Summary of last semester and summer plan

Zhuoyan Xu, Jiaxin Hu

Date: 2019.7.24

1 Model

1.1 Unsupervised

Considering a binary tensor $Y \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, the unsupervised factorization model with Bernoulli assumption is:

$$\begin{aligned} &Assumption: \ Y_{ijk} \sim Ber(p_{ijk}) \ independently; \\ &\mathcal{U} = logit(\mathbb{E}Y) = logit(p_{ijk}) = \log \frac{p_{ijk}}{1 - p_{ijk}} \\ &\mathcal{U} = \mathcal{C} \times_1 M_1 \times_2 M_2 \times_3 M_3 \end{aligned}$$

where $\mathcal{C}$ is core tensor, M_1, M_2, M_3 are factor matrices. $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times r_3}, M_1 \in \mathbb{R}^{d_1 \times r_1}, M_2 \in \mathbb{R}^{d_2 \times r_2}, M_3 \in \mathbb{R}^{d_3 \times r_3}$.

Comments
1. **Degree of freedom** In this model, we consider the number of parameters as the degree of freedom, which is :

$$r_1 \times r_2 \times r_3 + r_1 \times d_1 + r_2 \times d_2 + r_3 \times d_3$$

1.2 Semi-Supervised

Consider the semi-supervised tensor factorization, with covariate. The general model is:

$$\mathcal{U} = \mathcal{C} \times_1 N_1 \times_2 N_2 \times_3 N_3$$

Where $\mathcal{U}$ indicates the hidden structure of the given tensor $\mathcal{Y}$. N_1, N_2, N_3 are the factor matrices, $\mathcal{C}$ is the core tensor. And with additional co-variate matrix X.

1.2.1 Gaussian case

Under Gaussian assumption, our model can be written as:

$$\mathcal{Y}^{d_1 \times d_2 \times d_3} = \mathcal{U}^{r_1 \times r_2 \times r_3} \times_1 N_1^{d_1 \times r_1} \times_2 N_1^{d_2 \times r_2} \times_3 N_3^{d_3 \times r_3} + [\epsilon_{ijm}]$$

Consider the covariate matrix $X^{d_1 \times p}$. We use MSE loss for our general optimization. There are 2 models to solve it.

Treat X as predictor Under this condition, we just replace A with X(or XW):

$$\min_{U,N_1,N_2,N_3} ||Y - U \times_1 X \times_2 N_2 \times_3 N_3||_F^2$$

Where $U \in R^{P \times r_2 \times r_3}$, $X \in R^{d_1 \times P}$. We just use iteration method to get result.

Under sparse condition, we add a penalty:

$$\min_{U,N_1,N_2,N_3} ||Y - U \times_1 N_1 \times_2 N_2 \times_3 N_3||_F^2 + \alpha||N_1 - X||_F^2 + \lambda||N_1||_{2,1}$$

Where α, λ is penalty parameter, the last penalty is regularization requires $N_1 \in R^{d_1 \times p}$ to be sparse.

Treat X as response If we suppose $W \in R^{d_1 \times p}$(divide individuals into p classes), there is a matrix coefficients $W \in R^{r_1 \times p}$ connect N_1 and X such as:

$$X = N_1 W$$

we have:

$$\min_{U,N_1,N_2,N_3} ||Y - U \times_1 N_1 \times_2 N_2 \times_3 N_3||_F^2 + \alpha||N_1 W - X||_F^2 + \lambda||W||_{2,1}$$

This is concretely discussed in [`Bokai Cao, Chun-Ta Lu`,*Semi-supervised Tensor Factorization for Brain Network Analysis*]. This paper use ADMM to compute the result.

1.2.2 Binary case

Under Bernoulli assumption, where response X is binary, the transformation need to be used on mean(or expectation) of response.

$$X_{ijm} \sim \quad Bernoulli(\mu_{krg})$$

Where X_{ijm} are independent from each other.

$$logit(\mathbb{E}[\mathcal{Y}]) = log(\frac{\mathbb{E}[\mathcal{Y}]}{1 - \mathbb{E}[\mathcal{Y}]}) = \mathcal{U} \times_1 N_1 \times_2 N_2 \times_3 N_3$$

Consider the covariate matrix $X^{d_1 \times p}$. We use cross entropy loss for our general optimization. There are 2 models to solve it.

Treat X as predictors The first one, we treat X as predictors. To connect the information of $\mathcal{Y}$ and X, we use matrix product to connect X and one of $\mathcal{Y}$'s factor matrices.

The general form is:

$$logit\left\{\mathbb{E}\left[\mathcal{Y}^{d_1 d_2 d_3}\right]\right\} = \mathcal{C}^{r_1 r_2 r_3} \times_1 N_1^{d_1 r_1} \times_2 N_2^{d_2 r_2} \times_3 N_3^{d_3 r_3}$$
$$N_1^{d_1 r_1} = X^{d_1 p} W^{p r_1}$$

Where $\mathcal{C}$ is the low rank core tensor of factorization.N_1, N_2, N_3 are the factor matrices, . W is the regression coefficient matrix for X on N_1. Under this scenario.

The degree of freedom of this model is

$$r_1 r_2 r_3 + p r_1 + d_2 r_2 + d_3 r_3$$

The total sample size is

$$d_1 d_2 d_3 + d_1 p$$

Note that we need to choose $r_1 < rank(X)$.

There is another view for this model: Bilinear Model.

We can write down the model in another view, which helps to compute:

$$\begin{aligned} logit\left\{\mathbb{E}\left[\mathcal{Y}^{d_1 d_2 d_3}\right]\right\} &= \mathcal{B}^{p d_2 d_3} \times_1 X^{d_1 p} \\ \mathcal{B}^{p d_2 d_3} &= \mathcal{C}^{r_1 r_2 r_3} \times_1 N_1^{p r_1} \times_2 N_2^{d_2 r_2} \times_3 N_3^{d_3 r_3} \end{aligned}$$

There are two conclusions for this model:

1. Each update has unique solution, and each update increases log-likelihood.
2. The log-likelihood is invariant to orthogonalization.

Treat X as responses The general form is:

$$\begin{aligned} logit\left\{\mathbb{E}\left[\mathcal{Y}^{d_1 d_2 d_3}\right]\right\} &= \mathcal{C}^{r_1 r_2 r_3} \times_1 N_1^{d_1 r_1} \times_2 N_2^{d_2 r_2} \times_3 N_3^{d_3 r_3} \\ logit\{\mathbb{E}\left[X^{d_1 p}\right]\} &= N_1^{d_1 r_1} W^{r_1 p} \end{aligned}$$

Where $\mathcal{G}$ is the low rank core tensor of factorization. A, B, C is three factor matrices. W is the GLM regression coefficient matrix for A on Y.

The degree of freedom of this model is

$$r_1 r_2 r_3 + d_1 r_1 + d_2 r_2 + d_3 r_3 + p r_1$$

The total sample size is

$$d_1 d_2 d_3 + d_1 p$$

2 Algorithm checking

2.1 Unsupervised

2.1.1 Upgrading Algorithm

Algorithm 1 shows the explicit steps to find the estimate of $\hat{\mathcal{C}}, \hat{M}_1, \hat{M}_2, \hat{M}_3$ which achieves the maximum likelihood. And this algorithm would also work when the observation is of continuous entries if we replace the generalized regression model with ordinary linear regression model.

Algorithm 1 Unsupervised binary tensor decomposition

Input: Y, the shape of core tensor r_1, r_2, r_3; the maximum iteration time N; the link function f; significant increment criterion ϵ

Output: The estimate of factor matrices $\hat{M}_1, \hat{M}_2, \hat{M}_3$ and core tensor $\hat{\mathcal{C}}$.

Use Tucker decomposition to get the initial $\mathcal{C}^{(0)}, M_1^{(0)}, M_2^{(0)}, M_3^{(0)}$; Calculate the initial log-likelihood value $l^{(0)}$; Set the iteration time $t = 0$

while The increment of log-likelihood $l^{(t)} - l^{(t-1)} \geq \epsilon$ and $t \leq N$ **or** $t = 0$ **do**

    (a) Upgrade the iteration index $t = t + 1$;

    (b) Upgrade $\mathcal{C}^{(t-1)} \leftarrow \mathcal{C}^{(t)}$ by solveing one *glm* model with $r_1 \times r_2 \times r_3$ coefficients and link function f. The responses and predictors are formed by Y and $M_1^{(t-1)}, M_2^{(t-1)}, M_3^{(t-1)}$.

    for i = 1,2,3 **do**

        Upgrade $M_i^{(t-1)} \leftarrow M_i^{(t)}$ by solving d_i *glm* models. The responses and predictors are formed by Y and $M_j^{(t)}, M_l^{(t-1)}, j < i, l > i$.

    (c) Calculate the log-likelihood value $l^{(t)}$ with $\mathcal{C}^{(t)}, M_1^{(t)}, M_2^{(t)}, M_3^{(t)}$

    (d) After each upgrading step, orthogonalize M_is through *tucker* decomposition,

2.1.2 Algorithm Checking (May also happen in Semi-supervised method)

1. Likelihood drop: When we first implement the algorithm, there is always a situation that the likelihood will drop to extremely small after an upgrading step. Our remedy is to find a initial value for the *glm* function or keep the result from last iteration, for instance, $M_1^{(1)} = M_1^{(0)}$.

Further Extension: Although we solve this problem in the code, but according to the proof that the likelihood must increase after any step of upgrading, we still need to investigate why this phenomenon happen. Does this phenomenon only caused by the computational weakness of the R function?

2. About choice of distribution to generate $\mathcal{C}$: In the simulation, the choice of how to generate $\mathcal{C}$ would have a significant effect to the result. And denote that $\mathcal{P} = \mathbb{E}Y = inv.logit(\mathcal{U})$,$\mathcal{U} = \mathcal{C} \times_1 M_1 \times_2 M_2 \times_3 M_3$.

If we choose Gaussian entries, we use $rnorm(mean, sd)$ to generate the core tensor $\mathcal{C}$.

When we choose $mean = 0, sd = 1$, plot between vectorized $\mathcal{P}$ vs $\hat{\mathcal{P}}$ would become a "star-shape". As ($Figure1$) shows: That implies that, when we choose sd with very small value, the entries of probability matrix $\mathcal{P}$ would around 0.5, which would lead to the estimate hard. And that's why the plot is "star-shape" rather than a positive relative shape.

When we choose $mean = 0, sd = 10$, plot between vectorized $\mathcal{P}$ vs $\hat{\mathcal{P}}$ would become what we expected. As ($Figure2$) shows: When we choose larger sd, the probability of each entry would be more scattered, therefore gives a good result.

When we choose $mean = 10, sd = 1$, plot between vectorized $\mathcal{P}$ vs $\hat{\mathcal{P}}$ would become what we expected. As ($Figure3$) shows: It is weird that there seems has a horizon line shows that the estimate doesn't work.

If we choose uniform entries, the result shows the larger the interval of the uniform

distribution is, the better the result is:

When we choose interval to be [0,1], plot between vectorized $\mathcal{P}$ vs $\hat{\mathcal{P}}$ showed a erratic pattern, the $\mathcal{P}$ is more cluster to 0 than $\hat{\mathcal{P}}$. As Figure 4 shows: When we choose larger interval, the probability of each entry would be more scattered, therefore gives a good result.

When we choose interval to be [0,10], plot between vectorized $\mathcal{P}$ vs $\hat{\mathcal{P}}$ improved a little, still, the $\mathcal{U}$ is more cluster to 0 than $\hat{\mathcal{P}}$. As figure 5 shows:

When we choose interval to be [0,100], plot between vectorized $\mathcal{P}$ vs $\hat{\mathcal{P}}$ becomes more ideal. As figure 6 shows:

Further extension: In conclusion, it is important for us choose a proper distribution to generate the core tensor. We need to understand more deeply on the result respect to different distributions.

3.When to stop? It is reported that when we run more iterations, the result sometime seems not better or even worse.(As ($Figure7$) shows)
Further extension: Since the likelihood indeed is larger when we run more iteration, the question that why the result is worse still need to investigate. And how can we choose a threshold to stop the algorithm still need to discuss.

4.How to choose the rank for real data? As we use log-likelihood as the criterion of the upgrade algorithm, it is apparent that when we choose a larger rank for the core tensor, we will get a larger log-likelihood in the end.
Further extension: It needs to investigate that how to choose a proper rank of the core tensor when in real data problem.

5.How to deal with the outliers? Sometimes, when we check the real scale of $\hat{\mathcal{U}}$, there may some entries with extremely large values (As ($Figure$ 8) shows). To deal with the outlier, we can do some thresholding or scaling to the $\hat{\mathcal{U}}$ after each update.

2.2 Semi-Supervised

2.2.1 Treat X as predictor

The explict steps are shown below:

Algorithm 2 Semi-binary tensor decomposition

1: **Input** $\mathcal{Y}, X$, the shape of core tensor r_1, r_2, r_3, simulation times N
2: Use GLM to get $\mathcal{C}^{(1)}$, then use Tucker decomposition to get $N_1^{(1)}, N_2^{(1)}, N_3^{(1)}$
3: **for** $n = 1, 2, \ldots, N$ **do**
4: (a) Update N_2, N_3 use matrix form GLM, $N_2^{(n)} \leftarrow N_2^{(n+1)}, N_3^{(n)} \leftarrow N_3^{(n+1)}$, then orthogonalize N_2, N_3.
5: (b) Update $\mathcal{C}$ using vectorization of tensor, $\mathcal{C}^{(n)} \leftarrow \mathcal{C}^{(n+1)}$
6: (c) Update N_1 using vectorization of matrix, $N_1^{(n)} \leftarrow N_1^{(n+1)}$, then orthogonalize N_1.
7: Output $\mathcal{C}, N_1, N_2, N_3$.

2.2.2 Treat X as response

Update $\mathcal{C}, N_2, N_3$ as the same as before

Update N_1 when fixed $\mathcal{C}, N_2, N_3$

Consider

$$\begin{aligned}
\mathcal{U}_Y^{d_1 d_2 d_3} =& logit\left\{\mathbb{E}\left[\mathcal{Y}^{d_1 d_2 d_3}\right]\right\} = \mathcal{C}^{r_1 r_2 r_3} \times_1 N_1^{d_1 r_1} \times_2 N_2^{d_2 r_2} \times_3 N_3^{d_3 r_3} \\
=& \mathcal{C}_{23}^{r_1 d_2 d_3} \times_1 N_1^{d_1 r_1} \\
Then: \mathcal{U}_{Y(1)}^{d_1 \times d_2 d_3} =& N_1^{d_1 r_1} \mathcal{C}_{23(1)}^{r_1 \times d_2 d_3} \\
U_X^{d_1 p} =& N_1^{d_1 r_1} W^{r_1 p}
\end{aligned}$$

Where $U_{Y(1)}$ is the unfold of $\mathcal{U}_Y$ through mode-1. Thus, we have:

$$\left[U_{Y(1)}, U_X\right] = N_1 \left[\mathcal{C}_{23(1)}, W^{r_1 p}\right]$$

We can use matrix form GLM to realize it.

3 Summer plan

Here we give a simple/incomplete summary for our summer.

- In the algorithm check part, there are a lot of further extension problems need us to figure out. And we want to make the algorithm and the code more solid.
- There are also some theoretical properties need us to find.
  Consider unsupervised model, when we fixed size of core tensor, what will the MSE of our estimator $\hat{\mathcal{U}}$ change as data tensor size goes infinity? Is there a rate between $\|\hat{\mathcal{U}} - \mathcal{U}\| \longrightarrow 0$ and $d_1, d_2, d_3 \longrightarrow \infty$?
  We can consider a simple version:
  - Linear regression on Gaussian assumption, how would $\|\hat{\beta} - \beta\| \longrightarrow 0$ change in terms of n and σ^2?
  - Logistic regression on binary assumption, how would $\|\hat{\beta} - \beta\| \longrightarrow 0$ change in terms of n and p?

  And what's the properties of the estimate under the semi-supervised model also need to be considered.

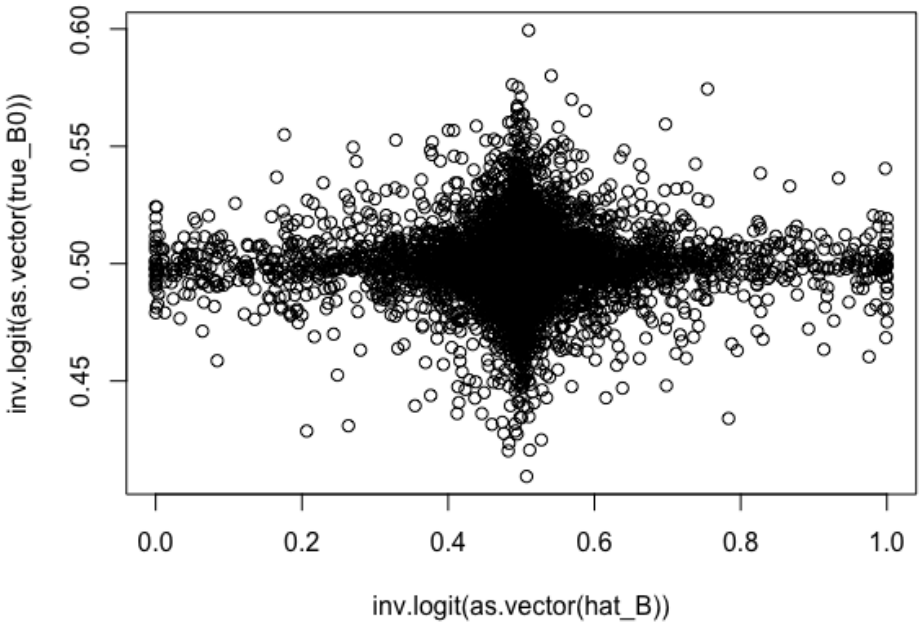

Figure 1: $\hat{\mathcal{U}}$ *vs* $\mathcal{U}$ when $mean = 0, sd = 1$

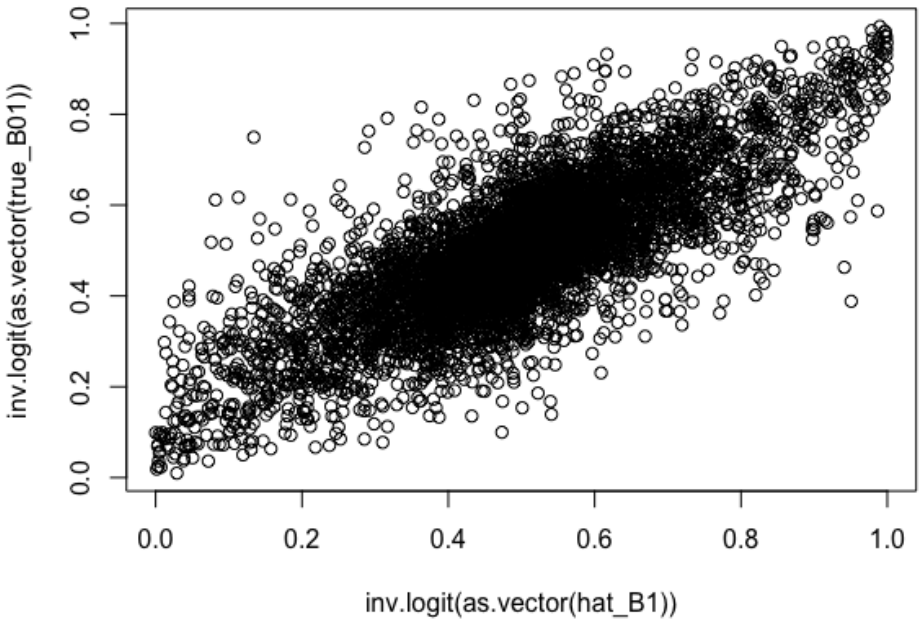

Figure 2: $\hat{\mathcal{P}}$ *vs* $\mathcal{P}$ when $mean = 0, sd = 10$

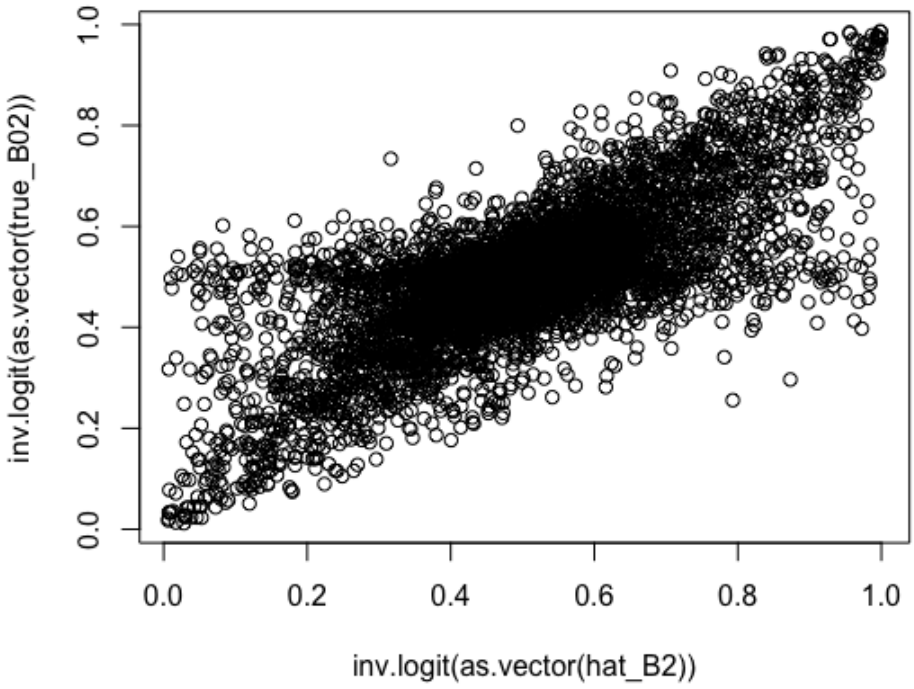

Figure 3: $\hat{\mathcal{P}}$ *vs* $\mathcal{P}$ when $mean = 10, sd = 10$

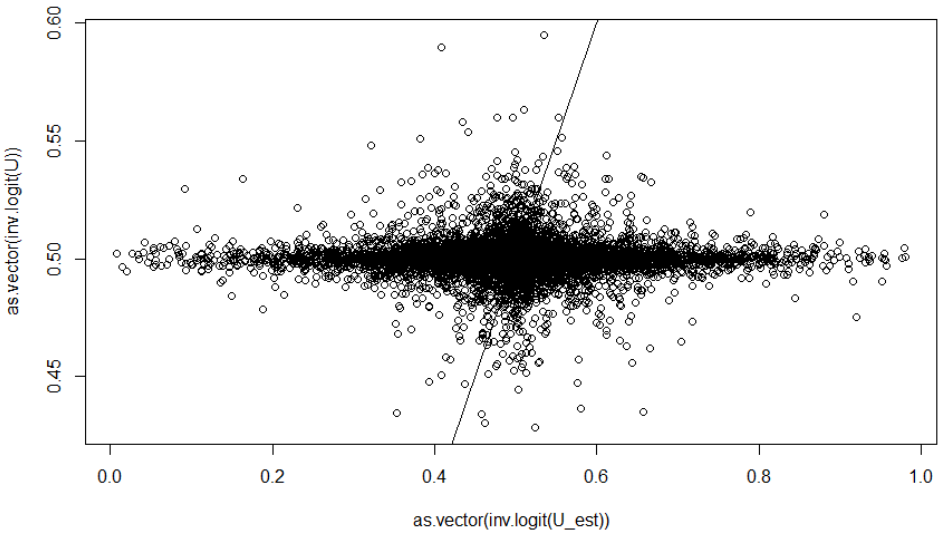

Figure 4: $\mathcal{P}$ *vs* $\hat{\mathcal{P}}$ when [0,1]

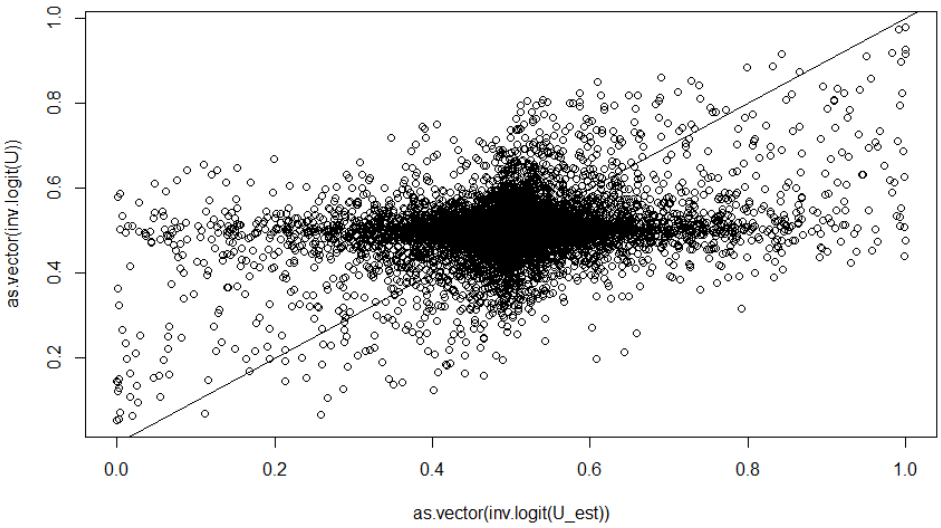

Figure 5: $\hat{\mathcal{P}}$ *vs* $\mathcal{P}$ when [0,10]

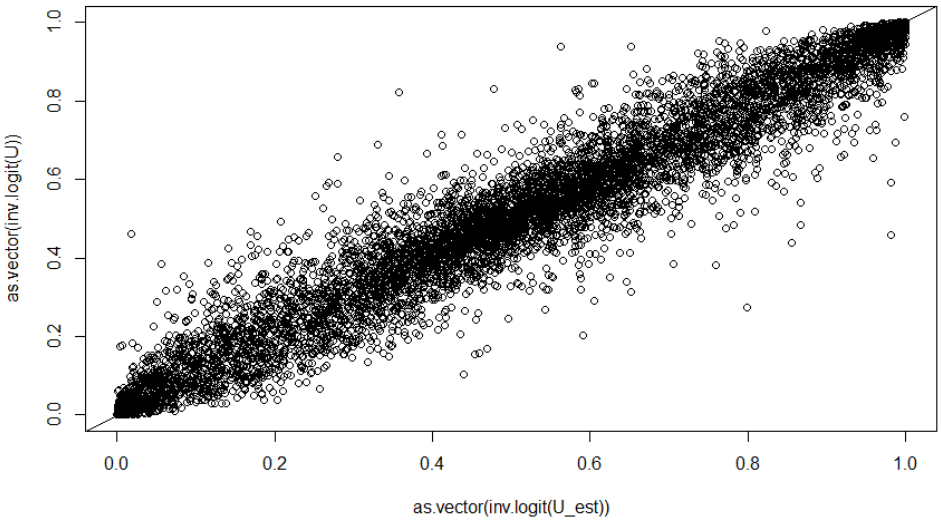

Figure 6: $\hat{\mathcal{P}}$ *vs* $\mathcal{P}$ when [0,10]

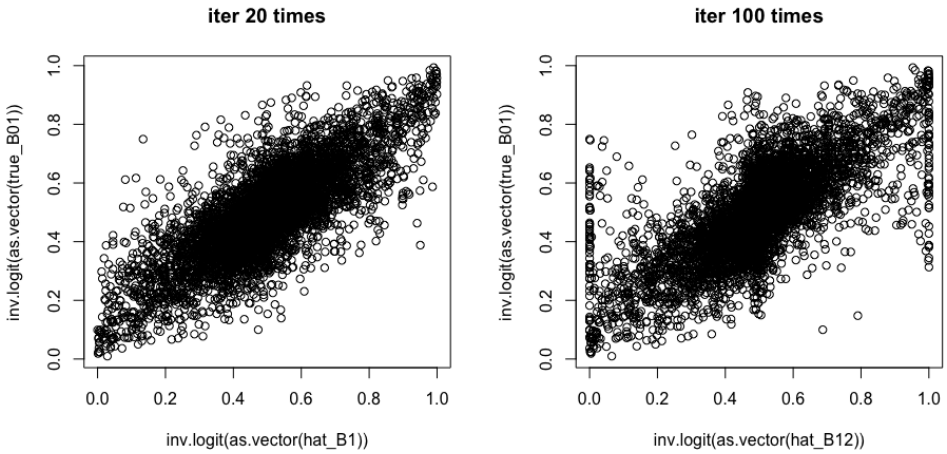

Figure 7: $\hat{\mathcal{P}}$ vs $\mathcal{P}$ when iter 20 times or 100 times

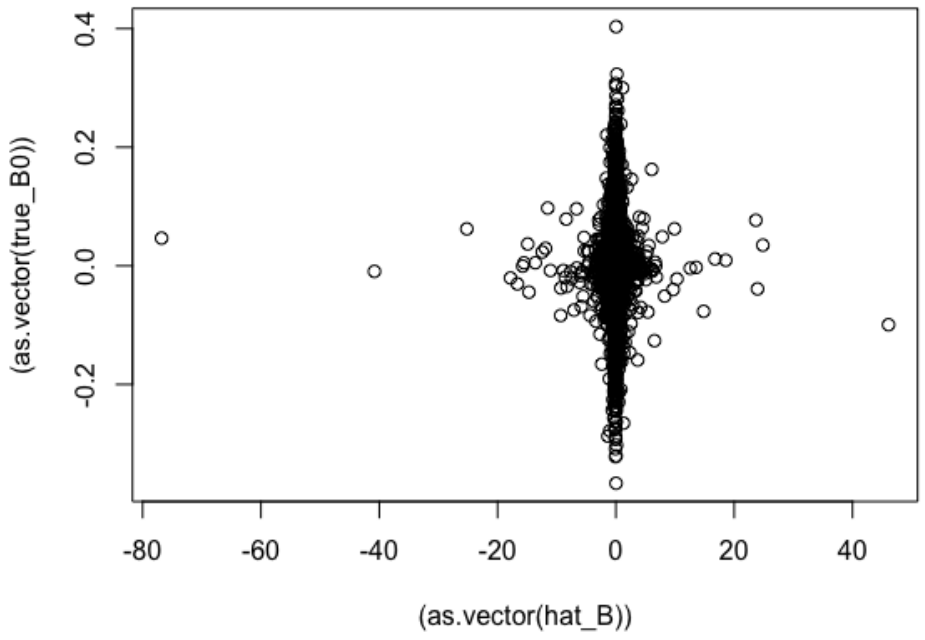

Figure 8: There are some extreme entries in $\hat{\mathcal{U}}$ with large absolute value.