Overview of Alkindi Recommendation Technology

March 23, 2001

Introduction: Collaborative Filtering and Clusters

Alkindi recommends products (say, movies) to a user by identifying other users whose tastes are similar to those of the user, then recommending products that those similar users have tended to enjoy. This approach is called collaborative filtering, because users collaborate to recommend products to each other, using one another's opinions to filter out all but the most relevant content.

A key difficulty with implementing robust collaborative filtering technology in real time has been scalability: accurately finding the most similar users to a given user as the user base grows has been viewed by some as computationally intractable. Alkindi avoids this problem by moving the most computationally intense calculations offline. We use a batch process to divide our user base up into groups, or clusters, of users of similar tastes, and to compute a range of statistics associated with these user clusters. To generate recommendations for a new user, we simply search through the clusters (rather than through the users) to find ones that reflect the user's tastes, then recommend based on the statistics associated with such clusters.

In other words, once the initial user cluster framework is set up offline, Alkindi can generate accurate recommendations in real time by the following simple steps:

- **Input:** collect data on the user's taste preferences
- **Clustering:** use the data to assign the user to appropriate user clusters
- **Output:** use the clusters to generate recommendations for the user

Offline Clustering Framework

Formation and Evaluation of User Clusters

The Data and the Model

Users express their tastes by assigning ratings to products. For example, users of the Alkindi movie recommendation engine rate movies on a scale from 1 (low) to 6 (high). To model users so that they can be grouped together, Alkindi represents them geometrically as vectors in a high-dimensional space. The coordinate axes of this space correspond to products; the coordinates of the point representing a user are that user's ratings of those products. Since it is impractical to ask users to rate every available product, an important issue is that, typically, only some of a user's coordinates in rating space are known. (In other words, our data is often sparse.)

The Clustering Algorithm

Alkindi partitions its existing user base into clusters using "*K*-means", a statistical algorithm that maximizes the geometric tightness of the clusters. Tightness depends on the way in which distance in rating space is measured. Alkindi has developed a novel metric that smoothly integrates all available data: in the case of movies, two users are close in rating space if they tend to see similar movies, and rate the movies they see the same way. This helps alleviate the sparse data problem, as instances of a particular user not having seen a particular movie now become usable data.

Evaluation of Cluster Quality

The purpose of clusters is to generate product recommendations, and Alkindi evaluates them with this purpose in mind. Once the clusters are formed, the software generates simulated recommendations, and estimates the quality of those recommendations by comparing them with existing user ratings. This process is repeated a number of times, and the clusters that generate the best simulated recommendations are the ones the system uses to generate real recommendations for real users. Carrying out this process offline ensures that clustering can continue until the clusters are known to generate effective recommendations.

Alkindex

In addition to evaluating overall cluster quality, Alkindi also estimates how accurately each particular user's cluster assignment represents the user's tastes. This is done by estimating the extent to which personalized recommendations based on the user's cluster membership improve on unpersonalized recommendations. This estimate is used to generate the Alkindex, a graphical gauge that shows users how well the system knows them. When the Alkindex is low, users are motivated to supply more data about themselves by rating more products, leading to better recommendations and enhanced user satisfaction. In user testing of Alkindi's movie recommendation engine, 79% of users said they were either strongly or very strongly motivated to rate more movies by the desire to improve their Alkindex.

Grouping Products Enables Granular Characterization of User Tastes

Typically, users' tastes overlap on some products but not others. (To take an extreme example, there is not much reason to believe that two people who like the same Westerns will also like the same computer books.) Consequently, rather than trying to group users based on all available products, Alkindi groups them based on subclasses, called product clusters, which appear to have predictive value. (Predictive value means we can reasonably expect agreement on some products in a product cluster to indicate agreement on others.)

Product clusters in the Alkindi recommendation engine currently correspond to movie genres: action films, comedies, etc. The user base is partitioned into user clusters associated with each product cluster. For example, each user belongs to an action user cluster, which agrees with the user on action movies and is used to recommend action movies to the user. Similarly, each user belongs to a comedy cluster, a drama cluster, and so on. The Alkindex then measures how effective all of the user's clusters are likely to be at recommending movies to the user.

Implementing more general techniques for grouping products (movies or otherwise) based on a range of criteria is an important step in the future development of the Alkindi engine.

Real-Time Clustering Process

Step 1: Data Collection

Data on Users

The sparse data problem forces us to ensure that the data we do collect is as useful as possible. Users can be expected to rate only a few products before asking for recommendations, and Alkindi chooses those products for maximum utility in characterizing the user's tastes. Initially, we ask users to rate popular products (for example, movies that most people have seen) whose ratings vary the most between different user clusters.

The Alkindex motivates users who have already rated some products to ask to rate more. Great care is taken to ensure that these additional products are appropriate to the individual user. The engine considers to which clusters the user belongs, and to which other clusters he or she might reasonably belong. It asks the user to rate products that the user is likely to have seen, and which differentiate most effectively between the different clusters to which the user could belong.

Data on Products

Each instance of a user rating a product adds to our knowledge of the user, and also to our knowledge of the product. A second key function of Alkindi's data collection system is to increase the amount of data we have on the available products in our database. This broadens the base of recommendable products, and allows us to more accurately match those products with users who might enjoy them.

Once our understanding of a user's taste passes a certain threshold, the engine periodically asks the user to rate products in the database that need more ratings. Once again, the products are chosen to enhance the likelihood that the particular user be able to rate them. In this process, products which are new to the database (for example, movies that have just been released on video) are favored, ensuring that we rapidly build up enough ratings for those products to begin recommending them.

Step 2: Clustering Individual Users

Anytime a user, whether new or existing, has rated some products, we know more about the user's tastes than we did before. Since our knowledge of the user's tastes is summed up in the user's cluster membership, we always update the user's cluster assignments in response to any new ratings.

The first time a user rates products in a particular product cluster, we assign them in real time to one of the user clusters associated with the product cluster. This is the cluster to which the user is geometrically closest, based on our rating space model.

If a user has already rated some products in a product cluster, he or she has already been assigned to a user cluster associated with that product cluster (either in the last batch process or since). A user who then rates some more products in that product cluster may be taken out of their current user cluster and assigned to a different one. This occurs if the user is closer to the new cluster than to the old one based on the updated ratings data.

In either case, we ensure that users are always assigned to the most appropriate user clusters based on all the ratings they have submitted. In other words, the system responds instantly to user feedback, updating its characterization of the user's tastes based on that feedback. Since clustering or reclustering a single user is a simple process, this instant response does not carry a high computational cost.

Step 3: Generating Recommendations

Stochastic Choice of Product Cluster

Once the user has been assigned to user clusters based on all of his or her ratings, we are ready to generate recommendations. We generate one at a time. The first step in generating a single recommendation is to pick a product cluster to recommend from. Dividing products into clusters allows us to measure the user's tendency to favor particular product clusters, and to target our recommendations accordingly. For example, if a user usually rates action movies low and romance movies high, the engine should recommend much more frequently from the romance cluster.

Each time Alkindi's engine is asked for a single recommended product, it uses a probabilistic (or "stochastic") model for choosing the product cluster from which to pick this product. A product cluster will be favored in this choice if:

- It has more products from which to recommend.
- The user tends to see products in the product cluster and rate them highly.
- The user is well clustered with respect to the product cluster.
- The user has responded positively in the past to recommendations from this product cluster.

Advantages of the Stochastic Approach

In addition to ensuring that recommended products typically come from product clusters favored by the user, the stochastic approach has several other advantages:

- **Variety.** Many competing recommendation engines tend to overweigh one kind of product (for example, recommending many Star Trek movies to someone who rated a single Star Trek movie highly). Taking explicit care to recommend from many product clusters avoids this. In user testing, 79% of the users polled by Alkindi were pleased with the diversity of the genres from which their movie recommendations came, and 68% were pleased with the mix of popular and obscure movies. No other service tested offered such a high level of satisfaction with overall diversity.
- **Robustness.** Recommending based on a single group of similar users results in all recommendations being poor if the group is inappropriate for the user. Using several user clusters makes it more likely that some are well chosen, ensuring some good recommendations and allowing the user to get as many good recommendations as desired by spending enough time on the system.

- **Responsiveness.** The stochastic method lets us distribute recommendations across genres according to our estimate of the user's tastes, while providing a correction mechanism in case our estimate happens to be wrong. (By design, unfavored genres will still appear once in a while, and positive user feedback will cause them to appear more in the future.) 73% of the users polled by Alkindi in user testing said they were satisfied with the engine's understanding of their taste by the time they finished using it. This number was higher for Alkindi than for any competing recommendation engine we tested.

Stochastic Choice of Recommending Criterion

Once a product cluster to recommend from is chosen, and similar users (who agree with the user on products in the product cluster) are identified, the engine must recommend a product liked by those similar users. This product can be chosen based on a range of different criteria: high average rating among the similar users, many high ratings, seen by many of the similar users, and so on. Each of these different recommendation algorithms will tend to produce different "types" of product from that cluster. For instance, using "many high ratings" will tend to produce more well-known products, and "high average rating" will tend to select niche products.

The engine chooses a criterion stochastically out of the available ones, favoring the criteria whose recommendations the user has liked in the past. The advantages of a stochastic approach are all present here, as the engine explores the range of possible recommendations (ensuring variety and robustness) and evolves to favor the kinds preferred by the user (responsiveness).

Recommendations and Predicted Ratings

Having picked a criterion on which to base its recommendation, the engine looks through the product cluster and recommends the highest scoring product according to the chosen criterion. (Products that the user has already rated, or that have already been recommended, are passed over.)

For each recommended product, the engine also returns a predicted rating --- the system's best guess for how a user will rate a product if he or she sees or purchases it. The engine computes this quantity based on the ratings of the user's cluster associated with the product cluster in which the product lies. If the product lies in several product clusters, then several user clusters are considered. (For example, to predict how a user will rate "Blazing Saddles," a comedy and a Western, we consider the opinions of those users who agree with our user on comedies and those who agree with our user on Westerns.) If the predicted rating is too low, the product is rejected.

Possible Further Work

Elements Not Yet Complete in Production System

The system described above has been fully tested as a prototype (including testing with real users, who compared the system with other, competing recommendation engines). While most elements are currently present in the production version currently available on the Alkindi web site (www.alkindi.com), the following are not yet fully implemented in production:

- **Full Alkindex**
- **General Distance Model** (current production version measures distance based on ratings only)
- **Predicted Ratings** (module that displays these is currently in testing)
- **Full Functionality of Selecting Products for Rating** (module that computes and uses a product's rating variances over some, but not all, user clusters has not yet been built)

In addition, many elements of the algorithm would benefit from additional tuning. There are many fixed parameters that play a role in the recommendation process. For example, only products that have been rated by a certain percentage of the users asked about them are eligible to appear on the first screen of products to rate. The exact percentage threshold is a parameter that can be adjusted. While we have good approximate values for all the parameters, and the most important ones have been optimized, we still need to tune the rest to ensure optimal performance.

Future Research and Development Projects

Recommendation From a Subset

Alkindi intends to build a module that allows the engine to function smoothly when the universe of products from which we can recommend is restricted. (For example, when working with a partner, we typically want to recommend only products offered by that partner). A prototype version of this module is currently in development.

Recommendation Based on Purchase Data

The Alkindi recommendation engine characterizes user tastes through rating data. Alkindi intends to build a module that allows us to make recommendations to users who have submitted no ratings, but for whom we do have purchase data. (For example, we could recommend videos to users for whom we have lists of previous rentals, without knowing how much they liked the videos they rented.) Prototype code that does this has been developed, and is currently being evaluated.

Cluster Initialization

The K-means clustering procedure we use to form clusters offline is iterative: we first guess at what the clusters might be, then improve our guess until it stabilizes. The outcome of the procedure can be improved by improving the quality of the first guess. Typically, the first guess is random. Inspired by clustering procedures used in bioinformatics, Alkindi has developed a novel way of making the first guess more representative of what the clusters are likely to be. A prototype clustering module based on this improved initialization procedure still needs to be developed.

Product Clustering

As mentioned earlier, Alkindi intends to develop general software for grouping together products that are likely to have predictive value; that is, where a user's rating for one product might reasonably predict that user's rating for another product. This project is at a very early stage and needs significant research.

Performance Metric

Optimization of Alkindi’s engine depends on having a way to evaluate its success. Standard methods do not attempt to measure the importance of 1) providing recommendations with proper attribute diversity; 2) instilling confidence in the user by demonstrating accurate predictions about the user's tastes; and 3) user comfort levels changing over time. User simulations are combined with empirical data and mathematical models in arriving at a single performance measurement. The algorithm is 75% complete, but as yet unimplemented.