Log-linear Model Specification

Eric Slud, UMD Math Dept and Census Bureau CSRM

To specify log-linear models, it helps to have two sets of notations and two ways of viewing the data structure for independent copies of a K-dimensional discrete random variable $\mathbf{X} = (X_1, \ldots, X_K)$, where $X_j \in \{1, \ldots, I_j\}$. The dataset we envision consists of n *iid* copies $\mathbf{X}_a$, $a = 1, \ldots, n$, of vectors of this type, exhibited as a tabular $n \times K$ discrete array $\mathbf{X}_a = (X_{a,1}, \ldots X_{a,K})$, combined into a set of multinomial counts

$$N_{\mathbf{x}} \;=\; \sum_{a=1}^{n} I_{[\mathbf{X}_a = \mathbf{x}]} \;=\; \sum_{a=1}^{n} I_{[X_{a,j}=x_j,\; j=1,\ldots,K]} \tag{1}$$

with cell probabilities

$$p_{\mathbf{x}} \;=\; P(\mathbf{X} = \mathbf{x}) \;=\; P(X_j = x_j,\;\; j = 1, \ldots, K) \tag{2}$$

The notations so far show the experimental-unit r.v.'s $\mathbf{X}_a$ with values formed into a 2-dimensional $n \times K$ array that could be put in a data-frame, while the aggregated multinomial counts $N_{\mathbf{x}}$ form a K-way array with multi-index $\mathbf{x}$ of dimensions $I_1 \times I_2 \times \cdots \times I_K$. The multinomial counts also could be put into the data-frame obtained by aggregating the n-vector of all 1's with respect to the values $\mathbf{x}$ of $\mathbf{X}_a$.

The background references we use for these definitions are, roughly, Bishop, Fienberg and Holland (1975) and Agresti (2013, 3rd ed.), but the cumbersome notations needed for absolutely general higher-order interactions are my own. Most treatments just create double or triple index notations for interactions up to second or third order at most.

The specification of a log-linear contingency table model is an equation expressing $\log(p_{\mathbf{x}})$ as a linear combination of separate coefficients for each of the subsets of the K-way multi-index $\mathbf{x}$. To make the specification clear, we need also a notation for ordered subsets $(j_1, \ldots, j_r)$ of dimension indices, where all $1 \le j_1 < j_2 < \cdots < j_r \le K$, and $r \le r^* \le K$ denotes the order of interaction described by the index subset. Then the loglinear model is

$$\log p_{\mathbf{x}} \;=\; \log P(X_j = x_j,\;\; j = 1, \ldots, K) \;=\; \mu + \sum_{r=1}^{r^*} \sum_{\mathbf{j}=(j_1,\ldots,j_r)} \alpha^{\mathbf{j}}_{(x_{j_1},\ldots,x_{j_r})} \tag{3}$$

subject to side-conditions (similar to those in linear-model theory) needed to ensure identifiability, namely for all $1 \le r \le r^*$ and $\mathbf{j} = (j_1, \ldots, j_r)$,

$$\text{for all} \quad k = 1, \ldots, r, \qquad \alpha^{\mathbf{j}}_{(x_{j_1},\ldots,x_{j_{k-1}},+,x_{j_{k+1}},\ldots,x_{j_r})} \;\equiv\; \sum_{x_{j_k}=1}^{I_{j_k}} \alpha^{\mathbf{j}}_{(x_{j_1},\ldots,x_{j_r})} \;=\; 0 \tag{4}$$

To keep track of the parameters, we need a list `indxList` of allowed combinations of r and $(j_1, \ldots, j_r)$, and then a vector of real coefficients $\alpha^{\mathbf{j}}_{(x_{j_1},\ldots,x_{j_r})}$ – not necessarily satisfying the conditions (4) – indexed in exactly the same order as the list components. For each such parameter θ consisting of μ and (underdetermined) coefficients $\alpha^{\mathbf{j}_r}_{\mathbf{x}_r}$, there exists a unique parameter whose set of α coefficients satisfies (4). For each such (underdetermined) parameter θ with unspecified μ parameter, it is easy to compute the multi-way of right-hand sides of (3) without μ and thereby the multi-way array of $p_{\mathbf{x}}$ probabilities with unspecified multiplicative factor e^{μ}. Then μ is determined by renormalizing this probability array to sum to 1.

A. Recovering Parameter's from $p_{\mathbf{x}}$'s

If we had a complete array of $p_{\mathbf{x}}$ values, for all $\mathbf{x} \in I_1 \times I_2 \times \cdots \times I_K$, then the method of recovering the parameters $\mu, \alpha^{\mathbf{j}}_{(x_{j_1},\ldots,x_{j_r})}$ functionally, subject to side-conditions (4), is fairly direct. First,

$$\text{for} \quad \mathbf{i} = (i_1, i_2, \ldots, i_s) \subset \mathbf{j} = (j_1, \ldots, j_r), \qquad \text{define} \qquad m(\mathbf{i}, \mathbf{j}) \;=\; \prod_{t:\, 1 \le t \le r,\; j_t \notin \mathbf{i}} I_{j_t}$$

with $m(\mathbf{j}, \mathbf{j}) \equiv 1$ by convention. Then by (3) and (4), $\mu \;=\; \sum_{\mathbf{x}} \log(p_{\mathbf{x}}) / \prod_{t=1}^{K} I_t$, and for a fixed r and $\mathbf{j} = (j_1, \ldots j_r)$ and $(y_{j_1}, \ldots, y_{j_r}) \in I_{j_1} \times I_{j_2} \times \cdots \times I_{j_r}$,

$$\sum_{\mathbf{x}:\; x_k = y_k \;\forall\; k \in \mathbf{j}} \log p_{\mathbf{x}} \;=\; \sum_{\mathbf{i}:\; \mathbf{i} \subset \mathbf{j}} m(\mathbf{i}, \mathbf{j})\, \alpha^{\mathbf{i}}_{(y_{i_1},\ldots,y_{i_s})} \tag{5}$$

The $\alpha^{\mathbf{i}}_{(y_{i_1},\ldots,y_{i_s})}$ parameters can be extracted from the right-hand sides of (5), recursively, by applying (5) first with all singletons $\mathbf{j} = \{j_1\}$, then all doubletons $\mathbf{j} = \{j_1, j_2\}$, then triplets, etc.

Remark 1 *While this operation of mapping $p_{\mathbf{x}}$ to θ is conceptually simple, it would be slightly laborious to code. There must be an existing* `R` *package to do it.* □

B. Recovering $p_{\mathbf{x}}$ Arrays from Specified Marginals of $p_{\mathbf{x}}$'s

In many applications, the simplest way to specify a multi-way array of probabilities $p_{\mathbf{x}}$ would be to draw a compatible set of marginal proportions $p^{\mathbf{i}}_{\mathbf{y}} \;=\; \sum_{\mathbf{x}:\; x_k = y_k \;\forall\; k \in \mathbf{j}} p_{\mathbf{x}}$ from existing known population-tables, and then create an array $p_{\mathbf{x}}$ of the form (3)-(4) with a minimal set of non-zero parameters $\alpha^{\mathbf{j}}_{(x_{j_1},\ldots,x_{j_r})}$ that has the specified marginals. Operationally, this would be done by **raking** or *Iterative Proportional Fitting*, and is known to lead to a unique solution when all of the specified marginal probabilities are non-zero, and in some other cases too (Winkler 1993; Fienberg and Rinaldo 2007, 2012).

Here again, writing this down and coding it in full generality would be tedious. But in this instance, there are effectively programmed `R` functions (e.g., `calibrate` in T. Lumley's `survey` package) to do raking for specified variables based on specified 'target' proportions. The data structure is as follows. Start with the aggregated data-frame structure with one row for each $\mathbf{x}$ combination, and each row also has the variables $x_1, \ldots, x_k, N_{\mathbf{x}}$. Now consider augmenting the data-frame with one dummy column of 0's and 1's for the indicators of each categorical value c for each raking variable Z_l, and assume for simplicity that all of the raking variables are functions of the variables $X_1, \ldots X_K$. For each such raking variable Z_l (with corresponding *iid* observed values $Z_{a,l}$ for experimental units $a = 1, 2, \ldots, n$), denote by $z_l(\mathbf{x})$ the value of $Z_{a,l}$ whenever $\mathbf{X}_a = \mathbf{x}$. Then there will be an augmented column in the data-frame for each distinct categorical value c for Z_l, and the entry of that column in the $\mathbf{x}$ row of the data-frame will be $I_{[z_l(\mathbf{x})=c]}$. The set of known 'population totals' for the corresponding augmented dummy column for $Z_l = c$ is the population total N (arbitrary, because it will cancel out) times the proportion of the population for which the raking variable Z_l takes the value c. Although the `calibrate` function is written with survey weights in mind, it does raking as envisioned when the weight vector $\mathbf{w}$ is specified to have all entries 1.

References

Fienberg, S. and Rinaldo, A. (2007), Three centuries of categorical data analysis: Log-linear models and maximum likelihood estimation. *Jour. Statist. Plann. Inference* **137**, 3430–3445.

Fienberg, S. and Rinaldo, A. (2012), Maximum likelihood estimation in log-linear models, *Annals of Statistics* bf 40, 996-1023.

Winkler, W. (1993). On Dykstra's iterative fitting procedure, *Ann. Probab.* **18**, 1410–1415.