Deep Learning Implementation of a Recommendation System for Restaurants

Nasreddine El Dehaibi
CS230 Spring 2019
Stanford University
Stanford, California
ndehaibi@stanford.edu

***Abstract*— The restaurant industry is on the precipice of revolutionizing itself given the wealth of information available from big data. The goal of this project is to build a recommendation system that leverages the power of deep learning to accurately recommend restaurants to users. The input to the algorithm is a collection of data for a given user and restaurant. I then use a neural network to output the predicted rating that a user would give to the restaurant. The data used in this still consists of 1161 samples of restaurant ratings, information about the restaurant, and information about the customer. I built a SoftMax classifier in Python using a three-layer neural network with TensorFlow and compared this to baseline model based on collaborative filtering and SVD. The baseline model generated an RMSE of 0.66 while the deep learning model generated an RMSE of 0.73. While the deep learning model did not perform better than the baseline, the results are still promising considering that the dataset is small and that the model is likely overfitting the training set. The results from this study encourage building deep learning networks using larger datasets**

Keywords—restaurants, deep learning, recommendation systems

I. Introduction

The restaurant industry is on the precipice of revolutionizing itself given the wealth of information available from big data. By merging information that restaurants already have with data from social media, reservation systems, review sites, and even weather reports, restaurants can acquire live information on sales, customers, staff performance, and competitors. This gives restaurants the ability to personalize customer experience in ways that were previously unimaginable. It also helps restaurants run a leaner operation, for example by optimizing food inventory on any given night [1].

Another area of interest for restaurants is using recommendation systems to drive demand growth. This is desirable for both the customers and the restaurants; customers can find the ideal dining experience that they are looking for and restaurants can benefit from an increased traffic growth to their establishments. Traditional recommendation systems are based on either collaborative filtering or content-based filtering that group together similar users or items based on user ratings to recommend different items to users [2]. These approaches are very popular for applications like movie recommendations, where a user's preference is relatively static and can be identified based on just ratings. For other applications however, such as restaurant recommendations, a user's preference can fluctuate much more on any given time or day. This makes it more challenging to build an effective recommendation system for this application.

A growing body of research on recommendation systems is enabling exciting opportunities that leverage the information from big data. Using approaches from deep learning, information on the users themselves can be considered such as ethnicity, marital status, preferred mode of transport, and so on. Similarly, data on restaurants can be considered such as type of cuisine, ambiance, parking availability, and so on. This level of granularity is similar to what humans use when recommending restaurants to each other and can help make recommendation systems for restaurants much more effective. The goal of this project is therefore to build a recommendation system that leverages the power of deep learning to accurately recommend restaurants to users. The input to the algorithm is a collection of data for a given user and restaurant (see section 3). I then use a neural network to output the predicted rating that a user would give to the restaurant.

II. Related Work

Gupta and Singh proposed a location-based recommendation system that considers both user's location and demographics to recommend restaurants [3]. Zhang et al. [4] use conditional random field and hidden Markov model to predict a user's next dining based on historical dining patterns, user profile, and restaurant features.

In addition to location, other data such as user ratings, reviews, and reservations were also studied to improve restaurant recommendation. Gao et al. [5] model the relationship between user ratings and restaurants with regression based on topic modeling of user reviews. Sun et al. [6] integrated multiple sources of information, i.e., users' tasks, their friends' preferences, and mobility patterns, into the matrix factorization framework for personalized restaurant recommendation. This is particularly clever as it factors in historical data from multiple sources and is arguably the state of the art. Fu et al. [7] proposed a generative probabilistic model to describe restaurants in in terms of location, customer attributes, and restaurant attributes.

Instead of explicit user ratings, Kuo et al. [8] relied on users' restaurant booking history to recommend restaurants.

Although recommender systems have been widely studied, most studies rely on collaborative filtering methods or on hybrid methods with classifier methods. Few studies have looked into deep learning in the context of recommendation systems as it is still a novel area of research. This paper aims to select a highly practical application, restaurant recommendations, to investigate the feasibility of deep learning recommendation systems for restaurants.

III. Dataset and Features

The dataset consists of restaurant data with consumer ratings spanning 9 different data files. It was created by Rafael Medellín and Juan Serna at the Department of Computer Science in the National Center for Research and Technological Development in Mexico [9].

The data from each of the 9 files is explained below:

- Customer-Restaurant-Rating
    - 1161 customer ratings for food, service, and overall on a scale of 0 to 2. Only overall ratings were considered in this study (rating_final.csv)
- Customers
    - Preferred cuisine for 330 customers (usercuisine.csv)
    - Preferred payment method for 177 customers (userpayment.csv)
    - User profile (location, smoker, religion, etc.) for 138 customers (userprofile.csv)
- Restaurants
    - Methods of payment accepted at 1314 restaurants (chefmozaccepts.csv)
    - Type of cuisine for 916 restaurants (chefmozcuisine.csv)
    - Business hours and days for 339 restaurants (chefmozhours4.csv)
    - Parking availability for 702 restaurants (chefmozparking.csv)
    - Location and other information (smoking section, dress code, ambience, etc.) for 130 restaurants (geoplaces2.csv)

I compiled the dataset into one file to allow quicker processing of the information. The file contains 1161 rows of customer ratings from a total of 138 different customers on 130 different restaurants. There is a total of 50 columns that include customer and restaurant IDs, ratings, and information on the corresponding customer and restaurant from the dataset. If information is not available for any given cell, there is a "?" in place of the cell. Figs. 1-3 show the distributions of the dataset.

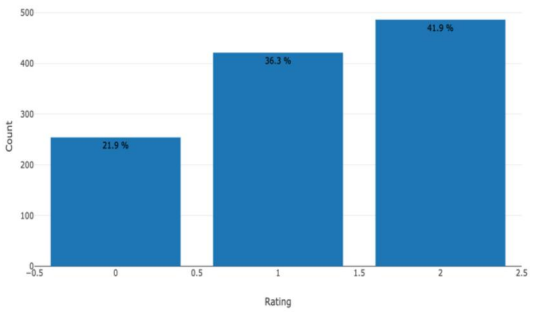

Figure 1: Distribution of 1161 restaurant ratings

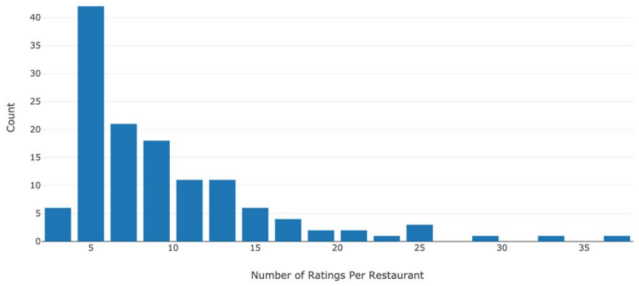

Figure 2: Distribution of Number of Ratings per Restaurant

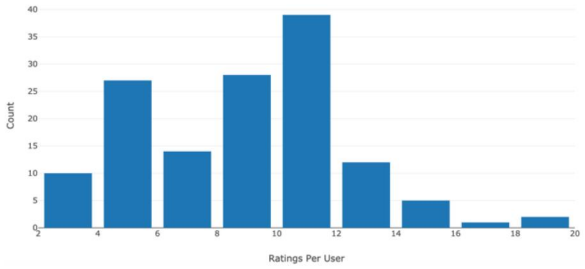

Figure 3: Distribution of number of ratings per user

Figure 1 shows the distribution of restaurant ratings on a scale of 0 to 2. There are a similar number of 1 and 2 ratings, but there are only about half as many 0 ratings. This might bias the models and is something to look out for. Figure 2 shows the number of ratings per restaurant. Most restaurants have around 5 ratings, while some have over 35 ratings. Having multiple ratings per restaurant helps the model capture more information from different users about each restaurant. Finally, Fig. 3 show shows the number of ratings per user and is a roughly normal distribution with most users giving about 11 ratings. This is also helpful for the model to learn about the variety of user preferences.

IV. Methods

The methods used consist of two parts, the first is identifying a baseline model, and the second is building a deep learning model. Both parts are discussed below.

A. Building a Baseline Model

There are a number of potential collaborative filtering-based baseline models from literature. I used the Surprise Scikit package in Python to build a number of these models using the following prediction algorithms [10]:

- SVD: Equivalent to probabilistic matrix factorization
- SVD++: An extension of SVD that considers implicit ratings
- Baseline Only: predicts the baseline estimate for a given user and item
- Co Clustering: a collaborative filtering algorithm based on co-clustering
- NMF: a collaborative filtering algorithm based on non-negative matrix factorization
- KNN Basic: a basic collaborative filtering algorithm
- KNN Baseline: basic collaborative filtering algorithm, taking into account a baseline rating
- KNN with Means: basic collaborative filtering algorithm, taking into account the mean ratings of each user
- KNN with Z Score: basic collaborative filtering algorithm, taking into account the z-score normalization of each user
- Slope One: a simple yet accurate collaborative filtering algorithm
- Normal Predictor: predicts a random rating based on the distribution of the training set

To test each model, I split the data into a 75% training set and 25% test set. I then trained each model using three-fold cross validation. I then tested each model on the test set using root mean square error (RMSE) as the error metric.

B. Building a Deep Learning Model

I built a SoftMax classifier in Python using a three-layer neural network with TensorFlow. I explain the steps I took below.

I first loaded the data into Python as a Pandas DataFrame. I then one hot encoded each column and split the data into a training, validation, and test set with an 80%, 10%, 10% split respectively. I chose this split considering the small dataset I was working with; I wanted to include as much data to the training set as possible while also having enough data to reliably test the model. With this split, there was 928 training samples, 117 validation samples, and 116 test samples. After splitting the data, I extracted an X and Y matrix for each of the training, validation, and test sets. The Y matrix included the one hot encoded restaurant ratings and had 3 columns, one for each of the possible ratings (on a scale of 0, 1 and 2). The X matrix consisted of the remaining one hot encoded columns totaling 1268 columns.

As I was using a TensorFlow implementation, I created placeholders for X and Y for the tensor flow session. I then initialized parameters with 100 weights using Xavier initialization and initialized bias vectors to zero. I then implemented a forward pass of the neural network using a standard model: linear, ReLu, linear, ReLu, linear, softmax. The equations for linear, ReLu, and softmax are shown in Eqs. 1 -3 respectively. After that I calculate the cost function using Eq. 4.

$$Z = Wx + b \tag{1}$$

$$\sigma(z) = \max(0, z) \tag{2}$$

$$Y(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}} \tag{3}$$

$$J = -\frac{1}{m}\sum_{i=1}^{m}\left(y^i log\sigma\left(z^{|2|(i)}\right) + (1 - y^i)\log\left(1 - \sigma\left(z^{|2|(i)}\right)\right)\right) \tag{4}$$

where m is the number of data samples, y^i is the label for a given sample, z is the output of a hidden layer, and σ is the activation function.

After computing the cost function, I then implemented backpropagation using Adam Optimizer. I chose this optimizer as it is generally effective and has the benefits of multiple other optimizers such as RMSProp. It updates learning rates based on the average first moments as well as the average second moments of the gradients, effectively creating an exponential moving average of the gradient and squared gradient. I selected a learning rate of 0.0001, mini batch size of 32, and 1500 number of epochs. I manually tweaked these numbers according to the performance of the model on the validation set.

I then tested the model on the test and calculated the RMSE between the predicted and actual ratings to allow for comparison of performance to the baseline.

V. Results and Discussion

The RMSE values for the different baseline models are shown in table 1.

Table 1: RMSE for Potential Baseline Models

Algorithm	test_rmse	fit_time	test_time
SVDpp	0.668294	0.109525	0.009100
SVD	0.679372	0.042878	0.004645
BaselineOnly	0.696977	0.000994	0.001838
KNNWithMeans	0.698772	0.002014	0.006794
KNNWithZScore	0.702664	0.006269	0.007673
CoClustering	0.720291	0.031317	0.002127
SlopeOne	0.736563	0.003677	0.004313
NMF	0.742113	0.047225	0.002896
KNNBaseline	0.759941	0.003650	0.006853
KNNBasic	0.849223	0.000602	0.004808
NormalPredictor	1.003145	0.000735	0.001739

The algorithm with the lowest RMSE score was the SVD++ algorithm which takes into account implicit ratings. The resulting RMSE from this prediction is 0.6712. The highest RMSE came from the Normal Predictor with an error of 1.00. This algorithm predicts a rating based on the normal distribution of the ratings. I decided to be ambitious and choose the lowest RMSE of 0.6712 as the baseline. It is likely however that this baseline will be challenging to beat considering the limited amount of data that the deep learning model can use.

Figure 4 shows the expected behavior of the cost function as a function of iteration number. The cost function exponentially decreases until about 100 iterations and then plateaus at around 0 as the network finds a minimum value. The training set accuracy was a perfect 1.0 while the validation dataset accuracy was about 0.598. This is reasonable considering the low number of data available in the datasets which makes it likely that the neural network model is overfitting the data.

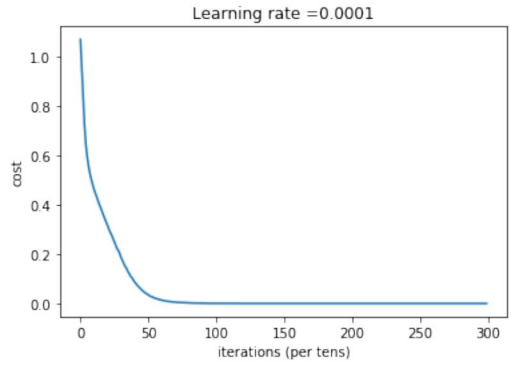

Figure 4: Cost Function versus number of iterations

Testing the model on the test set yielded an RMSE of 0.735. This is worse than the chosen SVD++ baseline that yielded 0.668. The deep learning model still did better than about half of the potential baseline models shown in table 1. This is a promising result since the dataset only had 1161 data samples and deep learning really shines when there are much larger datasets available. The likely cause of the limited performance is due to overfitting of the training data.

VI. Conclusions and Future Work

In this project I implemented a deep learning model for building a restaurant recommendation system. I used a dataset from with 1161 data samples of restaurant ratings, restaurant information, and user information. I first built several baseline models that based on collaborate filtering including SVD, KNN, and normal predictor. I chose the best performing model as the baseline which was the SVD++ with an RMSE of 0.66. I then built a SoftMax classifier in Python using a three-layer neural network with TensorFlow. I made an 80%, 10%, 10% split in the data for a training, validation, and test set respectively. The lowest RMSE was 0.73, which is not as good as the chosen baseline but is better than about half of the other baselines considered. This is a promising result since the dataset only had 1161 data samples and deep learning really shines when there are much larger datasets available. The likely cause of the limited performance is due to overfitting of the training data. For future work, the results from this study encourage building deep learning networks using larger datasets on the scale of millions. This will leverage the capabilities of deep learning and help personalize restaurant recommendations for users.

References

[1] https://www.nytimes.com/2017/08/25/dining/restaurant-software-analytics-data-mining.html

[2] https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed

[3] https://ieeexplore.ieee.org/abstract/document/6637223

[4] https://dl.acm.org/citation.cfm?id=2741095

[5] https://link.springer.com/chapter/10.1007/978-3-319-18123-3_33

[6] https://dl.acm.org/citation.cfm?id=2767818

[7] https://epubs.siam.org/doi/abs/10.1137/1.9781611973440.54

[8] https://link.springer.com/article/10.1007/s11280-017-0437-1

[9] https://www.kaggle.com/uciml/restaurant-data-with-consumer-ratings/version/1

[10] https://surprise.readthedocs.io/en/stable/index.html