COGS 118B Final Project Report

Team: CCXXXIII
Derek Chen, Xianglong Wang, Chen Yang, Xiaoshuo Yao, Fei Yin, Futian Zhang

June 5, 2021

1 Introduction and Motivation

In today's extremely busy world, recommendation systems are becoming increasingly important as people are always looking for ways to save more time from a myriad of tasks they seek to accomplish. With a system that makes the right decision for its users on which anime to watch, the users may save both time and effort. The purpose of this system is simple, it searches for contents that are likely of interest to each individual user. This involves an understanding of the users as well as a complete list of anime.

In this project, the goal for our team is to make an anime recommendation system that predicts each user's ratings on new anime that the user has not watched before. Combined with what we learned in class, we decided to approach this problem with both an accuracy-oriented supervised learning setup as well as an exploration-oriented unsupervised learning setup. Both setups utilize principal component analysis (PCA) for dimensionality reduction.

2 Related Work

In the *Recommendation systems: Principles, methods and evaluation* section of *"Egyptian Informatics Journal"* [2], it is stated that "[an explicit feedback] system normally prompts the user through the system interface to provide ratings for items in order to construct and improve his model". This is the approach that we followed in our supervised learning setup, where we maximize accuracy with explicit feedback in the form of user ratings.

In the *An improved collaborative movie recommendation system using computational intelligence* section of *"Journal of Visual Languages & Computing"* section [3], the authors show different usages of PCA in their paper. One example is the PCA-GAKM based collaborative filtering framework that further improves their recommendation system. This framework is a hybrid cluster-based model that helps to improve movie prediction accuracy. Such usage of PCA is similar to what we have done in our Method 2, which will be explained in detail in the Methods section.

3 Data Cleaning

All data used are from the *Anime Recommendations Database* from *Kaggle* [1]. This data comes in two parts: 1) a CSV file where each entry records the attributes of an anime (anime id, name, genre, type, number of episodes, average rating, and number of ratings), and 2) another CSV file where each entry records the attributes of an individual user rating (user id, anime id, and user rating). We name the former df_anime and the latter df_rating. To prepare this data for subsequent use, we perform the following cleaning operations to the data:

1. Remove entries in df_anime with missing data.
2. One-hot encode the categorical attributes (genre and type) in df_anime.
3. Remove inappropriate genre attributes and all anime entries under those genres in df_anime.

4. Normalize the numerical attributes in df_anime such that all attributes are given the same weight.
5. Remove entries in df_rating corresponding to all the removed entries in df_anime.
6. Remove entries in df_rating where the same user rates the same anime multiple times, a user rates less than 10 anime, or a user rates an anime with less than 200 ratings.

The resulting df_anime contains 10256 entries, each with 52 attributes; the resulting df_rating contains 5233173 entries, each with 3 attributes, belonging to 54004 distinct users.

4 Methods

4.1 Overview

To achieve our goal of accurate anime recommendation, we develop a system that predicts user ratings for anime utilizing PCA as the major algorithm. Given the attributes of all rated anime and the existing ratings from each user, our intention is that the system would predict user ratings for unseen anime, thus achieving the purpose of anime recommendation. To make our system testable, we adopt a supervised learning setup with 80-20 train-test split.

We explore two methods to solve this problem, namely, two ways of implementing PCA. Although both approaches follow the 80-20 train-test split, the split is done in different ways as the data is used differently. Both methods are explained in detail below.

In addition, we implement a K-means based exploratory unsupervised learning model to make actual recommendations. We use existing data for clustering then create small datasets by rating anime ourselves. We adjust the model through verification of the anime recommended to each of us.

4.2 Method 1: An Anime-space PCA

One way of PCA implementation for this problem is to do it in an anime-by-anime fashion. Since the covariance between every two anime is likely non-zero (the distribution of covariance between every two anime is shown in Figure 1), we can use these covariances to predict each user's rating for an anime based on that user's rating for other anime.

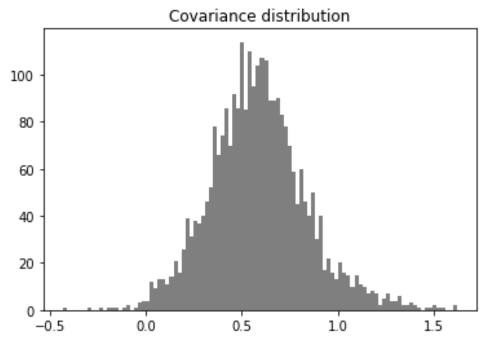

Figure 1: Covariance distribution between every two anime.

In practice, we first split the data set into training and testing sets. This split is visualized in Figure 2. The training set is represented as an $N_a \times N_{u_{train}}$ matrix, M_{train}, where $N_{u_{train}}$ is the number of users in the training set and N_a is the number of anime. Each entry in M_{train} represents a rating from a user for an anime. We subtract the mean rating of each anime μ_a from M_{train} to zero-center it. Note that if a user has not rated an anime, we adjust M_{train} such that the corresponding entry in M_{train} remains 0 after zero-centering. Next, we find V as the eigenvectors of the covariance matrix of M_{train}. V is an $N_a \times N_a$ matrix. We then use V to transform the original data into new coordinates $C = V^T M$ and form $\bar{V}$ and $\bar{C}$ as V and C corresponding to the top k principal component, respectively. Note that M contains both M_{train} and M_{test}, the testing set. ($\bar{V}$ is $N_a \times k$ and $\bar{C}$ is $k \times N_u$). Finally, we regenerate the data as $R = \bar{V}\bar{C} + \mu_a$.

Since R contains the prediction for missing anime, we intentionally exclude pre-existing ratings from M as we calculate $C = V^T M$ so that we may test the accuracy of our prediction by comparing against the ground-truth.

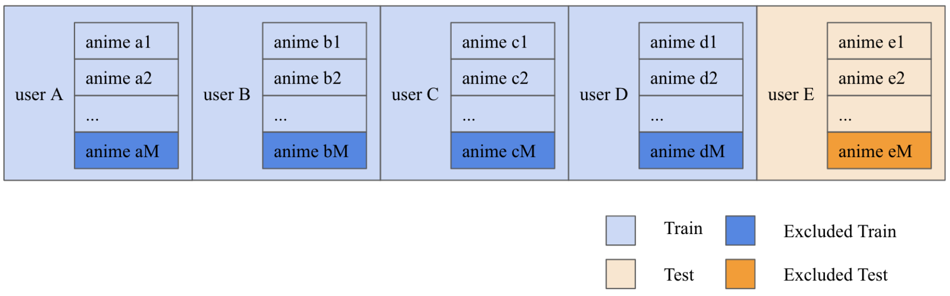

Figure 2: Visualization 80-20 train-test split for Method 1. Train: used in PCA. Excluded Train & Excluded Test: used in calculating training and testing RMSE, respectively.

4.3 Method 2: A User-space PCA

Another way of PCA implementation for this problem is to do it in a user-by-user fashion. That is, for each user, we assemble all anime rated by that user into a subspace space, which we call a user-space. Each user-space is an $M_u \times 50$ matrix where M_u is the number of anime rated by a particular user, and 50 comes from the fact that there are 50 attributes (aside from name and anime id) for each anime after one-hot encoding. The train-test split for Method 2 is visualized in Figure 3.

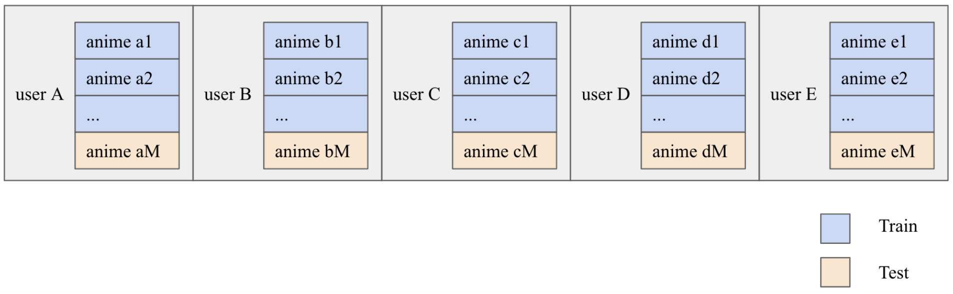

Figure 3: Visualization 80-20 train-test split for Method 2.

For each user-space, we apply PCA to reduce the number of dimensions of attributes to $min(0.8M_u, k)$. That is, for a user-space with more than k ratings (i.e. $M_u > k$), we reduce its dimensionality to k. Otherwise, we reduce it to $0.8M_u$ dimensions where $0.8M_u$ comes from the fact that we use 80% of the data in a user-space for training. We find that $k = 20$ gives sufficient PCA-explained variance ratio, which we will explain more in the Results section.

Next, we map both the training and testing parts of each user-space to its dimension-reduced eigen user-space. Within each eigen user-space, we find the sample mean μ_i and sample covariance Σ_i for each rating i. Since the rating from a user is an integer from 1 to 10, i.e. $i \in 1, ..., 10$, we can easily cluster the training part of user-space by ratings and assign to each testing data the rating corresponding to the cluster with the smallest Mahalanobis distance to that testing data, i.e. $\hat{i} = \arg\min_i (\mu_i - \mu)^T \Sigma^{-1} (\mu_i - \mu)$. Note that this clustering process is similar to applying one iteration of the K-means algorithm with both cluster means and covariances. The assigned ratings for the testing data can then be compared against the actual ratings for calculation of error.

4.4 clustering: a recommendation system

Based on the results from Method 2, we develop a clustering method to recommend anime to users based on their previous ratings. We assume that an anime's possibility of being recommended is based on its 43 genres, which we reduce to 3 dimensions with PCA. Then, we apply K-means clustering on the data in the 3 dimensional space. Through testing, we decide to set k to 15, resulting in the following clusters:

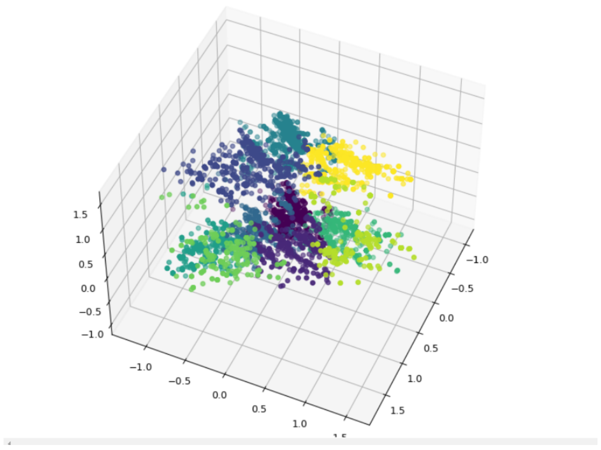

Figure 4: Visualization of clusters

For our recommendation algorithm, we first calculate a user's mean rating, μ_u. We decide that the user likes an anime if the user rates it above μ_u. We then calculate the mean position, $\hat{\mu}_u$, of the anime liked by the user in the 3 dimensional space. To recommend anime to the user, we first find the cluster i with its center closest to $\hat{\mu}_u$. Then, we recommend to the user the top 10 anime closest to $\hat{\mu}_u$ within that cluster.

5 Results

5.1 Results for Method 1 (An Anime-space PCA)

Method 1 achieves a top RMSE of 1.1591 on training set and 1.2853 on testing data where RMSE calculates the difference between the ground-truths and predicted ratings. This is an improvement on simply using the average rating for prediction, which scores 1.4199. Note that when we regenerate our predicted rating matrix, we make any rating that is greater than 10 equal to 10 and any ratings that is smaller than 1 equal to 1.

During method development, we hypothesized that two variables would significantly influence our final RMSE: 1) the number of top principle components used and 2) a multiplication factor, λ, in regenerating the predicted rating matrix, i.e. $R = \lambda \bar{V}\bar{C} + \mu_a$. We tuned these two variables. The training RMSE for top N principle components with $\lambda = 1$ are shown in Table 1. We see that setting $N = 1000$ yields the best result. Then, with $N = 1000$, we tuned the λ factor, with the training RMSE shown in Table 2.

# principle components (N)	RMSE
100	1.3601
200	1.3457
500	1.3148
1000	1.2937
2000	1.4180

Table 1: RMSE with different numbers of principle components.

λ factor	RMSE
1	1.2936
2	1.2166
3	1.1726
4	1.1591
5	1.1712
6	1.2023

Table 2: RMSE with different λ factors.

Although we cannot conclude on the optimal hyperparameter combination without an extensive grid search, we find that setting $N = 1000$ and $\lambda = 4$ yields a relatively low RMSE.

5.2 Results for Method 2 (A User-space PCA)

Method 2 achieves a top RMSE of 1.4510 over all users and 1.2783 over users with 1000 ratings or more. This is a fair result but not particularly good either.

We suspected that the number of ratings a user makes would affect how good the predictions are for that user since having more ratings means more data. We tested this hypothesis by calculating the RMSE for users with various rating number thresholds. The results shown in the RMSE column of Table 3 proves the hypothesis wrong. The takeaway from this aspect is that the ratings from a user either do not heavily depend on the attributes of anime or do not depend on all the attributes equality. To test whether the latter is true, a separate run of Method 2 with each anime's average rating normalized to 5 (previously all attributes are normalized to 1) is executed. The results are shown in the rightmost column of Table 3, where we see a clear increasing trend of RMSE as the number of ratings per user decreases. This aspect justifies the hypothesis. The interpretation is that, a user's rating for an anime not only depends on the latter's genre, but is more affected by the anime's overall quality and popularity, which is an expected phenomenon.

# Ratings	# Users	RMSE	RMSE 5x Avg Rating
>= 1000	43	1.4368	1.2783
>= 500	730	1.4274	1.3188
>= 100	17223	1.3999	1.3717
>= 50	30438	1.4418	1.4247
>= 10	54004 (all)	1.4591	1.4510

Table 3: RMSE with different thresholds of numbers of ratings per user

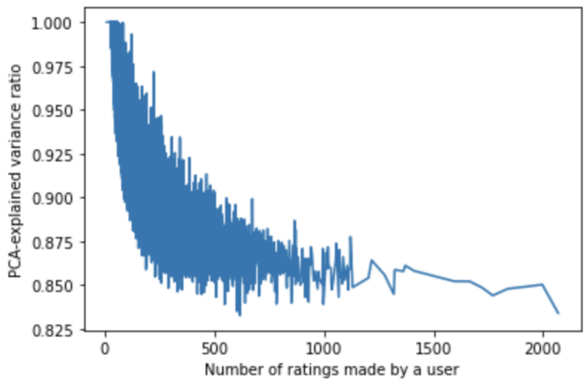

Figure 5: PCA-explained variance ratio for all users under Method 2

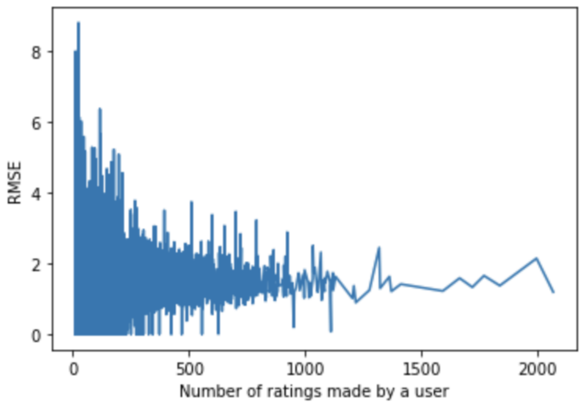

Figure 6: RMSE for all users under Method 2

Moreover, Figure 6 reveals that, although having more ratings does not mean better results, it does yield more stabilized results; on the other hand, results from users with few ratings fluctuate significantly. Also, Figure 5 shows that eigen user-spaces with less ratings have higher PCA-explained variance ratio. This is as expected because the dimension D_e of an eigen user-space is set to $min(0.8M_u, 20)$, which means the large M_u is, the smaller the ratio $\frac{D_e}{M_u}$, which results in larger differences in reconstructing the original space.

5.3 Results for Method 3 (Recommendation Based On K-means Clustering)

We visualize our clustering method with a set of pie charts that show the genre distributions in the original data for each cluster. This is shown as Figure 7. We see that some genres account for the majority in some clusters, showing proper functioning of our 3D K-means implementation. On the other hand, we do not see a genre that occupies the entirety of any cluster. This is as expected because each anime usually belongs to more than one genre.

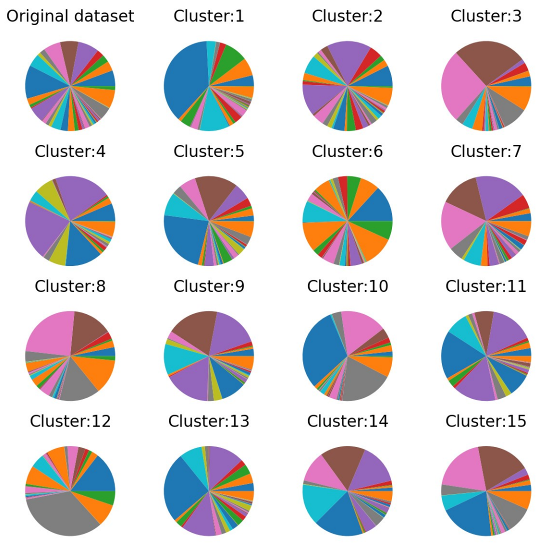

Figure 7: Genre distributions in resulting clusters

To test Method 3, we found several anime enthusiasts and asked each of them to rate about 10 anime. With their ratings, we use Method 3 to recommend anime to them. According to their feedback, the anime recommended fit the genre they liked. Therefore, Method 3 achieves the goal of anime recommendation based on user history. However, since the algorithm does not account for each anime's overall rating, it occasionally recommends anime that fit the genre requirement but are relatively unpopular.

6 Discussion

From the results of Method 1 and 2, we learned that PCA is useful for the recommendation system, but the final score is not very promising, indicating that PCA alone is not enough for making highly accurate recommendations to the users. We have explored the approach with a combination of PCA and clustering in Method 3. With more time, we would integrate Method 1 and 3 to create a more robust system.

From the results of Method 2, we see that the ratings from a user do not depend on all the attributes equality, which means there is a lot of space for improvement if we could individually tune the weight of each attribute. Due to limited time and computational resources, we only briefly explored this aspect, but we expect to see great improvements if we are able to accomplish such a task. Likewise, in the case of Method 1, an extensive grid search could improve the performance. Since we currently tune the parameters individually, we are not sure whether these parameters are the best for our model.

Tuning the weights of attributes is important not only for PCA, but also for clustering. During the process of using K-means and EM, we found that, by giving different weights to attributes, the system creates vastly different recommendations. One way to improve this system is to wrap it in a supervised-learning setup such that wrong weights would result in negative feed backs, and vice verse. Otherwise, we could gather small data sets from more actual people like ourselves, produce recommendations for them, and obtain feedback from those people. The former is a more efficient way to start the tuning process whereas the latter is more suited for subsequent minor adjustments.

7 Contributions

Derek Chen: data cleaning, introduction & motivation, related work, discussion

Xianglong Wang: Method 3 recommendation algorithm and Method 3 results section

Chen Yang: Method 3 K-means clustering implementation and visualization

Xiaoshuo Yao: Method 3 PCA and data visualization, Method 3 method

Fei Yin: data cleaning, all Method 2 related contents, discussion, video editing

Futian Zhang: data cleaning, all Method 1 related contents

8 Code

`https://github.com/FeiYin99/CCXXXIII---COGS-118B-Final-Project.git`

References

[1] CooperUnion. "Anime Recommendations Database". In: (2017). URL: `https://www.kaggle.com/CooperUnion/anime-recommendations-database`.

[2] F.O. Isinkaye, Y.O. Folajimi, and B.A. Ojokoh. "Recommendation systems: Principles, methods and evaluation". In: *Egyptian Informatics Journal* 16.3 (2015), pp. 261–273. ISSN: 1110-8665. DOI: `https://doi.org/10.1016/j.eij.2015.06.005`. URL: `https://www.sciencedirect.com/science/article/pii/S1110866515000341`.

[3] Zan Wang et al. "An improved collaborative movie recommendation system using computational intelligence". In: *Journal of Visual Languages & Computing* 25.6 (2014). Distributed Multimedia Systems DMS2014 Part I, pp. 667–675. ISSN: 1045-926X. DOI: `https://doi.org/10.1016/j.jvlc.2014.09.011`. URL: `https://www.sciencedirect.com/science/article/pii/S1045926X14000901`.