Project-II by Group Rome

Cimen Gokcen
EPFL
gokcen.cimen@epfl.ch

Angelopoulos Vasileios
EPFL
vasileios.angelopoulos@epfl.ch

Petrescu Viviana
EPFL
viviana.petrescu@epfl.ch

Abstract

This report present

1 Music Recommendation

1.1 Music recommendation

Collaborative filtering is the part of recommender systems that predicts users' preferences for particular items. The major challenge of these methods in predicting users'listening count is that the available user-artist count are usually too few and their performance relies on a good initial estimation of the unknown ratings. This is why we implemented ALS which uses only the known listetning counts and avoids dependencay on initial estimations.

1.2 Data description

The training data consists in a matrix $Ytrain$ of size 1774x15082, corresponding to 1774 users and 15802 artists. Element $Ytrain(i, j)$ expresses how many times user i has listened to the artist j. If the value is 0, it means we do not have information for that (user, artist, count) triple.

We are also given the friendship graph of the 1774 users storead as an adjacency matrix.

1.3 Exploratory Data Analysis

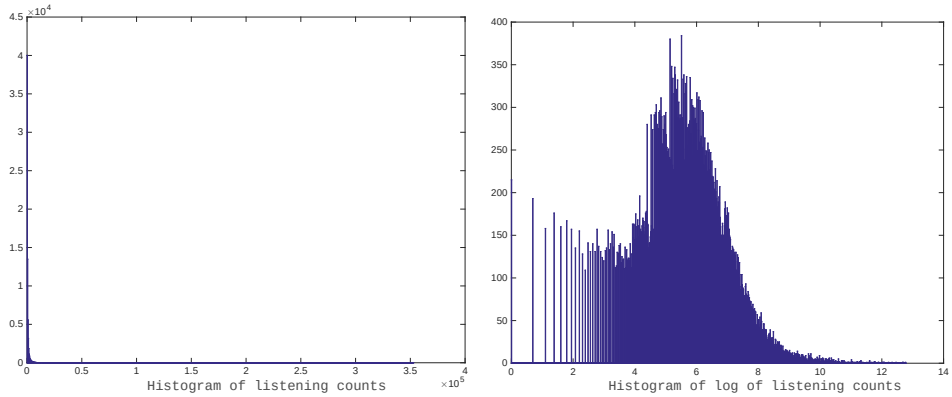
The listening counts matrix is very sparse with a density of only 0.0026%. The highest count is 352698, the average listening count per user is 5.52 and the average listening count per artist is 5.46.

A histogram of counts, shown in Fig1 tells us that the listening counts follow a heavy tail distribution. The long tail contains a small number of popular items, the well-known artists, and the rest are located in the heavy tail. One method to transform skewed data such that it becomes more gaussian distributed is to use the Box-Cox transform. In our case, we can choose $lamda = 0$ since for us y values are very high and ¿ 0. This transformation will make the distances between listening counts much smaller and will reduce the influence on RMSE of the (user,artist,count) triples in the long tail.

1262 artists had 0 listening counts.

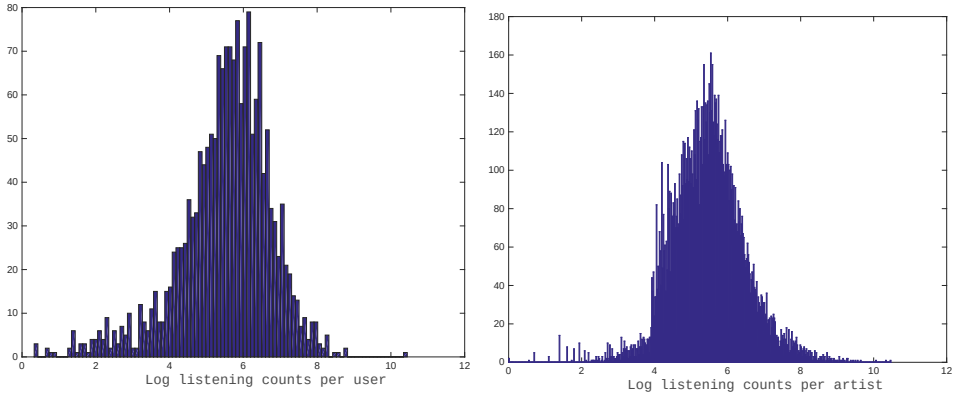
1.4 Task 1

20 per cent of the entries in Ytrain were kept for testing and the rest of the entries were kept for training and validation using 10-fold cross validation. Splitting the data was more difficult since we needed to make sure we do not remove completely the counts of an user or the counts of an artist.



(a) Before data transformation (b) Negative image.

Figure 1: After data transformation



(a) (b)

Figure 2: Per user and per artist log of listenining count distribution

1.4.1 KNN

1.4.2 ALS

1.4.3 logALS

1.4.4 Kmeans

1.5 Task 2

1.5.1 Friendship information

Regarding the friendshiop information, we noticed there are 22 connected components, but many of them contained. 1776 nodes and 22904 edges. Using Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre, Fast unfolding of communities in large networks, in Journal of Statistical Mechanics: Theory and Experiment 2008 (10), P1000 and the Gephi tool to find properties of the graph, it suggested it has about 32 communities, of which 8 had more than 150 members and the others being very small. (22 connected components, of which 20 are very small ¡ 50 users.) This was run to give us an idea of the number of clusters we could use in Kmeans.

1.6 Summary

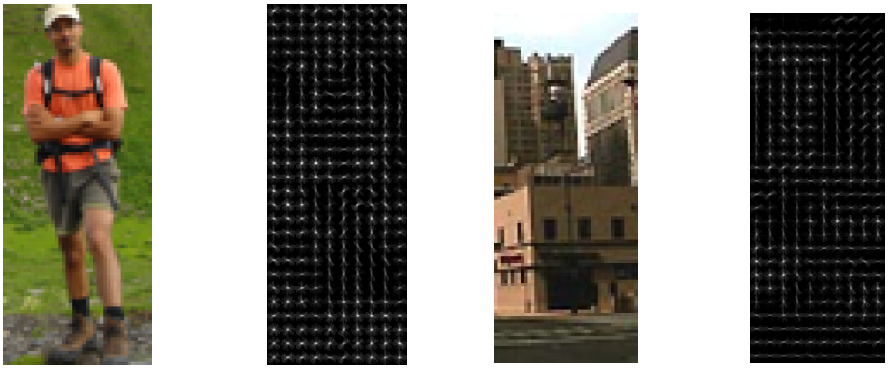
2 People Detection

2.1 Data description and visualization

The original training data $imgs$ contains 8545 color images with size 105x43. From every image a HOG descriptor with size 26x10x36 was extracted. All of these descriptors are saved at $feats$ and form the training data that we will work on. To process this HOG features we have converted them into a vector of total length 9360.

We notice that there are 7308 images without presence of people (negative images) and 1237 with (positive images). Figure 3 presents a typical example from each category accompanied with the relevant feature descriptor.

Our task is to train various classifiers, so that we will be able to detect the presence of people in unknown cases. For these purpose we evaluate these classifiers, measuring the accuracy of their estimations using Receiver Operating Characteristics (ROC) curves.



(a) Positive image. (b) Negative image.

Figure 3: Examples of positive and negative images accompanied with their feature descriptors.

2.2 Data preprocessing and used methods

Working on the extracted features, we split the data in order to perform our experiments. In our case we used K-fold validation with K = 5 to evaluate the performance of each classifier. Next step is to normalize the data so they will be centered with 0 mean and standard deviation 1. Here, we have to mention that we assume that there are no outliers, since the images are manually annotated and they have only two possible values for the classes.

The methods that were tested throughout this experiment are the naive Bayes classifier, the logistic regression and its penalized version, the Support Vector Machine (SVM), the K-NN classifier, and the Neural Network classifiers. All of them are compared also with the random guess of the class in the last subsection of this task.

Moreover, we have to say that at the beginning of the experiments we developed and tried PC Analysis, but, except the huge computational time that was necessary to perform it, it did not give almost any improvements to the dimensionality of our data.

2.3 Naive Bayes classifier

In this part we tried to fit a simple Bayes classifier to our data. The assumptions that we made were that we have data that follow the normal distribution and the estimates of the prior probabilities are empirical and were made from the relative frequencies of the classes in training.

The results from the K-fold validation were not very satisfying, since the average TPR was very low (0.031). The reason that may be hidden behind that are the strong assumptions that we made in order to fit this model. Naive Bayes model assumes that the data that will process every time follow a specific distribution (in our case Gaussian), but also that the value of a particular feature is unrelated to the presence or absence of any other feature, given the variable of the class. In our case, this is possibly wrong, so the classifier cannot fit a appropriate model.

2.4 Logistic and Penalized Logistic Regression

Our purpose in this part was to fit the logistic regression and its penalized version to our data. The value for the parameter α that we used was 10^{-2}, since all of the values close to this performed well enough for our experiments. Moreover, regarding the value for λ in case of the penalized logistic regression, we have to mention that all of the values that were used performed the same. This is the reason that both of the method give the same results, with average TPR equal to 0.754.

During penalization the idea is to avoid the overfitting by imposing penalty on large fluctuations of the model parameters and reduce the influence of the outliers to our model. This is possibly the reason that both models perform equivalently. The data, as we mentioned before, do not have outliers and the fluctuations at the model parameters are avoided by the nature of our data.

2.5 Support Vector Machines

For the certain part of the experiment we used the LIBSVM 3.20 toolbox that includes SVM using various kernels. In our case, we performed analysis and K-fold validation on linear-kernel SVM, polynomial-kernel SVM with degrees 2 and 3, real-basis-kernel SVM and sigmoid-kernel SVM. In all of the cases we used the scores that were given by the program to compute the TPR and the FPR. A comparison of the various cases is given in Figure 4a.

As we can observe radial-basis-kernel SVM have significantly good performance. A fact that can be seen not only by the average TPR, which is equal to 0.893, but also observing the whole range of the FPR, where the classifier gives better results than all of the other SVM cases. The second best in performance is the polynomial SVM with degree 2, which has very similar results with the RBF case for FPR greater than $210e{-}3$. The rest of the classifier perform well for FPR greater than $210e{-}2$, with the exception of the linear case, which has quite good results for lower rates, too.

Moreover, in Table 1 we present the average percentage of the successful classifications that were made during the K-fold validation, as they are given by the program.

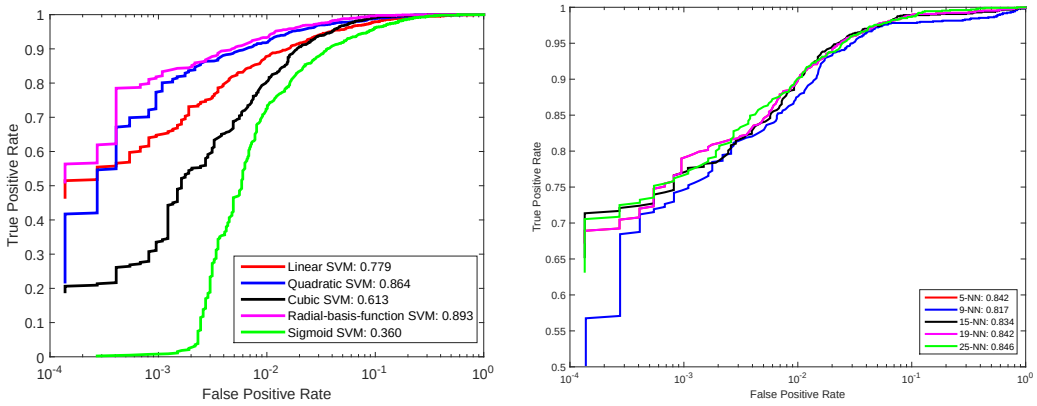
SVM kernel	Linear	Quadratic	Cubic	RBF	Sigmoid
Average success rate (%)	94.81	97.22	91.45	98.43	84.68

Table 1: Average percentage for the success rate for the various kernels of the SVM classifier, as they are given by the program of the LIBSVM 3.20 toolbox.

2.6 K-NN classifier

Using the K-NN classifier the most important parameter that we have to specify is the number of the neighbouring samples that will be used as a reference for the method so it will decide for any given sample the class that it belongs to. For our experiments we used a ready function, which we modified a lot in order to compute and give in output the scores of the prediction, except of the class labels. We varied the number of neighbours used among 5, 9, 15, 19 and 25. In addition, the euclidean distances were used as the way to specify the distances among the neighbouring samples.

Figure 4b presents the resulting ROC curves that were created using the scores got from the K-fold validation process. There we observe that all of the classifiers show very good performance, especially, for FPR greater than 10^{-3}. Only the 9-NN classifier seems to perform a bit worse for FPR between 10^{-4} and 10^{-3}. These observations can be explained by the distribution of the data, where most probably they are distributed in the N-dimensional (N = 9360) space with such a way that their neighbours can specify, with high probability, the class that they belong to.



(a) (b)

Figure 4: ROC curves created using: (a) SVM with various kernel types (linear, quadratic, cubic, real-basis-function, sigmoid), (b) K-NN classifier for various number of neighbours for the computations (5, 9, 15, 19, 25).

2.7 Neural Network

2.8 Convolutional Neural Network

2.9 Comparison

Acknowledgments

We would like to thank the course teaching assistants that helped us a lot during this project preparation, not only during the exercise session, but also at their office hours. Furthermore, the help that our teacher Mohammad Emtiyaz Khan offered us was very important, and we would want to thank him, too. All of the code used and submitted along with this report was written by equally by all of our group members.