Predicting User Ratings of an Item from Individual and Collective Data

Sitan Chen (sitanchen@college), Sami Ghoche (samighoche@college), Matt Rauen (rauen@college), Tom Silver (tsilver@college)

Brief Overview:

We aim to design an e-commerce recommendation algorithm that makes product recommendations to users based on other usersÕ ratings of similar products and the userÕs own past history of purchases. This idea emerged from the idea of constructing more precise information on the sizes of clothing items purchased online, so we will assume that product quality can range between two abstract extremes: too small and too big. In particular, we envision representing the space of all products on the site by a graph where each node represents an item, and connections exist between items that have been purchased by the same user. Edges are weighted by some metric indicating the degree of similarity between the nodes, determined by the distribution of ratings of those two nodes (e.g. a node consistently rated as Òtoo bigÓ and another rated as Òtoo smallÓ will be considered highly dissimilar). Edges will also have a confidence metric, which reflects the number of users that have rated both items and the variation between the differences of their ratings. The confidence metric (uncertainty value) will be computed such that a high number of ratings and low variation in the differences of ratings will yield a low uncertainty metric (this will allow us to use a shortest-path algorithm to traverse the graph). When a user is browsing for a specific item, the algorithm will look for the ÒshortestÓ path in the graph from that item to an item the user has already purchased, where length of a path is some function of the confidence between edges. The output is a function of the similarities along the shortest path, indicating the algorithmÕs prediction of how well the original item fits relative to the item that the user has already purchased. We hope to use machine learning to compare a userÕs rating of a purchased item to our predicted rating, and update the function that calculates confidence accordingly.

Feature Details:

User Satisfaction Prediction Algorithm: Given a collection of arrays, where each array represents the set of ratings associated with an item, weight the edges of a graph of nodes representing these items with a tuple $(f(\sigma, n), g(s))$, where σ is the standard deviation of the set of differences in ratings between two goods by each user that has purchased both goods, n is the number of users that have purchased both goods, s is the mean of these differences, and f, g are transformations where f is increasing in σ and decreasing, and g is increasing in s, which output uncertainty of our measurement of similarity, and our estimate of the similarity between the two goods. Given an adjacency list of edges with edge weights, as well as a node representing the item the user is interested in, the algorithm determines the minimal path from this node to a node representing an item that the user has already purchased and returns the length of this path and the similarity of the two items, obtained from combining (possibly multiplying) the similarity weights of the edges along the minimal path. In the case that a user is interested in selecting an item that is not in the same connected component as any of the items he/she has already purchased, a naive implementation of DijkstraÕs would not return any results. To fix this issue, we will consider at every step a Òvirtual

edgeÓ between the given node and the target nodes. The uncertainty value for that edge will be determined by a machine learning algorithm that will take into account brands and past successes of the algorithm, according to the correlation between user ratings and recommendation results. Machine Learning Algorithm for Projecting Virtual Paths To project a virtual link between a given node and target node with appropriate certainty, we can look at correlation between brands as an important indicator of the confidence of our jump. For example, suppose the current item of some brand A has a nominal size ÔaÕ and the target item of another brand B has a nominal size ÔbÕ. Suppose further that we know sizes for brands A and B have a strong correlation (meaning that size units are equivalent for both brands). Then we can return a low uncertainty value for this projected virtual link. This correlation metric can increase when more people buy items from those two brands and rate them according to what the nominal sizes would predict. Once we have an uncertainty value, we can compare that to the uncertainties of the other direct edges of the current vertex and continue our implementation of DijkstraÕs. Once our prediction algorithm is completed, we compare the correlation between our prediction and user ratings for non-jump predictions and for jump predictions and use that information to change our weights given to the number of people and the correlation between sizes needed for brands to be considered correlated. In the long run, we should be as confident about the uncertainty values produced by this algorithm as we are for the regular uncertainties of real edges.

Machine Learning to Calculate Uncertainty Values The uncertainty value for a given edge will be a reflection of the number of ratings and the variation among the differences of those ratings. The amount which the former is weighted relative to the latter will change as the program gathers more data about its success rate. The function to calculate the cummulative uncertainty value of a path will be linear. The scale factor in this function will be a reflection of how much uncertainty increases from edge to edge. This scale factor may also change from edge to edge based on the difference between the similarities of the nodes. For example, we may be more certain when we compare items of similar size than items of vastly different size. Recommendations for users A useful feature would be to recommend to users items that we predict would fit them nicely. For that purpose, the first step is to look for items they have already purchased and that they have rated above some threshold value r. Then we can traverse the adjacency lists for each of those items, looking for items whose fit algorithm prediction is higher than some threshold value f, and recommend those items to the user. A machine learning algorithm will make our choices of r and f return results that users will give better ratings to.

Recommendations for vendors Given an item (or a brand) and a list of users with size ratings, we would like to be able to recommend users for whom that item would fit well. IsnÕt this just the same as number 4 except we search for users that have the items that are found?

Draft Technical Specification

Graph

a. Build/update graph

Build: To build our database of items, we will define an empty graph to which we can insert nodes representing items.

Insert: A node will exist when a user purchases an item and rates it for the first time and will be inserted into an empty graph.

Update: If a node already exists and another user buys it and rates it, we want to update the graph of that item so that the adjacency list will contain the item in question and a list of all items previously purchased by the user (sorted by uncertainties). This also means merging two graphs.

Merge: To merge two graphs, we will add all nodes with all their lists in the smaller graph to the bigger graph and also sort the new graph items by popularity.

Traverse

Jumping

Fibonacci Heap Push: Add first node in lists of connections for the current node in the graph (because items in unordered lists are sorted)

Pop: Return the most confident path

Global variables (MENTION CONSTANT TIME ACCESS)

Sorted Lists of popularity of Brands

Sorted Lists of popularity of Items Overall Standard Deviation

Overall Connections/number of graphs/number of nodes/number of connections per node

Nominal Sizes of Items Nominal sizes of items will originally be obtained from the sizes that the vendor gives. These sizes will be updated as users give feedback for items. For example, if we have a ÒsmallÓ Abercrombie t-shirt, denote that size 1, and if a user rates this item as Òtoo largeÓ, we will update that number to 1.25 or something similar based on the number of users that give that rating. Updating nominal sizes will allow for more accurate virtual link projections (see Machine Learning Algorithm for Projecting Virtual Paths above).

ML Function Adjustments

Build correlation between brands matrix

Recommendations (for users)

Recommendations (for items)