Numerical Implementation of REL

Zhentao Shi
first draft: September, 2014

1 Concavity

The original Lagrangian associated with the primal problem is

$$\mathcal{L}_n^{\tau}(\beta,p,\gamma_A,\gamma)=\frac{1}{n}\sum_{i=1}^{n}\log p_i+\log n+\gamma_A\left(1-\sum_{i=1}^{n}p_i\right)+\gamma'\left(\tau-\left|\sum_{i=1}^{n}p_ih_i(\beta)\right|\right). \tag{1}$$

If we can show

$$\widehat{\gamma}'\sum_{i=1}^{n}\widehat{p}_i(\beta)\,h_i(\beta)=\tau\left\|\widehat{\gamma}\right\|_1, \tag{2}$$

in the optimization of the Lagrangian, then the optimizer of (1) will be the same as that of

$$\mathcal{L}_n^{\tau}(\beta,p,\gamma_A,\gamma)=\frac{1}{n}\sum_{i=1}^{n}\log p_i+\log n+\gamma_A\left(1-\sum_{i=1}^{n}p_i\right)+\gamma'\sum_{i=1}^{n}p_ih_i(\beta)-\tau\left\|\gamma\right\|_1.$$

We now verify (2).

$$\begin{aligned}
& \widehat{\gamma}'\sum_{i=1}^{n}\widehat{p}_i(\beta)\,h_i(\beta)-\tau\left\|\widehat{\gamma}\right\|_1\\
= & \widehat{\gamma}'\sum_{i=1}^{n}\widehat{p}_i(\beta)\,h_i(\beta)-\tau\widehat{\gamma}'\mathrm{sign}(\widehat{\gamma})\\
= & \sum_{\{j:\widehat{\gamma}_j=0\}}\widehat{\gamma}_j\left(\sum_{i=1}^{n}\widehat{p}_i(\beta)\,h_{ij}(\beta)-\tau\mathrm{sign}(\widehat{\gamma}_j)\right)+\sum_{\{j:\widehat{\gamma}_j\neq0\}}\widehat{\gamma}_j\left(\sum_{i=1}^{n}\widehat{p}_i(\beta)\,h_{ij}(\beta)-\tau\mathrm{sign}(\widehat{\gamma}_j)\right)\\
= & \sum_{\{j:\widehat{\gamma}_j\neq0\}}\widehat{\gamma}_j\left(\sum_{i=1}^{n}\widehat{p}_i(\beta)\,h_{ij}(\beta)-\tau\mathrm{sign}(\widehat{\gamma}_j)\right)\\
= & \sum_{\{j:\widehat{\gamma}_j\neq0\}}\widehat{\gamma}_j\left(\sum_{i=1}^{n}\widehat{p}_i(\beta)\,h_{ij}(\beta)-\mathrm{sign}^2(\widehat{\gamma}_j)\sum_{i=1}^{n}\widehat{p}_i(\beta)\,h_{ij}(\beta)\right)=0
\end{aligned}$$

where the last equality follow by the Krush-Kuhn-Tucker condition that $\left|\sum_{i=1}^{n}\widehat{p}_i(\beta)\,h_{ij}(\beta)\right|=\tau$ for any moment j such that $\widehat{\gamma}_j\neq0$, so $\tau=\mathrm{sign}(\widehat{\gamma}_j)\left(\sum_{i=1}^{n}\widehat{p}_i(\beta)\,h_{ij}(\beta)\right)$.

2 Implementation

The literature suggests implementing the EL optimization through the inner loop and outer loop. Much care has to be executed in the programming of the high-dimensional inner loop. After experimenting with several solvers, we find `MOSEK`, a commercial solver specialized in convex programming, efficiently returns reliable results. We use the `MOSEK Matlab toolbox`[1] under its free academic license. In the box below is the chuck of `Matlab` code that formulates the problem.

```
% the criterion function
prob.opr = repmat('log', [n 1]);
prob.opri = zeros(n, 1);
prob.oprj = (1:n)';
prob.oprf = ones(n, 1);
prob.oprg = ones(n, 1);
% the constraints
prob.c = sparse( zeros(n, 1) );
prob.a = [ ones(1,n); h'] ; % data
prob.blc = [ 1; -tau*ones(m, 1) ]; % moment lower bound
prob.buc = [ 1; tau*ones(m, 1) ]; % moment upper bound
prob.blx = sparse( zeros(n, 1) ); % lower bound of pi's
prob.bux = ones(n, 1); % upper bound of pi's
```

Under a fixed trial value β, the probability mass $p = \{p_i\}_{i=1}^n$ is the parameter to be optimized. The first five lines tell the solver the criterion function is $\sum_{i=1}^n \log p_i$. The next six lines specify the constraints. `prob.a` corresponds to the data matrix in the center of (3) below, and `prob.blc` and `prob.buc` are associated with the lower bound and the upper bound of each restriction.

$$\begin{pmatrix} 1 \\ -\tau \\ \vdots \\ -\tau \end{pmatrix} \leq \begin{pmatrix} 1 & \cdots & 1 \\ h_{11}(\beta) & \cdots & h_{n1}(\beta) \\ \vdots & \ddots & \vdots \\ h_{m1}(\beta) & \cdots & h_{nm}(\beta) \end{pmatrix} \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} \leq \begin{pmatrix} 1 \\ \tau \\ \vdots \\ \tau \end{pmatrix} \tag{3}$$

`prob.blx` and `prob.bux` specify $0 \leq p_i \leq 1$ for every i. We feed all the ingredients of the optimization problem into the following command

```
res = mskscopt(prob.opr, prob.opri, prob.oprj, prob.oprf,...
      prob.oprg, prob.c, prob.a, prob.blc, prob.buc, prob.blx, prob.bux,...
      [], 'maximize' );
```

It is recommended to check the problem status and solution status after execution. If the solution status is `'OPTIMAL'`, we are safe to collect the results; otherwise, say if the constraints are infeasible, we can assign `Inf` to the profile log-likelihood function.

[1] `http://docs.mosek.com/7.0/toolbox.pdf`

```
if strcmp( res.sol.itr.solsta, 'OPTIMAL')
     L_hat = -res.sol.itr.pobjval; % the optimal value of the function
     gam = sparse( res.sol.itr.y/n ); % collect the Lagrangian multipliers
     gam(1) = []; % remove the 1st element associated with sum(pi) == 1
else
     L_hat = Inf; % specify Inf, or a large value to indicate infeasibility
end
```