CS373: Project Final Report

Liam Connery, Sam Javed, Theo Burkhart, Todd Griffin
December 1, 2018

Project Goal

For our project, we used a dataset containing user's movie reviews. Each entry contains the user's unique ID, the movie's unique ID, and the rating the user gave the movie (1-5). Our goal was to be able to predict whether a given user will like the movie Forrest Gump based off their reviews of other movies. We chose Forrest Gump because it had the most reviews in our dataset. We used perceptron as well as perceptron with myopic forward fitting as our machine learning algorithms.

Dataset: http://files.grouplens.org/datasets/movielens/ml-latest-small.zip

Cleaning the data

The main file we are analyzing is 'ratings.csv' which contains 3 columns, the user ID, movie ID, and rating. Our first step was to create new, unique move IDs since the given movie IDs did not follow numerical order. We wanted the movies to be numbered 1 through n, where n is the total number of movies. Our python script 'ratings_mod.py' did this by creating a dictionary where the key is the old movie ID and the value is the new movie ID, which we generate by incrementing a count by one for every new movie ID we encounter. We then just ran through all the movie IDs and replaced them with the value in our dictionary. From this script, we had a new dataset, 'updated_movie_rating.csv', with 3 columns (user ID, new movie ID, and rating). In the same script, we created a new CSV file ('new_movie_id_lookup.csv') that will allow us to lookup the old movie ID with our newly created ID. We needed this to be able to look up the movie titles from the provided, 'movies.csv'.

Lastly, we created a python script ('Transform.py') that transformed the data so each row represents a user, and each column a unique movie. We did this by creating a new matrix where the number of rows is the number of users, and the number of columns is the number of movies. We then filled in each column for each user with their rating of that movie. For example, the field (1, 3) would contain the rating user 1 gave movie 3. If a user has not reviewed a movie, we put a 0 in that field. Once we had this, we removed all the users who have not reviewed Forrest Gump. In the end, we had 328 users (rows) with 9723 movies (features). Our final dataset is 'final_test_data.csv' which we will use to train our perceptrons. The data was shuffled when initially read in to the Perceptron_run.py to ensure that the order of the samples in data set did not affect our model.

Initial Testing

Before beginning to train, we found that ~76% of users reported liking Forrest Gump, so we knew as a lower bound of performance, if our error was worse than 24%, our learning algorithm would be worse than chance. We then did training and testing on the entire data set to see the upper bound on performance. The initial testing with this method had a mean error of 0.0%. This was expected due not only to training and testing on the same data set, but also due to the number of features being much larger than the number of training samples. In our dataset, we have 610 users reviewing over 9000 movies. The model is likely too complex to train with this few of samples and demonstrates the value that myopic forward fitting with bring to our model.

One modification that was done to the perceptron algorithm was shuffling the order that the samples were tested against theta. Since the results from the perceptron algorithm are affected by the order of the samples, shuffling the order the features were tested was done to prevent overfitting during training. This was done by creating a range that leveraged the random library in python.

Two-Fold Cross Validation

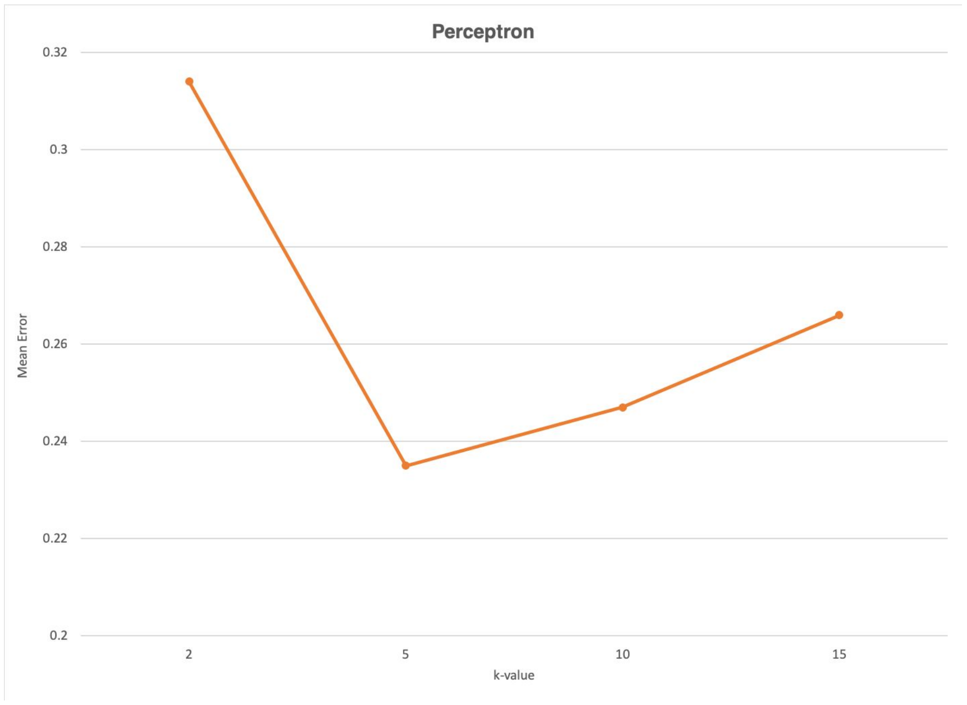
For our perceptron, our two inputs are a ***n*** by ***d*** matrix X and a ***n*** by ***1*** matrix y. N is equal to the number of different individuals that have provided reviews in our data set. D is equal to the number of different movies in the data set minus Forrest Gump which we will be testing on. The matrix y is generated by removing the column that represents Forrest Gump.

For feature selection, we used myopic forward fitting with each iteration of our k-fold cross validation. Which each new group generated by k-fold, we ran myopic forward fitting to find the best features when predicting whether a user will like Forrest Gump.

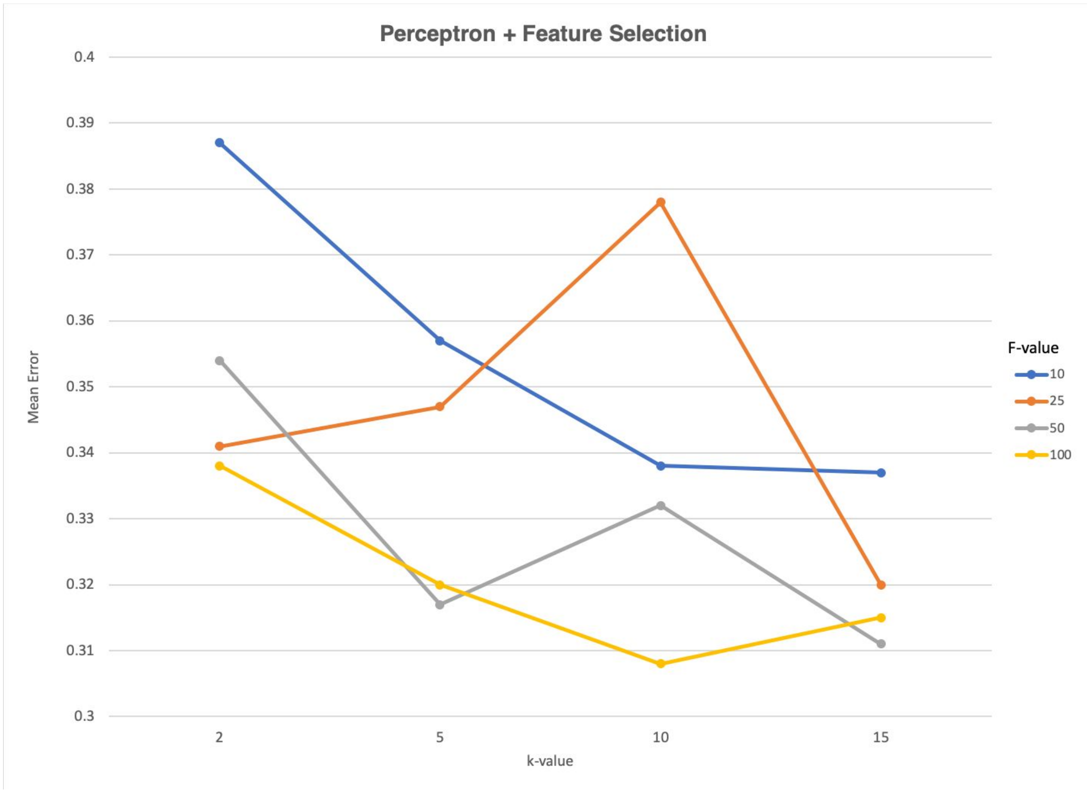
Using two-fold cross validation, and F=100 (number of selected features for myopic), we achieved a mean error of 31% for perceptron and 30% for perceptron with feature selection. Perceptron had a variance of 0 and perceptron with feature selection had a variance of 0.0015. Unfortunately, this is worse than chance error of 24%.

Results & Tuning

We began by tuning the number of iterations of perceptron. We found that increasing our number of iterations continuously improved our mean error up to 400 iterations. For perceptron we achieved a minimum mean error of 23.5%, and perceptron with feature selection a minimum mean error of 27.4%. After 400 iterations, our mean error did not improve or got worse.

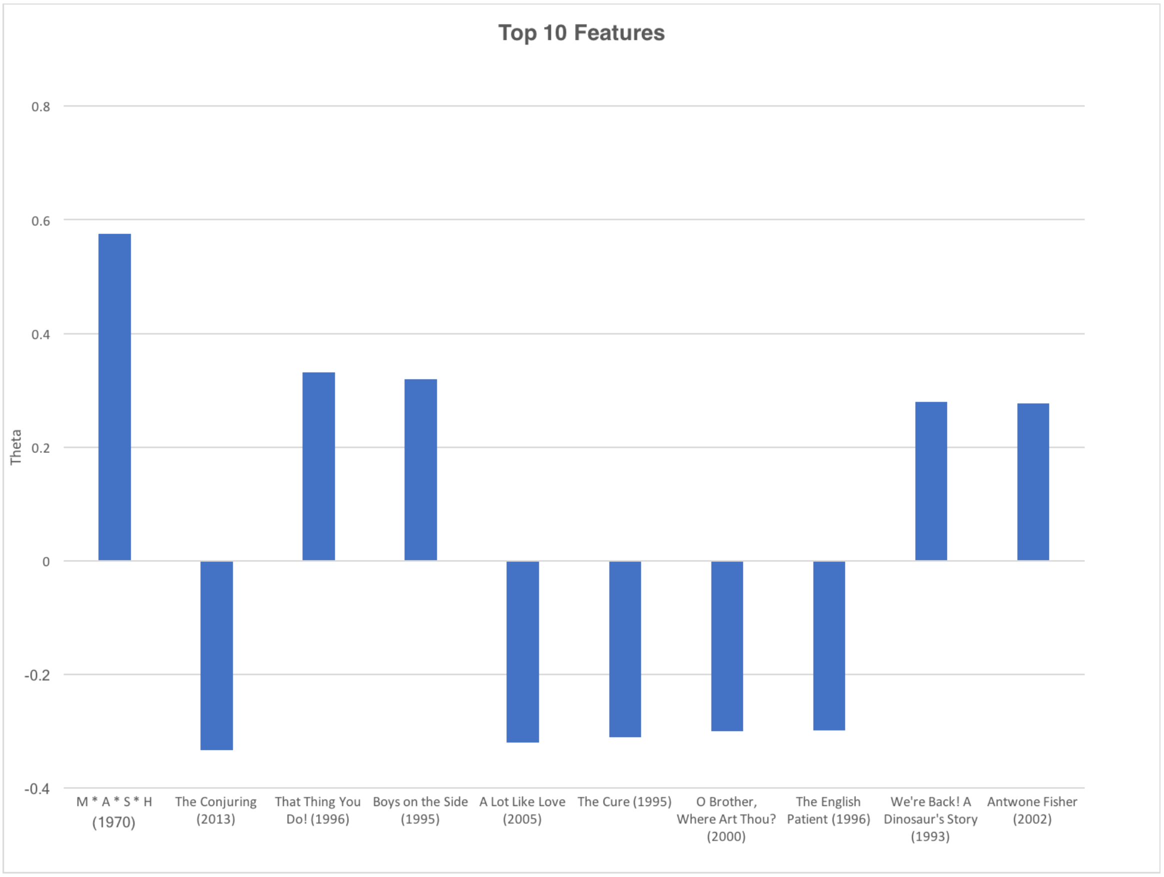


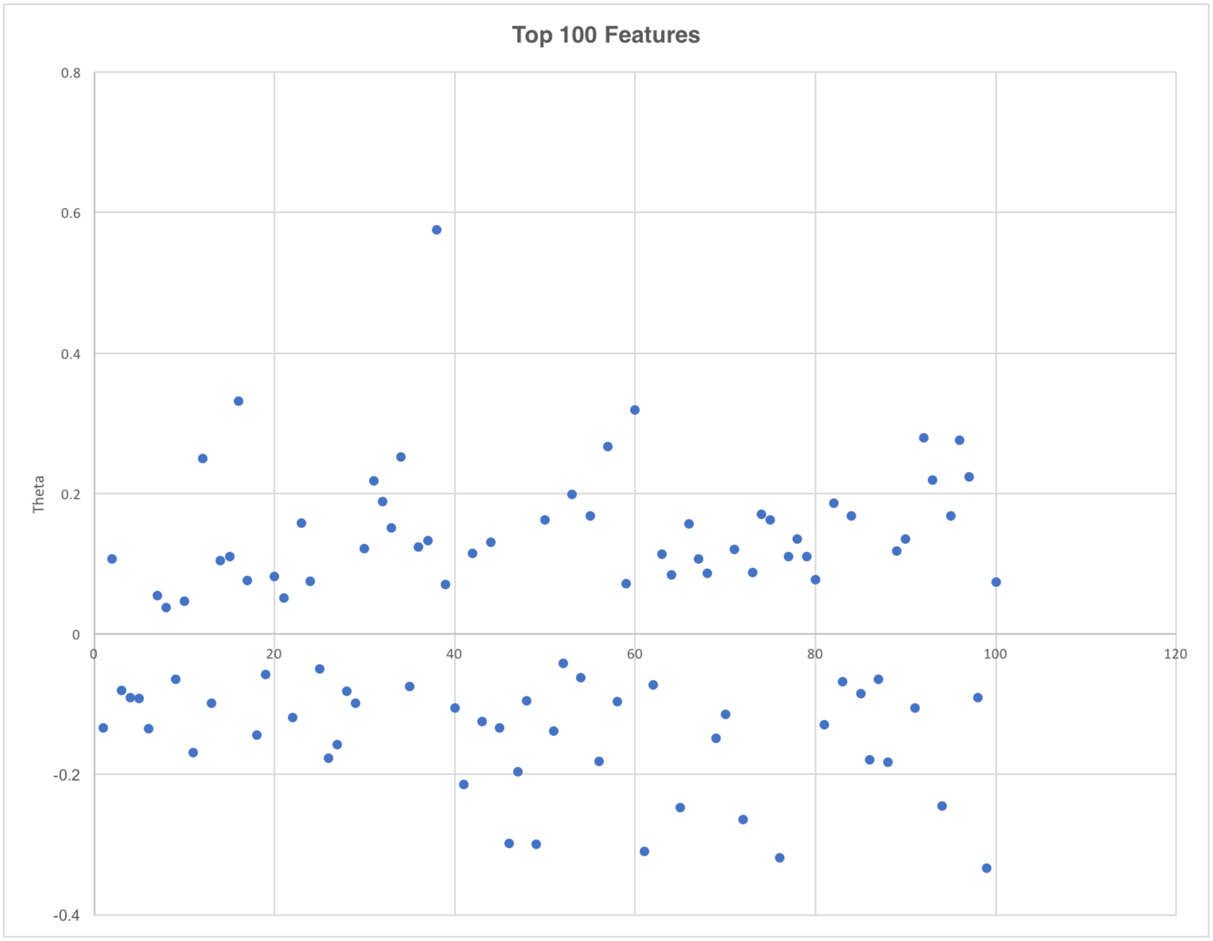
For k-fold cross validation, we wanted to tune our k-value for regular perceptron separately from perceptron with feature selection. For vanilla perceptron, our best results came from k=5 where we were able to achieve a mean error of 23.5% and variance of 0.002. Our worst results came from two-fold cross validation, which consistently had mean error of over 30% and variance of 0.0005.



For perceptron with feature selection we wanted to tune the k-value in conjunction with F. Analyzing our data, we had the best results with a F-value of 100 and a k-value of 10. With these values we had a mean error of 30.8% and variance of 0.006. Looking at the overall trend, a k-value between 10 and 15 had the best results while two-fold cross validation performed poorly.

Despite our hypothesis, perceptron outperformed perceptron with feature selection. Given we had 9723 features, we believed feature selection would greatly improve our results as it would narrow down the features to the best predictors. However it seems that all features collectively were important to the final predictions. Even as we increased F to values above 100, our mean error consistently worsened as our compute times increased exponentially. This is most likely a symptom of overfitting caused by feature selection.


Top 10 Features
Theta
0.8
0.6
0.4
0.2
0
-0.2
-0.4
M * A * S * H (1970)
The Conjuring (2013)
That Thing You Do! (1996)
Boys on the Side (1995)
A Lot Like Love (2005)
The Cure (1995)
O Brother, Where Art Thou? (2000)
The English Patient (1996)
We're Back! A Dinosaur's Story (1993)
Antwone Fisher (2002)



Myopic forward fitting was run to determine which features made the largest impact. The top 100 features shows how many of the features are clustered around -0.1 and 0.1. Additionally, since these are the features that make the largest impact, there is a large gap surrounding the x-axis. This is because features that do not make a large impact have a small magnitude. A few features stand out as being especially good predictors for this model. The top ten features and some information about that movie are displayed in the Top 10 Features. Some of these movies seem to make sense why they have a large positive magnitude. Several of the largest positive magnitudes are movies released within a couple years of Forrest Gump and of similar genres. One interesting find was the third largest magnitude from the Top 100 Features was a movie that starred Tom Hanks, the main actor in Forrest Gump.