Multinomial Logit and Variants From Scratch

Model files are standard alone * Standard Multinomial Logit: `mlogit.py` * Multinomial Logit With Latent Class: `mlogit_latent_class.py`

The `utils.py` file contains stable version of softmax function and numeric hessian approximation via the central finite difference. The Hessian matrix may not be invertible or the inverted matrix has non-positive values in diagonal, so standard error is calculated by SVD decomposition on the Hessian and inverted diagonal matrix.

Note, the `scipy.optimize.minimize` changes beta when calculating gradient and hessian, so caching forward calculation is not necessary.

Math equations are not displayed properly in Github. # Standard Multinomial Logit

Assume each observation is IID.

Notation

N is number of observation, K is number of choices, M is number of features, X: [N,T,K,M], Y: [N,T], and β: [M,]

Utility for choice k is

$$U_k = X_k\beta$$

Let $M = \max X_k\beta$. The probability of choosing k is

$$P_k = \frac{exp(U_k - M)}{\sum_{j=1}^{K} exp(U_j - M)}$$

The log loss is

$$\begin{aligned}
LL_i &= \sum_{k=1}^{K} Y_{ik} log(P_{ik}) \\
&= \sum_{k=1}^{K} Y_{ik}(U_{ik} - M) - \sum_{k=1}^{K} Y_{ik} log(\sum_{k=1}^{K} exp(U_{ik} - M))
\end{aligned}$$

The Hessian is

$$\begin{aligned}
H_j m &= -\frac{\partial}{\partial \beta_j}\frac{\partial LL}{\partial \beta_m} \\
&= -\sum_{k=1}^{K} \frac{\partial P_k}{\partial \beta_j} X_{km} \\
&= -\sum_{k=1}^{K} P_k \left(X_{kj} - \sum_{k} P_k X_{kj} \right) X_{km}
\end{aligned}$$

The standard error is

$$se = \sqrt{diag\left((-H)^T\right)}$$

or use SVD to decompose H and then take the inverse of the diagonal matrix D.

Latent Class

Notation: H is individual level to determine latent class probability. Q is number of latent class. P_{tkq} is predicted probability for choice k in latent class q at time t. Data: Y:[N,K], X:[N,K,M], H:[N,T,L] Parameter: β: [M,Q], γ:[L,Q] Model:

$$P_{tkq} = \frac{exp(X_{tk}\beta_q)}{\sum_{j=1}^{K} exp(X_{tj}\beta_q)}$$

$$W_q = \frac{exp(\sum_t H_t\gamma_q)}{\sum_{j=1}^{Q} exp(\sum_t H_t\gamma_j)}$$

$$L_{iq} = \prod_t \prod_k P_{tkq}^{Y_t k}$$

$$L_i = \sum_q^Q L_{iq}W_q$$

$$LL_i = log(Li)$$

$$LL = \sum_{i=1}^{n} LL_i$$

Gradient

$$\begin{aligned}
\frac{\partial LL}{\partial \beta_q} &= \sum_i^N \frac{1}{L_i} \sum_q W_q \frac{\partial L_{iq}}{\partial \beta_q} \\
&= \sum_i^N \frac{1}{L_i} W_q \frac{\partial L_{iq}}{\partial \beta_q}
\end{aligned}$$

where

$$
\begin{aligned}
\frac{\partial L_{iq}}{\partial \beta_q} &= L_{iq} \frac{\partial log(L_{iq})}{\partial \beta_q} \\
&= L_{iq} \sum_t \sum_k Y_{tk} \frac{1}{P_{tkq}} \left[X_{tk}^T P_{tkq}(1 - P_{tkq}) \right] \\
&= L_{iq} \sum_t \sum_k Y_{tk} X_{tk}^T (1 - P_{tkq}) \\
&= L_{iq} \sum_t \sum_k X_{tk}^T (Y_{tk} - P_{tkq})
\end{aligned}
$$

$$
\frac{\partial LL}{\partial \beta_q} = \sum_i^N \frac{1}{L_i} W_q L_{iq} \sum_t \sum_k X_{tk}^T (Y_{tk} - P_{tkq})
$$