Demo_RW

March 17, 2019

```
In [1]: suppressMessages(library(scDDboost))
        load("simulation_data/K7/sim_neg0.1_1_1.rds")
        load("simulation_data/K7/res_neg0.1_1_1.RData")
        load("simulation_data/K7/res_neg0.1_1_1D.RData")
```

1 Background

First, I want to briefly review the model for random weighting.

We assume weights $w_{i,j} \sim \text{Gamma}(a, b)$, let's focus on original way that $a = b$, $w_{i,j} \sim \text{Gamma}(a, a)$ with mean 1

The bayesian model for distance $d_{i,j}$ contains two parts

1.Prior of true distance $\Delta_{i,j}$ is inverse gamma distributed, i.e. $1/\Delta_{i,j} \sim \text{Gamma}(a_0, d_0)$

2.Conditional density $d_{i,j}|\Delta_{i,j} \sim \text{Gamma}(a_1, a_1/\Delta_{i,j})$ with mean $\Delta_{i,j}$

By conjugacy of gamma distribution, we know the posterior $1/\Delta_{i,j}|d_{i,j} \sim \text{Gamma}(a_0 + a_1, d_0 + a_1 * d_{i,j})$. We want our randomly generated distance $d^*_{i,j} = d_{i,j}/w_{i,j}$ to have similar distribution as the posterior $1/\Delta_{i,j}|d_{i,j}$

2 Estimation procedure

Goal: We want to estimate a by our estimated distance $d_{i,j}$.

We use two steps to do so

1. We infer (a_0, d_0, a_1) by $d_{i,j}$. I think this part is ambiguous and I will give one example to illustrate how I do it

2. After we estimated (a_0, d_0, a_1), we use them to determine a the paramter for weights. In the original way, $a = a_0 + a_1$

3 Fixing d_0 and estimating a_0 and a_1 by optimizing marginal likelihood of $d_{i,j}$

We are estimating (a_0, a_1) by optimizing the marginal likelihood of $d_{i,j}$. It is good to know the formula for marginal likelihood of $d_{i,j}$ is

$$P(d_{i,j}|a_0, a_1, d_0) = \frac{\Gamma(a_0 + a_1)}{\Gamma(a_0)\Gamma(a_1)} \frac{d_0^{a_0} d_{i,j}^{a_1 - 1} a_1^{a_1}}{(d_0 + a_1 * d_{i,j})^{a_0 + a_1}}$$

below is the function to calculate marginal log likelihood of $d_{i,j}$

```
In [2]: LL = function(param, x, d0){
          a0 = param[1]    #shape for prior
          #d0 = 1    #rate for prior, fixed
          a1 = param[2]   #shape for sampling model, consistent with dividing

          n = length(x)

          nc = ncol(x)

          I = matrix(1,nc,nc)

          I = I - diag(nc)

          C = d0 + a1 * x

          res = (n - nc) * (lgamma(a0 + a1) - lgamma(a0) - lgamma(a1) + a0 * log(d0) + a1 * log(

          return(-res)
        }
```

4 the part I screwed up is that I should put a bigger threshold for stopping criterion of nlminb function

There is a relative tolerance (reltol) threshold paramter in nlminb. The algorithm stops if it is unable to reduce the objective value by a factor of reltol * (abs(objective) + reltol). That is if we can not further reduce our objective function by at least some amount we should stop. The default value of relative tolerance is 1e-10

The optimizing result with default threshold is below:

We fix $d_0 = 1$ i.e. the prior of $1/\Delta_{i,j} \sim \text{Gamma}(a_0, d_0)$ assuming the mean (a_0/d_0) and the variance (a_0/d_0^2) are the same

```
In [3]: nlminb(start = c(1,1), objective = LL,
                  x = D_c, d0 = 1, lower = c(0,0),upper = c(Inf,Inf))
```

$par 1. 5.49546155733852 2. 127118.822540337

$objective -186675.51260376

$convergence 1

$iterations 43

$evaluations **function** 61 **gradient** 108

$message 'false convergence (8)'

We found $a_0 = 5.49$ and $a_1 = 127118$ objective at -186675 and false convergence message from nlminb

If we give a bigger threshold, let the relative tolerance be 1e-3, the optimizing result is

```
In [4]: ctrl = list()
        ctrl$rel.tol = 1e-3
        nlminb(start = c(1,1), objective = LL,
               x = D_c, d0 = 1, lower = c(0,0),upper = c(Inf,Inf), control = ctrl)
```

$par 1. 5.48370042144545 2. 115.708722010479

$objective -185857.605247714

$convergence 0

$iterations 16

$evaluations **function** 18 **gradient** 34

$message 'relative convergence (4)'

We get $a_0 = 5.48$, $a_1 = 115$ and objective at -185857, which is not big different from the one we get with default threshold of relative tolerance

Previously, I used the default threshold and typically get large estimations of a_1, so I put upper bound for a_1 which is not correct. By setting a bigger threshold would yield

We know the randomness of $d_{i,j}$ under the bayesian framework coming from two parts, one is the prior of the true distance $\Delta_{i,j}$, the other is the variation of given $d_{i,j}|\Delta_{i,j}$. Intuitively, it is hard to estimate the variations corresponding to those two parts by only optimizing the marginal density of $d_{i,j}$ So we fix d_0 and only estimating two paramters (a_0, a_1)