Machine Learning Engineer Nanodegree

Capstone Project

Sam Ariabod
September 03, 2018

I. Definition

Project Overview

The amount of information on the internet is growing exponentially. 90% of the data on the internet has been created since 2016[1].

Recommender systems in the internet have become so normal that we tend to forget how critical they are[2]. From shopping, to music, to education – the amount of options available is staggering. Netflix posted a Kaggle competition with a reward of $1,000,000.00 USD to build a recommender model that performed better than the one they were using or surpassed the previous year winner[3].

Connecting users to content that is relevant in a timely fashion is important to providing a good user experience. This can be the difference between a user that never comes back to a repeat customer.

The origin of this project stems from a client I work with that offers online training. The courses number in the hundreds across a dozen different categories. In order to help a user find a course that is relevant we will leverage the power of machine learning to analyze historical and current course/user data to build a recommendation engine.

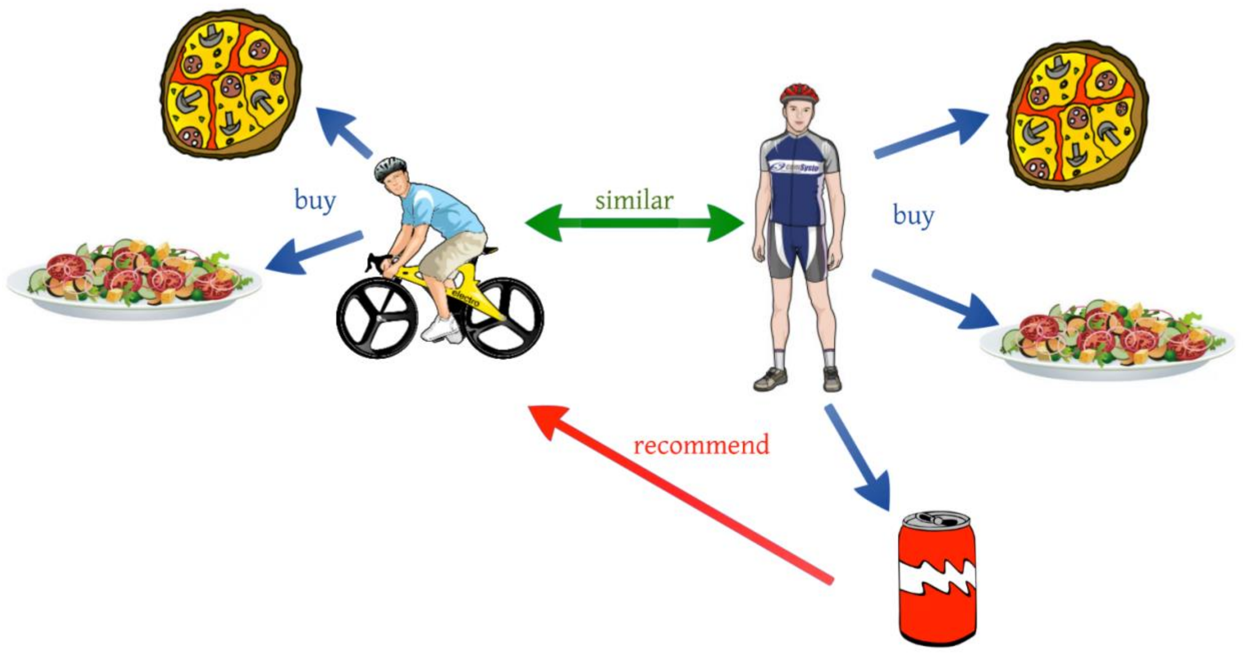
Once the model is built it will be served up via a RESTful API then integrated into the live site.

[1] https://public.dhe.ibm.com/common/ssi/ecm/wr/en/wrl12345usen/watson-customer-engagement-watson-marketing-wr-other-papers-and-reports-wrl12345usen-20170719.pdf
[2] http://www.commons.fm/?p=25017
[3] https://en.wikipedia.org/wiki/Netflix_Prize

Problem Statement

The eLearning Course provider has too many courses spread over a dozen categories for users to reasonably go through. The site offers course reviews and offers a text search for users to try and narrow down the offerings. Neither of these account for any relationship between users or the format of the courses (some are text heavy, some are video based, some are interactive).

The solution for this problem is based on the assumption that people tend to like things similar to other things they like, and things that are liked by other people with similar taste.



[4]

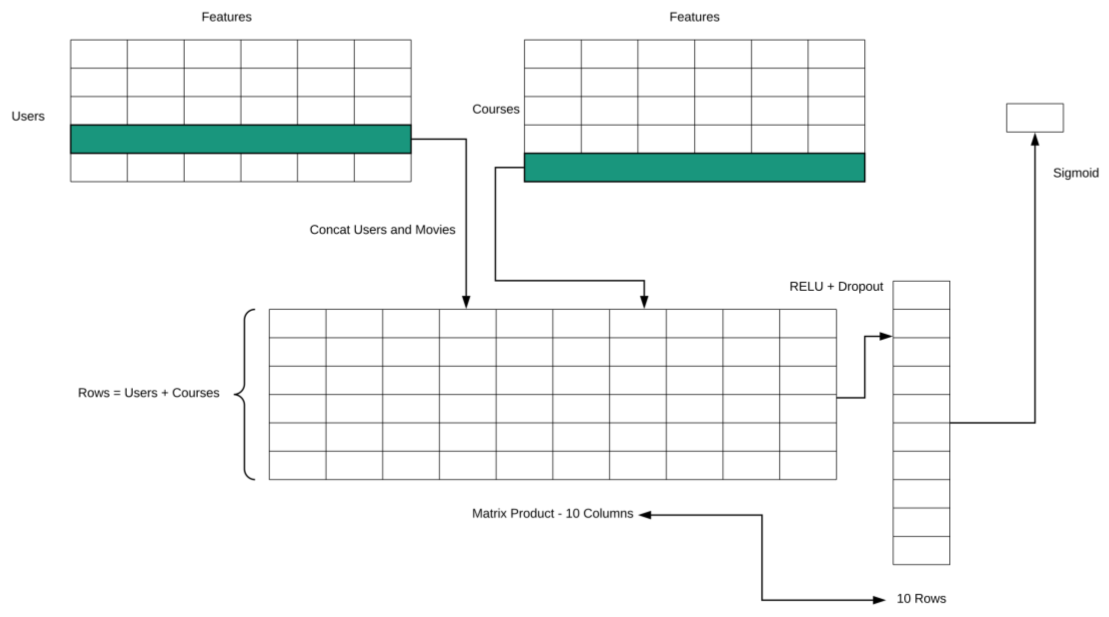
To solve this, we will be taking all user profiles that have completed and rated courses and all course ratings and use collaborative filtering to find hidden relationships.

The technique we will be using for this collaborative filtering model is Matrix (or Tensor) Factorization:

"Matrix factorization can be used to discover latent features underlying the interactions between two different kinds of entities. (Of course, you can consider more than two kinds of entities and you will be dealing with tensor factorization, which would be more complicated.) And one obvious application is to predict ratings in collaborative filtering."[5]

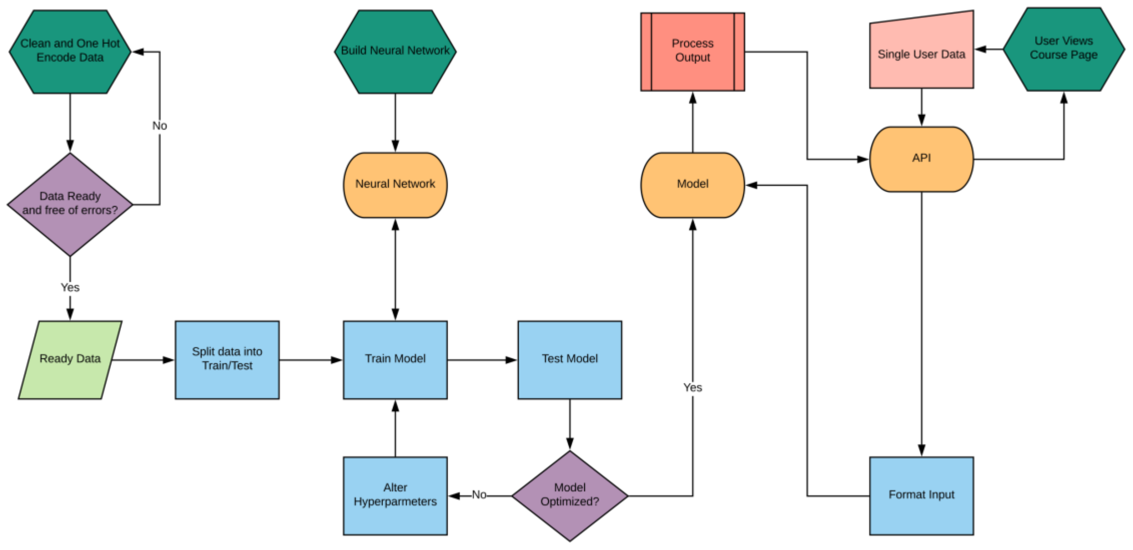
[4] https://medium.com/@tomar.ankur287/user-user-collaborative-filtering-recommender-system-51f568489727
[5] http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/

Matrix Factorization with a Neural Net Basic Flow:



Once the model is built, we will be able to pass in the user information and the system will be able to calculate ratings for missing entries (courses the user has not taken). We can then serve this up through an API.

Birds eye view of the process from data analysis to going live with API:



Metrics

To evaluate the model accuracy, we will be using a very standard Root Mean Squared Error:

$$RMSE = \sqrt{\frac{\sum_{n=1}^{n=N} \left(P_n - O_n\right)^2}{N-1}}$$

"Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results."[6]

As we train the model, we will try to minimize the error rate of prediction ratings vs actual ratings.

II. Analysis

Data Exploration

The data includes 6 features (all categorical) and 1 target variable.

Features:

User – This is the user that rated the course (cardinality: 52118)

Course – Course that the rating was given on (cardinality: 219)

Category – The category the course belongs to (cardinality: 14)

Job – This is the job category of the user (cardinality: 45)

Institution – This is the institution type that the user belongs to (cardinality: 15)

State – This is the state the user resides in (cardinality: 60)

Target:

[6] http://www.statisticshowto.com/rmse/

Rating – This the overall rating for a course, it will have a potential range of 0 – 5 where 0 is on the low end and 5 being a perfect score. This will be what we are trying to predict given a user and a course.

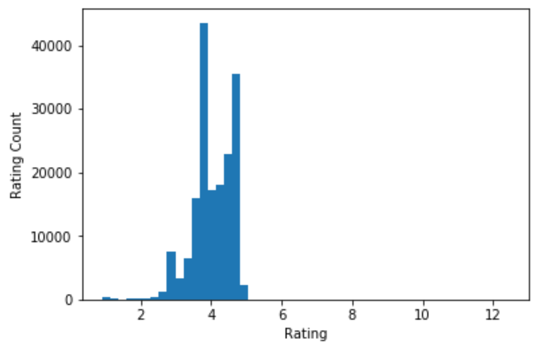
Exploratory Visualization

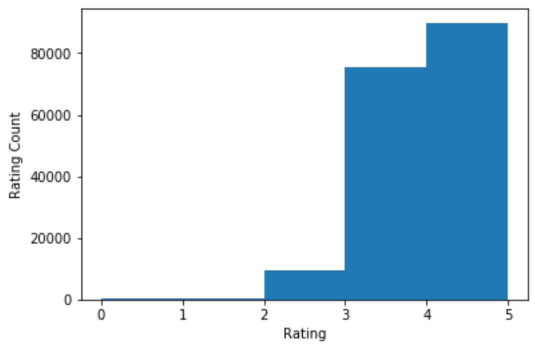
Rating Stats:

	rating
count	175347.000000
mean	4.030533
std	0.571618
min	0.910000
25%	3.690000
50%	4.000000
75%	4.590000
max	12.450000

Looking at the max, it looks like we have some bad data that we need to clean up. A rating of 12.45 is just not possible.

Rating Distribution:





Excluding outliers, a majority of the ratings are between 3-5 with a high spike right around 4.

Mean: 4.03053282919012 - Median: 4.0

Overall, the records are clean. No missing values across the different features.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175347 entries, 0 to 175346
Data columns (total 7 columns):
user           175347 non-null object
course         175347 non-null object
category       175347 non-null object
rating         175347 non-null float64
job            175347 non-null object
institution    175347 non-null object
state          175347 non-null object
dtypes: float64(1), object(6)
memory usage: 9.4+ MB
```

Algorithms and Techniques

The main technique we will be using is matrix factorization with embeddings.

Benchmark

In this section, you will need to provide a clearly defined benchmark result or threshold for comparing across performances obtained by your solution. The reasoning behind the benchmark (in the case where it is not an established result) should be discussed. Questions to ask yourself when writing this section:

- *Has some result or value been provided that acts as a benchmark for measuring performance?*
- *Is it clear how this result or value was obtained (whether by data or by hypothesis)?*

III. Methodology

(approx. 3-5 pages)

Data Preprocessing

In this section, all of your preprocessing steps will need to be clearly documented, if any were necessary. From the previous section, any of the abnormalities or characteristics that you identified about the dataset will be addressed and corrected here. Questions to ask yourself when writing this section:

- *If the algorithms chosen require preprocessing steps like feature selection or feature transformations, have they been properly documented?*
- *Based on the* ***Data Exploration*** *section, if there were abnormalities or characteristics that needed to be addressed, have they been properly corrected?*
- *If no preprocessing is needed, has it been made clear why?*

Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

- *Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?*
- *Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?*
- *Was there any part of the coding process (e.g., writing complicated functions) that should be documented?*

Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

- *Has an initial solution been found and clearly reported?*
- *Is the process of improvement clearly documented, such as what techniques were used?*
- *Are intermediate and final solutions clearly reported as the process is improved?*

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model's solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

- *Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?*
- *Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?*
- *Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?*
- *Can results found from the model be trusted?*

Justification

In this section, your model's final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

- *Are the final results found stronger than the benchmark result reported earlier?*
- *Have you thoroughly analyzed and discussed the final solution?*
- *Is the final solution significant enough to have solved the problem?*

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

- *Have you visualized a relevant or important quality about the problem, dataset, input data, or results?*
- *Is the visualization thoroughly analyzed and discussed?*
- *If a plot is provided, are the axes, title, and datum clearly defined?*

Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

- *Have you thoroughly summarized the entire process you used for this project?*
- *Were there any interesting aspects of the project?*
- *Were there any difficult aspects of the project?*
- *Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?*

Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

- *Are there further improvements that could be made on the algorithms or techniques you used in this project?*
- *Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?*
- *If you used your final solution as the new benchmark, do you think an even better solution exists?*

Before submitting, ask yourself. . .

- Does the project report you've written follow a well-organized structure similar to that of the project template?
- Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
- Would the intended audience of your project be able to understand your analysis, methods, and results?
- Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
- Are all the resources used for this project correctly cited and referenced?

- Is the code that implements your solution easily readable and properly commented?
- Does the code execute without error and produce results similar to those reported?