STA 250 Final Part2

Chen Zihao 915490404

Problem 2. Extreme classification

In the multi-label classification problem, given the data matrix $X \in \mathbb{R}^{nxd}$ (each row is an input data point) and label matrix $Y \in \mathbb{R}^{nxL}$. L is number of labels, and each row of Y is an L-dimensional 0/1 vector indicating the labels for a data point. We want to predict the label for a given new input data point. In "extreme" multi-label classification, number of labels can be extremely large (e.g. 10,000, or 1 million). Let's develop an algorithm for soving this problem.

We will test our algorithms using the dataset from

http://manikvarma.org/downloads/XC/XMLRepository.html.

We solve the following optimization problem to get the model W,H:

$$\min_{W \in \mathbb{R}^{dxk}, H \in \mathbb{R}^{Lxk}} \frac{1}{2}||Y - XWH^T||_F^2 + \lambda||W||_F^2 + \lambda||H||_F^2$$

Forthis problem we set k=50. After solving the optimization problem, for each testing data $x \in \mathbb{R}^d$, we predict the label $\tilde{y} \in \mathbb{R}^L$ by

$$\tilde{y} = x^T W H^T$$

This is supposed to be close to the true label vector y since we minimize the square loss $||XWH^T - Y||_F^2$ in the objective function.

We will evaluate the results using precision@1 and precision@5, where precision@k is

(number of ture labels in the top-k predictions)/k.

or formally,

$$P@k := \frac{1}{k} \sum_{i \in rank_k(\tilde{y})} y_l,$$

where $rank_k(\tilde{y})$ returns the k largest indices of the predictive label vector $\tilde{y}$, and $y_l = 1$ if it is a correct label.

There will be multiple testing samples, so the P@1,P@5 will be the average among those samples.

1.(10pt) Download the bibtex data. Transform the training data (only samples with training indices) into data matrix X and label matrix Y. Note that in "Bibtex_trSplit.txt" and "Bibtex_tstSplit.txt" there are 10 splits of training and testing data. We will only use the first split (first column in both files) to conduct the experiments.

I have try every mothod i know in R to read the data. But i failed. So i change my strategy.

With the help of windows notepad I replace all the space in the txt file with ";" and then delete all the ":1.000000". Read it in Excel and found it works well, then I save it as the new dataset.

In this method, I got a X and a Y excel.

It is not a elegent way, but i at least work.

```
X1=read.csv("C:\\X.csv",header = FALSE)
Y1=read.csv("C:\\Y.csv",header = FALSE)
bibtex_trSplit=read.table("C:\\bibtex_trSplit.txt")
bibtex_tstSplit=read.table("C:\\bibtex_tstSplit.txt")
```

```
X=matrix(0,nrow(X1),1836)

for (i in 1:nrow(X1)){
  for (j in X1[i,]){
      X[i,j]=1
  }
}

Y=matrix(0,nrow(Y1),159)
for (i in 1:nrow(Y1)){
  for (j in Y1[i,]){
      Y[i,j]=1
  }
}
```

Choose the training data and test data according to the splits.

```
Xtr=X[bibtex_trSplit[,1],]
Ytr=Y[bibtex_trSplit[,1],]

Xte=X[bibtex_tstSplit[,1],]
Yte=Y[bibtex_tstSplit[,1],]
```

2.(25 pt) Develop an algorithm for solving this problem. Describe your algorithm. What's the time complexity?

Use the coodinate descend to solve this problem.

first fix W to update H, and then Fix H to update W

$$F(W,H)=\frac{1}{2}||Y-XWH^T||_F^2+\lambda||W||_F^2+\lambda||H||_F^2$$

```
F=function(W,H){
  norm(Ytr-Xtr%*%W%*%t(H),type = "F")^2/2+lambda*norm(W,type = "F")^2+lambda*norm(H,type = "F")^2
}
```

first, fix W we get

$$\min_{H\in\mathbb{R}^{Lxk}}\frac{1}{2}||Y-XWH^T||_F^2+\lambda||H||_F^2$$

$$\nabla_H F(W,H)=(XWH^T-Y)^TXW+2\lambda H$$

```
FH=function(W,H){
  A=Xtr%*%W
  t(A%*%t(H)-Ytr)%*%A+2*lambda*H
}
```

similarly, we get

$$\nabla_W F(W,H)=X^T(XWH^T-Y)H+2\lambda W$$

```
FW=function(W,H){
  t(Xtr)%*%(Xtr%*%W%*%t(H)-Ytr)%*%H+2*lambda*W
}
```

For i=1,2,. . . ,t

$G = \nabla_H F(W^k, H^k)$

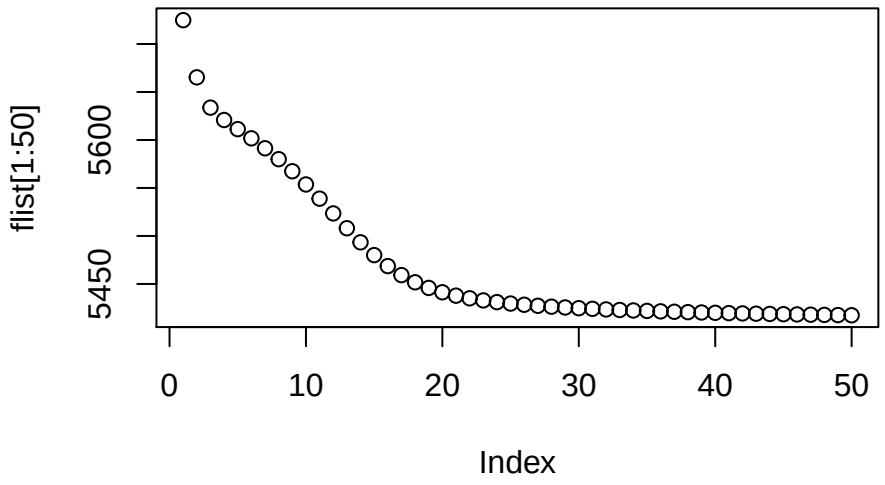
$H^{k+1} = H^k - \eta G$

$J = \nabla_W F(W^k, H^{k+1})$

$W^{k+1} = W^k - \eta J$

end

```
L=ncol(Ytr)
d=ncol(Xtr)
lambda=1/4
eta=10^(-4)
t=50
k=50
H=matrix(0.001,L,k)
W=matrix(0.001,d,k)
flist=numeric(t)
for (i in 1:t){
  H=H-eta*FH(W,H)
  W=W-eta*FW(W,H)
  flist[i]=F(W,H)
}
plot(flist[1:50])
```



the complexity: $o(dLn^2)$

3.Apply your algorithm to the bibtex data. Test your algorithm and compute P@1, P@5 of your algorithm. Test different λ values and compare the results. What's your finding? Also, you can compare your results with the state-of-the-art performance on the website http://manikvarma.org/downloads/XC/ XMLRepository.html (Table 6).

After training, time for the testing data.

```
y=Xte%*%W%*%t(H)
```

y is the $\tilde{y}$

This is the results of $\lambda = 1/4$

precision @1

```
a1=0
for (i in 1:nrow(y)){
  a1=a1+as.numeric((which(y[i,]==max(y[i,]),arr.ind = TRUE)%in%which(Yte[i,]==1,arr.ind = TRUE)))
}
a1
```

```
## [1] 382
```

P@5

```
a5=0
for (i in 1:nrow(y)){
  a5=a5+sum(as.numeric(order(y[i,],decreasing = TRUE))[1:5]%in%which(Yte[i,]==1,arr.ind = TRUE))
}
a5=a5/5
a5
```

```
## [1] 125.4
```

For $\lambda = 1/2$

```
L=ncol(Ytr)
d=ncol(Xtr)
lambda=1/2
eta=10^(-4)
t=50
k=50
H=matrix(0.001,L,k)
W=matrix(0.001,d,k)

for (i in 1:t){
  H=H-eta*FH(W,H)
  W=W-eta*FW(W,H)

}
y=Xte%*%W%*%t(H)
```

precision @1

```
a1=0
for (i in 1:nrow(y)){
  a1=a1+as.numeric((which(y[i,]==max(y[i,]),arr.ind = TRUE)%in%which(Yte[i,]==1,arr.ind = TRUE)))
}
a1
```

```
## [1] 382
```

P@5

```
a5=0
for (i in 1:nrow(y)){
  a5=a5+sum(as.numeric(order(y[i,],decreasing = TRUE))[1:5]%in%which(Yte[i,]==1,arr.ind = TRUE))
}
a5=a5/5
a5
```

```
## [1] 125.4
```

For $\lambda = 1/8$

```
L=ncol(Ytr)
d=ncol(Xtr)
lambda=1/2
eta=10^(-4)
t=50
k=50
H=matrix(0.001,L,k)
W=matrix(0.001,d,k)
for (i in 1:t){
  H=H-eta*FH(W,H)
  W=W-eta*FW(W,H)
}
y=Xte%*%W%*%t(H)
```

precision @1

```
a1=0
for (i in 1:nrow(y)){
  a1=a1+as.numeric((which(y[i,]==max(y[i,]),arr.ind = TRUE)%in%which(Yte[i,]==1,arr.ind = TRUE)))
}
a1
```

```
## [1] 382
```

P@5

```
a5=0
for (i in 1:nrow(y)){
  a5=a5+sum(as.numeric(order(y[i,],decreasing = TRUE))[1:5]%in%which(Yte[i,]==1,arr.ind = TRUE))
}
a5=a5/5
a5
```

```
## [1] 125.4
```

the difference is small with different λ

As i got a P@1=361,which is far more bigger than the number in table 6. If i calculate the P@1 right, my algorithm is working well.

If the 60 in Table 6 is the percentage. then i just got around 14% accurate, which is bad.

4.(Bonus,20pt) Try to scale your algorithm to a larger dataset AmazonCat-13K, Report your findings. If you can't scale to that data, you can still get partial credit by explaining why you

are not able to do that (which operations in your algorithm are too slow for larger datasets), and what's the potential way to speed up your code.

first of all, i did not find a good way to read the data. I manage the data by notepad and excel which was hard for a large scale data as excel is not good tool to manage big data.

Second, i use a fixed step size instead of the line search strategy. The step size η is tunned by myself. When doing a time comsuming data analysis, it is hard for me to get a good η

Third, I calculate the gradient in an exact way. It means I need to duel with the whole matrix every time. If i can change it to a inexact method, it will save more time.

What's more, I did not use any parallel tool in R. Such as the package "parallel", it will worsen the speed of R ,not to mention that R is known as a slow tool.