Finite Mixture Modeling

Shiro Kuriwaki[*]

September 2019

1 Setup

Index individuals by $i \in \{1 : N\}$ and the universe of races excluding the top of the ticket as $j \in \{1 : J\}$. The data we observe is categorical and at the individual-race level:

$$y_{ij} \in \{0, 1, 2\}.$$

where the M possible values encode

0	straight ticket vote with reference office (e.g. President)
1	split ticket vote
2	abstain

The probability is governed by a length-M simplex which we will denote as θ. The distribution is conditioned on the cluster, which is the random variable Z. We predetermine how many clusters there can be,

$$z_i \in \{1 : K\}.$$

2 Model 1

We model the outcome as coming from a categorical distribution:

$$(y_{ij} \mid Z_i = z) \sim \text{Categorical}(\theta_{z,j})$$

For example, suppose that there are three types of voters. They each have a given value of θ_{jz}:

$$\begin{cases} \text{Always straight} & \theta_{z,j} = (0.97, 0.01, 0.02) \text{ if } z = 1 \\ \text{Always split} & \theta_{z,j} = (0.01, 0.97, 0.02) \text{ if } z = 2 \\ \text{Random} & \theta_{z,j} = (0.49, 0.49, 0.02) \text{ if } z = 3 \end{cases}$$

[*] Thanks to Shusei Eshima, Sooahn Shin, and Shusei Eshima for their help.

For simplicity, let's assume this holds regardless of the office, i.e. $\theta_{z,j} = \theta_{z,j'}\ \forall j \neq j'$.

The cluster is also a categorical variable,

$$Z_i \sim \text{Categorical}(\psi),$$
$$\psi \sim \text{Dirichlet}(\alpha)$$

From substantive background knowledge, our prior is that most voters are straight ticket voters, so we can set the hyperparameter to

$$\alpha = (0.80, 0.10, 0.10)$$

3 Model 2

A simpler model may be to model the office type as a categorical variable separately. Let this variable be W. The main attribute of this variable is its level of office. Our main interest is whether some offices exhibit more split ticketting than others, so the simplest setup is to let

$$w_j \in \{0, 1\}.$$

where the values indicate

0	low propensity to generate split ticket
1	high propensity to generate split ticket

Later on, we can add a variable for candidate level incumbency.

Then, the simplex governing the parameter can be indexed as

$$(Y_{ij} | Z_i = z, W_j = w) \sim \text{Categorical}(\theta_{z,w})$$

The values of W_j for offices $j \in \{1 : J\}$ can be modeled as a categorical variable, or more simply a Bernoulli

$$W \sim \text{Bern}(\pi)$$

where $\pi \in [0, 1]$

4 Adding covariates

Later on, we will model θ_{jz} as a function of v_j, covariates of candidate j.

$$\theta_{jz} = \frac{\exp(v_j^\top \beta_{jz})}{\sum_{j'} \exp(v_{j'}^\top \beta_{j'z})}$$

For now, we will model three attributes of the candidate: whether the candidate is an incumbent, and whether the candidate is in an open-seat.

For example, assume we are talking about Republican voter:

		Republican Candidate		
j	Race	Name	Incumbent	Open-seat
1	HD 15	Samuel Rivers	1	0
2	HD 94	Con Chellis	0	1
3	HD 99	Nancy Mace	1	0
4	HD 110	William Cogswell	1	0
5	HD 112	Mike Sottile	1	0
6	HD 114	Lin Bennett	1	0
7	HD 115	Peter McCoy	1	0
8	HD 117	Bill Crosby	1	0
9	HD 119	Paul Sizemore	0	0

5 Stan Code

The Stan code:

```
data {
  int<lower=2> M;          // number of possible values of Y
  int<lower=1> K;          // number of clusters
  int<lower=1> J;          // number of offices
  int<lower=1> N;          // number of voters
  int  Y[N, J];          // observations
  vector<lower=0>[M] alpha;  // hyperparameter
}


parameters {
  simplex[M] theta[K*J];     // abstain, straight, split
  simplex[K] psi[J];         // mixing proportions
}

model {
  for (j in 1:J) {
      psi[j] ~ dirichlet(alpha);  // prior
  }

  for (j in 1:J) {
    for (n in 1:N) {
      vector[K] lps = log(psi[j]);
      for (k in 1:K) {
        lps[k] += multinomial_lpmf(Y[n, j] | theta[k*(j - 1) + k]);
      }
      target += log_sum_exp(lps);
    }
  }
}
```

6 Simulated Data

```
library(purrr)
library(rstan)
library(brms)

# dimensions
M <- 3L
K <- 5L
J <- 10L
N <- 400L

# hyperparameter
alpha <- c(2, 1.5, 1)

# priors
psi <- rdirichlet(1, alpha)
Z_table <- rmultinom(N, 1, psi)
Z <-  map_dbl(1:N,  ~which(Z_table[, .x] == 1) - 1)

# set parameters
theta_z <- list(
  ‘0‘ = c(0.97, 0.01, 0.02),
  ‘1‘ = c(0.01, 0.97, 0.02),
  ‘2‘ = c(0.49, 0.49, 0.02)
)

# Generate data
Y <- array(NA, dim = c(N, J))
for (j in 1:J) {
  theta_i <- map(as.character(Z), ~ theta_z[[.x]])
  y_j <- map_dbl(theta_i, ~which(rmultinom(1, 1, .x) == 1) - 1)
  Y[, j] <- y_j
}
```

7 Model Run

To compile and estimate the model we can add code like this at the end of the simulation script

```r
m <- stan_model("finite-mixture_stan-05-03.stan")
fit = sampling(m,
               data = list(M = M,
                           K = K,
                           N = N,
                           J = J,
                           Y = Y,
                           alpha = alpha))
```