial mixtures like the Random-Clumped model discussed in Morel & Neerchal (1998), direct numerical optimization works well

- Procedures involving the gradient (with/without hessian) can be effectively parallelized
- See Raim, Gobbert, Neerchal & Morel (submitted)
- But for general multinomial mixtures, approaches like EM and FSA seem more effective than direct optimization

References

[1] Minglei Liu. *Estimation for Finite Mixture Multinomial Models*. Phd thesis, University of Maryland, Baltimore County, Department of Mathematics and Statistics, 2005.

[2] J.G. Morel and N.K. Nagaraj. A finite mixture distribution for modelling multinomial extra variation. *Biometrika*, 80:363–371, 1993.

[3] N.K. Neerchal and J.G. Morel. Large cluster results for two parametric multinomial extra variation models. *Journal of the American Statistical Association*, 93:1078–1087, 1998.

[4] A.M. Raim, M.K. Gobbert, N.K. Neerchal, and J.G. Morel. Maximum likelihood estimation of the random-clumped multinomial model as prototype problem for large-scale statistical computing. 2010. Submitted.

Acknowledgement: The computational resources used for this work were provided by the High Performance Computing Facility at the University of Maryland, Baltimore County (UMBC). See `www.umbc.edu/hpcf` for information on the facility and its uses. Andrew additionally thanks the facility for financial support as an RA.