Paul FAYOUX
Valérian MENIN

Système multimédia

TP2

Dans ce TP, notre objectif principal était d'apprendre à utiliser des algorithmes de compression/décompression d'images

Nous avons programmé la compression et décompression d'une image avec une découpe par bloques.
Le programme prend en paramètre une image sur laquelle il va travailler.
Basé sur OpenCV, cv2 permet de faire de la manipulation d'images. Nous avons travaillé sur l'image "lena_std.tif" qui se trouvait sur les raspberry.

Nous avons créé une matrice permettant de représenter notre image.

Dans un premier temps, nous avons implémenté la compression par DCT (Discrete Cosine Transform) d'une matrice donnée puis la compression et décompression sur une image et enfin la compression et décompression rapide d'une image.

Paul FAYOUX
Valérian MENIN

Système multimédia

TP5

Dans ce TP, nous avions pour objectif de comprendre le fonctionnement de la recommandation de films de Netflix.

Dans un premier temps, il est important de lister un certain nombre de notation:

- RMSE (Root Mean Square Error) $e = \sqrt{\sum_{r_{u,i} connu} \frac{(r_{u,i} - \hat{r}_{u,i})^2}{C}}$ noté “e” dans notre code
- C: nombre de votes noté “nb_votes”
- r_ui_connu: note utilisateur u au Film i
- r^ui: note prédictive noté “votes_data”
- u.data: fichier de données: User id | Film id | score | timestamp
- r_barre: moyenne de toutes les notes
- N: nombre d’utilisateur noté “nbUser”
- M: nombre de films noté “nbMovie”
- Vu: nombre de votes utilisateur noté “bu[index utilisateur][1]”
- Vi: nombre de votes film noté “bi[index film][1]”
- bu: écart à la moyenne entre l’utilisateur u et sa moyenne personnelle noté “bu[u][0]”
- bi: écart à la moyenne entre le film i et la moyenne du film noté “bi[i][0]”

Partie 1

Voici le calcul de bu et bi

$$b_u = \frac{\sum_i r_{u,i}}{V_u} - \bar{r}$$

$$b_i = \frac{\sum_u ru, i}{V_i} - \bar{r}$$

```
# note moyenne du film d'index "movieIndex"
def avgMovieScore(votes_data, bi, movieIndex):
        bi[userIndex][0] = 0
        bi[userIndex][1] = 0
        for index_user in range(nbUser):
                if votes_data[index_user, movieIndex] != -1:
                bi[movieIndex][0] = bi[movieIndex][0] + votes_data[index_user][movieIndex]
                bi[movieIndex][1] = bi[movieIndex][1] + 1
```

Paul FAYOUX
Valérian MENIN

```
        bi[movieIndex][0] = bi[movieIndex][0]/bi[movieIndex][1] - r_barre
```

Le calcul de la matrice de prédiction est le suivant.

```
# calcul la matrice predicatrice
def predicateScores(votes_data):
        votes_predicator = numpy.zeros((nbUser, nbMovie))
        for index_user in range(0, nbUser):
                for index_movie in range(0, nbMovie):
                        if votes_data[index_user][index_movie] != -1:
                                votes_predicator[index_user][index_movie] = r_barre + bu[index_user][0] + bi[index_movie][0
        return votes_predicator
```

La méthode applique simplement la formule suivante $\hat{r}_{u,i} = \bar{r} + b_u + b_i$

Partie 2

On cherche dans une seconde partie à calculer les prédictions de ce qu'un utilisateur pourrait aimer. On va donc conseiller aux utilisateurs les L films les plus proches de ses goûts en prenant en utilisant la méthode du voisinage. Cette méthode va effectuer un rapprochement entre les films pour savoir si un utilisateur serait potentiellement intéressé par cet autre film.

$$s_{i,j} = \frac{\tilde{r}_i^t \tilde{r}_j}{\|\tilde{r}_i\|_2 \|\tilde{r}_j\|_2} = \frac{\sum_u \tilde{r}_{u,i} \tilde{r}_{u,j}}{\sqrt{(\sum_u r_{u,i}^2)(\sum_u r_{u,j}^2)}}$$