Internship 2019-2020

What metric spaces do machine learning methods learn?

at

INRIA Laboratory - Sophia Antipolis, France

Student:
Abdelkarim HAFID

Supervisors:
Giovanni Neglia
Tareq Si Salem

List of Figures

Chapter 1

Introduction

Recommendation systems are among the most popular applications in data science today. They are used to predict the "rating" that a user would give to an article. Today, tech companies have applied those recommendations systems to their daily work: Amazon uses recommendations systems to suggest products to customers, YouTube to decide which video to play on auto play, and Facebook to recommend products and pages to like and follow. In addition, for some companies like Netflix, the economic situation model and its success depend on and revolve around the power of their recommendations. One of the main methods used in the recommendations systems is the Collaborative filtering method that consist of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). There are two types of Collaborative filtering methods : The first one is user-based collaborative filtering and the second one is Item-based collaborative filtering. One key advantage of Item based collaborative filtering is the stability which is that the rating on a given item will not change significantly overtime. This Collaborative filtering method is based on the factorization of the matrix of users and items . In fact, this factorisation tells us how far a user is aligned with a set of features and how far a movie fits into this set of latent features. In this internship, we will first focus on understanding the fundamentals of one first ranked method of recommendations systems on Netflix, then we will run the code of this method to understand the meaning of the results and then to retrieve the users embedding of each user in the datasets of Netflix and MovieLens 20M . Moreover, we will try to understand the distribution of our users embedding in one high dimensional space and finally we will focus on similarity caching to understand if it might be used as an extension of a trained model.

Chapter 2

Recommendation systems on Netflix

2.1 An overview of the RecVAE method : a New Variational Autoencoder for Top-N Recommendations with Implicit Feedback

We have selected this method of RecVAE because it is the first ranked method of the recommendation systems on Netflix, the table 1 of the paper [1] shows the difference on scores regarding other methods of recommendation systems on Netflix. The RecVAE method is a model for collaborative filtering with implicit feedback, it means that the method considers the amount of data available for a given user in the processing stage. This method is based on the Mult-VAE approach, in the paper [1] on section 3.1 the Mult-VAE approach is well explained. The RecVAE method introduce important novelties regarding other approaches as you can see in the table 2 of the paper[1]. This novelties are first a new architecture, a composite prior, one technique of 'Beta rescaling', alternating training and finally a decoder denoising. The table 2 in the paper[1] shows how those new novelties influence the results of the training on the Movielens 20M and Netflix datasets. The models performance of the recommendation systems on Netflix are evaluated based on 2 indices: NDCG@k and Recall@k. Both metrics compares top-k predictions of a model with the test set X_u^t of user feedback for user u. To obtain recommendations for RecVAE and similar models, they sort the items in descending order of the likelihood predicted by the decoder and they exclude items from the training set, and denote the item at the n-th place in the resulting list as R_n^u. In this notation, evaluation metrics for a user u are defined as follows, The two evaluation metrics NDCG@k and Recall@k for a user u are defined as follows:

$$Recall@k(u) = \frac{1}{\min(M, |X_t^u|)} \sum_{n=1}^{k} 1[R_u^n \in X_t^u]$$

$$DCG@k(u) = \sum_{n=1}^{k} \frac{2^{1[R_u^n \in X_t^u]} - 1}{\log(n+1)} \text{ and } NDCG@k(u) = (\sum_{n=1}^{|X_t^u|} \frac{1}{log(n+1)})^{-1} \mathbf{DCG@k(u)}$$

NDCG@k(u) is defined as DCG@k(u) divided by its highest theoretically possible value. Recall@k accounts for all top-k items equally, while NDCG@k assigns larger weights to top ranked items. As a result, it is natural to choose a larger value of k for NDCG@k.

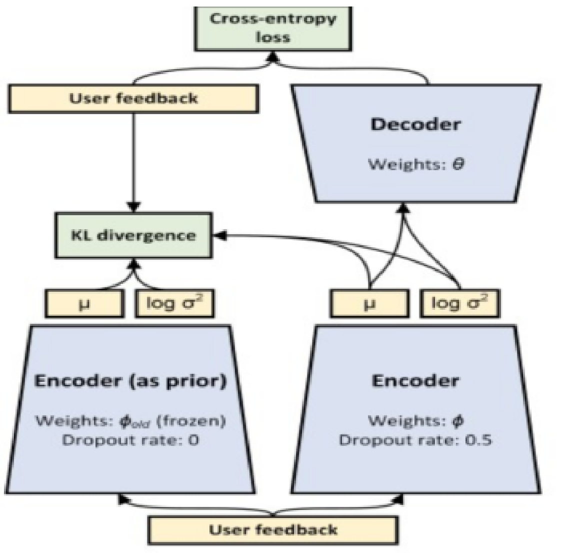


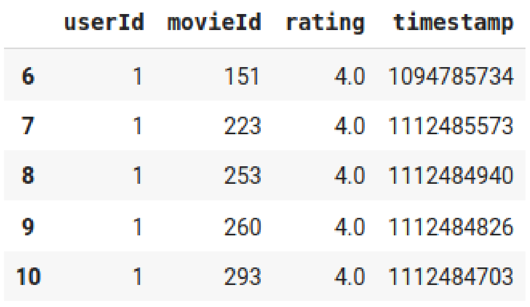
Figure 2.1: RecVAE architecture . Source:[1]

The figure 2.1 above explain the global idea of the RecVAE Method: in this figure we have two types of encoders:First, one encoder 'as prior' with dropout rate = 0 it means that we will not drop any neuron of the neural network on the training stage of the model because in this type of encoder they have already implemented another technique to avoid over-fitting. The second type of encoders is with dropout rate = 0,5 it means that we will drop some neurons in the training of the model to reduce the number of parameters and then to avoid over-fitting. In fact, each user after encoding his feedback will be represented as one random Gaussian variable of mean 'mu' and variance 'logvar': N (mu, logvar) in one high dimensional space, the dimension of this space will be known after running the code of this method. The RecVAE method use the 'mu' and 'logvar' as the users embeddings to train their model but only 'mu' in the testing and the evaluation of their model. As in the figure 1 the user-embedding calculated with the encoder of dropout-rate=0,5 will be passed to the decoder then we will calculate the cross-entropy loss to measure how far the results predicted are from the test user feedback. In an another hand they will use the two types of embeddings calculated by the two types of encoders in the high dimension to measure the KL divergence with the user feedback in the low dimension.

2.2 An overview on the results after running the RecVAE code on the MovieLens 20M and Netflix datasets

2.2.1 Running RecVAE code on the MovieLens 20M dataset

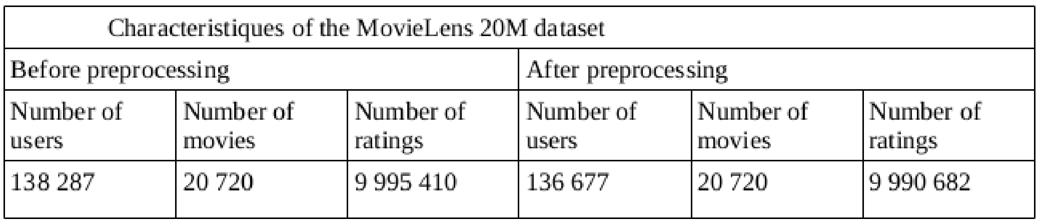
First we preprocessed the MovieLens 20M dataset by following the method of preprocessing used in the paper[2] then we runned the code of RecVAE method on this preprocessed dataset. The Movielens 20M dataset as a dataframe is shown in the following figure.



	userId	movieId	rating	timestamp
6	1	151	4.0	1094785734
7	1	223	4.0	1112485573
8	1	253	4.0	1112484940
9	1	260	4.0	1112484826
10	1	293	4.0	1112484703

Figure 2.2: The first rows in the Movielens 20M dataset

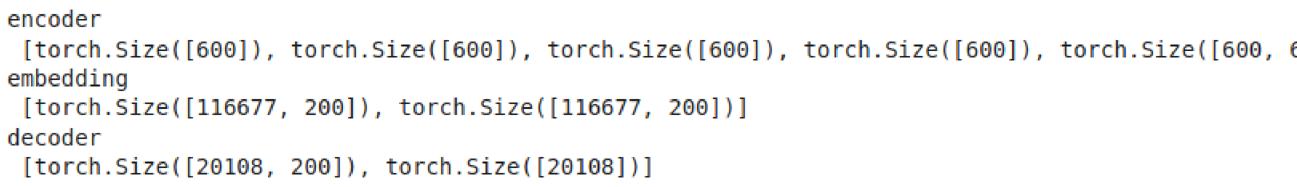
Preprocessing the MovieLens20M dataset means that we will only keep the triplets for items which were clicked on by at least 1 user. Moreover, we will only keep the triplets for users who clicked on at least 5 items and after doing this, some of the items will have less than 5 users, but should only be a small proportion. The following table summarizes the characteristics of the Movielens 20M dataset before and after preprocessing.



Characteristiques of the MovieLens 20M dataset					
Before preprocessing			After preprocessing		
Number of users	Number of movies	Number of ratings	Number of users	Number of movies	Number of ratings
138 287	20 720	9 995 410	136 677	20 720	9 990 682

Figure 2.3: Characteristics of the MovieLens20M dataset

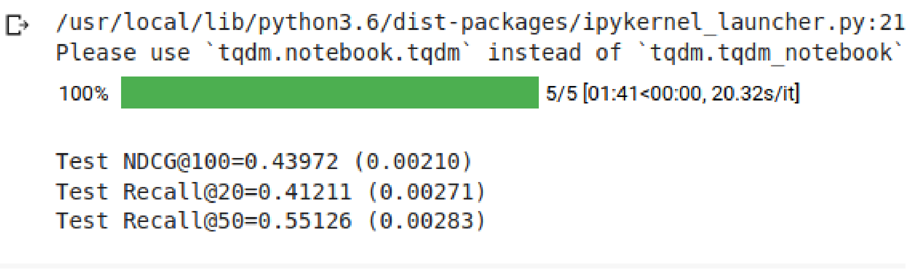
The first results that we have got after running the code of RecVAE on the preprocessed dataset of Movielens 20M are depicted in the following caption



```
encoder
 [torch.Size([600]), torch.Size([600]), torch.Size([600]), torch.Size([600]), torch.Size([600, 6
embedding
 [torch.Size([116677, 200]), torch.Size([116677, 200])]
decoder
 [torch.Size([20108, 200]), torch.Size([20108])]
```

Figure 2.4: size of MovieLens20M embeddings

The main information from this results is the dimension of our users embeddings: 200. The 116677 is the number of users on the training dataset, it means that each user in this training dataset will be represented in the high dimension space by one embedding of dimension 200. After completing the running of the code of RecVAE method we can conclude from NDCG@100 that in 100 ratings on the test user feedback 43,9 are well predicted, then from Recall@20 we can deduce that in 100 ratings 41,2 of the ratings are well predicted on the test user feedback and finally from Recall@50 we can conclude that in 100 ratings on the test user feedback 55,126 are well predicted.

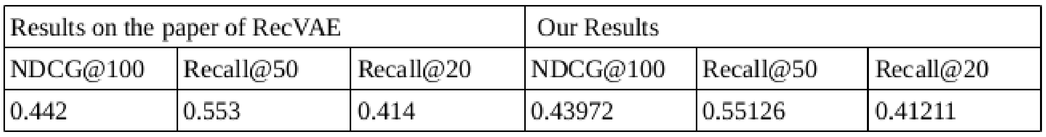


```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:21
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
100%                                  5/5 [01:41<00:00, 20.32s/it]

Test NDCG@100=0.43972 (0.00210)
Test Recall@20=0.41211 (0.00271)
Test Recall@50=0.55126 (0.00283)
```

Figure 2.5: Performance indices of RecVAE on the MovieLens 20M dataset

The following table summarizes the differences between the values of NDCG@k and Recall@k that we have founded after running the code of RecVAE on the Movie Lens 20M dataset and those that are founded on the paper of RecVAE that you can see on the link [3].



Results on the paper of RecVAE			Our Results		
NDCG@100	Recall@50	Recall@20	NDCG@100	Recall@50	Recall@20
0.442	0.553	0.414	0.43972	0.55126	0.41211

Figure 2.6: Comparison between the results of the paper[1] and our results

At the end, we have used T-SNE and UMAP like two dimensionality reduction techniques to better visualise the users embeddings in two dimensions. From the visualisation below we can notice that the users embeddings on the Movie Lens 20M dataset are separated approximately into 3 clusters.

```
<matplotlib.collections.PathCollection at 0x7ff9df8ad5f8>
```

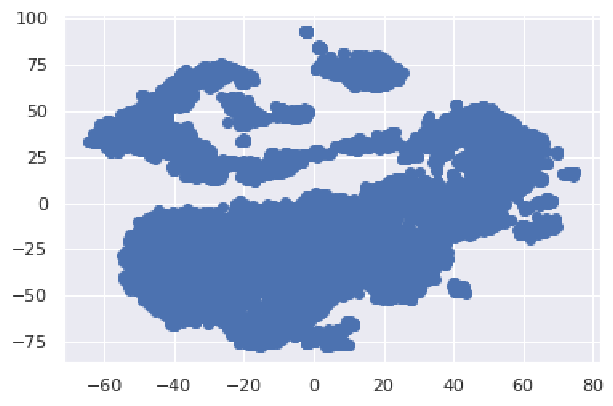


Figure 2.7: T-SNE visualisation of the users-embeddings on the MovieLens 20M dataset

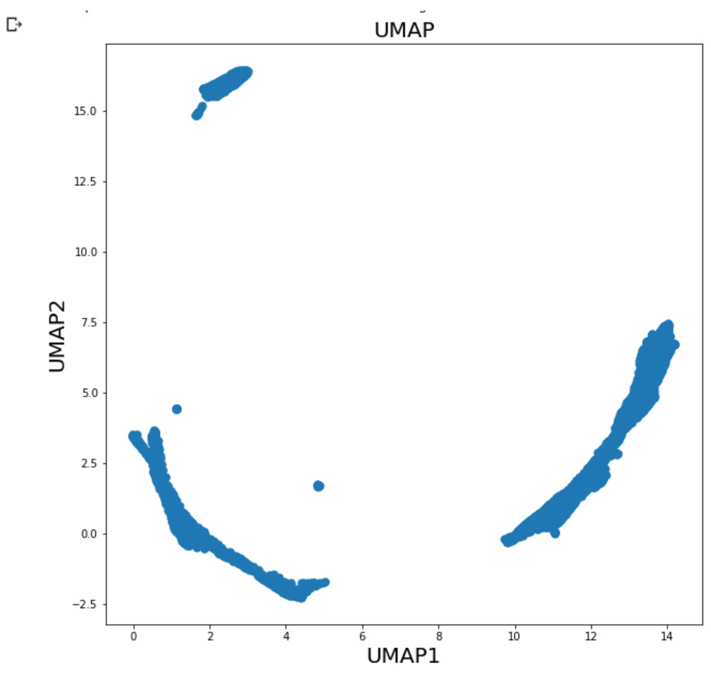
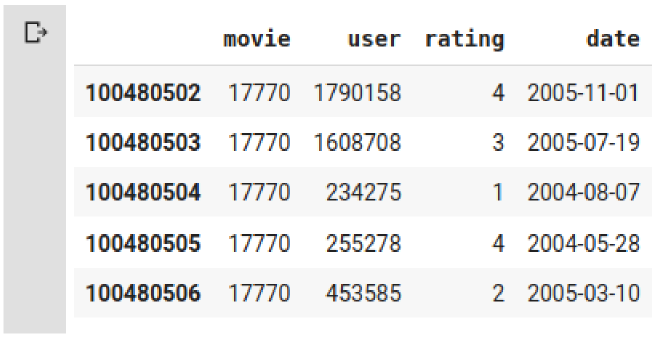


Figure 2.8: UMAP visualisation of the users-embeddings on the MovieLens 20M dataset

2.2.2 Running RecVAE code on the Netflix dataset

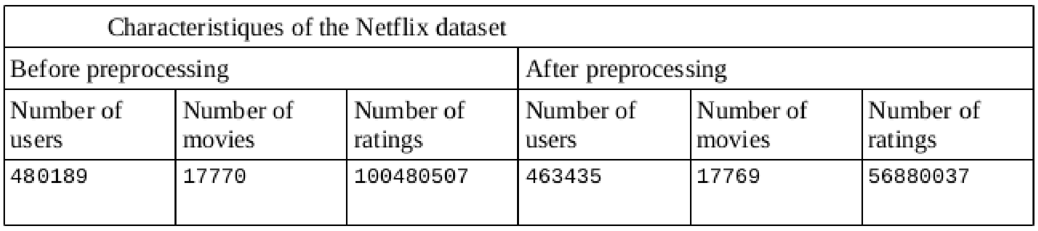
After preprocessing the Netflix-prize dataset by following the method used on the paper[2], we reproduced the same results founded in the RecVAE paper[1] by running the code of this method on this preprocessed dataset.The Netflix prize dataset as a dataframe is shown in the following figure.



	movie	user	rating	date
100480502	17770	1790158	4	2005-11-01
100480503	17770	1608708	3	2005-07-19
100480504	17770	234275	1	2004-08-07
100480505	17770	255278	4	2004-05-28
100480506	17770	453585	2	2005-03-10

Figure 2.9: The first rows in the Netflix dataset

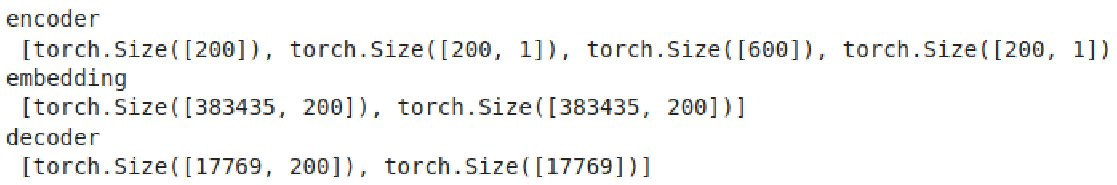
The preprocessing method of the dataset of Netflix is similar to the one of MovieLens 20M dataset. The following table summarizes the characteristics of the Netflix dataset before and after preprocessing it.



Characteristiques of the Netflix dataset					
Before preprocessing			After preprocessing		
Number of users	Number of movies	Number of ratings	Number of users	Number of movies	Number of ratings
480189	17770	100480507	463435	17769	56880037

Figure 2.10: Characteristics of the Netflix dataset

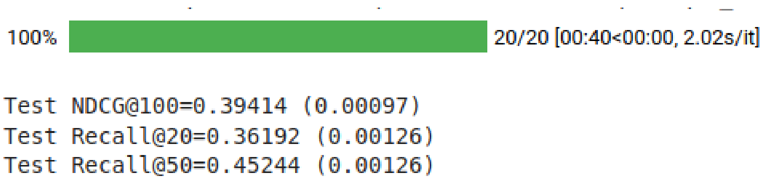
The first results that we have got after running the code of RecVAE method on the prepossessed dataset of Netflix are depicted in the following caption:



```
encoder
 [torch.Size([200]), torch.Size([200, 1]), torch.Size([600]), torch.Size([200, 1])
embedding
 [torch.Size([383435, 200]), torch.Size([383435, 200])]
decoder
 [torch.Size([17769, 200]), torch.Size([17769])]
```

Figure 2.11: The size of the users embeddings on the Netflix dataset

So as the MovieLens 20M dataset, the embeddings associated to each user among the 383435 users on the training dataset of Netflix are of dimension 200. Moreover the second results depicted in the following caption are the values of the indices NDCG and Recall. From NDCG@100 we can conclude that the Model of RecVAE can predict correctly 39,414% of the test user feedback on the dataset of Netflix then from Recall@20 we can deduce that 36,192% of the ratings are well predicted on the test user feedback and finally from the Recall@50 we can conclude that 45,244% are well predicted on the test user feedback.

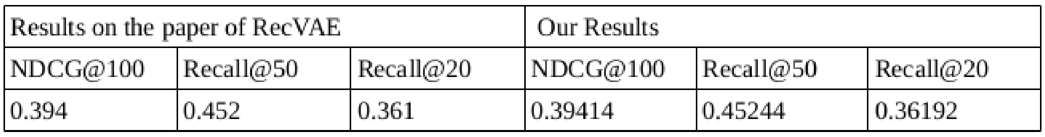


```
100% 20/20 [00:40<00:00, 2.02s/it]

Test NDCG@100=0.39414 (0.00097)
Test Recall@20=0.36192 (0.00126)
Test Recall@50=0.45244 (0.00126)
```

Figure 2.12: The Performance indices of the RecVAE method on the Netflix dataset.

The following table gives the comparison between the results that we have found after running the code of RecVAE on the Netflix dataset and the results founded on the paper of RecVAE that you can see on the link [3].



Results on the paper of RecVAE			Our Results		
NDCG@100	Recall@50	Recall@20	NDCG@100	Recall@50	Recall@20
0.394	0.452	0.361	0.39414	0.45244	0.36192

Figure 2.13: Comparison of our results and those of RecVAE paper on the Netflix dataset

As in the MovieLens20M dataset, we have used T-SNE and UMAP like two dimensionality reduction techniques to visualise the users-embeddings in two dimensions. From the two visualisations below we can notice that the users embeddings on the Netflix-Prize dataset are not separated into one specific number of clusters because of the huge size of the Netflix dataset comparing to the MovieLens20M dataset. So we can conclude that the visualisation with T-SNE and UMAP are not enough to discover the type of the embeddings distribution on the Netflix dataset.

```
100%  40000/40000 [04:07<00:00, 161.61it/s]

<matplotlib.collections.PathCollection at 0x7f70316f2748>
```

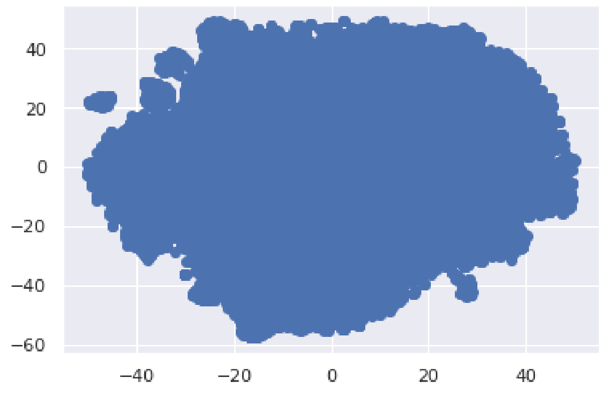


Figure 2.14: T-SNE visualisation of the users-embeddings on the Neflix dataset

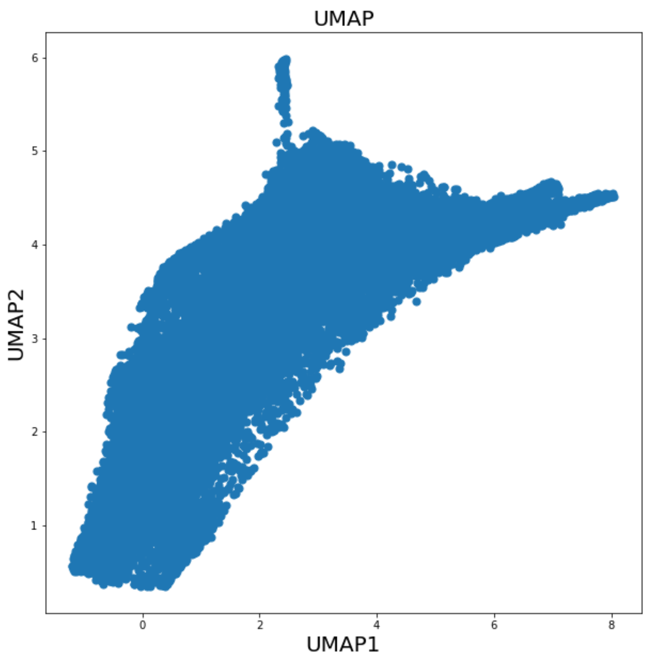
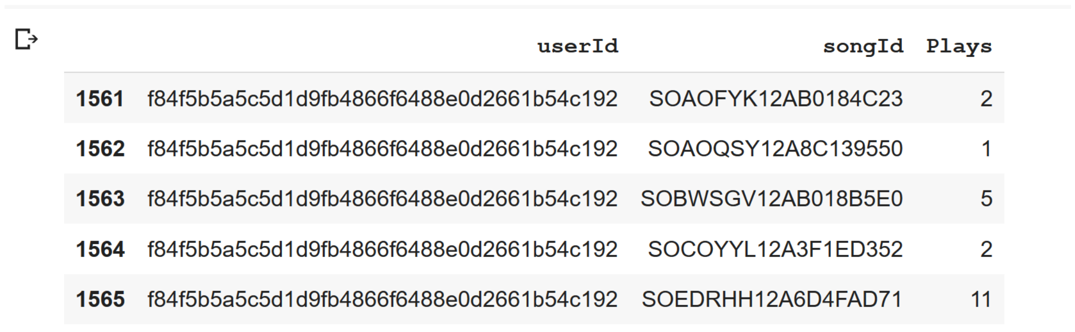


Figure 2.15: UMAP visualisation of the users-embeddings on the Neflix dataset

2.2.3 Running RecVAE code on the Million Songs dataset

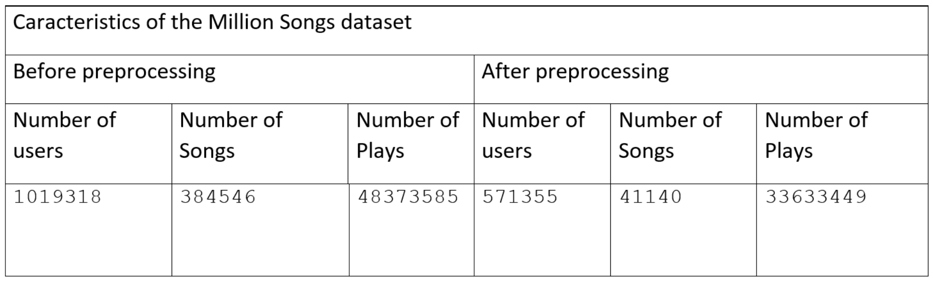
After preprocessing the Million Songs dataset by following the method used on the paper[2], we reproduced the same results founded in the RecVAE paper[1] by running the code of this method on this preprocessed dataset.The Million Songs dataset as a dataframe is shown in the following figure.



	userId	songId	Plays
1561	f84f5b5a5c5d1d9fb4866f6488e0d2661b54c192	SOAOFYK12AB0184C23	2
1562	f84f5b5a5c5d1d9fb4866f6488e0d2661b54c192	SOAOQSY12A8C139550	1
1563	f84f5b5a5c5d1d9fb4866f6488e0d2661b54c192	SOBWSGV12AB018B5E0	5
1564	f84f5b5a5c5d1d9fb4866f6488e0d2661b54c192	SOCOYYL12A3F1ED352	2
1565	f84f5b5a5c5d1d9fb4866f6488e0d2661b54c192	SOEDRHH12A6D4FAD71	11

Figure 2.16: The first rows in the Million Songs dataset

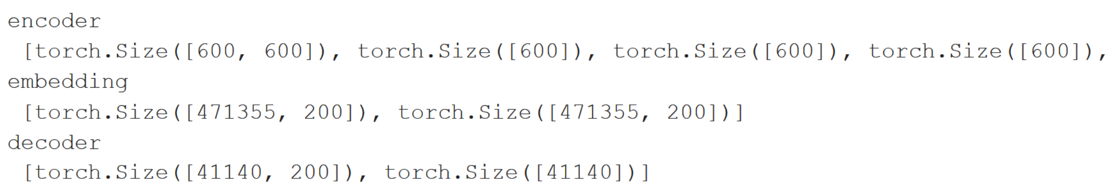
The preprocessing method of the Million Songs dataset is similar to the one of MovieLens 20M and Netflix datasets. The following table summarizes the characteristics of the Million Songs dataset before and after preprocessing it.



Caracteristics of the Million Songs dataset					
Before preprocessing			After preprocessing		
Number of users	Number of Songs	Number of Plays	Number of users	Number of Songs	Number of Plays
1019318	384546	48373585	571355	41140	33633449

Figure 2.17: Characteristics of the Million Songs dataset

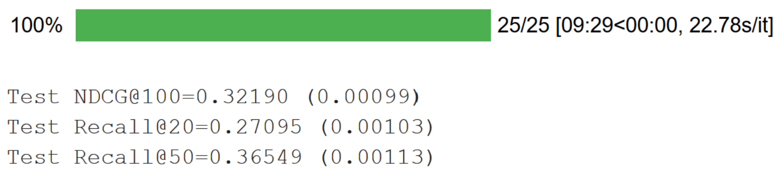
The first results that we have got after running the code of RecVAE method on the prepossessed dataset of Million Songs are depicted in the following caption:



```
encoder
 [torch.Size([600, 600]), torch.Size([600]), torch.Size([600]), torch.Size([600]),
embedding
 [torch.Size([471355, 200]), torch.Size([471355, 200])]
decoder
 [torch.Size([41140, 200]), torch.Size([41140])]
```

Figure 2.18: The size of the users embeddings on the Million Songs dataset

So as the MovieLens 20M dataset and Netflix datasets , the embeddings associated to each user among the 471355 users on the training dataset of Million Songs are of dimension 200. Moreover the second results depicted in the following caption are the values of the indices NDCG and Recall. From the NDCG@100 value we can conclude that the Model of RecVAE can predict correctly 32,19% of the test user feedback on the dataset of Million Songs then from Recall@20 we can deduce that 27,095% of the ratings are well predicted on the test user feedback and finally from the Recall@50 we can conclude that 36,549% are well predicted on the test user feedback.

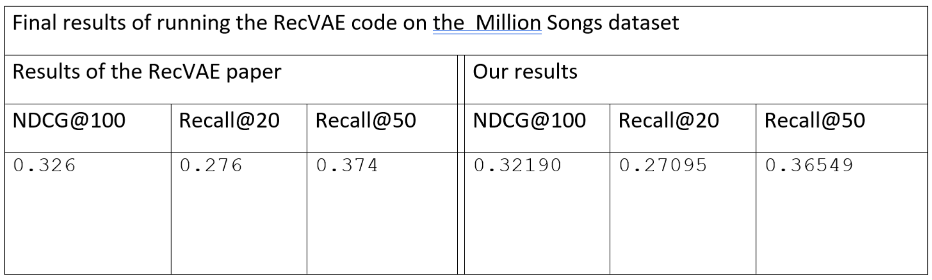


```
100% 25/25 [09:29<00:00, 22.78s/it]

Test NDCG@100=0.32190 (0.00099)
Test Recall@20=0.27095 (0.00103)
Test Recall@50=0.36549 (0.00113)
```

Figure 2.19: The Performance indices of the RecVAE method on the Million Songs dataset.

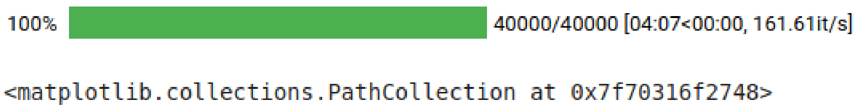
The following table shows the comparison between the results that we have found after running the code of RecVAE on the Million Songs dataset and the results founded on the paper of RecVAE that you can see on the link [3].



Final results of running the RecVAE code on the Million Songs dataset					
Results of the RecVAE paper			Our results		
NDCG@100	Recall@20	Recall@50	NDCG@100	Recall@20	Recall@50
0.326	0.276	0.374	0.32190	0.27095	0.36549

Figure 2.20: Comparison between our results and those of the RecVAE paper on the Million Songs dataset

As in the MovieLens20M and Netflix datasets, we have used T-SNE and UMAP like two dimensionality reduction techniques to visualise the users-embeddings in two dimensions. From the two visualisations below we can notice that the users embeddings on the Million Songs dataset have a gaussian appearance as UMAP illustrate in the second plot. Moreover, we can not conclude definitively by this visualisation with T-SNE and UMAP the type of the distribution of the users embeddings .



```
100% 40000/40000 [04:07<00:00, 161.61it/s]

<matplotlib.collections.PathCollection at 0x7f70316f2748>
```

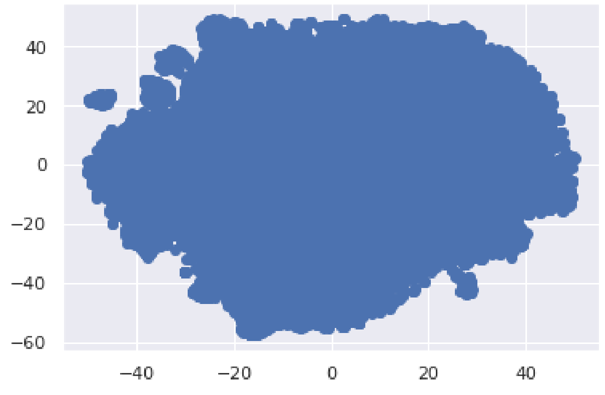


Figure 2.21: T-SNE visualisation of the users-embeddings on the Million Songs dataset

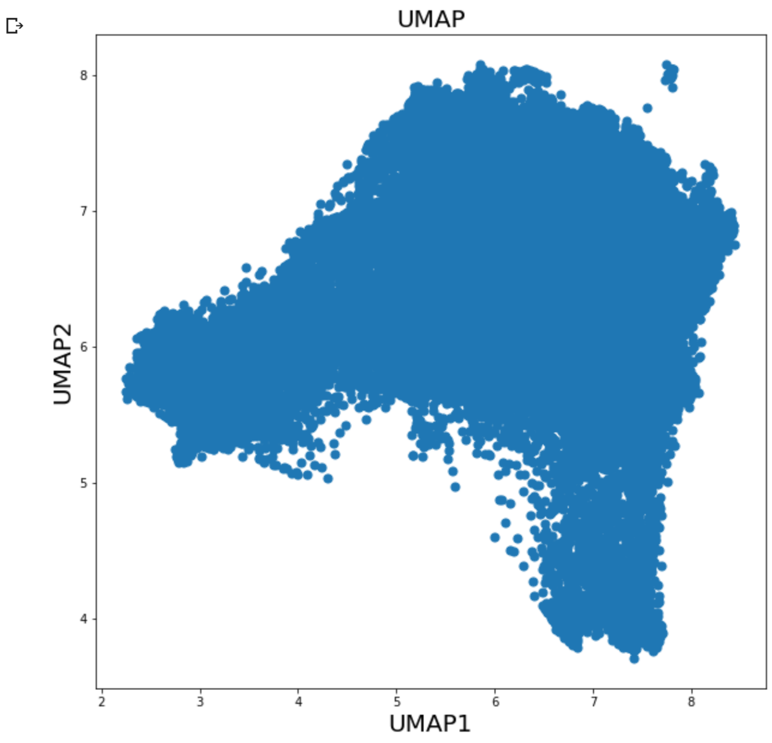


Figure 2.22: UMAP visualisation of the users-embeddings on the Million Songs dataset

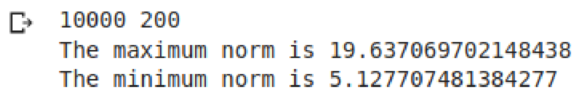
.

Chapter 3

Netflix, MovieLens-20M and Million Songs embeddings distribution type

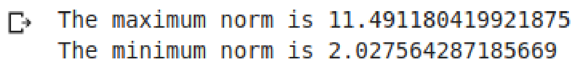
3.1 Users-embeddings inside a 200-dimensional sphere of radius R

Netflix, MovieLens20M and Million Songs users-embeddings are of dimension 200, it means that they are distributed in one 200 dimensional space. The idea is that we will do a search with l2 norm with a given threshold R to find how many points are inside a 200-dimensional sphere of Radius R. We will take this R from 0 until covering all our data-set : it means that we will calculate the maximum of the l2 norms of all our users-embedding and then we will variate the R from 0 to this maximum value of norms to cover all the embeddings. The maximum and the minimum values of the norms on the MovieLens20M, Netflix and Million Songs datasets are depicted in the following caption :



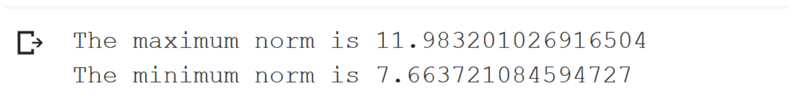
```
10000 200
The maximum norm is 19.637069702148438
The minimum norm is 5.127707481384277
```

Figure 3.1: Maximum and minimum values of the embedding-norms on the MovieLens20M dataset



```
The maximum norm is 11.491180419921875
The minimum norm is 2.027564287185669
```

Figure 3.2: Maximum and minimum values of the embedding-norms on the Netflix dataset



```
The maximum norm is 11.983201026916504
The minimum norm is 7.663721084594727
```

Figure 3.3: Maximum and minimum values of the embedding-norms on the Million Songs dataset

Approach 1 : let's verify this relation (1): $\log R(k) = \frac{1}{d} log(k) + const \quad (1)$ with k is the k-th neighbor of a query point and R(k) is the distance to this neighbor. We will sort the norms of the users embeddings and then we will associate this ordered norms to the indices of users in our test dataset.

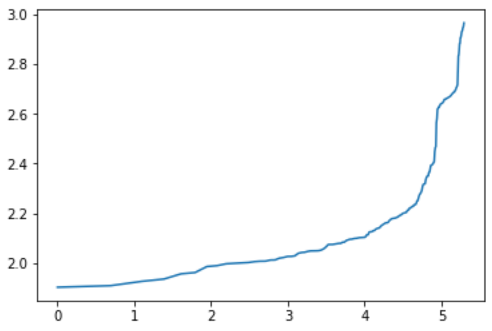


Figure 3.4: Verification of the relation (1) on the MovieLens 20M dataset

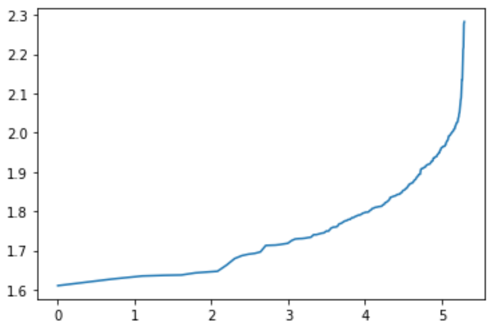


Figure 3.5: Verification of the relation (1) on the Netflix dataset

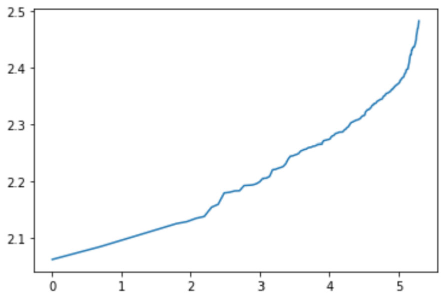


Figure 3.6: Verification of the relation (1) on the Million Songs dataset

To verify the relation (1) on one empirical uniform dataset we have generated 40000 points uniformly distributed inside one sphere of 200 dimensions and then we sorted the norms of those samples to associate them to the indices of samples. Then, we have tried also to generate one empirical Gaussian dataset by using one method named Gaussian random projection implemented in sklearn to reduce the dimension of one random matrix through Gaussian random projection. The components of the random matrix are drawn from N(0, 1/n-components), the n-components is the dimension of the target projection space, the target projection space is determined by the number of input samples (embeddings) and one strictly positive float named 'eps' to control the quality of embedding. Smaller values of 'eps' lead to better embedding. For us the target projection space is 4000 embedding of 200 dimensions to simulate the dimension of Netflix ,MovieLens20M and Million Songs embedding. This 4000 embedding are drawn from the N(0,1/200) on one 200 dimensional space. The two following figures show the verification of the relation (1) on the uniform and Gaussian datasets generated.

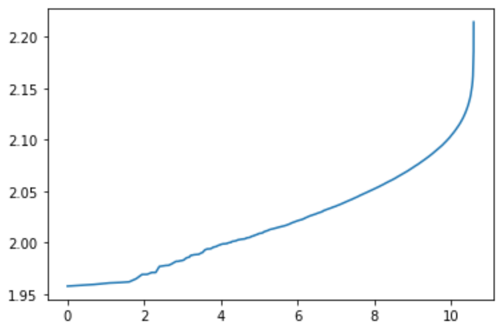


Figure 3.7: Verification of the relation (1) on one random dataset uniformly distributed

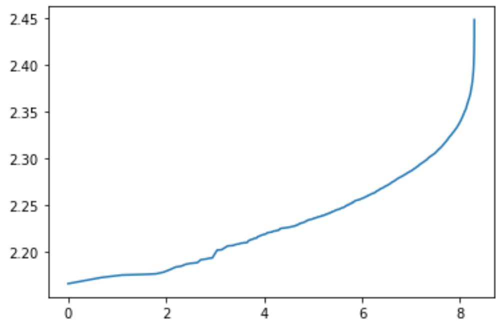


Figure 3.8: Verification of the relation (1) on one random dataset that have a Gaussian distribution

Comment: This two last experiments show that the distribution of the embedding on the empirical uniform and gaussian datasets seems to be similar and they are slightly different from the the Neflix, MovieLens 20M and Million Songs dataset. So, this approach is not enough to discover the type of the distribution of the embeddings on the 3 datasets.

Approach2: This approach consists in looking at the distribution of the norms of the users embeddings . First we will discover the distribution of the norms on the MovieLens 20M dataset, the Netflix dataset, the Million Songs dataset, in one empirical uniform dataset and finally in one empirical gaussian dataset.

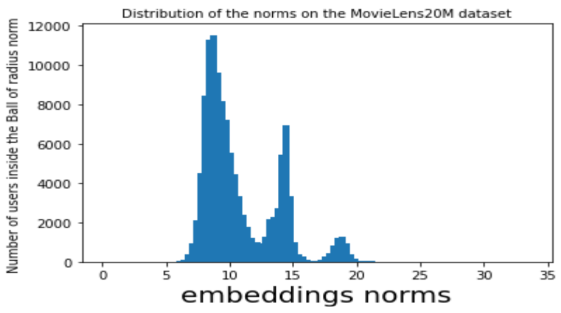


Figure 3.9: Distribution of the norms on the MovieLens20M dataset

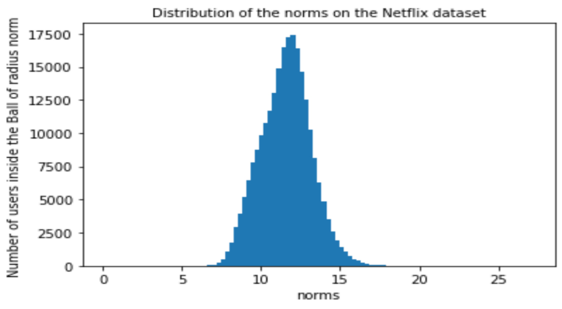


Figure 3.10: Distribution of the norms on the Netflix dataset

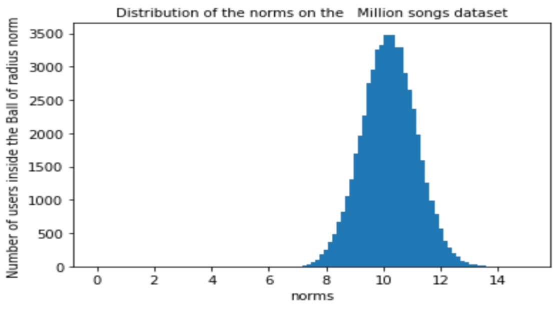


Figure 3.11: Distribution of the norms on the Million Songs dataset

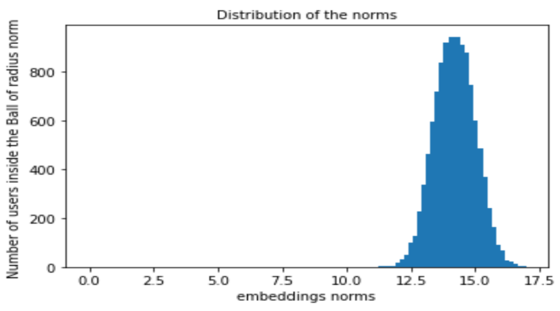


Figure 3.12: Distribution of the norms on the empirical uniform dataset

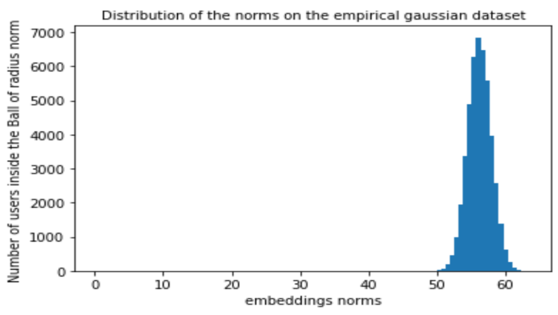


Figure 3.13: Distribution of the norms on the empirical gaussian dataset

Comment : The distribution of the norms on the uniform and gaussian plots looks approximately similar. Furthermore, the norms of the embeddings on the MovieLens 20M dataset are concentrated decreasingly on the norms 7.5 , 14 and finally 19. Moreover, the embeddings on the Netflix dataset are uniformly distributed in the interval of norms [4.2 , 10] and for the Million Songs dataset the norms are concentrated in the interval [8,12]. To sum up, from this approach we can not decide if the distribution of the embeddings on the 3 datasets is uniform or gaussian.

Approach3 : In this approach we will keep changing the centre of the ball that we will use to measure the distribution of our embeddings norms, the radius r of our ball is fixed and at each time we will change the centre of this ball to measure the variation of the distribution of the norms in function of the variation of the center of the ball. As a result, the number of plots that we can have is equal to the number of centers generated. We have selected randomly 3 centers to see how the distribution of the norms change with the variation of the centers.

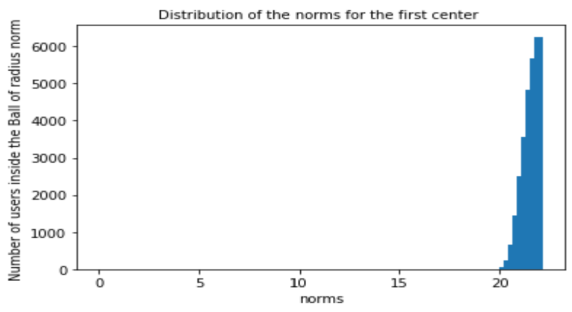


Figure 3.14: Distribution of the norms on the MovieLens 20M with the first randomly selected center of the Ball of radius r

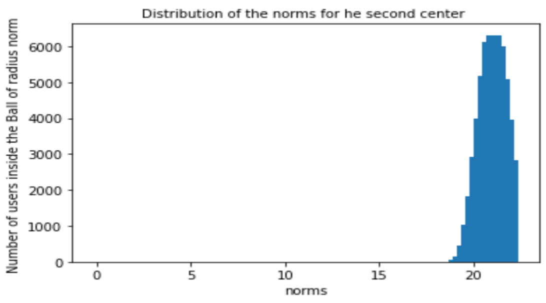


Figure 3.15: Distribution of the norms on the MovieLens 20M with the second randomly selected center of the Ball of radius r

Comment : From the 2 plots of the MovieLens 20M dataset we might conclude that the norms of our users-embeddings are concentrated on the interval of norms [19,22] and the form of the two plots are clearly different so the distribution of the norms of our embeddings can not be uniform because when we change the centre of the ball the form of the distribution also change.

Approach4 : This approach consists in looking at the distribution of the angles between the 200 canonical base vector of our 200 space and the vectors of our users embeddings. It means that at each iteration among the 200 directions we will fix one base vector of size 200 that looks like [0,0,. . . ,1,0,0,0] and we will measure the distribution of the angles between this canonical vector and all the vectors of our embeddings.The angle between any two vectors is calculated by using the dot-product and the arccos of this dot-product. At the end we will have 200 plots, because each plot is associated to one canonical based vector. We will select Randomly 3 plots and in each one of them we will see the distribution of the angles on the axis='x' in function of the Count of users on the axis='y'.

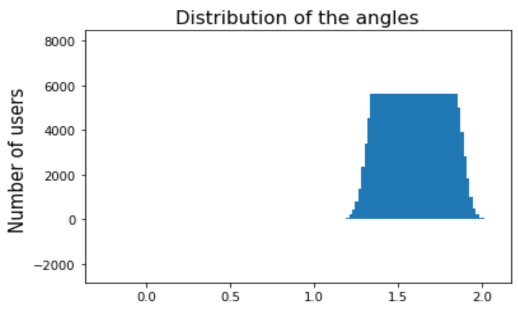


Figure 3.16: distribution of the angles on the Netflix dataset

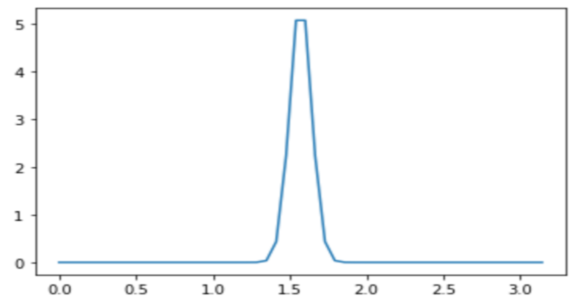


Figure 3.17: the empirical distribution of the angles described by (2)

Comment : First, we have to notice that there is no difference between considering two random vectors or considering a random vector and a canonical vector. Moreover, the experiments on the MovieLens20M, Netflix and Million Songs datasets prove that our embeddings are approximately concentrated between $\pi/8 = 0.4$ and $\pi/5 = 0.62$. On the paper [4] it is shown that, when the dimension p is fixed like for us p=200,as n the number of samples takes high values, the empirical distribution of the angles converges to a distribution with the density function given by:

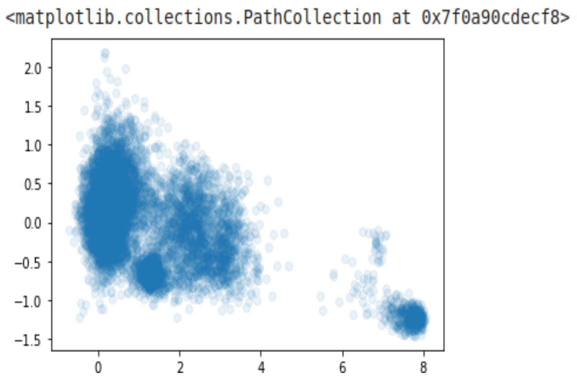
$$h(\theta) = \frac{1}{\sqrt{\pi}} \frac{\Gamma(p/2)}{\Gamma(p-1/2)} \sin(\theta)^{p-2} \quad (2)$$

When the dimension is high, most of the angles are concentrated around $\pi/2$. We have to notice that when p=2, $h(\theta)$ is the uniform density on [0,π] and when $p > 2$, h() is uni-modal. The concentration becomes stronger around $\pi/2$ as the dimension p grows since $\sin(\theta)^{p-2}$ converges to zero more quickly for $h(\theta) \neq \pi/2$.

3.2 Random projection of two random pair of embeddings coordinates

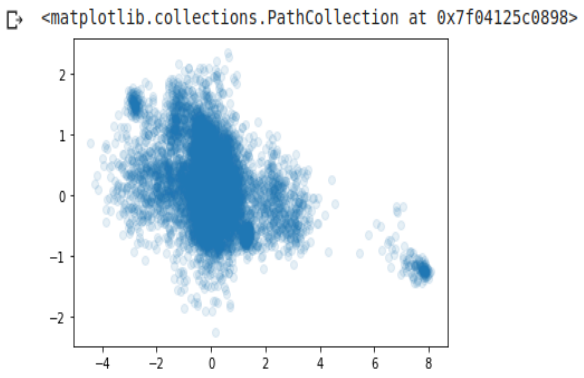
We will select randomly two features of embeddings from the 200 features that we have and then we will plot one feature vector of dimension the number of users-embeddings in function of the second feature.

3.2.1 Random projections on the Netflix and MovieLens20M datasets



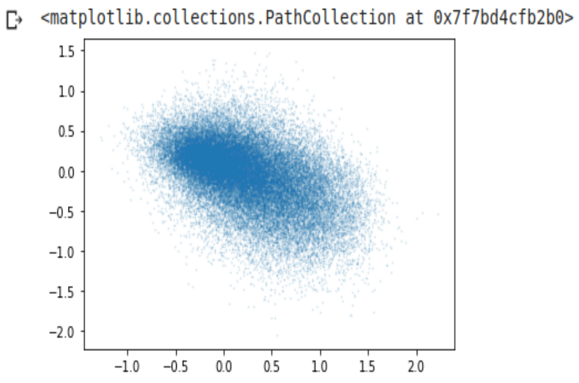
```
<matplotlib.collections.PathCollection at 0x7f0a90cdecf8>
```

Figure 3.18: First random projection on the MovieLens20M dataset



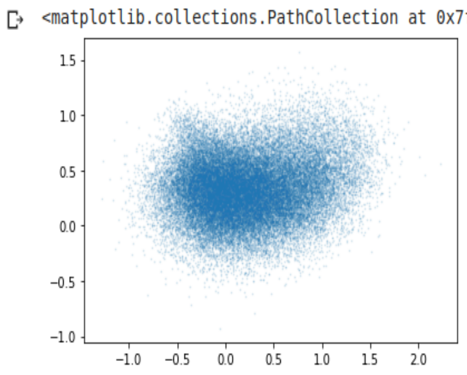
```
<matplotlib.collections.PathCollection at 0x7f04125c0898>
```

Figure 3.19: Second random projection on the MovieLens20M dataset



```
<matplotlib.collections.PathCollection at 0x7f7bd4cfb2b0>
```

Figure 3.20: First random projection on the Netflix dataset



```
<matplotlib.collections.PathCollection at 0x7f7bd5dcee80>
```

Figure 3.21: Second random projection on the Netflix dataset

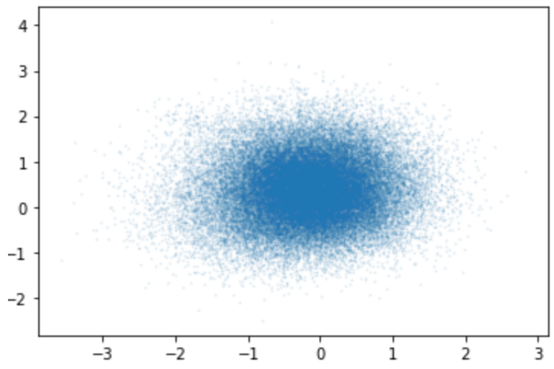


Figure 3.22: First random projection on the Million Songs dataset

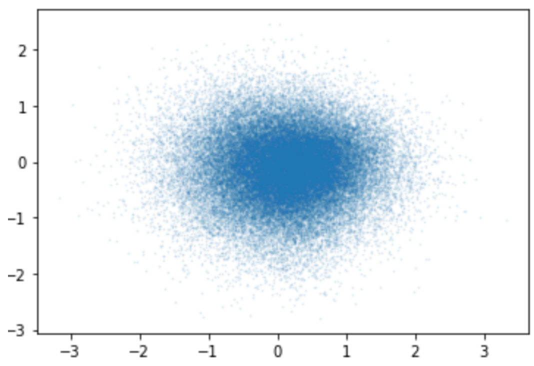


Figure 3.23: Second random projection on the Million Songs dataset

3.2.2 Random projections on the empirical uniform and gaussian datasets

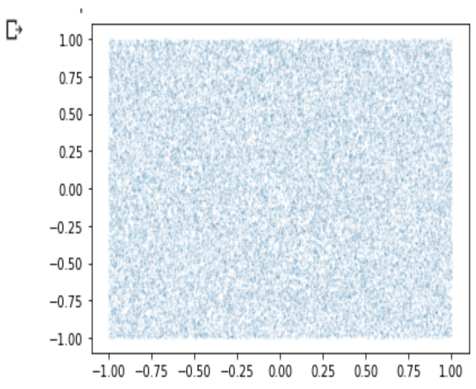
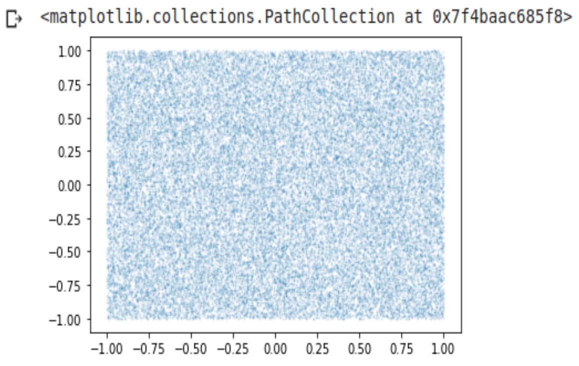
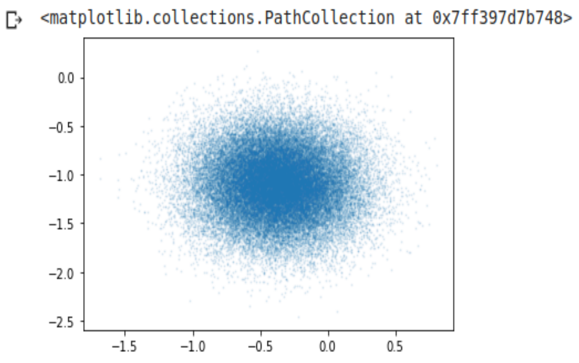


Figure 3.24: First random projection on the empirical uniform dataset



```
<matplotlib.collections.PathCollection at 0x7f4baac685f8>
```

Figure 3.25: Second random projection on the empirical uniform dataset



```
<matplotlib.collections.PathCollection at 0x7ff397d7b748>
```

Figure 3.26: First random projection on the empirical gaussian dataset

```
<matplotlib.collections.PathCollection at 0x7ff39605a4a8>
```

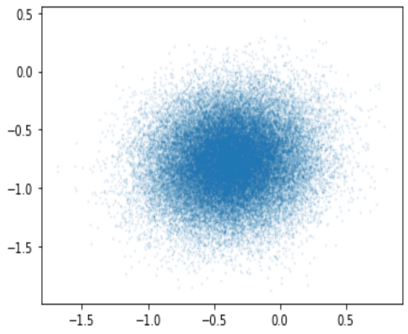


Figure 3.27: Second random projection on the empirical gaussian dataset

Comment: from the random projection on the 5 datasets we can conclude that the distributions of the embeddings on the MovieLens 20M , Netflix and Million Songs datasets are near to be a gaussian distribution than to be a uniform distribution.

3.3 The Gaussian Mixture Model technique on the Netflix and MovieLens 20M datasets

3.3.1 An overview on the gaussian Mixture Model technique

We have choosed this technique of clustering because it is the most efficient technique to classify our embeddings into clusters to have an idea about the distribution of our embeddings , this technique is the generalisation of the K-means technique. In fact K-means is the Gaussian Mixture model when the variance of the Random gaussian variables on this mixture is Identity. In the RecVAE model they modelize the distribution of their embeddings by one Gaussian mixture model but the variance of their gaussian variables in this mixture is not Identity. This is why K-means will not give a relevant informations about the classification of the embeddings.Furthermore,K-means has two main drawbacks as the lack of flexibility in cluster shape and lack of probabilistic cluster assignment. As a result we have choosed to fit our embeddings by using this technique of Gaussian Mixture Model with different number of components and we verify the quality of the fit for each value of number of components used to get an idea about the uniformity of the distribution of our user-embedings. The BIC:Bayesian information criterion is a criterion for model selection among a finite set of models , the model with the lowest BIC is preferred because the BIC estimates the relative amount of information lost by a given model: the less information a model loses, the higher the quality of that model. There is another criterion similar to BIC is AIC: Akaike information criterion like an estimator of the quality of each model, relative to each of the other models. AIC provides a means for model selection. In estimating the amount of information lost by a model, AIC deals with the trade-off between the goodness of fit of the model and the simplicity of the model. In other words, AIC deals with both the risk of overfitting and the risk of underfitting. When fitting models, it is possible to increase the likelihood by adding parameters, but doing so may result in overfitting. Both BIC and AIC attempt to resolve this problem by introducing a penalty term for the number of parameters in the model, the penalty term is larger in BIC than in AIC this is why we have choosed to work with BIC in our experiment. So After fitting the embedding-array with this model we have measured the BIC values associated to each value of components. The plot below illustrate the variation of the BIC in function of the number of components.

3.3.2 Gaussian Mixture Model technique on the Netflix, MovieLens 20M and Million Songs datasets

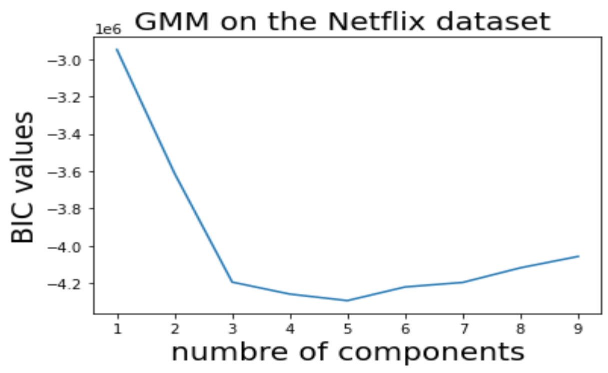


Figure 3.28: Gaussian Mixture Model on the Netflix dataset

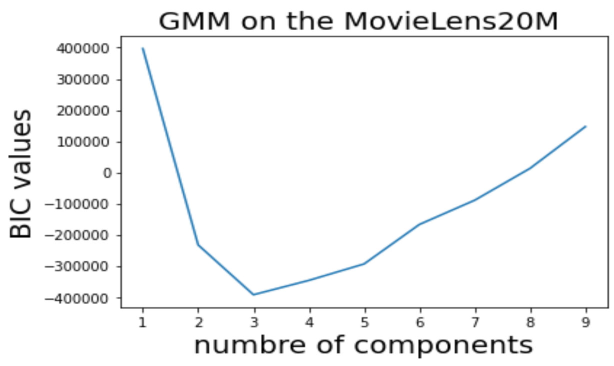


Figure 3.29: Gaussian Mixture Model technique on the MovieLens 20M dataset

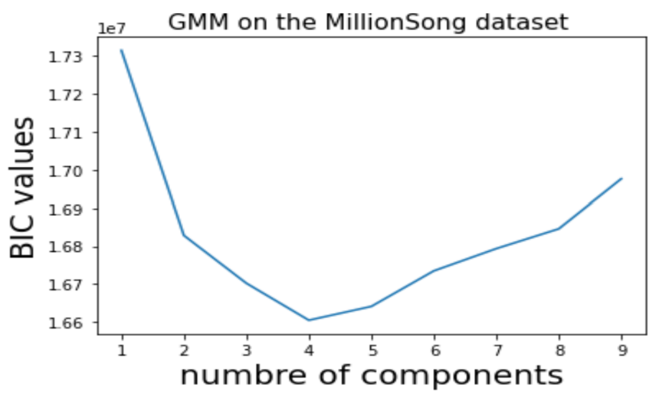


Figure 3.30: Gaussian Mixture Model technique on the Million Songs dataset

Comment : At the beginning, we have to notice that in the figure 2.1 that explain the architecture of RecVAE method the encoder in the left named encoder as prior is a mixture of 3 gaussians: this information is clearly proved in the class GaussianMixturePriorWithAprPost that you can find in the code of RecVAE method in the link [4]. Moreover, this class is called in the time of the setting and the calculation of the users embeddings and another time in the calculation of the KL divergence to reduce the loss function between the encoder as prior in the left of figure 2.1 and the encoder in the right. So finally at the end of the training of the model the distribution of the embeddings encoded by the encoder in the right with dropout rate = 0.5 will converge to the same distribution of the embeddings in the encoder as prior : Mixture of Gaussian's distribution, this is the reason of our decision to approximate the number of components of each mixture of gaussians in the 3 datasets. From the first plot of the Netflix dataset we noticed that the BIC is minimal for n-components=5, which means that the distribution of the Netflix embeddings is a mixture of 5 gaussians. Moreover, from the second plot of the MovieLens20M dataset we can conclude that distribution of the MovieLens20M embeddings is a mixture of 3 gaussians because the minimal value of BIC is on n-components = 3. Finally, from the third plot of the Million Song dataset we can conclude that the distribution of the embeddings on the Million Songs dataset is a mixture of 4 gaussians, because the minimal value of BIC is on n-components = 4. The number of components in the Mixtures of gaussians of the 3 datasets is different because of the different number of users in each dataset and also this method of Gaussian Mixture Model still an approximation and it does not give 100% exact results. To sum up, the distribution of the embeddings on the MovieLens20M,Netflix and Million Songs datasets is a mixture of gaussians.

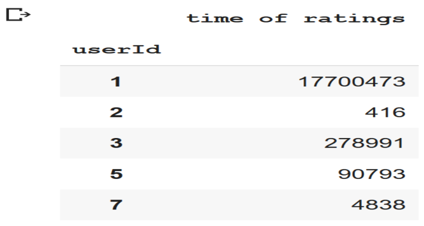
Chapter 4

Trace of the embeddings on the MovieLens20M and Netflix datasets

In this chapter we will try to study the users embeddings trace on the MovieLens20M dataset. After analyzing the users ratings distribution, we have fixed the following strategy to embed the users: If a user have less than 50 ratings then we will embed it just one time. However if the user have more 50 ratings then We will use 80 % of his ratings to calculate the first embedding and for each 10 ratings we will embed another time this user. In the MovieLens20M trace, we will associate 522947 embeddings to 116677 users.

4.1 Trace of the embeddings on the MovieLens20M dataset

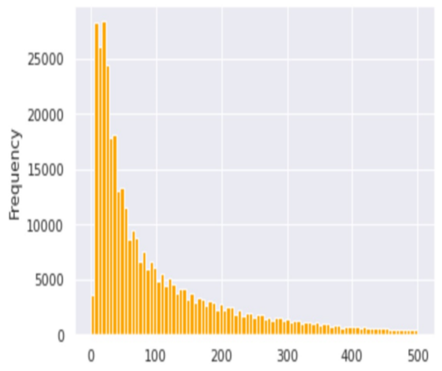
The trace of the users-embeddings is a table that contain the columns userId, user-embedding and finally the timestamps associated to each user. As a result, the table will contain the number of ratings as number of rows and 202 columns: one column for the userId and 200 columns for the users-embeddings and one column for the timestamps. The following figure illustrate the duration of the rating of each user it mains that we will calculate the difference on time between the first and the last rating for each user on the MovieLens20M dataset.

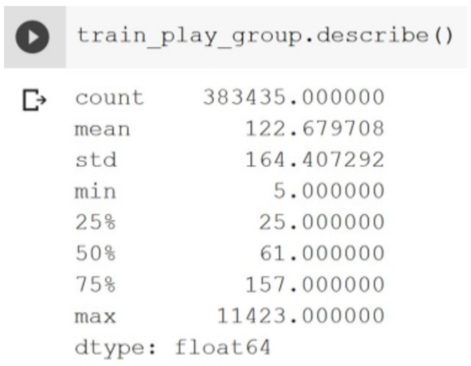


userId	time of ratings
1	17700473
2	416
3	278991
5	90793
7	4838

Figure 4.1: The ratings duration of each user in the MovieLens 20M dataset

The figure below illustrate the users ratings distribution description.



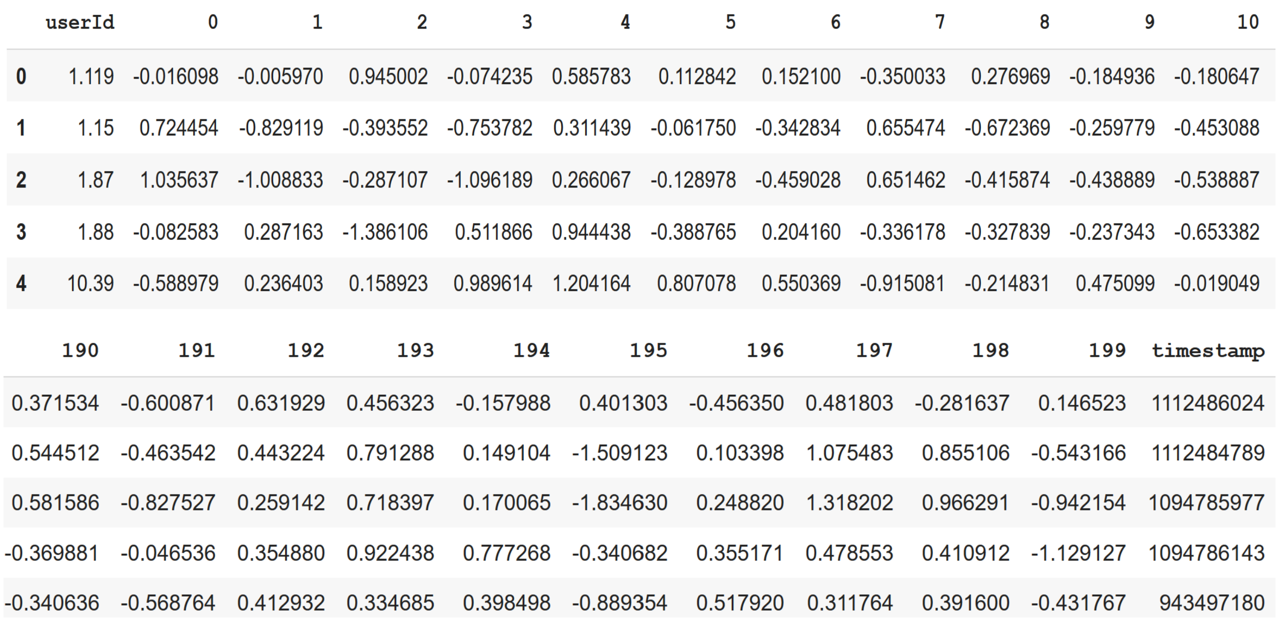


```
train_play_group.describe()

count    383435.000000
mean        122.679708
std         164.407292
min           5.000000
25%          25.000000
50%          61.000000
75%         157.000000
max       11423.000000
dtype: float64
```

Figure 4.2: The rating distribution of the MovieLens20M dataset

based on the description of the ratings distribution of the MovieLens 20M users we have fixed this strategy to associate to each user one specified number of embeddings and not only one embeddings. The first part of the userId column is to refer to the original user on the original dataset. The second part of the userId is just to separate the embeddings associated to each user in he original dataset . For instance, the first user has 4 embeddings sorted by their timestamps.



	userId	0	1	2	3	4	5	6	7	8	9	10	190	191	192	193	194	195	196	197	198	199	timestamp
0	1.119	-0.016098	-0.005970	0.945002	-0.074235	0.585783	0.112842	0.152100	-0.350033	0.276969	-0.184936	-0.180647	0.371534	-0.600871	0.631929	0.456323	-0.157988	0.401303	-0.456350	0.481803	-0.281637	0.146523	1112486024
1	1.15	0.724454	-0.829119	-0.393552	-0.753782	0.311439	-0.061750	-0.342834	0.655474	-0.672369	-0.259779	-0.453088	0.544512	-0.463542	0.443224	0.791288	0.149104	-1.509123	0.103398	1.075483	0.855106	-0.543166	1112484789
2	1.87	1.035637	-1.008833	-0.287107	-1.096189	0.266067	-0.128978	-0.459028	0.651462	-0.415874	-0.438889	-0.538887	0.581586	-0.827527	0.259142	0.718397	0.170065	-1.834630	0.248820	1.318202	0.966291	-0.942154	1094785977
3	1.88	-0.082583	0.287163	-1.386106	0.511866	0.944438	-0.388765	0.204160	-0.336178	-0.327839	-0.237343	-0.653382	-0.369881	-0.046536	0.354880	0.922438	0.777268	-0.340682	0.355171	0.478553	0.410912	-1.129127	1094786143
4	10.39	-0.588979	0.236403	0.158923	0.989614	1.204164	0.807078	0.550369	-0.915081	-0.214831	0.475099	-0.019049	-0.340636	-0.568764	0.412932	0.334685	0.398498	-0.889354	0.517920	0.311764	0.391600	-0.431767	943497180

Figure 4.3: Trace of the embeddings on the MovieLens 20M dataset

References :

[1] RecVAE: a New Variational Autoencoder for Top-N Recommendations with Implicit Feedback 24 Dec 2019 , Ilya Shenbin , Anton Alekseev , Elena Tutubalina , Valentin Malykh , Sergey I. Nikolenko

[2] Variational Autoencoders for Collaborative Filtering Dawen Liang Rahul G. Krishnan Netflix Los Gatos, CA dliang@netflix.com MIT Cambridge, MA rahulgk@mit.edu Matthew D. Hoffman Tony Jebara Google AI San Francisco, CA mhoffman@google.com

Netflix Los Gatos, CA tjebara@netflix.com