Machine Learning Engineer Nanodegree

Capstone Proposal

Sam Ariabod
July 15th, 2018

Proposal – Course Recommendation System

Domain Background

Recommender systems in the internet have become so normal that we tend to forget how critical they are (commons.fm). From shopping, to music, to education – the amount of options available is staggering. Having a way to accurately make suggestions has become critical to gaining and keeping customers. 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 years winner (Wikipedia).

I have been involved in online adult education since 2000. I am a course junkie and a big believer in continuing education - anything I can do to make the learning experience better for the user is a big win. Being able to accurately help users find content they wil like is critical to the learning journey. The dataset I am using is from an online continuing education platform and is being used with permission. The model will be implemented on the live site at the conclusion of this project.

Problem Statement

The site has too many courses for user to reasonably browse to find one they like. Also, when they do decide on a course, many times it is not a good fit and they ask to be transferred, or worse, ask for a refund. This creates customer support overhead and also a poor user experience and potentially a lost patron.

To solve this problem, we need to create an intelligent recommender that can recommend courses to new users but also adapt as the user takes courses and rates them.

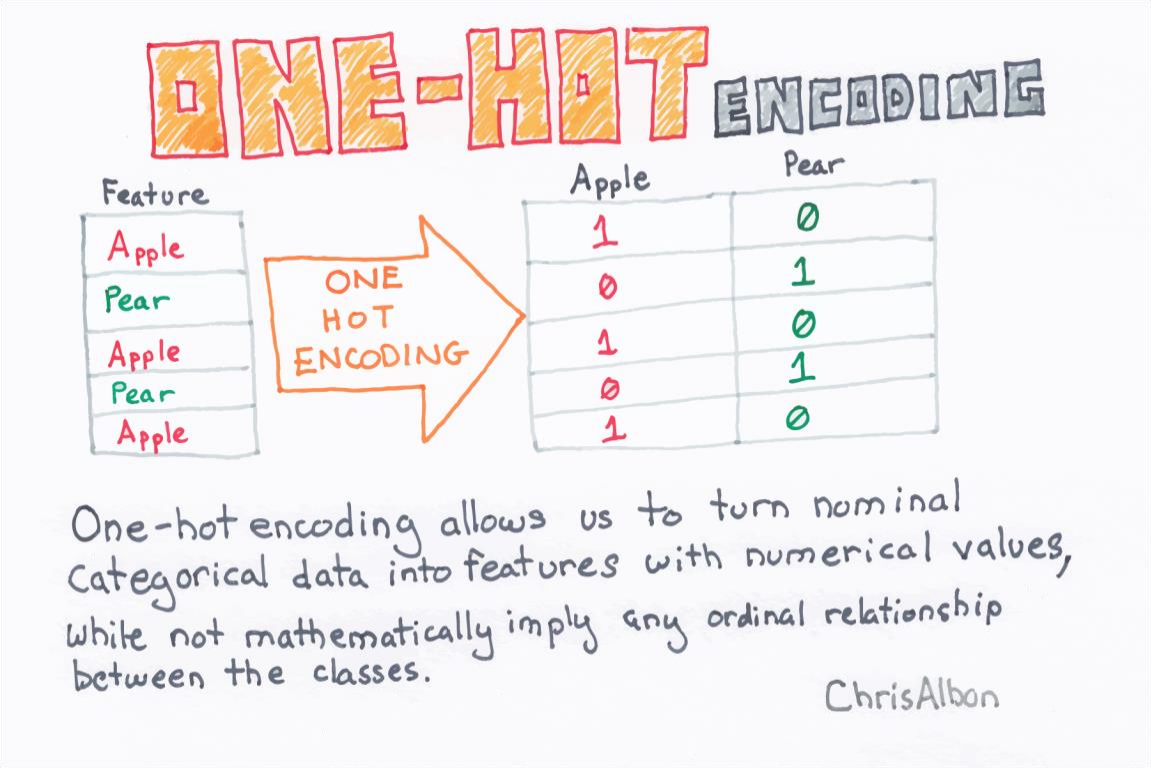
This is measurable in the long term by tracking refunds and course transfers after the system goes live.

Datasets and Inputs

The data set comes from a live site and is fully anonymized using a SHA2 512-bit hash to protect the users. The only human readable value is the rating that is based on a 5-point scale where the higher the number, the better the rating. I have also included some profile details in addition to the course ratings. I will be using this to help match like users.

The dataset is already clean since I was the one the exported it using SQL. The only thing that needs to be done is building out the data frames by breaking out the categorical data.

The categorical data will be spread out into features to create the matrices using one hot encoding. Example:



One hot encoding image from: https://chrisalbon.com/machine_learning/preprocessing_structured_data/one-hot_encode_nominal_categorical_features/

Once the model is finished we can enter in a user matrix including any course ratings then predict how they will rate a specific course. We can use this to build out a recommended course table or even specify the prediction right on the individual course pages.

Solution Statement

The solution for this is to build a model using Collaborative Filtering (Wikipedia). The concept is simple: people tend to like things similar to other things they like, and things that are liked by other people with similar taste.

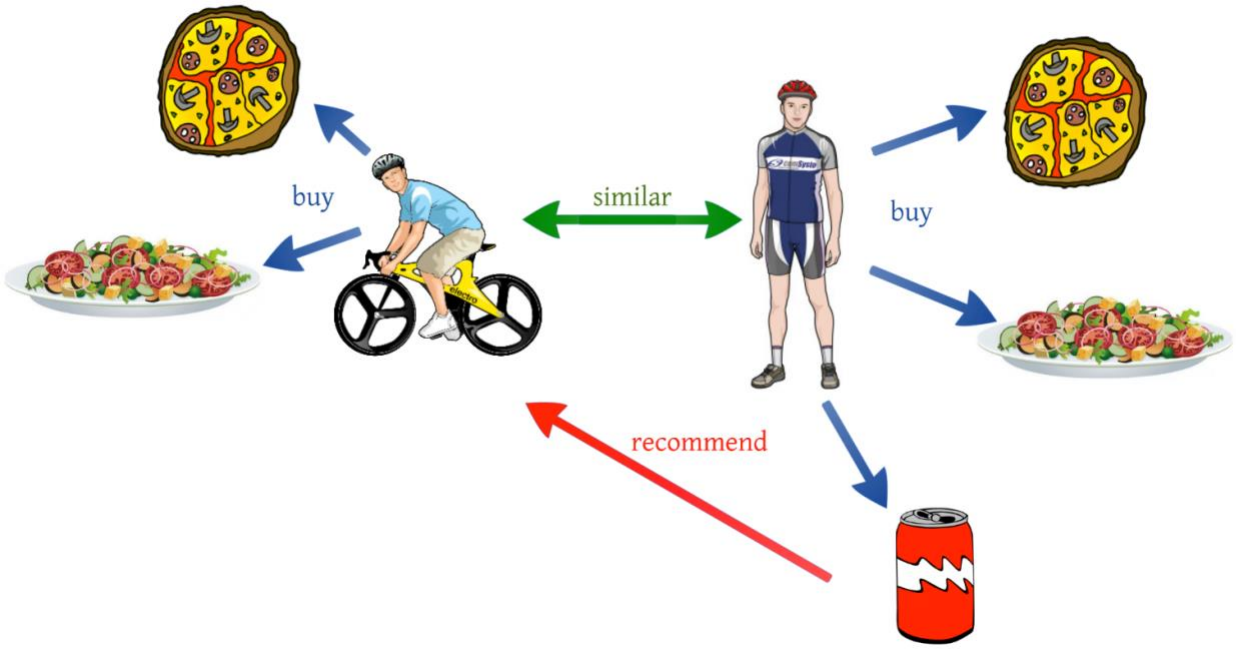


Image from: https://medium.com/@tomar.ankur287/user-user-collaborative-filtering-recommender-system-51f568489727

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." (source)

Essentially, we will be taking all the data and converting them into matrices with embedding layers then using matrix factorization to fill in the gaps that will allow us to make predictions for a user.

Benchmark Model

We will benchmark the model against itself by tuning hyper parameters until we achieve an acceptable loss. Some of the parameters that we can tune are the number neural network layers, the learning rate, size of embeddings, dropout rates, and the loss function.

Evaluation Metrics

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

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

"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." (*source*)

As we train the model, we will try to minimize the error rate between predictions and actual between the train/test sets.

Project Design

Below is a high-level workflow for the building of this model:

1. Clean the data (this has already been done but adding here for completeness)
2. One hot encode all categorical data
3. Load the data into a dataframe in order to create training/testing sets
4. Build out the neural network
5. Start the train/test cycle while optimizing hyper parameters
6. Save the final accepted model
7. Create a API to use the model

Flow chart going over the steps: