Summary of Short-term Research Objectives

Detian Deng

April 13, 2015

1 Model Specification

Let L be a K-dimensional Bernoulli random variable denoting the true state. Consider the general log linear model:

$$f(l;\Theta) = \exp\{\Theta_1^T l + \Theta_2^T u_2 + \ldots + \Theta_K^T u_K - A^*(\Theta)\}$$

where U_k is a $\binom{K}{k} \times 1$ vector of k-way cross-products, $k = 1, \ldots, K$, and $\Theta = (\Theta_1, \ldots, \Theta_K)$ contains the the natural parameters, which is a $(2^K - 1) \times 1$ vector.

Model restrictions, let $\tilde{l} = (l, u_2, \ldots, u_K)^T$, and $S = \sum_{j=1}^K L_j = s$ has some fixed pmf

$$\begin{aligned} \pi(s) :=& P(S = s) \\ =& \frac{1}{A(\Theta)} \sum_{\tilde{l}:S=s} \exp\{\Theta^T \tilde{l}\} \ , \ s = 0, 1, \ldots, K \end{aligned} \tag{1}$$

$$A(\Theta) = \sum_{\tilde{l}:l \in \{0,1\}^K} \exp\{\Theta^T \tilde{l}\} \tag{2}$$

2 Research Objectives

Suppose $\{L_i\}$ are i.i.d random variables with density $P(L_i;\Theta)$ defined above, our goal is to find a re-parameterization from Θ to $(\boldsymbol{\pi},\boldsymbol{\gamma})$ such that $P(L_i;\Theta) = P(L_i;\boldsymbol{\pi},\boldsymbol{\gamma})$, where $\boldsymbol{\pi} = (\pi_0,\pi_1,\ldots,\pi_s,\ldots,\pi_K)^T$, with $\sum_{s=0}^{K}\pi_s = 1$.

3 Results

3.1 Definition of New Parameters

Using previous notations, with some properly defined $\boldsymbol{\gamma}$, we have:

$$\begin{aligned} P(L_i;\boldsymbol{\pi},\boldsymbol{\gamma}) =& P(L_i,S_i;\boldsymbol{\pi},\boldsymbol{\gamma}) \\ =& P(L_i|S_i;\boldsymbol{\pi},\boldsymbol{\gamma})P(S_i;\boldsymbol{\pi},\boldsymbol{\gamma}) \end{aligned}$$

where $P(L_i|S_i;\boldsymbol{\pi},\boldsymbol{\gamma}) = P(L_i|S_i;\boldsymbol{\gamma})$ because $1(S_i = s)$ is the sufficient statistic for π_s, furthermore S_i is the sufficient statistic for $\boldsymbol{\pi}$. Also $P(S_i;\boldsymbol{\pi},\boldsymbol{\gamma}) = \pi_{S_i}$ by definition.

Therefore, we have

$$P(L_i;\boldsymbol{\pi},\boldsymbol{\gamma}) = P(L_i|S_i;\boldsymbol{\gamma})\pi_{S_i}$$

Then we define the following parameters:

$$\begin{aligned} \gamma_{j_1,\ldots,j_s} =& P(L_{ij_1} = \ldots = L_{ij_s} = 1|S_i = s) \qquad (3) \\ \boldsymbol{\gamma} =& (\gamma_1,\gamma_2,\ldots,\gamma_{12},\ldots,\gamma_{1\ldots K})^T, \text{ where } \sum_j \gamma_j = \sum_{j\neq j'} \gamma_{jj'} = \sum_{j\neq j'\neq j''} \gamma_{jj'j''} = \ldots = \gamma_{1\ldots K} = 1 \end{aligned}$$

Therefore $(\boldsymbol{\pi},\boldsymbol{\gamma})^T$ is a vector of length $2^K + K$ with degrees of freedom $2^K - 1$.

Let $J_i = \{j : L_{ij} = 1\}$. We have,

$$P(L_i;\boldsymbol{\pi},\boldsymbol{\gamma}) = \gamma_{J_i}\pi_{S_i} \qquad (4)$$

The relation between $(\boldsymbol{\pi},\boldsymbol{\gamma})$ and Θ is defined by equation (2) together with:

$$\gamma_{J_i} = \frac{\exp(\Theta^T\tilde{l}_i)}{\sum_{l:l^T1=S_i}\exp(\Theta^T\tilde{l})} \qquad (5)$$

with $2^K - 1 - K$ degrees of freedom.

Therefore (1), (3) and (5) together define $2^K - 1$ non-linear equations for $2^K - 1$ unknowns. If there exists a unique root for the above non-linear system, then there is a one-to-one mapping between $(\boldsymbol{\pi},\boldsymbol{\gamma})$ and Θ, which provides the re-parameterization.

3.2 Find the Re-parameterization

3.2.1 Quasi-Newton Method

Numerically solve the system defined by (1), (3) and (5). As the dimension of L grows ($K > 6$), multiple sets of starting values are needed to reach the solution. Also, solutions to high order Θ are subject to larger error.
See code GammaToTheta() in the appendix.

3.2.2 Multiple Iterative Proportional Fitting

- Usage: Given a K-dimensional contingency table, find the unique MLE of cell probability subject to some marginal constrains.
- The relation between marginal constrains with log linear model parameterization.

3.2.3 Restrict Θ to QE model

Setting all high order interaction parameter to 0, using only the equations defined by $pi_0, pi_1, \gamma_1, \ldots, \gamma_{K-1}$ and $\gamma_{11}, \ldots, \gamma_{K-2,K}$, which are in total $\frac{K(K+1)}{2}$ equations, we can solve for Θ for larger value of K.
See code GammaToTheta.QE() in the appendix.

Appendix

```
GammaToTheta = function(Gamma0, L.All.expd0, k0, maxNpar0=30)
{
      Equation = function(Theta, Gamma, L.All.expd, k)
      {
            fn = rep(NA,2^k-1)
            theta = matrix(Theta,ncol=1)
            L.s = apply(L.All.expd[,1:k],1,sum)
            gamma.order = order(L.s)
            L.All.expd = L.All.expd[gamma.order,]
            L.s = L.s[gamma.order]

            potentials = exp(L.All.expd%*%theta)
            A = sum(potentials)

            # k unique equations derived from pi
            fn[1] = 1/A - Gamma[1]
            for (j in 2:k)
            {
                  fn[j] = sum(potentials[L.s==j-1])/A - Gamma[j]
            }

            # 2^k - 1 - k equations derived from gammas
            j = k + 1; t = 1
            for (s in 1:(k-1))
            {
                  for (i in 1:(choose(k,s)-1))
                  {
                        fn[j] = potentials[t+i]/sum(potentials[L.s==s]) - Gamma[t+i+k]
                        j = j + 1
                  }
                  t = t + choose(k,s)
            }
            fn
      }
      iters = 1
      par0 = rep(0,2^k0-1)
      result = nleqslv(par0, fn=Equation,
                       Gamma = Gamma0, L.All.expd = L.All.expd0, k = k0,
                       method="Broyden",global="dbldog",xscalm="auto",
                       control = list(maxit=1500,cndtol=1e-15,ftol=1e-10))
      print(paste("Starting Value Set",iters,":",result$message))
      while(result$termcd>2 & iters<maxNpar0)
      {
            iters = iters + 1
            par0 = rnorm(2^k0-1,sd=0.1)
            result = nleqslv(par0, fn=Equation,
                        Gamma = Gamma0, L.All.expd = L.All.expd0, k = k0,
```

```
                        method="Broyden",global="dbldog",xscalm="auto",
                        control = list(maxit=1500,cndtol=1e-15,ftol=1e-10))
            print(paste("Starting Value Set",iters,":",result$message))
      }
      result
}

GammaToTheta.QE = function(Gamma0, L.All.expd0, k0, maxNpar0=30, Full.designMatrix)
{
      if (Full.designMatrix)
      {
            Equation = function(Theta.qe, Gamma, L.All.expd, k)
            {
            N.qe = k*(k+1)/2
            fn = rep(NA,N.qe)
            theta = matrix(c(Theta.qe,rep(0,2^k-1-k*(k+1)/2)),ncol=1)
            L.s = apply(L.All.expd[,1:k],1,sum)
            gamma.order = order(L.s)
            L.All.expd = L.All.expd[gamma.order,]
            L.s = L.s[gamma.order]

            potentials = exp(L.All.expd%*%theta)
            A = sum(potentials)
            A1 = sum(potentials[L.s==1])
            A2 = sum(potentials[L.s==2])

            # 2 unique equations derived from pi
            fn[1] = 1/A - Gamma[1] # pi0
            fn[2] = A1/A - Gamma[2] # pi1

            # k-1 + (k choose 2)-1 equations derived from gammas
            j = 3; t = 1
            for (s in 1:2)
            {
                  As = ifelse(s==1, A1, A2)
                  for (i in 1:(choose(k,s)-1))
                  {
                        fn[j] = potentials[t+i]/As - Gamma[t+i+k]
                        j = j + 1
                  }
                  t = t + choose(k,s)
            }
            fn
      }
      }
      else
      {
            Equation = function(Theta.qe, Gamma, L.All.expd, k)
            {
            N.qe = k*(k+1)/2
```

```
      fn = rep(NA,N.qe)
      theta = matrix(Theta.qe,ncol=1)
      L.s = apply(L.All.expd[,1:k],1,sum)
      gamma.order = order(L.s)
      L.All.expd = L.All.expd[gamma.order,]
      L.s = L.s[gamma.order]

      potentials = exp(L.All.expd%*%theta)
      A = sum(potentials)
      A1 = sum(potentials[L.s==1])
      A2 = sum(potentials[L.s==2])

      # 2 unique equations derived from pi
      fn[1] = 1/A - Gamma[1] # pi0
      fn[2] = A1/A - Gamma[2] # pi1

      # k-1 + (k choose 2)-1 equations derived from gammas
      j = 3; t = 1
      for (s in 1:2)
      {
            As = ifelse(s==1, A1, A2)
            for (i in 1:(choose(k,s)-1))
            {
                  fn[j] = potentials[t+i]/As - Gamma[t+i+k]
                  j = j + 1
            }
            t = t + choose(k,s)
      }
      fn
}
}

iters = 1
par0 = rep(0,k0*(k0+1)/2)
result = nleqslv(par0, fn=Equation,
                 Gamma = Gamma0, L.All.expd = L.All.expd0, k = k0,
                 method="Broyden",global="dbldog",xscalm="auto",
                 control = list(maxit=3500,cndtol=1e-15,ftol=1e-10))
print(paste("Starting Value Set",iters,":",result$message))
while(result$termcd>2 & iters<maxNpar0)
{
      iters = iters + 1
      par0 = rnorm(k0*(k0+1)/2,sd=0.1)
      result = nleqslv(par0, fn=Equation,
                       Gamma = Gamma0, L.All.expd = L.All.expd0, k = k0,
                       method="Broyden",global="dbldog",xscalm="auto",
                       control = list(maxit=3500,cndtol=1e-15,ftol=1e-10))
      print(paste("Starting Value Set",iters,":",result$message))
}
```

```
        result
    }
```