Hierarchical Poisson Factorization for Movie Recommendation Systems

Dataset

The data used for this project was extracted from “The Movies Dataset”, from the Kaggle Platform. (The Movies Dataset (kaggle.com)). It provides extensive information about a variety of movies, including their Title, Series, if that’s the case, Genres, Budgets and the score on the well known IMDB platform.

Another very important feature in the dataset is the list of interactions between a set of over 270.000 users and 170.000 movies. Each interaction is represented by a rating, between 1 and 5, with a step of 0.5, that the user gave the movie after watching it. We also have access to the timestamp of the rating, but it is not really relevant for this task. For simplicity, we doubled all the ratings so they can be natural, and halved them for metrics computing.

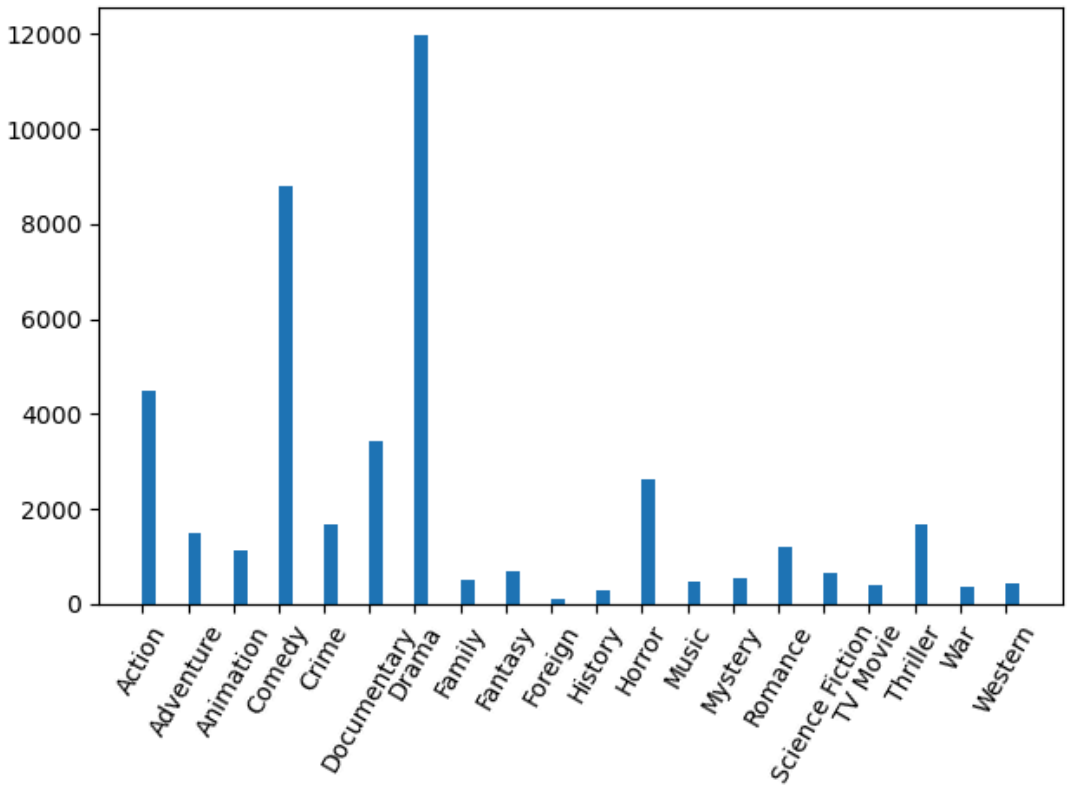
Sample extraction

Since the used models require a high computational and memory cost (which I, unfortunately, do not have access to), representative subsamples of the dataset were chosen to, ideally, replicate the distribution of the whole dataset on a smaller amount of data. Two such datasets were chosen, one serving as a “**Sanity Check**” for routine checking and hyperparameter testing with a low computing time and one serving as a “**Main Dataset**”, in which the true performance of the model is evaluated.
The dimensions of the datasets are the following:

	Number of Users	Number of Movies	Number of Reviews
Sanity Check	20	20	184
Main Dataset	200	200	18320

Random sampling of the data, however, is not enough to capture the true distribution of the main dataset, so, the **Stratified Sampling** technique was used, a technique primarily used for polls.

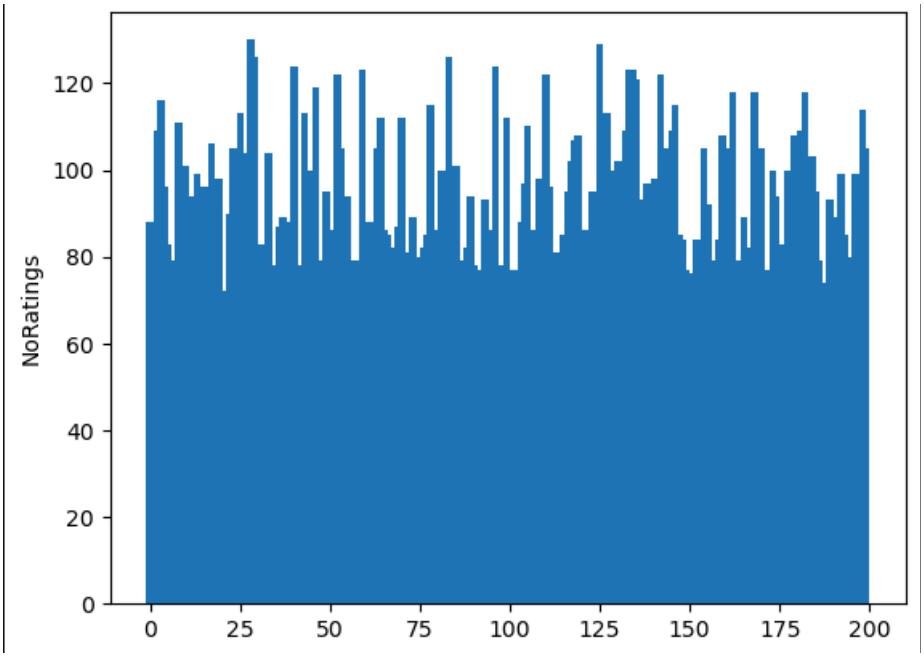
First, the desired number of movies were chosen according to the proportion of their main Genre (the first one that appears in the genre set) in the full dataset. Ignoring very small values, which are insignificant (“Carousel Productions”, “Aniplex” and “Odyssey Media”), the distribution based on the main genre is the following.



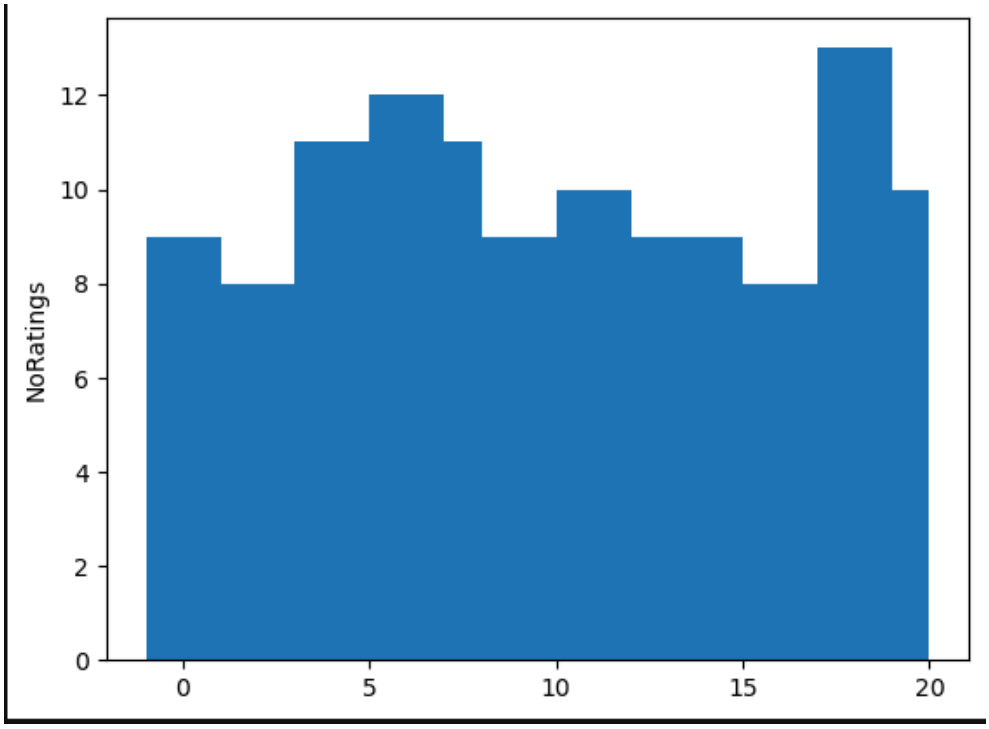
In order to replicate the distribution, the number of movies for each genre was chosen proportional to the main distribution. Thus, the number of movies which have the main genre as “g”, was **Ng = f * N**, where f is the percentage of movies with g as their main genre in the full dataset, and N is the number of movies in the sampled dataset.

Once the number of movies for each genre was set, a random sampling with these imposed limits could be easily done.
After sampling the movies, the right users had to be chosen. A good dataset for the approached tasks is one in which there are no users biased towards a small subset of movies. The tasks would be, then, better approached using a linear model.
To avoid such a situation, the chosen users had to have watched between 35% and 65% of the chosen movies. The number of user-movie relationships is, then, dense enough to gasp the models’ true capacities. To sample the desired number of users, we choose at each step, the user that adds the most new movies to the set of the ones already viewed.

The distribution on the number of interactions per user in the Main Dataset is the following:



And for the sanity check:



Thus, no watched movie can “give away” directly some users’ other preferences.

Approached tasks

I. Binary Recommendation System

The main task that was approached was recommending accurate movies for a user, based on its previous interactions. For this task, positive examples were labeled with 1 and signified an interaction between a user and a movie. For negative sampling, random interactions were chosen from the missing values in the interaction matrix, as described in the support article. As such, a number equal to 20% of the training data of negative samples were randomly chosen and appended to the training set.

The chosen train-test balance was 80%-20%, and the training set was also chosen using Stratified Sampling. 80% of each user’s interactions were chosen for the training set, the rest being left for the test set.

The chosen models predict a score associated with each (user, movie) pairing. Our predictor architecture returns the 5 maximum scoring movies for a certain user.

The metrics used to evaluate this model were **Mean Precision-at-K and Mean Normalized Recall-at-K.** In this case, K = 5.

The **Mean Precision** computes how many of our predictions were accurate, while the **Mean Recall** computes how many of the accurate predictions we have seen.

As our datasets were sampled, each user will have at least K=5 valid predictions in the test data, however that can be more so in the computation of the recall, we only consider the first 5 variants. This is how many online streaming services work, only being interested in providing the best matches for a user, rather than some good, but not the best, predictions.

They are similar to normal precision and recall metrics, but they also take into account the number of predicted values.

Thus:

$$MP@5 = \frac{\sum_{u} \frac{Correct_u}{5}}{N_{users}}$$

$$MR@5 = \frac{\sum_{u} \frac{Correct_u(first\ 5)}{min(5, len(GroundTruth))}}{N_{users}}$$

II. Rating Regression System

For this task, we use the train data to estimate the rating that a user would give to a certain movie. As this is a regression task, we make use of the actual ratings in the training set. An unwatched movie is represented by a value of 0, signifying the movie was not interesting enough for the user to even consider watching it.

The metrics used were **MAE**, **MSE** and **RMSE**.

$$\text{MAE} = \frac{\sum_u |y_u - p_u|}{N}$$

$$\text{MSE} = \frac{\sum_u (y_u - p_u)^2}{N}$$

$$\text{RMSE} = \sqrt{MSE}$$

The MSE metric tends to penalize more predictions that have a very high difference to the correct answer, while MAE penalizes misclassified examples more equally.

Models used.

I. Hierarchical Poisson Factorization

Model architecture

An interesting probabilistic approach for recommendation systems. This model makes use of probability distributions to assess the attributes of the users and movies involved and approximates their pattern.

Each user and each movie will get its own **latent vector of features**, similar to a usual machine learning model's embedding, which represents a **feature vector** that defines the significance of that user (or movie) as figured out from the train data.

Each of these features are sampled from a Gamma distribution, which is dependent on its hyperparameters (a', b', c', d', a, c). The embedding vectors all have the same fixed size, **k**, which can also be adjusted as a hyperparameter to increase or decrease its dimensionality.

The final estimation of the probability of interaction between a user u and a movie m is represented using a **Poisson** distribution with the mean represented by position u,m in the dot product of the matrices of features (for the users and for the movies respectively).

1. For each user u:
   (a) Sample activity $\xi_u \sim \text{Gamma}(a', a'/b')$.
   (b) For each component k, sample preference

$$\theta_{uk} \sim \text{Gamma}(a, \xi_u).$$

2. For each item i:
   (a) Sample popularity $\eta_i \sim \text{Gamma}(c', c'/d')$.
   (b) For each component k, sample attribute

$$\beta_{ik} \sim \text{Gamma}(c, \eta_i).$$

3. For each user u and item i, sample rating

$$y_{ui} \sim \text{Poisson}(\theta_u^\top \beta_i).$$

Prediction

In order to predict a user's interaction with a certain movie after training, we just have to measure the **Expected Value** of the **Poisson** distribution associated with that point. As of the definition of the Poisson distribution, the Expected Value is just the mean value in itself.

If we denote as L, the dot product of the theta and beta matrices, we can order the preferences of a certain user by decreasingly ordering the values of L[user]. We can then set the number of predictions we want to display.

In order to predict a numerical value for the score prediction, we just have to output **L[user][movie]** for the (user, movie) pair we want to predict.

II. Nonparametric Poisson Factorization

Model architecture

Similar to the previous model, this model also considers a vector of latent variables which signify an embedding of a user or an item respectively. The main difference arises from the assumption that latent features for users are not independent

between each other. The latent features represent a product of Gamma and Beta distributions rather than just Gamma ones.

The final score matrix is also represented by a Poisson distribution, so the prediction is similar to the previous model.

1. For each user $u = 1, \ldots, N$:
   (a) Draw $s_u \sim \text{Gamma}(\alpha, c)$.

   (b) Draw $v_{uk} \sim \text{Beta}(1, \alpha)$, for $k = 1, \ldots, \infty$.
   (c) Set $\theta_{uk} = s_u \cdot v_{uk} \prod_{i=1}^{k-1}(1 - v_{ui})$, for $k = 1, \ldots, \infty$.
2. Draw $\beta_{ik} \sim \text{Gamma}(a, b)$, for $k = 1, \ldots, \infty$, $i = 1, \ldots, M$.
3. Draw $y_{ui} \sim \text{Poisson}(\sum_{k=1}^{\infty} \theta_{uk}\beta_{ik})$, for $u = 1, \ldots, N$, $i = 1, \ldots, M$.

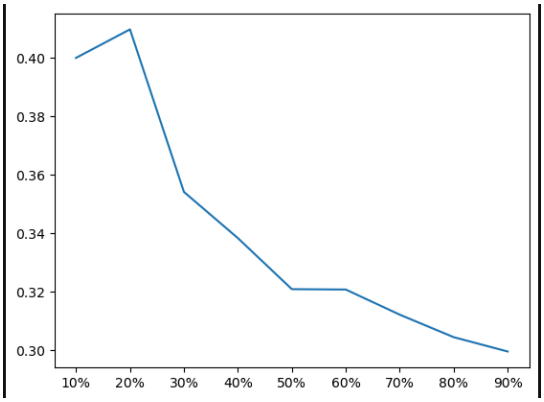
Experiments and Results

All the experiments were done using the **No U-Turn Sampler (NUTS)** from the pymc library.

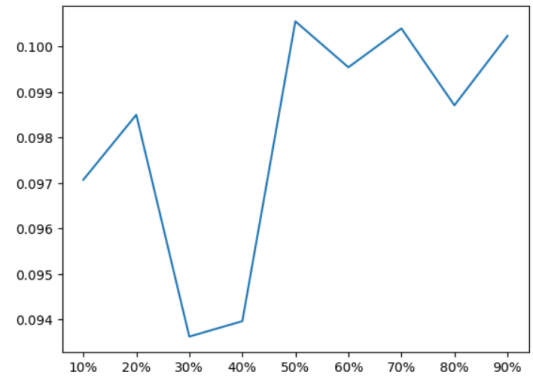
For the first task (binary):

Model	a'/c'	b'/d'	a/c	K	MP@5	MR@5
HPF	0.3	0.3	0.3	20	31.2%	10.902%
HPF	0.1	0.2	0.1	20	28.5%	10.476%
HPF	0.2	0.1	0.2	20	29.3%	10.615%

Similar to the paper, the graphs were computed for percentages of users, relative to the frequency of their activities.



P@5



R@5

Model	a’	a	b	c	K	MP@5	MR@5
NPF	0.3	0.3	0.3	0.3	10	13.99 %	9.76%
NPF	0.1	0.1	0.1	0.1	10	13.42 %	8.88%
NPF	0.2	0.2	0.2	0.2	10	13.68 %	9.12%

For the second task (regression):

Model	a’/c’	b’/d’	a/c	K	MAE	MSE	RMSE
HPF	0.3	0.3	0.3	10	0.838	1.143	1.069
HPF	0.1	0.2	0.1	10	0.921	1.203	1.096
HPF	0.2	0.1	0.2	10	0.934	1.187	1.089

Conclusions

Hierarchical Poisson Factorization is a very powerful model that manages to learn from sparse data such as a table of user - item interactions. It is based on

probabilistic distributions so it is not as prone to overfitting on small datasets as other machine learning models.