Movie Recommendation Project

Al Hazwani Ibrahim
Matr. N. 18556

Programming for Data Analytics
M.Sc. Computational Data Science
Winter semester 2019

Contents

1 The problem

The purpose of the Kaggle competition named "Movie Recommendation Project" was about to recommend to 10 users 10 movies based on the previous ratings they gave.
The code of my solution is available on GitHub at this link `https://github.com/ibrahimalhazwani/Movie-Recommendation-Project`.
Once understood what was the problem I decide to review and studying the lectures done in class. For that I create a Jupyter Notebook in which using the train-data I have compared all the models seen in class and the result of that has shown that the best model to use was the SVD.
With this result, I start searching on Google how to build an efficient recommender system base on SVD. While reading some articles about it, I found an article with a captivating title: "How Turi Create is disrupting the Machine Learning Landscape".
After reading this article I was curious about it, so I decided to use the TuriCreate library for creating the recommended system for my project.

2 Turi Create

Turi Create is an Apple open-source library that simplifies the development of custom machine learning models based on GraphLab.
Turi Create is:

1. Easy-to-use: focus on tasks instead of algorithms;
2. Visual: built-in, streaming visualizations to explore your data;
3. Flexible: supports text, images, audio, video and sensor data;
4. Fast and Scalable: work with large datasets on a single machine;
5. Ready To Deploy: export models to Core ML for use in iOS, macOS, watchOS, and tvOS apps.

3 My solution

The data for the challenge were in three CSV file named *train-PDA2019.csv* for the training set, *test-PDA2019.csv* for the test set and *content-PDA2019.csv* that contained supplemental metadata about the movie items (visual features, genre, tags, etc.).
There was also a file called *sampleSubmission-PDA2019.csv* that shows a sample of submission file in the correct format.

Created the Jupyter Notebook, I imported all the library for creating the recommender system: *Pandas* for reading the CSV files and *turicreate* for creating the recommender system.
Once imported the three CSV I have displayed the first data from the train set and after using the command *turicreate.SFrame* I made the ready to be used for training and testing the model that I am going to create.
SFrame is a scalable, tabular, column-mutable data frame object. The data in SFrame is stored column-wise and is stored on persistent storage to avoid being constrained by

memory size. Each column in an SFrame is a size-immutable SArray, but SFrames are mutable in that columns can be added and subtracted with ease.
Now that the data are correctly imported and in the correct format. We can star build the recommender system. Before doing this I explored a little bit of the data and the power of Turi Create by creating a popular model using the method *turicreate.popularity_recommender*. The result of this first model shows what are the most popular film for a fixed number of user.
After this basic exploration, I started building the recommender system. Turi Create lets you build a recommender system based on:

1. similarity, a model that ranks an item according to its similarity to other items observed for the user in question;

2. factorization, earns latent factors for each user and item and uses them to make rating predictions;

3. nearest neighbours, create a nearest neighbour model, which can be searched efficiently and quickly for the nearest neighbours of a query observation.

I first try to build a recommender system using the factorization, but the computational weight of this approach was to high, so I decide to switch, build and use a recommender system based on the similarity.
This model computes the similarity between items using the observations of users who have interacted with both items. There are three choices of similarity metrics to use: 'jaccard', 'cosine' and 'pearson'.
You can choose which of these three similarity metrics to use by setting the parameter of the function 'similarity_type'. I try every choice and in the end, the best one was the cosine similarity.
The command *turicreate.recommender.item_similarity_recommender.create* has multiple parameters that you can set. The ones I have set are the following:

1. observation_data, the dataset to use for training the model. It must contain a column of user ids and a column of item ids. Each row represents an observed interaction between the user and the item;

2. user_id, the name of the column in observation_data that corresponds to the user id;

3. item_id, the name of the column in observation_data that corresponds to the item id;

4. target, column of scores representing ratings given by the users;

5. similarity_type, similarity metric to use. In my case I set this to cosine;

6. training_method, the internal processing is done with a combination of nearest neighbour searching, dense tables for tracking item-item similarities, and sparse item-item tables. In particular, I choose to use a sparse matrix to store item-item interactions as a lookup, and may do multiple passes to control memory requirements. This is solution is better if the data has many infrequent items. Other possible option

7. threshold, predictions ignore items below this similarity value.

Now the model for recommending movie is created and trained, we can test it using our test data.
For doing that I used the command *item_sim_model.recommend* which take as a variable the test data set and a parameter k that represent the number of recommendation to calculate. In our case the test data are the test[user] and k is equal to 10.
The result is in the format SFrame for that reason by using command *.to_dataframe* I convert it to a dataframe.
To create the CSV in the correct format for the submission I first grouped by all the itemID recommended before in a list and then using a for loop I create the correct layout for the submission. Once all the values were ready to be saved in a CSV I used the command *.to_csv*.