TP5 Système de recommandation de films

PENG Hanyuan
YAN Wenli

1. Objectif

Faire des recommandations automatiques de films utilisées dans les systèmes multimédia comme Netflix et Youtube. Par des notes données par des utilisateurs avec la mesure d'erreur RMSE.

2. Mise en œuvre

Selon le cahier des charges de ce TP, nous avons réussi d'établi cinq prédicteurs. On a analysé des performances avec des donnés "ml-100k.zip", et on a fait une comparaison à la fin.

2.1. **Prédicteur aléatoire**

Dans ce cas là, on a créé des donnés (notes) par cette méthode:

```
def aler():
    scorealea=np.random.random()*5
```

j'ai utilisé:

```
def creatPredictionsArray():
    res = np.random.randint(1,5,size=[nbuser,nbfilm])
```

```
Matrice de notes aléatoires: [[3 4 2 ... 1 3 4]
 [2 1 3 ... 4 3 1]
 [3 4 3 ... 3 3 4]
 ...
 [2 4 2 ... 2 4 4]
 [2 2 1 ... 4 2 3]
 [2 3 2 ... 2 2 3]]
```

Et puis, on a fait le RMSE: 1.89

2.2. **Prédicteur basique**

Dans le modèle basique, on suppose que chaque utilisateur u note tous les films qu'il aregardé avec un écart à la moyenne bu qui lui est propre. Aussi, chaque film a un écart àla moyenne qui lui est propre bi. On a créé le prédicteur basique avec des formules donnés:

$$\hat{r}_{u,i} = \bar{r} + b_u + b_i \qquad b_u = \frac{\sum_i r_{u,i}}{V_u} - \bar{r} \qquad b_i = \frac{\sum_u ru,i}{V_i} - \bar{r}$$

Quand on calcul la moyenne des notes, on a ignoré des valeurs qui sont "-1" par numpy.ma.mask().

```
res = readdata('./ml-100k/u.data',943,1682)
res = np.ma.masked_values(res, -1)
```

```
def predi_basique(data, uid, fid , r_):
    # print(data[uid,:])
    bu = data[uid,:].mean() - r_
    bi = data[:,fid].mean() - r_
    res = r_ + bu + bi
    return res
```

Et puis, on a fait le RMSE: 0.9383

Nous avons stocké la matrice de résultat de prédicteur basique dans un fichier ***prediction_basic.npy.***

2.3. **Prédicteur avec méthode voisinage**

Avec cette méthode, on veut prédire la note que l'utilisateur va mettre pour un film en cherchant à établir une notion de similarité entre utilisateurs.

D'abord, on a crée matrice R: $\tilde{r}_{u,i} = r_{u,i} - (\bar{r} + b_u + b_i)$ et $\tilde{R} = (\tilde{r}_{u,i})_{u,i})$

```
R: [[ 1.26147538e+00  1.93907808e-01  1.19390781e+00 ... -3.60609219e+00
  -4.05609219e+00 -3.94894933e+00]
 [-1.28230507e-01 -4.19579807e+00 -4.19579807e+00 ...  4.20192588e-03
   5.54201926e-01 -3.38655217e-01]
 [-3.75323051e+00 -2.82079807e+00 -2.82079807e+00 ... -2.62079807e+00
  -3.07079807e+00 -2.96365522e+00]
 ...
 [-3.12823051e+00 -2.19579807e+00 -2.19579807e+00 ... -1.99579807e+00
  -2.44579807e+00 -2.33865522e+00]
 [-4.81004869e+00 -3.87761626e+00 -3.87761626e+00 ... -3.67761626e+00
  -4.12761626e+00 -4.02047340e+00]
 [ 9.96769493e-01 -4.07079807e+00 -7.07980741e-02 ... -3.87079807e+00
  -4.32079807e+00 -4.21365522e+00]]
```

Ensuite, on a calculé le matrice S (Similarié entre des films):

```
S: [[1.         0.72148694 0.73463192 ... 0.72373609 0.71166905 0.71864455]
 [0.72148694 1.         0.9172118  ... 0.91284743 0.93758173 0.90942422]
 [0.73463192 0.9172118  1.         ... 0.92750611 0.93501683 0.91156341]
 ...
 [0.72373609 0.91284743 0.92750611 ... 1.         0.95836139 0.94799367]
 [0.71166905 0.93758173 0.93501683 ... 0.95836139 1.         0.95377478]
 [0.71864455 0.90942422 0.91156341 ... 0.94799367 0.95377478 1.        ]]
```

Pour un film donnée, on a fait le sort() par l'ordre décroissant et choisi les L(ici L = 10) premiers voisins .

Et avec tout ça, on a obtenu le nouvel prédicteur:

RSME: 1,257

```
reducing the number of element by 6
res:
note 6854
res_length 157
res_width 280
mean: 3.8042019258826962
Prédicteur_voisinage_RMSE: 1.2572439098778425
```

2.4. **Prédicteur basique par régression linéaire: résolution analytique**

Pour cette méthode, la matrice A est de forme K * (M+N).

Nous avons fait cette méthode mais on trouve que la matrice A^TA n'est pas inversible (Singular Matrix), donc nous avons continué à faire l'algorithme suivant.

2.5. **Prédicteur basique par régression linéaire: résolution par la descente de gradient**

Nous avons utilisé 1764 votes pour entraîner notre algorithme. Nous n'avons pas normaliser les données donc nous avons utilisé un learning rate alpha petite:

alpha = 0.00000003

Nous avons choisi de faire 10000 fois d'itération pour l'entraînement et nous avons stocké le paramètre ***b*** dans un fichier ***b_value.npy*** avec un gradient proche de 0 (environ ±0.02 pour chaque dimension).

Nous avons utilisé 499 votes dans la base pour tester et on trouve que le RMSE: 0.8768

```
rmse for gradient descent
0.8768622160798086
```

Les codes sont dans le fichier ***gradient.py***

Vous pouvez lancer le fichier ***main.py*** pour vérifier le résultat de prédicteur basique et prédicteur avec la descente de gradient.