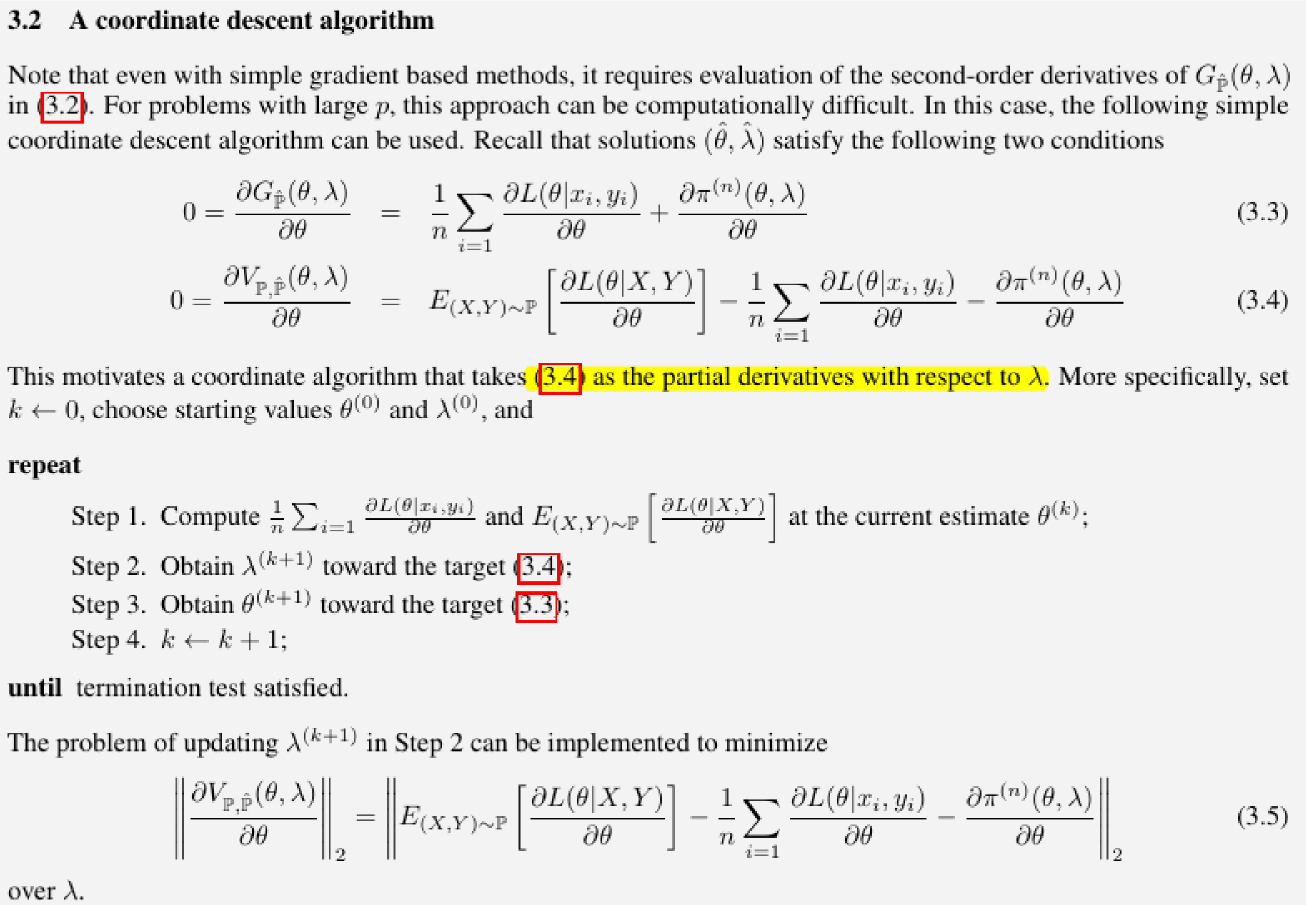
Auto-modeling on Many-Normal-Means Example

> *Lijun Wang (2022.07.31)*

Here is my attempt on reproducing the results of auto-modeling on many-Normal-means examples.

The main algorithm is



3.2 A coordinate descent algorithm

Note that even with simple gradient based methods, it requires evaluation of the second-order derivatives of $G_{\hat{\mathbb{P}}}(\theta, \lambda)$ in (3.2). For problems with large p, this approach can be computationally difficult. In this case, the following simple coordinate descent algorithm can be used. Recall that solutions $(\hat{\theta}, \hat{\lambda})$ satisfy the following two conditions

$$0 = \frac{\partial G_{\hat{\mathbb{P}}}(\theta, \lambda)}{\partial \theta} = \frac{1}{n}\sum_{i=1} \frac{\partial L(\theta|x_i, y_i)}{\partial \theta} + \frac{\partial \pi^{(n)}(\theta, \lambda)}{\partial \theta} \tag{3.3}$$

$$0 = \frac{\partial V_{\mathbb{P},\hat{\mathbb{P}}}(\theta, \lambda)}{\partial \theta} = E_{(X,Y)\sim\mathbb{P}}\left[\frac{\partial L(\theta|X,Y)}{\partial \theta}\right] - \frac{1}{n}\sum_{i=1} \frac{\partial L(\theta|x_i, y_i)}{\partial \theta} - \frac{\partial \pi^{(n)}(\theta, \lambda)}{\partial \theta} \tag{3.4}$$

This motivates a coordinate algorithm that takes (3.4) as the partial derivatives with respect to λ. More specifically, set $k \leftarrow 0$, choose starting values $\theta^{(0)}$ and $\lambda^{(0)}$, and

```
repeat
    Step 1.  Compute $\frac{1}{n}\sum_{i=1}\frac{\partial L(\theta|x_i,y_i)}{\partial\theta}$ and $E_{(X,Y)\sim\mathbb{P}}\left[\frac{\partial L(\theta|X,Y)}{\partial\theta}\right]$ at the current estimate $\theta^{(k)}$;
    Step 2.  Obtain $\lambda^{(k+1)}$ toward the target (3.4);
    Step 3.  Obtain $\theta^{(k+1)}$ toward the target (3.3);
    Step 4.  $k \leftarrow k + 1$;
until  termination test satisfied.
```

The problem of updating $\lambda^{(k+1)}$ in Step 2 can be implemented to minimize

$$\left\|\frac{\partial V_{\mathbb{P},\hat{\mathbb{P}}}(\theta, \lambda)}{\partial \theta}\right\|_2 = \left\|E_{(X,Y)\sim\mathbb{P}}\left[\frac{\partial L(\theta|X,Y)}{\partial \theta}\right] - \frac{1}{n}\sum_{i=1} \frac{\partial L(\theta|x_i, y_i)}{\partial \theta} - \frac{\partial \pi^{(n)}(\theta, \lambda)}{\partial \theta}\right\|_2 \tag{3.5}$$

over λ.

- Take another perspective of (3.3), it is just to minimize $G_{\hat{\bbP}}(\theta, \lambda)$. Thus, we can directly take advantage with existing nonlinear programming solvers, such as Ipopt (Interior Point Optimizer).
- For (3.4) and (3.5), since $\theta_2, \cdots, \theta_m \geq 0$, then

m m

$$\pi^{(n)}(\theta,\lambda) = \sum_{k=2} \lambda_k |\theta_k| = \sum_{k=2} \lambda_k \theta_k \tag{1}$$

and hence

$$\frac{\partial \pi^{(n)}(\theta,\lambda)}{\partial \theta} = [0 \quad \lambda_2 \quad \lambda_3 \quad \cdots \quad \lambda_m \quad 0 \quad \cdots \quad 0]^T \tag{2}$$

And note that $\lambda_i \geq 0$, it follows that

$$\lambda_i = \max(0, d_i), i = 2, \ldots, m \tag{3}$$

where d_i is the i-th element of

$$E_{(X,Y)\sim\mathbb{P}}[\frac{\partial L(\theta \mid X,Y)}{\partial \theta}] - \frac{1}{n}\sum_{i=1}^{n} \frac{\partial L(\theta \mid x_i, y_i)}{\partial \theta} \tag{4}$$

For many-Normal-means example,

$$\begin{aligned} L_i \triangleq L(\theta \mid y_i) &= -\log \sum_{k=1}^{m} \alpha_k \exp\left(-\frac{(y_i - \eta_k)^2}{2}\right) \\ &= -\log \sum_{k=1}^{m} \theta_{m+k} \exp\left(-\frac{(y_i - \sum_{j=1}^{k}\theta_j)^2}{2}\right), \end{aligned}$$

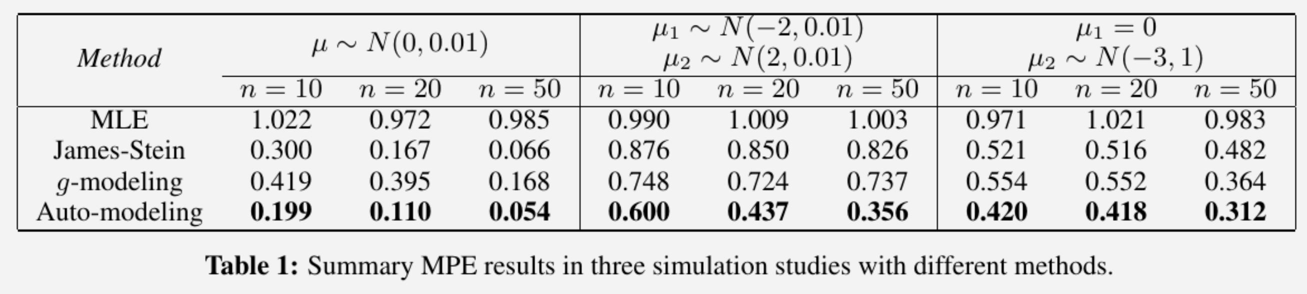
then

$$\frac{\partial L_i}{\partial \theta_\ell} = \begin{cases} -\frac{\sum_{k=\ell}^{m}\theta_{m+k}\exp\left(-\frac{(y_i - \sum_1^k \theta_j)^2}{2}\right)(y_i - \sum_1^k \theta_j)}{\sum_{k=1}^{m}\theta_{m+k}\exp\left(-\frac{(y_i - \sum_1^k \theta_j)^2}{2}\right)} & \ell \leq m \\ -\frac{\exp\left(-\frac{(y_i - \sum_1^{\ell-m}\theta_j)^2}{2}\right)}{\sum_{k=1}^{m}\theta_{m+k}\exp\left(-\frac{(y_i - \sum_1^k \theta_j)^2}{2}\right)} & \ell \geq m+1 \end{cases} \tag{5}$$

For simplicity, I skip the step of approximation of $\mathbb{P}$ by bootstrap samples, and instead directly use the true $\mathbb{P}$.

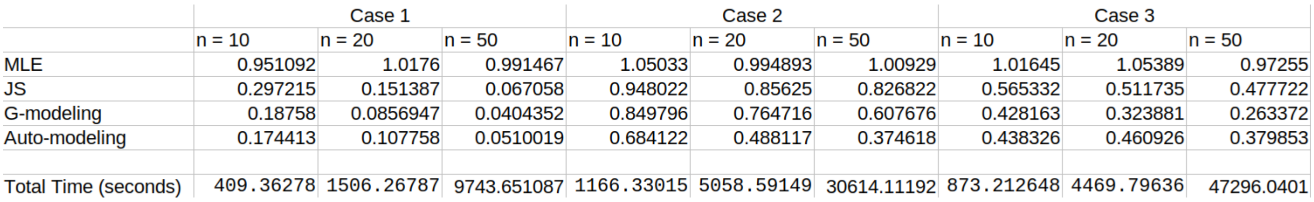
As for g-modeling, just call `deconv` function from the R package `deconvolveR`.

Like in the paper, I repeat 200 times, then report the average mean square error, the results are as follows



Method	$\mu \sim N(0, 0.01)$			$\mu_1 \sim N(-2, 0.01)$ $\mu_2 \sim N(2, 0.01)$			$\mu_1 = 0$ $\mu_2 \sim N(-3, 1)$		
	$n = 10$	$n = 20$	$n = 50$	$n = 10$	$n = 20$	$n = 50$	$n = 10$	$n = 20$	$n = 50$
MLE	1.022	0.972	0.985	0.990	1.009	1.003	0.971	1.021	0.983
James-Stein	0.300	0.167	0.066	0.876	0.850	0.826	0.521	0.516	0.482
g-modeling	0.419	0.395	0.168	0.748	0.724	0.737	0.554	0.552	0.364
Auto-modeling	**0.199**	**0.110**	**0.054**	**0.600**	**0.437**	**0.356**	**0.420**	**0.418**	**0.312**

Table 1: Summary MPE results in three simulation studies with different methods.



	Case 1			Case 2			Case 3		
	n = 10	n = 20	n = 50	n = 10	n = 20	n = 50	n = 10	n = 20	n = 50
MLE	0.951092	1.0176	0.991467	1.05033	0.994893	1.00929	1.01645	1.05389	0.97255
JS	0.297215	0.151387	0.067058	0.948022	0.85625	0.826822	0.565332	0.511735	0.477722
G-modeling	0.18758	0.0856947	0.0404352	0.849796	0.764716	0.607676	0.428163	0.323881	0.263372
Auto-modeling	0.174413	0.107758	0.0510019	0.684122	0.488117	0.374618	0.438326	0.460926	0.379853
Total Time (seconds)	409.36278	1506.26787	9743.651087	1166.33015	5058.59149	30614.11192	873.212648	4469.79636	47296.0401

Here are some observations:

- Except the results of g-modeling, others are close to the reported results in the paper.
- In my experiments, g-modeling can outperform auto-modeling, but I did not deliberately select its parameters. The paper also did not discuss how they choose the parameters for this method.
- Currently, the program is quite slow, and the computational burden is mainly Step 3 in the coordinate descent algorithm. Is there any speed-up strategies?