ISSN 2250-3153

Implementing Cluster based Recommendation algorithm through Split Inversions

Shalabh Tewari , Rohan Joshi

Department of Computer Science and Engineering, Galgotia's College of Engineering and Technology

Abstract- Recommendation systems are important business applications with significant economic impact. In recent years, a large number of algorithms have been proposed for recommendation systems. We evaluate a wide range of recommendation algorithms. These algorithms include the popular user-based, simple search, collaborative filtering and item-based filtering algorithms. We have also devised a method to form clusters through split inversions.

Index Terms- Split inversions, clusters , value , collaborative clustering

I. INTRODUCTION

Recommendation Algorithm

The recommendation algorithms are of huge importance to online stores the more accurate they are, the more the online store will sell. Consider though the problems that must be solved by such a recommendation algorithm. A large online store like Amazon may have millions of customers and millions of items in stock. New customers will have limited information about their preferences, while more established customers may have too much. The data on which these algorithms work is constantly updated and changed. Customers are browsing the site and the prediction algorithm should take the recently browsed items into consideration, for example - it doesn't help if anyone is looking for a toy and all he get recommendation for Harry Potter books.
The biggest and most important criterion for these systems (apart from accuracy) is speed. The recommendation algorithm must produce suggestions within a second or so. After all, the user is in the process of displaying the store's home page where the ***recommendations*** will appear.

Traditionally, these recommendation algorithms have worked by finding similar customers in the database. In other words, they work by finding a set of customers who have bought or rated the same items that you have. Throw out the items you've already purchased and then recommend the rest.

One of the biggest innovations in online shopping - first introduced by Amazon - was the automatically generated recommendation.

Some of the currently used recommendation algorithm are:-

Item-to-Item filtering

This algorithm matches each of the current customer's purchased and rated items to similar items and then builds a list from those matched items. First of all, then, the web site must build a 'similar items table' by analysing the items customers tend to purchase together.

Here's how this works: for each item X in the catalog, find all customers C who purchased X. For each of those customers, find all items Y purchased by C and record that a customer bought X and Y. Then, for all pairs X and Y, calculate the similarity between X and Y in the same manner as for the collaborative filtering algorithm.

Although this calculation is fairly computationally expensive, it can be done beforehand.

Simple search

The next traditional algorithm is a fairly simple search algorithm. For example, if I buy Inferno by Dan Brown, the search algorithm would query the items database for other books by Dan Brown, thriller books by other authors, DVDs of movies made from Dan Brown books, and so on so forth.However this kind of recommendation algorithm is least used as it is very inefficient.

Collaborative Filtering

One of the earliest such algorithms is known as collaborative filtering. In essence, the algorithm represents each customer as a vector of all items on sale. Each entry in the vector is positive if the customer bought or rated the item, negative if the customer disliked the item, or empty if the customer has not made his or her opinion known.

Most of the entries are empty for most of the customers. Some variants factor in the popularity of the items to bump up the significance of items that are less popular or familiar. The algorithm then creates its recommendations by calculating a similarity value between the current customer and everyone else. The most acceptable way to do this is to calculate the angle between the vectors - the simplest method being to calculate the cosine using the dot product divided by the product of the vector lengths. The larger the cosine, the smaller the angle, and therefore the more similar the customers.

This process is computationally expensive. There are usually a lot of customers and a lot of calculations have to take place very quickly. There are techniques to reduce the computation (by sampling the customer base or ignoring unpopular items, for example), but in general it's always going to be expensive to calculate recommendations this way.

II. RESEARCH ELABORATION

Cluster Based Recommendation algorithm

Another traditional prediction algorithm involves the use of cluster models. Here the goal is to prepare the customer base by dividing it into clusters and then to assign the current customer to

one of the clusters, in theory choosing the cluster with the most similarity. Once the cluster has been identified, the recommendations come from the purchases and ratings from other customers in that particular cluster.

Although the choice of cluster works in roughly the same way as the classification algorithm (we assume that we can calculate a characteristic vector that describes the cluster in much the same way that there is a vector per customer), the real meat of the algorithm is in the creation of the clusters.

In general, clustering of customer data is done through a heuristic: start off with some number of empty clusters, assign a randomly selected customer to each, and then assign the other customers to the clusters according to similarity. Since the initial clusters are essentially randomly created, sub-algorithms must be used to merge or split clusters as they are being built up.

Using cluster models is less computationally intensive at the point where you need to make recommendations quickly for a customer. After all, there's less work to be done to find a similar cluster rather than a similar customer. If you like, most of the work is done up front in the creation of the clusters themselves.

Unfortunately, this particular method tends to result in low quality recommendations since the purchases/ratings are averaged out within a cluster. No longer is a particular customer matched to the most similar customer, but instead to the average of a large group of customers. Certainly the number of clusters can be increased to refine the matching, but then you run into the possibility of increasing computation time. To remove the disadvantage of assigning randomly selected customers to empty clusters we have come with solution to this problem. The assignment of customers to a cluster can be based on a parameter . This parameter can be computed through some user input that would be common for all customers. Eg In our case the parameter is vote .User has to vote for the best book among some most famous books shown to him. There are 5 books in our case :-

Based on voting for the best book by individuals

BOOKS	NO. OF VOTES	RANK
The Hobbit	1000	1
Inferno	800	2
The Lost Symbol	500	3
Hardy boys	250	4
Godfather	125	5

USER 1

BOOKS	RANK
The Hobbit	2
Inferno	3
The Lost Symbol	1
Hardy boys	4
Godfather	5

2 3 1 4 5 number of split inversions =2
that are (2,1),(3,1)
2/c(n/2)=2/10=0.5
value(V)=0.5

USER 2

BOOKS	RANK
The Hobbit	2
Inferno	3
The Lost Symbol	1
Hardy boys	5
Godfather	4

2 3 1 5 4 number of split inversions =3
that are (2,1),(3,1),(5,4)
3/c(n/2)=3/10=0.33
value(V)=0.3

USER 3

BOOKS	RANK
The Hobbit	1
Inferno	3
The Lost Symbol	2
Hardy boys	4
Godfather	5

1 3 2 4 5 number of split inversions =1
that are (3,2)
1/10=0.1
VALUE(v)=0.1

USER 4

BOOKS	RANK
The Hobbit	1
Inferno	3
The Lost Symbol	2
Hardy boys	4
Godfather	5

1 3 2 4 5 number of split inversions =10

that are (5,1)(5,2)(5,3)(5,4)
(2,1)(3,1)(3,2)(4,1)(4,2)(4,3)

VALUE(v)=1

USER 5

BOOKS	RANK
The Hobbit	1
Inferno	2
The Lost Symbol	3
Hardy boys	4
Godfather	5

1 2 3 4 5 number of split inversions =0

VALUE(v)=0

The values received are 0.5 ,0.3, 0.1, 1, 0 . V will always lie between 0 and 1 . Best Case is that which has no split inversions Worst Case has split inversions equal to C(n,2)

III. RESULTS

The above model was shown for 5 users but there are millions of users accessing the same website such as amazon ,flipkart . The above method can be used as Books recommendation algorithm since the value will always lie between 0 and 1.

USER	VALUES
user 1	0.5
user 2	0.3
user 3	0.1
user 4	1
user 5	0
user 6	0.1
user 7	0.15
user 8	0.3
user 9	0.4

Let the above users have the above values as given in the table

How the recommendation system will work? User3 and User 6 have the same values for V .It means they have the same liking .Hence Auto-Suggestion will suggest the books to User3 that have been liked by User6 or even bought by User6.

Similarly in case of user2 and user 8 both have 0.3. It means they have the same liking .Hence Auto-Suggestion will suggest the books to User2 that have been liked by User8 or even bought by User8. It can happen vice versa also.

How the value is obtained?

The value (V) is obtained by V = (Number of split inversions) / (Total split inversions possible)

How to calculate split inversions?

The split inversions forms an important part of our algorithm. We can calculate these by merge sort. Merge sort leads to interesting applications besides sorting. One of the most interesting is counting the number of split inversions between two arrays. Assuming there are two arrays, the number of split inversions counts the number of swaps one array needs to transform into the other array. A split inversion count forms a space over the two arrays and therefore one can accomplish measurements with it.

Counting Inversions: Brute Force

A brute force algorithm would be to compare each element in both arrays to each other, and determine if they would need to be swapped. This is O(n2) (quadratic in n). Can we do better?

Counting Inversions: Divide And Conquer

In fact, we can do much better. The key point to note that is that we can get this information for free during the merge stage of the Merge Sort.

In the merge stage, two sorted arrays are merged into one sorted array. Lets call these arrays A and B. Then the number of split inversions involving an element y belong to B of the second array is precisely the number elements lefts in the first array A when y is copied into the temporary buffer. Adding up these split inversions during the merge stage will result in the total number of split inversions.

REFERENCES

[1] http://www.techradar.com/news/internet/how-recommendation-algorithms-know-what-you-ll-like-1078924

[2] http://en.wikipedia.org/wiki/Main_Page

AUTHORS

First Author – Shalabh Tewari , Student , Galgotia's College of Engineering and Technology , shalabh.hsc@gmail.com
Second Author – Rohan Joshi , Student , Galgotia's College of Engineering and Technology , rohanjoshi81@gmail.com