4-13

XU, Xin

目录


```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(Rcpp)
```

1 Data Pre-processing

```
Ratings <- read.csv("archive/Ratings.csv")
Ratings_by_users <- group_by(Ratings, User.ID)
user.Rating <- Ratings_by_users %>% summarise(
  num = length(Book.Rating),
  avg.rating = mean(Book.Rating)
)
num.Rating <- group_by(user.Rating, num) %>% summarise(
  users = length(avg.rating)
)
```

1.1 ISBN

```
N <- max(Ratings$User.ID)
book_idx <- unique(Ratings$ISBN)
M <- length(book_idx)
book_idx <- 1:M
names(book_idx) <- unique(Ratings$ISBN)
train <- read.csv("archive/train.csv")
train$ISBN <- book_idx[train$ISBN]
test <- read.csv("archive/test.csv")
```

```
test$ISBN <- book_idx[test$ISBN]
```

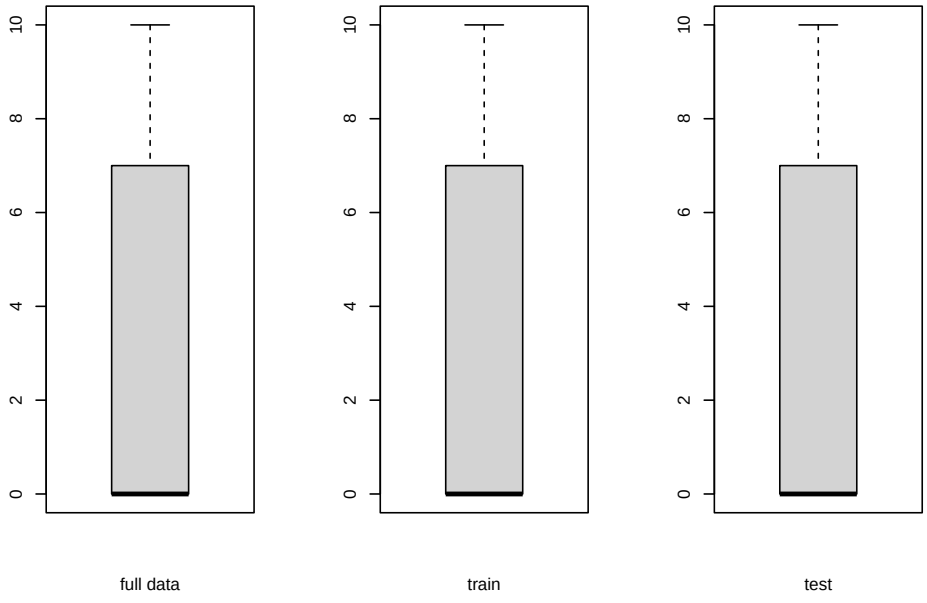
1.2 Romve Some Entries

```
train.book.avg <- group_by(train, ISBN) %>% summarise(
  avg.rating = mean(Book.Rating)
)
test <- test[which(test$ISBN %in% train.book.avg$ISBN),]
```

```
## There are 278854 users and 340556 books.
```

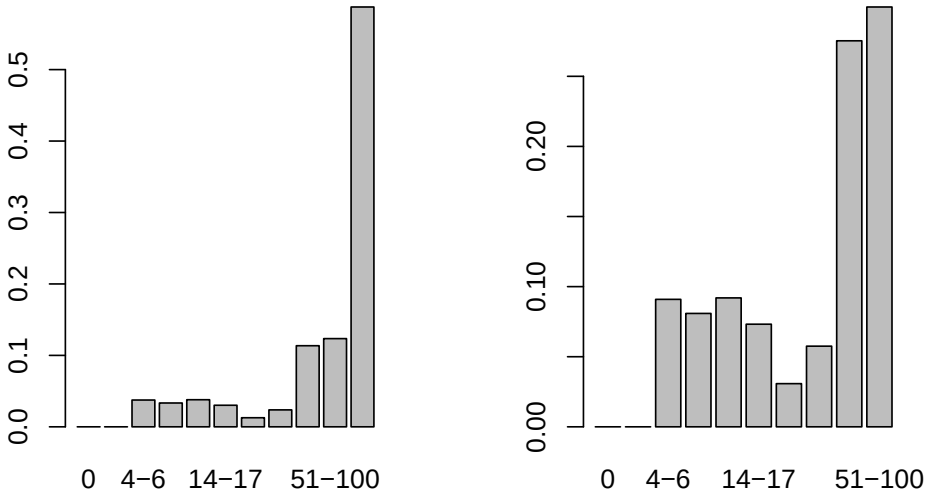
```
## There are 847349 examples in the training set and 240117 examples in the test set.
```

```
par(mfrow = c(1, 3))
boxplot(Ratings$Book.Rating, xlab = "full data")
boxplot(train$Book.Rating, xlab = "train")
boxplot(test$Book.Rating, xlab = "test")
```



1.3 User Groups

```
par(mfrow = c(1, 2))
barplot(grp_num/sum(grp_num))
barplot(grp_num[-11]/sum(grp_num[-11]))
```



2 book average

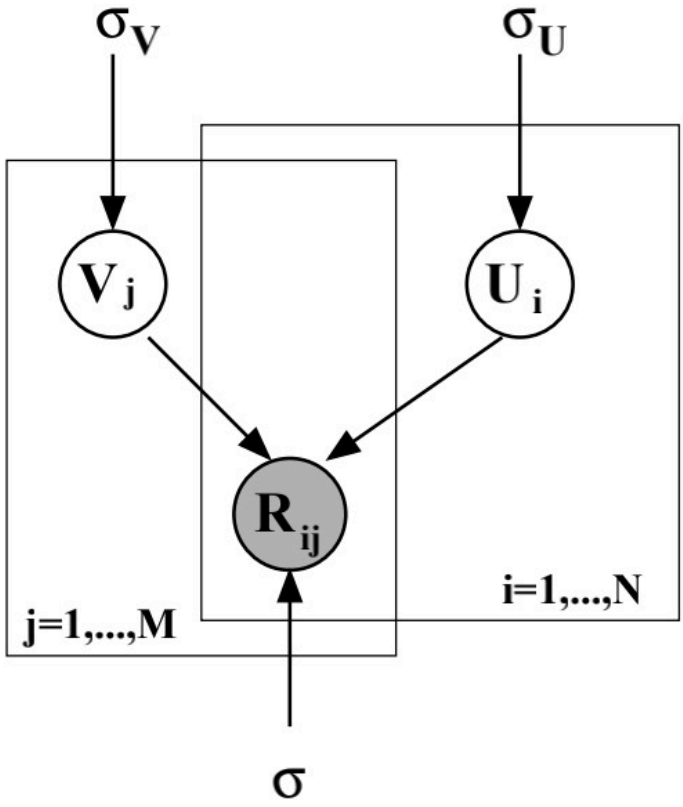
```
train$Book.Rating <- train$Book.Rating/10
test$Book.Rating <- test$Book.Rating/10
train.book.avg <- group_by(train, ISBN) %>% summarise(
  avg.rating = mean(Book.Rating)
)
book_avg <- train.book.avg$avg.rating
names(book_avg) <- train.book.avg$ISBN
book_idx1 <- names(book_avg)
Rcpp::sourceCpp("cpp/pred_bookavg.cpp")
bookavg_rslt <- pred_bookavg(test, book_avg, book_idx1)
save(bookavg_rslt, file = "bookavg_rslt.Rda")
```

```
## MAE: 0.3266  RMSE:  0.4104
```

```
head(cbind(test[, -1], bookavg_rslt$prediction), 10)
```

```
##    User.ID ISBN Book.Rating bookavg_rslt$prediction
## 1        8 8974           0                     0.4
## 2        8 8975           0                     0.2
## 3        8 8979           0                     0.0
## 4        8 8982           5                     0.0
## 5        8 8984           0                     0.5
## 6       14  453           0                     0.1
## 7       17 8994           0                     0.3
## 8       17 8998           0                     0.3
## 11      53 9017           3                     0.5
## 13      67 9028           0                     0.1
```

3 PMF



$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|U_i^T V_j,\sigma^2)\right]^{I_{ij}},$$

$$p(U|\sigma_U^2) = \prod_{i=1}^{N}\mathcal{N}(U_i|0,\sigma_U^2\mathbf{I}), \qquad p(V|\sigma_V^2) = \prod_{j=1}^{M}\mathcal{N}(V_j|0,\sigma_V^2\mathbf{I}).$$

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \parallel U_i \parallel_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \parallel V_j \parallel_{Fro}^2,$$

3.1 Implementation

```
// [[Rcpp::export]]
List grad_pmf(DataFrame df,
              NumericMatrix Ut,
              NumericMatrix Vt,
              double u_lam,
              double v_lam) {
  NumericVector user = as<NumericVector>(df["User.ID"]);
  NumericVector book = as<NumericVector>(df["ISBN"]);
  NumericVector rate = as<NumericVector>(df["Book.Rating"]);
  NumericVector u, v, du, dv;
  NumericMatrix grad_Ut =  clone(Ut), grad_Vt =  clone(Vt);
  grad_Ut = u_lam * grad_Ut;
  grad_Vt = v_lam * grad_Vt;
  int i, j, r;
  double aux;
  for(int k = 0; k < df.rows(); k++){
    i = user[k];
    j = book[k];
    r = rate[k];
    u = Ut.row(i-1);
    v = Vt.row(j-1);
    //du = grad_Ut.row(i-1);
    //dv = grad_Vt.row(j-1);
    aux = 0;
    for(int l = 0; l < u.size(); l++){
      aux += u[l]*v[l];
    }
    aux -= r;
    grad_Ut.row(i-1) = aux*v + grad_Ut.row(i-1);
    grad_Vt.row(j-1) = aux*u + grad_Vt.row(j-1);
  }
  return List::create(
    Named("Ut") = grad_Ut,
    Named("Vt") = grad_Vt
  );
}
```

```
double loss_pmf(DataFrame df,
                NumericMatrix Ut,
                NumericMatrix Vt,
                double u_lam,
                double v_lam) {
  NumericVector user = as<NumericVector>(df["User.ID"]);
  NumericVector book = as<NumericVector>(df["ISBN"]);
  NumericVector rate = as<NumericVector>(df["Book.Rating"]);
  NumericVector u, v;
  int i, j, r;
  double sum = 0, sum_u = 0, sum_v = 0;
  double aux;
  for(int k = 0; k < df.rows(); k++){
    i = user[k];
    j = book[k];
    r = rate[k];
    u = Ut.row(i-1);
    v = Vt.row(j-1);
    aux = 0;
    for(int l = 0; l < u.size(); l++){
      aux += u[l]*v[l];
    }
    sum += (r-aux)*(r-aux);
    //std::printf("%f\n", sum);
  }
  for(int k = 0; k < Ut.size(); k++){
    sum_u += Ut[k]*Ut[k];
    sum_v += Vt[k]*Vt[k];
  }
  sum_u *= u_lam;
  sum_v *= v_lam;
  sum += (sum_u + sum_v);
  sum /= 2;
  return sum;
}
```

```
List pred_pmf(DataFrame df,
              NumericMatrix Ut,
              NumericMatrix Vt) {
  NumericVector user = as<NumericVector>(df["User.ID"]);
  NumericVector book = as<NumericVector>(df["ISBN"]);
  NumericVector rate = as<NumericVector>(df["Book.Rating"]);
  NumericVector u, v;
  IntegerVector pred(rate.size());
  int i, j, r;
  double sum = 0, sum1 = 0;
  double aux;
  for(int k = 0; k < df.rows(); k++){
    i = user[k];
    j = book[k];
    r = rate[k];
    u = Ut.row(i-1);
    v = Vt.row(j-1);
    aux = 0;
    for(int l = 0; l < u.size(); l++){
      aux += u[l]*v[l];
    }
    pred[k] = floor(aux);
    if(pred[k] > 10){
      pred[k] = 10;
    }
    if(pred[k] < 0){
      pred[k] = 0;
    }
    sum += (r-aux)*(r-aux);
    if(r-aux > 0){
      sum1 += (r-aux);
    }else{
      sum1 += (aux-r);
    }
  }
  return List::create(
    Named("prediction") = pred,
```

3.2 Simulation

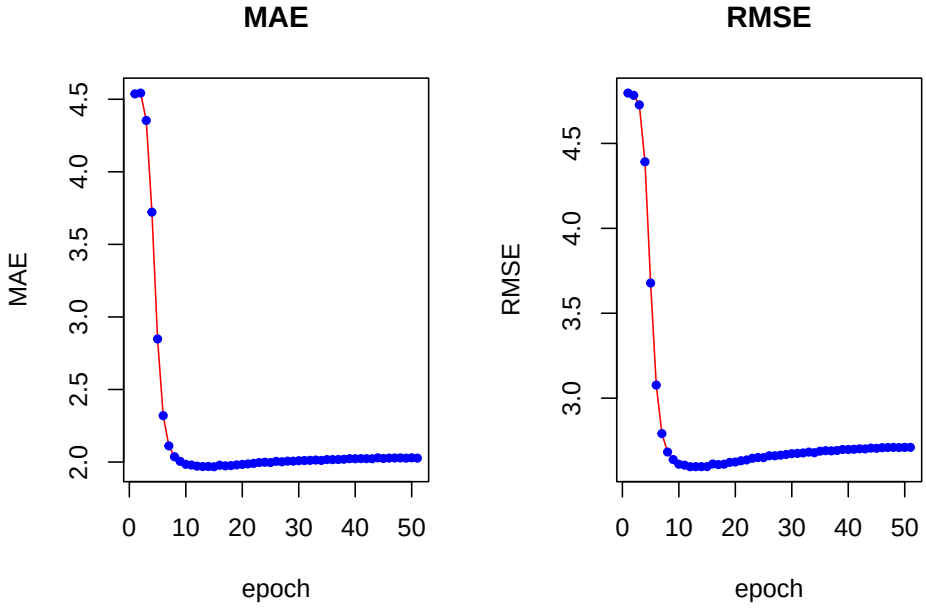
```
set.seed(123)
A <- matrix(rnorm(1e4*4, sd = sqrt(1)), ncol = 1e4)
B <- matrix(rnorm(1e2*4, sd = sqrt(1)), ncol = 1e2)
E <- matrix(rnorm(1e4*1e2, sd = sqrt(1e-2)), ncol = 1e2)
```

```
aux <- t(A) %*% B
aux <- 1/(1+exp(-aux))
obs <- aux + E
obs <- ifelse(obs < 0, 0, obs)
obs <- ifelse(obs > 1, 1, obs)
obs <- floor(10*obs)/10
obs <- 10*obs
W <- matrix(sample(0:2, 1e4*1e2, prob = c(0.9, 0.07, 0.03), replace = TRUE), ncol = 1e2
strain <- data.frame(User.ID = c(), ISBN = c(), Book.Rating = c())
stest <- data.frame(User.ID = c(), ISBN = c(), Book.Rating = c())
for(i in 1:1e4){
  for(j in 1:1e2){
    if(W[i, j] == 1){#train
      strain <- rbind(strain, c(i, j, obs[i, j]))
    }
    if(W[i, j] == 2){#test
      stest <- rbind(stest, c(i, j, obs[i, j]))
    }
  }
}
colnames(strain) <- c("User.ID", "ISBN", "Book.Rating")
colnames(stest) <- c("User.ID", "ISBN", "Book.Rating")
strain.book.avg <- group_by(strain, ISBN) %>% summarise(
  avg.rating = mean(Book.Rating)
)
stest <- stest[which(stest$ISBN %in% strain.book.avg$ISBN),]
write.csv(strain, file = "archive/strain.csv")
write.csv(stest, file = "archive/stest.csv")
```

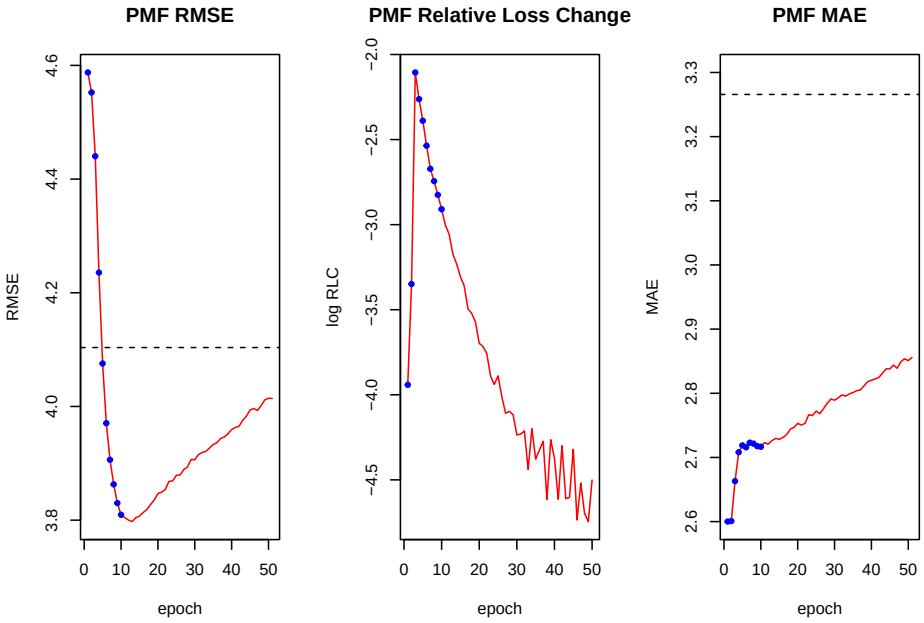
```
strain <- read.csv("archive/strain.csv")
stest <- read.csv("archive/stest.csv")
strain$Book.Rating <- strain$Book.Rating/10
stest$Book.Rating <- stest$Book.Rating/10
```

```
strain.book.avg <- group_by(strain, ISBN) %>% summarise(
  avg.rating = mean(Book.Rating)
)
sbook_avg <- strain.book.avg$avg.rating
names(sbook_avg) <- strain.book.avg$ISBN
sbook_idx1 <- names(sbook_avg)
Rcpp::sourceCpp("cpp/pred_bookavg.cpp")
sbookavg_rslt <- pred_bookavg(stest, sbook_avg, sbook_idx1)
```

```
## MAE: 0.2638  RMSE:  0.3098
```



3.3 Real data set



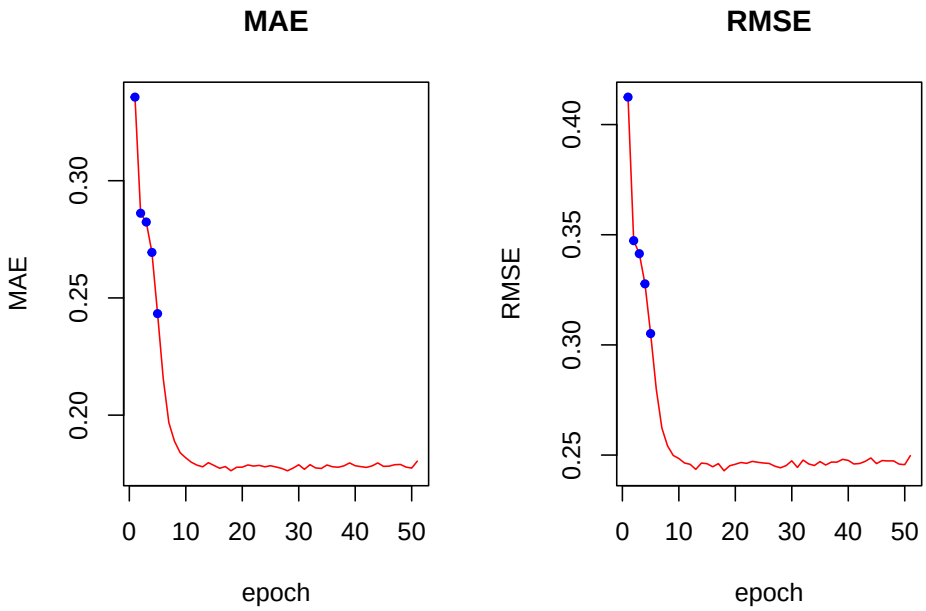

```
##
## PMF  MAE: 2.726  RMSE: 3.7974

##
## Book Average      MAE: 3.2657      RMSE: 4.1037
```

4 Logistic PMF

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|g(U_i^T V_j),\sigma^2)\right]^{I_{ij}}.$$

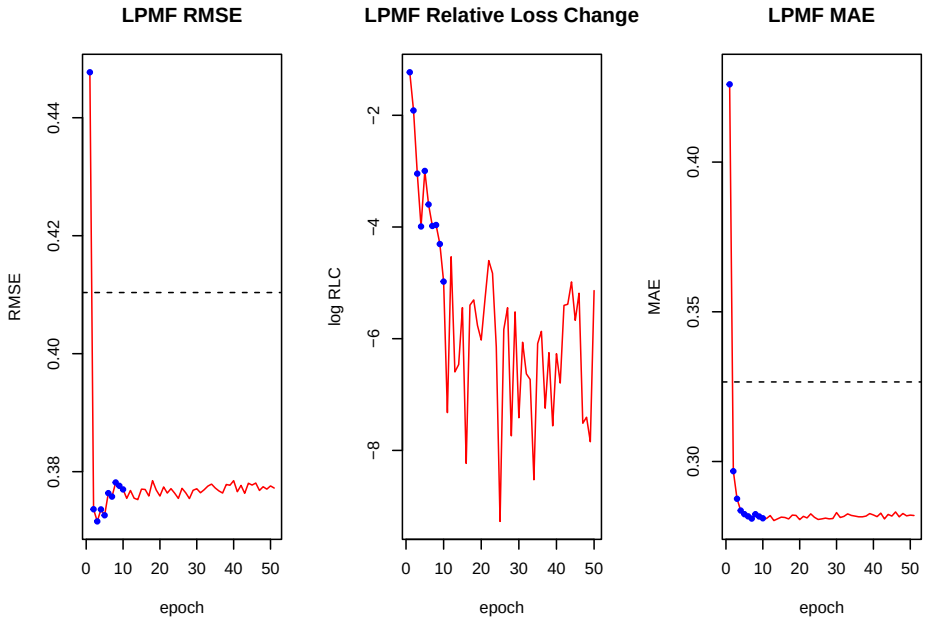
We map the ratings $1,...,K$ to the interval $[0,1]$ using the function $t(x) = (x-1)/(K-1)$, so that the range of valid rating values matches the range of predictions our model makes. Minimizing

4.1 Simulation



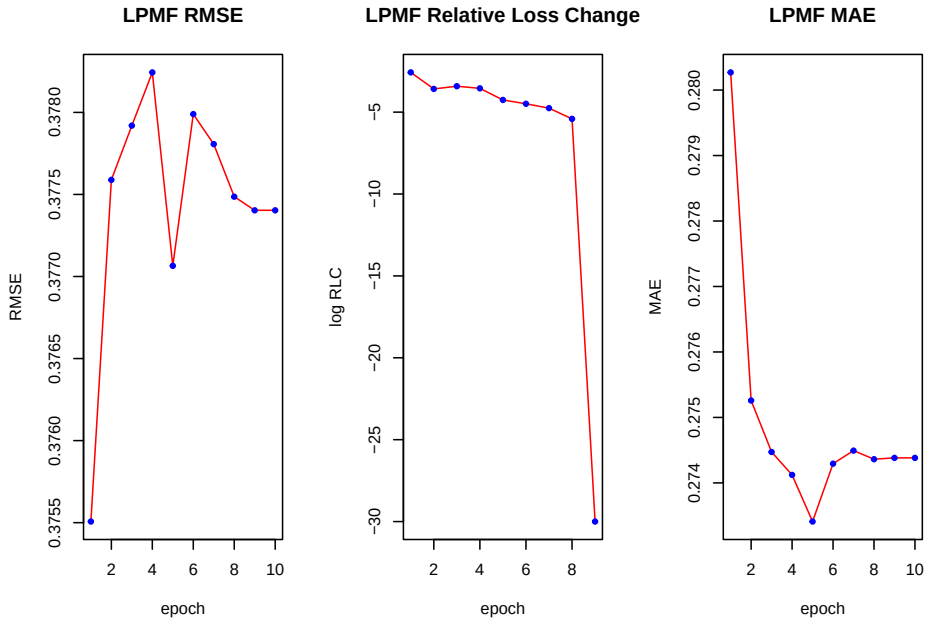
4.2 Real data set

4.2.1 Momentum



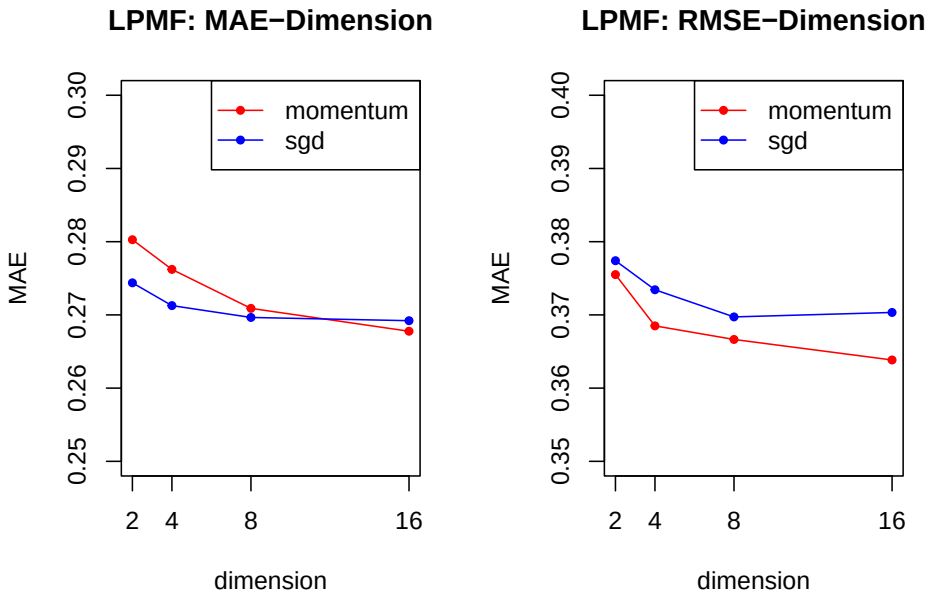

```
##
## LPMF Momentum    MAE: 0.2803     RMSE: 0.3755

##
## Book Average     MAE: 0.3266     RMSE: 0.4104
```

4.2.2 SGD

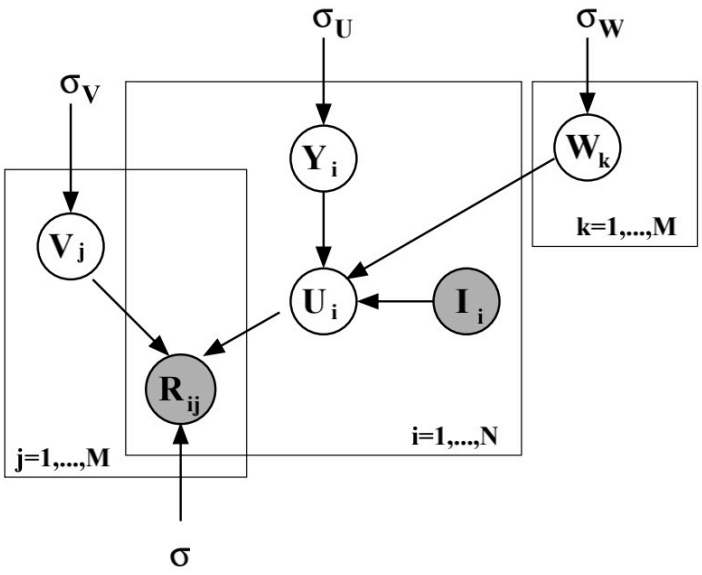



```
##
## LPMF SGD     MAE: 0.2744     RMSE: 0.3774

##
## Book Average     MAE: 0.3266     RMSE: 0.4104
```

4.2.3 Dimension of the Feature Vectors



5 Constrained PMF



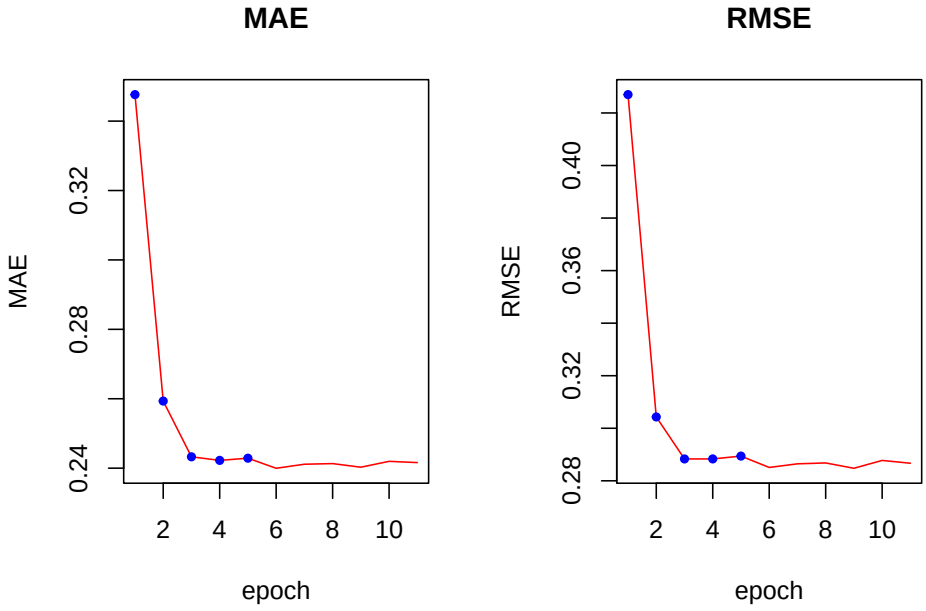
$$p(W|\sigma_W) = \prod_{k=1}^{M} \mathcal{N}(W_k|0, \sigma_W^2 \mathbf{I}).$$

$$U_i = Y_i + \frac{\sum_{k=1}^{M} I_{ik} W_k}{\sum_{k=1}^{M} I_{ik}}.$$

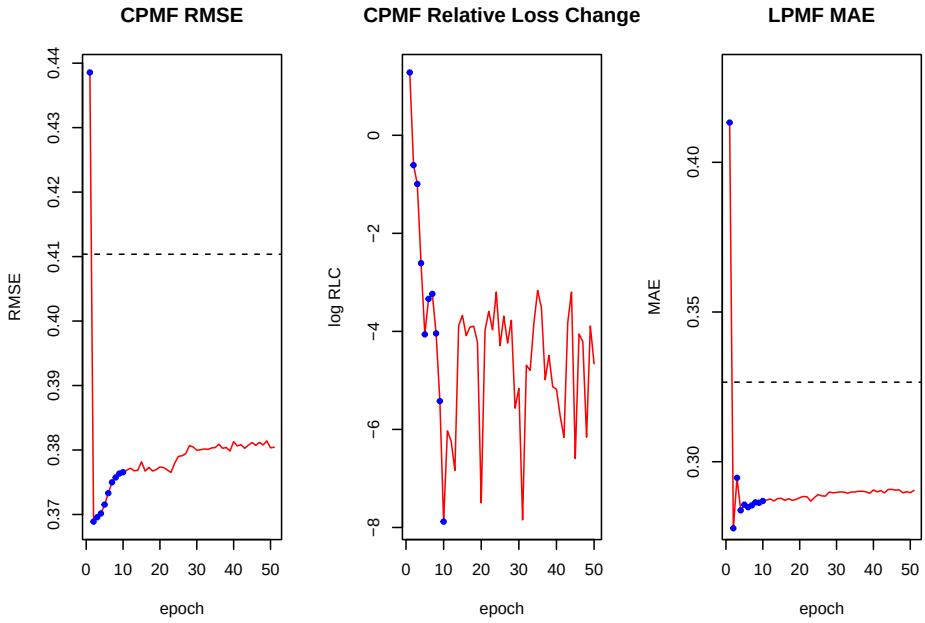
$$p(R|Y, V, W, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij}|g([Y_i + \frac{\sum_{k=1}^{M} I_{ik} W_k}{\sum_{k=1}^{M} I_{ik}}]^T V_j), \sigma^2) \right]^{I_{ij}}.$$

$$
\begin{aligned}
E &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\big(R_{ij} - g\big(\big[Y_i + \frac{\sum_{k=1}^{M} I_{ik}W_k}{\sum_{k=1}^{M} I_{ik}}\big]^T V_j\big)\big)^2 \\
&\quad + \frac{\lambda_Y}{2}\sum_{i=1}^{N} \parallel Y_i \parallel_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \parallel V_j \parallel_{Fro}^2 + \frac{\lambda_W}{2}\sum_{k=1}^{M} \parallel W_k \parallel_{Fro}^2,
\end{aligned}
$$

5.1 Simulation



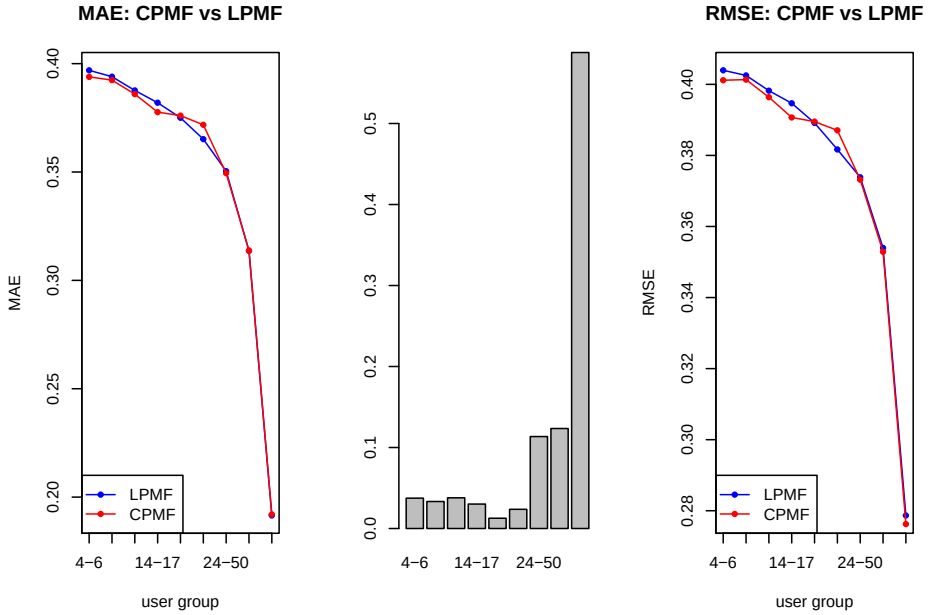
5.2 Real Data




```
## CPMF     MAE: 0.2778     RMSE: 0.3689

##
## LPMF     MAE: 0.2762     RMSE: 0.3685

##
## Book Average     MAE: 0.3266     RMSE: 0.4104

##
## > dim = 2
##
## > Y <- matrix(rnorm(N * dim), nrow = dim)
##
## > W <- matrix(rnorm(M * dim), nrow = dim)
##
## > aaa <- user_cpmf(read, t(Y), t(W))

##
## > dim = 2
```

```
##
## > Y <- matrix(rnorm(N * dim), nrow = dim)
##
## > V <- matrix(rnorm(M * dim), nrow = dim)
##
## > W <- matrix(rnorm(M * dim), nrow = dim)
##
## > Ut <- user_cpmf(read, t(Y), t(W))
##
## > pred_cpmf(test, Ut, t(V))$mae
## [1] 2.744441
```

```
##
## > dim = 2
##
## > U <- matrix(rnorm(N * dim), nrow = dim)
##
## > V <- matrix(rnorm(M * dim), nrow = dim)
##
## > pred_lpmf(test, t(U), t(V))$mae
## [1] 2.745198
```


MAE: CPMF vs LPMF
MAE
0.20
0.25
0.30
0.35
0.40
LPMF
CPMF
4−6
14−17
24−50
user group
0.0
0.1
0.2
0.3
0.4
0.5
4−6
14−17
24−50
RMSE: CPMF vs LPMF
RMSE
0.28
0.30
0.32
0.34
0.36
0.38
0.40
LPMF
CPMF
4−6
14−17
24−50
user group