Non-convex optimization in RNA-seq models

Kushal K Dey

December 29, 2015

Overview

RNA-seq is a popularly used technique in genetics/genomics to map reads (small nucleotide sequences)coming from a sample (may be single cell or a tissue sample) to the genome. The number of reads mapped to each gene is recorded, resulting in counts data. The number of samples or repetitions of this experiment are in the range of 100 to 1000, whereas the number of genes range from 20,000 to 50,000. The reads data may be statistically analyzed as it is (Poisson framework for counts) or may be normalized to RPKM values which is amenable to continous distribution fitting (typically Normal framework).

Under topic model settings, we use a mixed membership approach to determine the proportion of reads coming from different cell subpopulations (tissue level data) or cell phases (single cell data). In simple words, we wish to soft cluster the samples into groups and then record the proportional membership of the sample to each cluster orr group.

Model

Reads model If we model the counts of reads, then we may use the following Poisson framework. Let c_{ng} be the number of counts of reads from sample n that get mapped to gene g. Suppose there are N samples and G genes. Then for nth sample and g the gene, assume the model

MODEL 1 - Idnetity link model

$$c_{ng} \sim c_{n+} Poi(\sum_{k=1}^{K} \omega_{nk}\theta_{kg}) \qquad \sum_{k=1}^{K} \omega_{nk} = 1 \forall n \qquad \sum_{g=1}^{G} \theta_{kg} = 1 \forall k$$

where c_{n+} is the sum of counts across G features. The prior for ω may be considered as

CHOICE 1

$$(\omega_{n1}, \omega_{n2}, \cdots, \omega_{nK}) \sim Dir_K(\frac{1}{K}, \frac{1}{K}, \cdots, \frac{1}{K})$$

CHOICE 2

$$(\omega_{n1}, \omega_{n2}, \cdots, \omega_{nK}) \sim Dir_K(1, 1, \cdots, 1)$$

For θ, choose the prior

$$(\theta_{k1}, \theta_{k2}, \cdots, \theta_{kG}) \sim Dir_G(\frac{1}{KG}, \frac{1}{KG}, \cdots, \frac{1}{KG})$$

RPKM model If we are provided with the RPKM values, which are log transformed and scaled values obtained from the reads or counts values, we can fit the following model.

Let r_{ng} be the RPKM for sample n and gene g. Then we can write

$$r_{ng} \sim N(\bar{r}_n + \sum_{k=1}^{K} \omega_{nk}\alpha_{kg}) \qquad \sum_{k=1}^{K} \omega_{nk} = 1 \quad \forall\, n \qquad \sum_{g=1}^{G} \alpha_{kg} = 0 \quad \forall\, k$$

where $\bar{r}_n$ is the mean RPKM value for n th sample. The prior choice for ω taken similar to the reads model described above and treat α more like a hyperparameter with constraint.

Simulation experiments

I am providing the simulated data for the three models so that they are ready to be applied the model and tested for whether the method works on these examples (the estimated ω and α should match with the true values from which the data has been simulated from).

MODEL 1 - Reads level Poisson model

```
K=4;
G=100;
N=500;

alpha_true=matrix(rnorm((K)*G,0.5),nrow=(K)); ### the matrix of fixed effects

library(gtools)
T=10;
omega_true=matrix(rbind(rdirichlet(T*10,c(3,4,2,6)),rdirichlet(T*10,c(1,4,6,3)),
            rdirichlet(T*10,c(4,1,2,2)),rdirichlet(T*10,c(2,6,3,2)),
            rdirichlet(T*10,c(3,3,5,4))), nrow=N);

theta_true <- matrix(rbind(rdirichlet(1,c(2,2,rep(1,G-2))),rdirichlet(1,c(rep(2,G-2),1,1)),
                  rdirichlet(1, c(1, rep(2,G-2),1)),rdirichlet(1, c(1,1,2,2,2,rep(1,G-5)))),
                  nrow=4);


###  generating the table

read_counts_model=matrix(0,N,G);
for(n in 1:N)
{
    for(g in 1:G)
    {

        mean=1000*omega_true[n,]%*%theta_true[,g];
        read_counts_model[n,g]=rpois(1,mean);
    }
}
```

MODEL 2 - RPKM Normal model

```
K=4;
G=100;
N=500;

alpha_true=matrix(rnorm((K)*G,0.5),nrow=(K)); ### the matrix of fixed effects

library(gtools)
T=10;
omega_true=matrix(rbind(rdirichlet(T*10,c(3,4,2,6)),rdirichlet(T*10,c(1,4,6,3)),
            rdirichlet(T*10,c(4,1,2,2)),rdirichlet(T*10,c(2,6,3,2)),
            rdirichlet(T*10,c(3,3,5,4))), nrow=N);

###  generating the table

rpkm_model=matrix(0,N,G);
for(n in 1:N)
{
    for(g in 1:G)
    {

        mean=omega_true[n,]%*%alpha_true[,g];
        rpkm_model[n,g]=rnorm(1,mean);
    }
}
```

Discussion

- Of the parameters, only ω are constrained, they sum to 1 for each n. The idea would be to de-simplex them using

$$\nu_{nk} = log(\frac{\omega_{nk} + 1e-06}{\omega_{n1} + 1e-06}) \qquad k = 2, 3, , \cdots, K \qquad \forall n$$

$$\psi_{kg} = log(\frac{\theta_{kg} + 1e-06}{\theta_{k1} + 1e-06}) \qquad g = 2, 3, , \cdots, G \qquad \forall k$$

Carry out the optimization on ν_{nk} and then reverse transform them.

- The time and computational expense is an important issue in this model fit. Note that the real data model will have 100 s of samples and $20,000$ to $50,000$ genes. So, need to see how fast the algorithm is on this toy dataset provided above, and how it scales for increase in number of samples and genes. So, report the time expenditure (using *system.time()*). We may need to use variable selection methods to select genes if it is found to be slow.
- We shall apply the method on real data (both reads and RPKM) once the method has been validated on the above simulation examples.
- The model may have overdispersion effects and also possible technical effects, which would make the model slightly more complicated, but we are not worrying about these as of now. First, we focus on making the method work on the simple set ups.