Gaussian Process Regression for Movie Recommendations

Group 7
National University of Singapore

Abstract

In recent years, online streaming and video on-demand services have grown exponentially. The online nature of such services necessarily multiplies the number of products that can be shown to the consumer. Consequently, there is a need for such services to determine a selection which should be recommended. In order to produce accurate predictions, there is a need to consider users' preferences, item profiles and correlations between items and users. In this paper, we attempt to apply Gaussian Process (GP) regression on movie ratings to generate recommendations.

1 Introduction

Today, it is easy to obtain online information pertaining to a movie and its critics through a search on Google, IMDB, Rotten Tomatoes, etc. Hence, modern day movie watchers would often research online about a movie before buying tickets in their local cinema or paying for online streams on movie streaming platforms such as Netflix, Amazon Video, HBO, or Hulu.

The rise of on-demand video services requires an accurate movie recommendation system in order to help users purchase the movies they like. As movie-watching shifts into the online world, consumers are faced with more choices as compared to the traditional way of browsing limited movies available in physical dvd rental stores or the theatre. Within the online ecosystem, services like Netflix are faced with the double-edged sword of being able to provide virtually any movie in existence to the consumer. Letting the consumer conduct an exhaustive search on this massive search space is certainly a computational burden that we would like to avoid.

2 Motivating Application

Moreover, motivating reasons such as word of the mouth, reviews and advertising are insufficient in the context of digitized movies. To cater to the consumer who faces the paradox of choice, there is a need for quality recommender systems which can enable movie watchers to discover movies they love on their own.

Our application firstly exploits the predictive mean and uncertainty provided by the Gaussian process model. In particular, it is important for our application to obtain a ranking of movie that we should recommend. The Gaussian process model is particularly well-suited in this context as compared to traditional regression models. This is because our selection can be made on the basis of the predictive uncertainty that the Gaussian process model usefully provides. For example, we might obtain similarly high ratings for 2 movies, A and B. If the predictive uncertainty given by the Gaussian process model is such that $\sigma_A^2 > \sigma_B^2$, then we are more likely to recommend movie B, or rank it higher on our recommendation list. Additionally, it is possible to use multiple models and weight our predictions inversely proportional to the variance. As a consequence, users will benefit by deriving greater enjoyment.

Moreover, movie preference is highly personal and complex to model. Director, script, cast, score, visual effects and genres are just of some obvious features of a movie. To give a trivial example, an individual might not like mainstream A-listers action movies such as Fast & Furious, Bourne or James Bond, but he may like Marvel action movies because he likes superhero comics. Even then, he might not like the newer Spiderman movies because of the main lead. He might love DC Comic Batman trilogy directed by Christopher Nolan but still be disappointed by Batman v Superman: Dawn of Justice. The usage of Gaussian process models in this context is particularly helpful in enabling us to exploit the correlations between users with similar preferences. Since users with similar tastes are likely to give similar ratings to a specific movie, we can make use of these correlations to recommend movies.

It is clear that modelling human behaviour and preference is complex. Using a parametric model could unintentionally constrain our model. Since the Gaussian process model is a distribution over an infinite number of functions, we are better able to model the full spectrum of highly variable human preferences by exploiting the non-parametric characteristic of Gaussian processes.

Finally, let us suppose that our Gaussian process model can generate relatively accurate recommendations. With increased user satisfaction and hence higher utilisation of the system, we are likely to obtain even more data from new ratings. This will allow us to further exploit the correlations

among user preferences, as well as the Bayesian nature of Gaussian process models by updating the model with new data.

In conclusion, to tackle the problem of movie recommendation, we propose a Gaussian process (GP) model. Given a list of movies that an individual has not rated before, a GP model will compute a predictive mean rating that the individual is likely to give and a predictive uncertainty which gives us insight on how accurate our prediction is. These can then be made use of to generate movie recommendations.

3 Technical Approach

A Gaussian process is a collection of random variables, any finite subset of which have a multivariate Gaussian distribution. It is completely specified by a mean function $\mu(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$. For a real process $f(\mathbf{x})$:

$$\mu(\mathbf{x}) = E[f(\mathbf{x})]$$
$$k(\mathbf{x}, \mathbf{x}') = E[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}')]$$

The GP can then be denoted as:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$

We assume that our data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_i, y_i)\}$ are such that y_i are noisy observations originating from a GP-distributed random function $f(\mathbf{x}_i)$ such that:

$$y_i = f(\mathbf{x}_i) + \epsilon_i$$
$$\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$$

Given $\mathbf{y} = [y_1 y_2 \ldots y_i]^\top$, suppose we have a new input $\mathbf{x}_*$ for which we would like to obtain a prediction for. In other words, we would like to infer $p(f_* \mid \mathbf{y})$. Then, the predictive mean and variance from the GP can be given by:

$$E[f_* \mid \mathbf{y}] = \mu(\mathbf{x}_*) + \mathbf{k}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} (\mathbf{y} - \boldsymbol{\mu})$$
$$V[f_* \mid y] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_*$$

Problem Definition

Traditionally, there are 2 approaches to implementing recommender systems:

1. Content-based systems make recommendations based on item attributes. For example, a user who tends to watch many movies of the horror genre will be recommended a movie from that genre.
2. Collaborative filtering systems analyse the similarities between users and/or items to make recommendations. Users will be recommended items that are preferred by users with similar tastes.

The task is thus rating prediction, given item and user attributes. In particular, suppose we are given a set of movies, $\mathcal{M}$, and a set of users, $\mathcal{U}$. We formulate an input matrix X based on 2 types of models, per-user and per-movie, to predict user ratings $y = [y_1, y_2, \ldots, y_n]^\top$, where X is given by

$$X = \begin{bmatrix} x_1(1) & x_2(1) & \cdots & x_d(1) \\ \vdots & & \ddots & \vdots \\ x_1(N) & x_2(N) & \cdots & x_d(N) \end{bmatrix}$$

Model Definition

In this paper, we assume that user ratings follow a Gaussian distribution. We formulate two different types of models, each exploiting a different set of features:

1. **Per-user:** The per-user model can be viewed as follows – imagine the entire universe of movies, $\mathcal{M}$ – the known ratings by this particular user consist of a subset of movies, $\mathcal{M}_i$. By utilising the item profiles of these $|\mathcal{M}_i|$ movies, the Gaussian process predicts the ratings for the unknown subset of movies, $\mathcal{M}_u$, where $\mathcal{M}_i \cup \mathcal{M}_u = \mathcal{M}$. The item profile consists of movie attributes such as year of release and average critic's rating.
2. **Per-movie:** The per-movie can be viewed using a similar analogy, but flipping the roles of users and items. Given the user profiles of all known user ratings for a specific movie, we predict the ratings that the remaining users are likely to give, based on the correlations between the user profiles. The user profile consists of user attributes such as age and gender.

Here is an example of how a prediction is made from these two models. Suppose Steve is a user in the system and he has provided ratings for a few movies. From these ratings provided by Steve, we train a per-user model for Steve. In addition we provide these ratings to their respective per-movie model so as to further train the models. One movie that Steve has not rated is Jurassic Park and we are interested in predicting his rating for that movie. To do so, we first provide Steve's per-user gp model features from the movie Jurassic Park (1993, Action and Thriller) which then returns a prediction for the rating. Likewise, we provide the Jurassic Park's per movie-model with Steve's features (age, gender) and obtain another prediction for the rating. Finally, we take a weighted average of the prediction to obtain the final prediction. The approach for the weighted average will be explained under experimental evaluation.

The reason for having these two models is so that we can exploit both personal and demographic preferences. The per-user model tells us a user's personal preference for certain movie features such as genre or year of release whearas the per-movie model tells us demographic preferences for a particular movie. Another advantage of having the prediction provide by two models is that we can still make reasonable predictions when data is sparse. A new user would not provide many ratings on the system but we can give more weights to the per-movie model's prediction. Likewise for a new movie, we give more weights to the per-user model's prediction.

Choosing Model Parameters

The GP function is mainly characterized by its covariance function after normalizing the data to attain a mean of 0. The covariance function produces a covariance matrix which is utilised by the Gaussian process model for inference. Our choice of kernel for both experimental models, per user and per movie, was the radial basis function(RBF) kernel

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_i - x_i')^2}{\ell_i^2})$$

where the hyperparameters σ^2 is the variance, ℓ is the lengthscale, d is the dimension of the input vector x_i.

The choice of the rbf kernel was due to its ability to produce similar predictive ratings for two inputs with similar features based on their distance. If two users have contrasting preferences, it is highly unlikely that both will assign the same rating to their opposing's favorite movie.

Learning the hyper-parameters

Lemma: The product of two Gaussians gives another (unnormalized) Gaussian.

The log marginal likelihood of the gaussian process can be computed by marginalizing the likelihood with respect to function **f** and by using the lemma we obtain

$$\log p(\mathbf{y}|X,\theta) = -\frac{1}{2}\mathbf{y}^T(K+\sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log|K+\sigma_n^2 I| - \frac{n}{2}\log 2\pi$$

where the $\theta = \{\ell, \sigma_n^2, ...\}$ are the hyper-parameters which can be tuned by Markov Chain Monte Carlo methods or through Bayesian Optimisation (Neal 1997; Sundararajan and Keerthi 2001). Learning the hyperparameters is possible as the log likelihood partial can be derived. Letting $\mathcal{Q}$ to be $K + \sigma_n^2 I$,

$$\frac{\partial}{\partial\theta}\log p(\mathbf{y}|X,\theta) = -\frac{1}{2}\mathrm{tr}(\mathcal{Q}^{-1}\frac{\partial\mathcal{Q}}{\partial\theta_i}) + \frac{1}{2}\mathbf{y}^T\mathcal{Q}^{-1}\frac{\partial\mathcal{Q}}{\partial\theta_i}\mathcal{Q}^{-1}\mathbf{y}$$

We used a python library GPy to optimize our per user and per movie models which implements the bayesian optimization framework for maximizing the likelihood function. Through the batch gradient descent method, the maximum aposteriori(MAP) estimation gives us the maximum likelihood(Rasmussen, 1997).

This feedback to us critical information on the which are the important features in the per-user model through the Automatic Relevance Determination(ARD) represented by the lengthscale ℓ. Suppose ℓ is short for the per user model input vector dimension "genre", then this suggests that "genre" is an important feature for predicting the rating.

4 Experimental Setup

Dataset

Our primary dataset is the MovieLens[1] 1M Dataset which is a stable benchmark dataset widely used for evaluating recommender systems. The Dataset consists of three files, namely "ratings.dat", "movie.dat" and "user.dat" which contains around 1,000,000 ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000. Each rating is accompanied by a movie_id and a user_id so that we can map the rating to the respective user and movie.

Besides ratings, the dataset also provides movie attributes. The following attributes were extracted from "movie.dat":

movie_id movie identification number

title movie title

genres a list of genres that the movie belongs to

We also utilised user demographic information such as age, gender and occupation available from "user.dat":

user_id user identification number

gender user's gender

age user's age

occupation user's occupation

To supplement this dataset, we combined it with the MovieLens+IMDb/Rotten Tomatoes Dataset released by HetRec 2011. Our motivation for doing so was to obtain movie critic ratings from Rotten Tomatoes. As the project progressed, we also extracted the relevant directors and actors for feature engineering purposes. The notable features extracted from this dataset include:

AllCriticsRating average critics' rating for the movie

AudienceRating average audience rating for the movie

CriticsNumRatings number of critics' ratings for the movie

AudienceNumRatings number of audience ratings for the movie

By combining all these datasets, we obtained a processed dataset consisting of user ratings augmented with user attributes and movie attributes.

Procedure

The pre-processed dataset contains both numerical and categorical features. We experimented on simple models based solely on numerical features as well as models that utilizes categorical features. We also experimented with different kernels such as the linear kernel and cosine kernel in addition to the RBF kernel. In order to test the performance of our models, the data was split using stratified sampling into train and test sets[2].

The per-movie model uses of the following numeric features: user age, user gender[3] .

The basic per-user model uses of the following numeric features: year of movie release, average critics and audience rating, number of critics and audience rating.

In the next section, we shall explain the different approaches taken to integrate non-numeric features into the models.

[1] MovieLens is a movie recommender system created by GroupLens Research.

[2] A 70-30 split was used. Stratification was done over the user id

[3] Encoded as Female: 0 Male: 1

Feature Engineering

From our dataset, we were only able to obtain 3 demographic features: age, gender and occupation. However, the dataset contains a much richer set of features, including title and genres. We were also able to obtain the directors and actors by merging the original dataset with the supplementary dataset from HetRec. To utilise these non-numeric features so that we could input them into the Gaussian process, we have to first transform them into numerical features.

One approach was to use one-hot encoding over the categorical features. We attempted this on the genre feature where a column is created for each of the genres. For each movie, the value is 1 if the movie belongs to that particular genre, 0 otherwise. However, this technique does not scale very well as the input dimension increases with the number of categorical classes. Since there are 20 movie genres in total, we sought to reduce the number of dimensions for the input matrix.

Through the use of Word2vec, we could transform the remaining text data into useful numerical features and reduce the dimensions of the input. Word2vec enables us to create word embeddings, or vector representations of words with a given corpus. Similar words are constructed such that they are close to each other within the vector space. We thus extracted the textual data from our datasets and processed them into a corpus suitable for training a Word2vec model on. The Word2vec model is then queried for vectors, $x = [x_1, \cdots, x_k]^\top$ representing each film. Additionally, we were able to obtain vectors representing each distinct genre present in the dataset. In training the model, it was found that a value of $k = 8$ produced reasonable results from a test of similarity between selected films and genres. The following table gives some of the cosine similarities between genre vectors most similar to each other:

Genre	Genre most similar to	Similarity
Drama	Crime	0.9132
Thriller	Mystery	0.9834
Animation	Children	0.9621
Sci-fi	Fantasy	0.8729

Another approach to reduce the number of dimensions is to select several genres as "buckets" and calculate the probability of a movie's genres given a bucket. This probability gives the intuition of how similar two genres are. For example, if a movie has the following 3 genres, adventure, animation, family, and the given bucket is comedy, we calculate the numerical feature as follows:

$$P(adventure, animation, family \mid comedy) = \max \begin{Bmatrix} P(adventure \mid comedy) \\ P(animation \mid comedy) \\ P(family \mid comedy) \end{Bmatrix}$$

Since the conditional independence assumption does not necessarily hold, we instead take the max of the individual probabilities. The individual probabilities are calculated based on their various counts in the training set. In addition, the probabilities are smoothed using Witten-Bell smoothing to ensure non-zero probabilities.

$$P(animation \mid comedy) = \frac{C(animation, comedy)}{C(comedy)}$$

Choosing from the top popular genres (based on their counts), we identify 9 buckets: drama, comedy, crime, action, thriller, horror, fantasy, family and animation. This probabilistic approach allows us to compare the distance between genres and buckets. It also reduces the dimension of the input vector significantly.

5 Experimental Evaluation

As mentioned, our approach involves combining the predictions from the per-user and per-movie models. We thus train a GP model for each of the 6,040 users and each of the 3,090 movies using the processed data. The number of ratings per user ranges from 18 to 2264 with a median of 94. The number of ratings per movie ranges from 1 to 3428 with a median of 135. Given that the maximum training samples per GP model is only in the thousands, there was no need to use any techniques for handling large datasets in GP model.

Evaluating the accuracy of the models is slightly more involved as we have to feed the right entry to the right model. Suppose we want to evaluate the accuracy of the per-user models using the test set. For each entry on the test set, we pass the entry to the per-user model that matches the entry's user_id and obtain the prediction for that entry. We do this for every entry on the test set and that gives us the full prediction for the test set. Likewise for per-movie models.

As mentioned, we experimented with various combinations of features and kernels. Below is a subset of the experimental results[4] for the different combinations: The Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for the models trained are given as follows:

Type	Features	Kernel	MAE	RMSE
Per-Movie	Numeric	RBF	0.7823	0.9785
	Numeric	Cosine	0.7823	0.9786
	Numeric	Linear	0.7823	0.9786
Per-User	Numeric	RBF	0.8128	1.019
	Word2vec Genres		0.8210	1.026
	Probabilistic		0.8204	1.027
	Word2vec Movies		0.8278	1.033

We observed that when predicting with just per-movie or per-user models by themselves, per-movie models out performs per-user models. This could be due to per-movie models having more training points on average.

Combining the models

Inverse-Variance Weighting

6 References

Note: max 6 pages

[4] Full result is provided in the appendix