# Report

Xinzhou Yan CDSOR A20376898

April 27, 2018

### 0. Claim:

I claim that all the materials is finished by myself. There is no plagiarism in the reports or code. Especially the code, which was uploaded to the GITHUB, was carefully written and debugged by myself. So for any similar code, it is not related to mine.

### 1. Title and author of paper:

This paper was published by American Statistical Association in 1993, whose author is Edward I. George and Robert E. McCulloch. Edward I. George ,who achieved Ph.D. in Statistics field of Stanford University 1981, is teaching in the University of Pennsylvania, The Wharton School.His research interest are hierarchical modeling, model uncertainty, shrinkage estimation, treed modeling, variable selection, wavelet regression. Robert E. McCulloch holds MS and PhD degrees in statistics from the University of Minnesota.Now he is teaching courses in the Data Science field in the Uchicago. The background survey shows us the paper's scholar tune. The paper was published more than 20 years ago, two years before Windows 95 was released. The materials are organized via carefully old-school statistic deduction. So there are two potential problems with the content of the paper." Can the model deal with high dimensional big data?" "Will modern methodology simplify the process?"

### 2. Summary:

The paper suggests a model $\beta|\gamma \sim N_p(0, {D_\gamma}^T R D_\gamma), \quad Y|\beta, \sigma^2 \sim N_n(X\beta, \sigma^2 I)$ for the regression situation. It is a hierachical model for variable selection which all weights $\beta$ will depend on $\gamma$. If $\gamma_i$ is 0, then model will not include $\beta_i$, else if $\gamma_i$ is 1, the model will include $\beta_i$. This paper try to get posterior distribution $f(\gamma|Y)$, which is realized in 4 steps. Firstly, assume the prior distribution of all model parameters $[\gamma, \beta, \sigma^2]$. Secondly, achieve posterior distribution of all model parameters $[\gamma, \beta, \sigma^2]$. Thirdly,tune all the relative parameters and prove feasibility. Finally, count the frequency of $f(\gamma|Y, model)$ via Gibbs Sampling and output the results ranking. The key of the Gibbs Sampling is to compare different possibility of $f(\beta_i|\gamma_{(i)}, \gamma_i = 1)$ and $f(\beta_i|\gamma_{(i)} p_i, \gamma_i = 0)(1 - p_i)$. This value will decide whether this turn changes $\gamma_i$ 's value. In next section, the four sections will be carefully discussed.

## 3. Methodology:

3.1 Prior distribution
For $\gamma$ , which tells whether the model includes some feature.

$$f(\gamma) = \prod {p_i}^{\gamma_i}(1-p_i)^{\gamma_i}$$

This formula is simplified via symmetry consideration and implies the inclusion of feature is independent of other feature. Because $f(\gamma)$ is multiple of Bernoulli distribution. Here $p_i$ is 0.5 as each feature has the same prior distribution.

For $\beta$ , the weights of the model,comes from a mixture of two normal distributions.

$$f(\beta_i|\gamma_i) \sim (1-\gamma_i)N(0,{\gamma_i}^2) + \gamma_i N(0,{c_i}^2{\gamma_i}^2)$$

This is the key of the prior part. If $\gamma_i = 1$, $f(\beta_i|\gamma_i) \sim N(0,{c_i}^2{\gamma_i}^2)$.
If $\gamma_i = 0$, $f(\beta_i|\gamma_i) \sim N(0,N(0,{c_i}^2{\gamma_i}^2)$. This is the trick of the model. If feature is not in the model, the weight will be more likely drown from a small variance distribution around zero.If feature is in the model, the weight will be more likely drown from a larger variance distribution around zero, which the variance is tuned via c. The c is the key parameter decide the different pdf from two distribution. The tuning part will be explained in detail in 3.3.

For $\sigma^2$ , the noise amount of the model,comes from a inverse-gamma distribution.

$$\sigma^2|\gamma \sim IG(v_r/2, v_r\lambda_r/2)$$

$v_r$ , $\lambda_r$ is the degree of freedom which depends on $\gamma$.

3.2 Posterior distribution

For $\beta$,

$$\beta^j \sim f(\beta^j|Y,\sigma^{j-1},\gamma^{j-1})$$

$$N_p({A_\gamma}^{j-1}(\sigma^{j-1})^{-2}X^TX\tilde{\beta}_{LS},{A_\gamma}^{j-1})$$

where:

$${A_\gamma}^{j-1} = ((\sigma^{j-1})^{-2}X^TX + D^{-1}_{\gamma^{j-1}}R^{-1}D^{-1}_{\gamma^{j-1}})^{-1}$$

Each time, Gibbs Sampler will generate one set of $\beta$ vectors given current $\sigma$, $\gamma$, Y. The $\beta$ vector will be used to update $\sigma$ in this iteration.

For $\sigma$:

$$\sigma^j \sim f(\sigma^j|Y,\beta^j,\gamma^{j-1})$$
$$IG(\tfrac{n+{v_r}^{j-1}}{2},\tfrac{|Y-X\beta^j|^2+{v_r}^{j-1}{\lambda_r}^{j-1}}{2})$$

In the paper, the $v_r$ is set to zero, the $\lambda_r$ is set to 1, so the Inverse-gamma function relates little to the prior distribution. Reversely,the result of $\sigma$ is very close to the data.

For $\gamma$:

$$\begin{aligned}{\gamma_i}^j &\sim f({\gamma_i}^j|Y,\beta^j,\sigma^j,{\gamma_{(i)}}^j)\\ &= f({\gamma_i}^j|\beta^j,\sigma^j,{\gamma_{(i)}}^j)\end{aligned}$$

This is the key part of the Gibbs Sampling process, here Gibbs Sampling process diverges into two different results, two posterior distribution a,b of $\beta^j$ given $\gamma$ in this iteration.

For a:

$$a = f(\beta^j|{\gamma_{(i)}}^j,{\gamma_i}^j=1)p_i$$

For b:

$$b = f(\beta^j|{\gamma_{(i)}}^j,{\gamma_i}^j=0)p_i$$

Since each distribution is Bernoulli with probability:

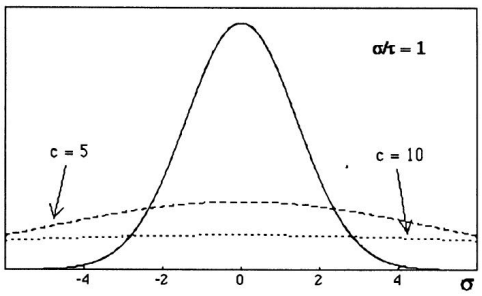
$$p({\gamma_i}^j=1|\beta^j,\sigma^j,{\gamma_{(i)}}^j) = \tfrac{a}{a+b}$$

The Gibbs sampler will reject ${\gamma_i}^j = 1$ if the p<=0.5, will accept ${\gamma_i}^j = 1$ if p>0.5. The reason is embedded in the posterior distribution of $\beta$. In figure 1, there prior has different distribution over $N(0,\sigma^2+\tau^2)$ and $N(0,\sigma^2+c^2\tau^2)$, when c is larger, the difference will be clearer. But the $|\sigma/\tau|$ will be also larger. The trade off of the tuning will be discussed in next part.

3.3 Tuning

R is the covariance matrix of X, which can be easily calculated in the python. If R is assumed as I, it is equal to assume each feature is independent of each other.
$v_r$, $\lambda_r$ is the constant in the inverse gamma prior function, where $v_r$ is the number of observation and $v_r/(v_r\text{-}2)\lambda_r$ is the prior estimation of $\sigma^2$. In the experiment, the $v_r$ is set to zero to represent ignorance.

$\tau_i$, $c_i$ appear in the $\beta$ 's prior distribution.




b

Figure 1: The Marginal$N(0, \sigma^2 + \tau^2)$ and $N(0, \sigma^2 + c^2\tau^2)$

$$f(\beta_i|\gamma_i) \sim (1 - \gamma_i)N(0, {\gamma_i}^2) + \gamma_i N(0, {c_i}^2{\gamma_i}^2)$$

First, the paper set $\tau_i$ small so if $\gamma_i$ =0. $\beta_i$ will be safely around zero. Then the paper set $c_i$ big so non-0 estimate of $\beta_i$ should probably be included in the final model. But it raises the problem, how big should $c_i$ be? If too small, two distribution will be very similar, if too big, $\beta_i$ with $\gamma_i = 1$ 's the probability will be higher in $N(0, {c_i}^2$ rather than $N(0, {c_i}^2{\gamma_i}^2)$. Because $|t_i|$ is far away from zero.In the example 1, author just select c value as 10, which is correspond to the t statistics 2.7. In the experiment, feature $\beta_4$ is nearly 2.7 which could be rejected as the high significance. feature $\beta_5$ is nearly 3.9 which will be also kept if running a stepwise regression model. It is tricky to set c, it is kind of arbitrarily tuning. But it will not be wrong set c correspond to t statistics.

3.4 Count the frequency

The experiment series follow these steps:

$$(\beta^0, \sigma^0, \gamma^0) \Rightarrow (\beta^1, \sigma^1, \gamma^1) \Rightarrow (\beta^j, \sigma^j, \gamma^j) \Rightarrow .....$$

The algorithm first set ups $\beta$ from the least squares estimation.Then, set up $\sigma$ according to the inverse gamma distribution. After than, set $\gamma$ list to 1, which means all $\beta$ in the model. After initialization, the algorithm runs into iterations. It iteratively sample $\beta$ based on last' iteration's information, then sample $\sigma$ base on $\beta$, $\gamma$. After than, it is possible to get $\gamma$ 's posterior values base on the $\beta$, $\sigma$ just sampled. After this iteration, it will start from $\beta$ again. The algorithm will converge after 5000 iterations. After trowing the burn out, the remaining record shows the result of experiment 1 like following.

[(array([0., 0., 0., 1., 1.]), 232),
(array([0., 0., 1., 1., 1.]), 225),
(array([1., 0., 0., 1., 1.]), 207),

(array([0., 1., 0., 1., 1.]), 199),
(array([1., 1., 0., 0., 1.]), 195)]

Each line is the record and frequency of record. E.g. [0., 0., 0., 1., 1.] shows $\beta 4$, $\beta 5$ is included in the model.232 is the how many time this record show in 2500 iterations. The first 3 records are similar in frequency as the setting up is different from the paper. In the experiment, R is retrieved from X's covariance matrix. In the paper , R is set up as diagonal matrix of 1, which assumes no correlation before experiment. This is the part I can not agree with.

In problem 2, $\beta 3$ and $\beta 5$ have high correlation .98.

[(array([0., 1., 1., 1., 1.]), 515),
(array([0., 0., 1., 1., 1.]), 503),
(array([1., 0., 1., 1., 1.]), 479),
(array([1., 0., 1., 0., 1.]), 434),
(array([1., 1., 1., 1., 1.]), 426)]

What is interesting is the probability of record is more dense than in the problem 1.

**4. Skill improvement:**

In this paper, it implements Gibbs Sampler to select features during linear regression. Last semester I implemented stepwise regression to select features for linear regression. The running time complexity is O(k$n^2$), where k is the number of features, n is the number of data. It is very slow when there is ton of data and many features. In the Gibbs sampling, it may do worse. As the running time complexity is O(M$n^2$+M*k), where M is the number of iteration. So Gibbs sampling will only do better when the data is super sparse matrix. But the SSVS framework is very flexible, it can be implemented to the non-linear classification while stepwise regression may not. Next time it is worthy a try to implement SSVS to logistic regression.

It is clear the skill summery can be divided into three parts. The first part linear regression: it is a very common algorithm. In the machine learning area, the developer will use ridge regression to simplify the model. A regularization term is adds to the model to penalize the feature complexity. During gradient descent, the non relative term's coefficients will be automatically drop to 0. Another method is PCA to reduce the dimension of the data.Model with more features will have less bias but more variance. Model with less features will have more bias but less variance. So it is a trade off, simple model will be more robust to data.

The second part is to learn Gibbs sampler. The Gibbs sampler can sample data

which are not conjugate. Firstly, Gibbs sampler divides features into different groups. Within each group, features are conjugate.Then Gibbs sampler iteratively update model parameters to sample the data. It is a MCMC method so normally it will converge and become stationary. Discard the first half records, the final data records the posterior distribution of parameters. Here, what looks promising is that it is possible to simulate the parameters in the distributed cloud computation if using metropolis hasting. When implementing the paper, I just scan the setting up of the SSVC and focus on the section three.As the other parts are almost talking about why the SSVS is superior and feasible. If author added the middle deduction from prior distribution to posterior distribution, it can not be better.

The third part is to talk about programming. In Python, the matrix manipulation is very trick. When two matrix a,b (two numpy.ndarray )multiple with each other, a*b, a.dot(b), numpy.matmul(a,b), it requires tons of carefulness to take care each formula having matrix manipulation. This calculation bug will not pop out error and it takes tons of time if any process has mistakes.What to do when the final values is wield, it is possible to set break point after each step during Gibbs sampling because each step depends on last step. If we insure the bug appears not in this step, we are sure no bug in previous step.Another common bug is try to write function by self, it wastes too much time to build wheel which is already existing. What is more, this self-made wheels are full of pitfall. Tons of bugs will hide in unbelievable corners in the scripts. Finally, it is very important to put each formula into function, it is very important to be a readable code as the code may need to be revised and change in the future. When the setting up changes, the clean and tidy code is like an old friend. During this semester, I have to add some applications to other person's code. His code is so confusing and without comment. Changing his code is like walking in the muddy swamp, bugs are like numerous alligators beneath water staring the game.From then on, I make up my mind to write code as clean and simple as possible like Juyi Bai's poet.Debugging in Python is not so comfortable as in Java, it is easy to debug in eclipse. However, the IDE for Python is much more worse, Jupyter notebook has no debugger. The debugger in Spyder is full of bugs itself. The feasible way is using pdb library to debug, try and catch exception in pdb.set_trace(), pdb.set_trace() has the same function as break point. So it is possible to catch bug conditionally.

In problem 1.1, the result of the simulation is good, the simulation successfully detect the relative feature. In problem 1.2, the result of the problem is not so satisfying, the simulation also include some other feature in the record which ranks first. In the problem 2.1 the out side data set, train linear model on 2016 NYC Yellow Cab trip record data. The data is recorded during the taxi trip in 2016 New York Manhattan area.
The features include distance,t_sin_hour,t_cos_hour,t_sin_day,t_cos_day, holiday,number_of_steps,total_distance,minimum temperature.

These data are from Kaggle competition to predict Manhattan's Taxi travel estimation. The first very important step is to normalize the data. The result turns out to be strange. Only two type of record are generated.

[(array([1., 0., 1., 1., 1., 1., 0., 1., 1.]), 2477),
(array([1., 0., 1., 1., 0., 1., 0., 1., 1.]), 23)]

The result is dubious as t_sin_hour,number_of_steps is excluded when c is 10. number_of_steps shows how many times the car turns direction, which can roughly estimate how many red light the car get.Let's try set c to 50, the result is still wield. To sum up, the algorithm is slow and not precise in big data and high dimension.I used XBGboost to do the training on the whole data set.(Here I use Gibbs sampler for only 5000 records.)The model can output coefficient quickly and precisely. Irreverent features will get a very low coefficient. My conclusion is the SSVS may be useful but the technology really advance in these 25s years.