RSS Feed Recommendation

Charles Chen charles.chu.chen@gmail.com

Introduction

Really Simple Syndication (RSS) Feeds allows users to access blogs and articles in an easy to read format. It cuts out the overhead of navigating websites for content and allows users to get information more quickly. Currently, the user is in total control of their RSS feeds, adding and deleting feeds according to their tastes. This requires the user to actively search out RSS feeds that interest them. The proposed recommendation algorithm will aid users in their search for new and exciting feeds. It will suggest feeds by extracting features from the user's subscription history and predict the feeds they will most likely enjoy.

Because there are many recommendation algorithms, it is important to choose an algorithm that is suitable for this problem and the data set. Therefore, I begin by describing the data set in Section I. Then I discuss the results of using a simple recommendation algorithm that utilizes K-Means Clustering in Section II. The disadvantages of clustering will motivate the use of Maximum Margin Matrix Factorization (MMMF) [1] for this problem in Section III.

Section I

For my data set, Alphonso Labs has generously provided a database of RSS feeds from the past four months. Furthermore, they have user activity logs for Pulse, their RSS reader application for the iPad and iPhone. This user data includes actions such as adding or deleting feeds from one's subscription list and articles read by that user. User activity is split into sessions in which the user logs on to browse some feeds and ends the session by logging off. Therefore it is possible to track the articles each user has read during each session.

There are many potential features that can be extracted from the data set. I quickly discovered that extracting features from individual stories would be difficult. The main issue is that the articles were saved directly from the web. Therefore the article content was still embedded within a sea of HTML markup. The task of extracting the content for each story was highly infeasible. Without access to the article content I had to focus on algorithms that did not require feature extraction of articles. This sidelined my original plan to use an algorithm such as TD-IDF to identify the unique terms for each story therefore creating a feature set for each article.

After ruling out feature extraction from stories, I focused on user activity. I noticed that the action of adding and deleting a feed was similar to rating that feed. If a user keeps a feed and never removes it, it signifies that he/she likes the content from that feed. This implies a high rating for that feed. However, if a user adds a feed and then subsequently removes it, then the content of that feed is not what the user wants. This implies a low rating for that feed. Therefore I collected all activity for each user that had to do with adding and removing feeds. A total of 80,000 unique feeds were found from the actions of 30,000 users. With this dataset I was now able to apply K-Means Clustering to group users together based on their "ratings".

Section II

K-Means Clustering is an unsupervised learning algorithm that groups the training examples into clusters. Training examples are assigned to the closest cluster based on the distance from their feature vector to the center of the cluster.

The idea behind using K-Means Clustering to recommend feeds comes from the intuition that users will enjoy feeds that similar users have added. Once users are in clusters, recommendations can be made based off of the feeds from other users within the same cluster.

A feature vector for each user, $u^{(i)}$, is chosen to be a N-dimensional vector, $f^{(i)}$, where the j-th element of $f^{(i)}$ is as follows:

$$f_j^{(i)} = \begin{cases} 1, & if\ u^{(i)} is\ subscribed\ to\ feed\ j \\ 0, & otherwise \end{cases} \quad for\ j = 1,2,\ldots,N$$

By choosing feeds that have at least 100 user subscribers, I limited the initial 80,000 unique feeds down to 200 "popular" feeds. Subsequently, this fixes the dimension of the feature vector to be N = 200. The reason for limiting the feeds down to more active feeds is that most of the feeds only had a few subscribers. This is not enough information for the learning algorithm to work with. Also it unnecessarily creates a higher dimension problem for the learning algorithm to solve. The final parameter needed is the number of clusters. I found through experimentation that with K = 15, the users were split into clusters of a couple hundred each with no clusters having too few users.

To set a metric for the ability of clustering to correctly recommend a feed to a user, I use a soft generalization approach [1]. Before clustering, I randomly choose 10 percent of the users and remove r feeds from their subscription list. Then I perform clustering on the entire data set including the users with feeds removed. After clustering, for each user with r feeds removed, I recommend a set of m feeds. This set of feeds is a collection of the highest m subscribed to feed by other users in the same cluster. For each removed feed for that user, I check if the feed is in the set of recommended feeds. If it is, then this is a successful recommendation, otherwise it is not. Recommendation Accuracy (RA) is calculated as follows

$$RA = \frac{\#\ of\ successful\ recommendations}{total\ \#\ of\ removals}$$

Figure 1 is a plot of RA for 10 runs of r = 1 with m varying from 1 to 10. The blue lines are the RA of recommending the most popular feeds within the same cluster that the user does not have. The red lines are the RA of recommending the most popular feeds overall that the user does not have. Because the blue and the red line are so close together, even overlapping, it suggests that clustering does not help the recommendation process especially when m is less than 4. However once m > 4, the clustering is able to predict more accurately the feed that the user is missing. Overall the performance of clustering for recommending feeds is very poor. It starts off at around 5% chance of recommending the removed feed with m = 1 to around a 50% chance of recommending the removed feed with m = 5. Furthermore, the plot does not differ by much in varying r and K.

I believe that there are a couple reasons that make this problem unsuitable for clustering. The first issue is that by assigning a 0 in the feature vector to feeds that a user is not subscribed to is equivalent to saying that this user does not like any of these feeds. It may be the case that the user has never seen this feed before and has yet to pass a judgment on it. In this representation, this cannot be distinguished from a true 0, where the user actually added and deleted the feed, signifying dislike. In

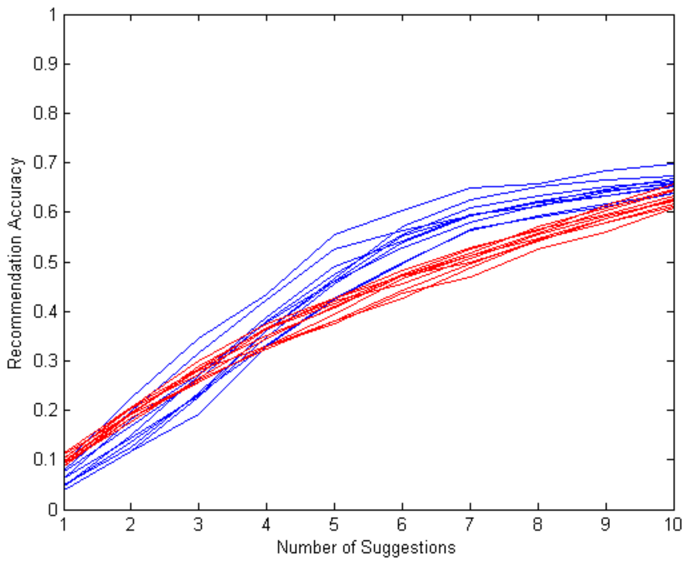

Figure 1: Clustering Results

other words, this feature vector does not have the ability to code an unknown rating (which can potentially be a good or poor rating) and naively labels it as a poor rating.

Also another issue is that grouping users together purely by what they have subscribed to may not be the correct thing to do. Other user features such as, user interest in sports, movies, fashion, technology, etc…, may serve to cluster similar users together. In the current setup, users that have the same tastes might not even end up in the same cluster. A difference in just one feed in their subscriptions may bounce them into separate clusters. Due to these concerns with clustering, I propose using MMMF to mitigate some of these problems.

Section III

Maximum Margin Matrix Factorization (MMMF) is an algorithm that is able to predict a user's ratings on a set of m items based solely on a few ratings provided by the user and other users. MMMF learns from users that rate similarly to the current user and predicts the current user's unknown ratings. Recommending the user a new item may be as simple as recommending the highest predicted rated items in his/her set.

MMMF is suitable for this problem because for a given user not all the ratings are known for each feed. Most users have only voted on a couple feeds, the ones that they are currently subscribed to and the ones that they have added and removed. Other feeds have not been voted on yet and MMMF is capable of filling in the ratings for these feeds. Another reason that MMMF is good for this problem is that it does not require any extra feature extraction. This is ideal for my data set since I only have user activity to work with (i.e. features to do with personality and interests will be difficult to determine).

In MMMF the observed ratings is represented as matrix $Y \in \mathbb{R}^{n \times d}$, where n is the number of users and d is the number of feeds. The non-zero elements of Y are the observed ratings. Each row represents the ratings for all feeds of one user. Since each user votes on a few feeds, this row will mostly be all zeros. It will be up to MMMF to fill in these zero elements with a predicted rating. MMMF finds a rank-k matrix $X \in \mathbb{R}^{n \times d}$ which is a product of two matrices $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{k \times d}$. The rows of U can be thought of the "feature vector" for each user and the columns of V as a linear predictor, predicting the entries in the corresponding column of Y based on the "features" in U. Parameter k is chosen to be relatively small (< 15) and represents the number of "features" in the "feature vector" for each user.

Both U and V are unknown and are estimated by MMMF by minimizing their Frobenius norms, the sum of squares of entries, of U and V and a chosen loss function L. In the simplest form, MMMF uses a regression based L which is the sum of the squared errors of user ratings in Y and the predicted values for those same ratings in X. The following equation is the optimization problem that MMMF solves

$$\min_{U,M} \frac{1}{2}\sum_{i=0}^{n}\sum_{j=0}^{d} S_{ij}\left(Y_{ij} - X_{ij}\right)^2 + \|U\|_{Fro}^2 + \|V\|_{Fro}^2 \quad where\ S_{ij} = \begin{cases} 1 & if\ user\ i\ rated\ item\ j \\ 0 & otherwise \end{cases}$$

The optimization problem is not jointly convex in U and V, but it is still convex in U if V is kept constant and vice versa. Using a solver such as one provided in the CoFiRank [2] software bundle enables MMMF to be performed efficiently on large data sets.

As in the clustering section, users have not actually rated the feeds. Therefore I have to infer a rating based on the user's actions. I have approached this rating inference in two ways which I will describe separately in the following two sections.

Section IV

In the first attempt at inferring a rating for each user I used a binary approach very much akin to that used in the K-means clustering section. However, instead of only keeping track of the feeds a user adds, I also include those that the user removes. Each element of matrix Y is now defined as follows

$$Y_{ij} = \begin{cases} 2 & if\ user\ i\ subscribes\ to\ feed\ j \\ 1 & if\ user\ i\ subscribed\ to\ and\ removed\ feed\ j \\ 0 & otherwise \end{cases}$$

Using this rating system, MMMF will predict for the feeds that the user has not yet subscribed to if he/she will add and keep the feed or add and remove the feed. This decision boundary is by default set at 1.5.

To evaluate the performance of MMMF, I again used a soft generalization approach. I randomly chose 10 percent of the users and for each user, set r non-zero elements in their ratings list to zero. Then I ran MMMF on the resulting Y matrix. The test accuracy is defined as the ratings that were removed earlier and correctly categorized by MMMF as a rating 1 or 2, given the decision boundary of 1.5, divided by the total number of ratings removed.

Using this metric with parameters r = 3 and k = 10, the test accuracy was averaged over ten runs to be 0.839. This means that the rank-k matrix X that approximates the observed matrix Y could predict with an approximate 84% accuracy the feeds that the user will keep and those that the user will add and remove.

There are a few shortcomings to this rating system. First of all, it assumes that the action of a user subscribing to a feed means that the user likes the feed. This is a weak assumption since some users may add a feed and never read from it. Also, users may add a feed on accident and remove it because it wasn't what they were looking for initially. This doesn't mean that the user doesn't like that feed, the user just made a mistake. Lastly, although the MMMF can achieve a high accuracy for predicting feeds that the user will keep and those that they will remove, it doesn't show how much a user will like the feed. There is no way to select out of the all the feeds that the user is predicted to keep which ones they will like the most. The following section will attempt to satisfy all of these shortcomings by inferring ratings of feeds for each user from the articles read by that user.

Section V

Inferring a rating purely through the actions of subscribing and removal of feeds was argued to be a weak indicator of the user's true preference for that feed. Looking back at the available data, it became evident that tracking the articles read by each user from each feed would yield a better indicator. The assumption here, which is quite reasonable, is that the user will read more from feeds that he/she prefers and less from feeds that he doesn't.

Because feeds update at different rates (articles per day) it is important to choose a rating system that does not favor feeds that have higher update rates than other feeds. Therefore, I chose to use a session based approach. As mentioned before in Section II, a session is one which the user logs on, reads some articles, and then logs off. For each feed, I record session reads, the number of sessions where the user reads from that feed. This prevents from over counting a feed just because it updates often. It is important to note that this format is susceptible to biasing against feeds that update at very low rates. If the user logs on often and his favorite feed has not released a new article he/she will not access that feed giving it a low session read count. This is one of the drawbacks of this format.

Given the session reads for each user, we can now infer a rating for each feed. A simple and linear choice would be to normalize for each user the session reads for a feed by the total sessions by that user. Then uniformly discretizing the normalized session reads into 3 buckets yields a rating system between 1 and 3.

Again, using the soft generalization approach, r feeds were removed from the ratings for 10 percent of users. Now, because the rating is no longer a binary classifier as it was in Section IV, the test RMSE (root mean squared error of the r ratings initially removed) is the metric used. With the linear ratings between 1 and 3, an average test RMSE of 1.20 was achieved.

The test RMSE seems to be on the high side so I looked for a more nonlinear way of mapping the session reads to a rating to bring down the RMSE value. The intuition exploited here is demonstrated in the following example. Take for example two feeds that differ by 5 session reads. If their session read values are both relatively high with respect to the user's total sessions the ratings should not differ much. In other words, a feed read 19 times versus another feed read 14 times out of 20 total sessions should still yield almost equivalent ratings. The user likes both and will generally read from both in a given session. On the other hand, for low session reads, 5 session reads makes a large difference. Looking at a feed 1 time versus another feed 6 times out of 20 total sessions should be distinguished more widely. The user dislikes articles of the feed that has only 1 session read but is mildly interested in the other feed.

The resulting nonlinear rating system is defined as follows

$$for\ user\ i, Y_{ij} = \begin{cases} 3 & 0.5 * total\ sessions < session\ reads\ for\ feed\ j \\ 2 & 2 < session\ reads\ for\ feed\ j < \ 0.5 * total\ sessions \\ 1 & session\ reads\ for\ feed\ j < 2 \\ 0 & otherwise \end{cases}$$

The breakpoints of 2 and 0.5* total sessions may seem a little arbitrary. I didn't have time to optimize these values but this was just used to demonstrate how a nonlinear mapping of session reads to ratings helps lower test RMSE. With the soft generalization approach, an average test RMSE of 0.76 was achieved. This is a significantly lower test RMSE than that achieved with a linear mapping.

Conclusion

RSS feed recommendation is similar in difficulty to movie recommendations. Extracting features from RSS feeds is quite difficult and is highly dependent on how and what you extract from the feeds. Therefore it seemed fit to use algorithms that depend on features that are easily accessible given the data set. This is why the K-means clustering algorithm was used where there was one feature per feed for each user, a binary value designating if the user subscribed to that feed or not. Clustering proved to be ineffective for many reasons, the main one being that it could not distinguish between feeds that the user has not yet seen and ones that he/she has seen and disliked. Also, the accuracy of the recommendation of the algorithm was determined to be too low.

MMMF was found to succeed the areas where the K-means clustering algorithm failed. MMMF is able to fill in missing ratings for each user based on users who rate similarly. This is a form of "collaborative filtering". The final issue addressed with MMMF was to determine a way to accurately infer a rating for each feed given the user's reading history. A nonlinear mapping of the number of session reads for each feed to a rating between 1 and 3 yielded the best average test RMSE of 0.76.

Completing the recommendation algorithm will require more testing of the nonlinear breakpoints. Also, it may benefit to explore finer nonlinear functions that span over more ratings such as 1 to 5. Finally once the test RMSE is at an acceptable low value, feeds that are predicted to be highly rated by the user can be recommended confidently.

[1] N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems* 17, Cambridge, MA, 2005, MIT Press.
[2] Weimer, M., Karatzoglou, A., Le, Q., & Smola, A. (2008). Cofi rank – maximum margin matrix factorization for collaborative ranking. In *Advances in neural information processing systems 20*. cambridge: MIT Press.