

Addressing Coldstart Problem in Recommender Systems

Shisagnee Banerjee

IIIT-D-MTech-CS-GEN-14-023

May 15, 2016

Indraprastha Institute of Information Technology Delhi
New Delhi

Thesis Advisor

Dr. Angshul Majumdar

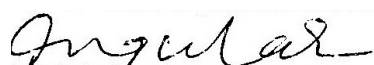
Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Computer Science

© Banerjee, 2016

Keywords : Recommender Systems, Cold-Start Problem, Demographic Data

Certificate

This is to certify that the thesis titled “**Addressing Coldstart Problem in Recommender Systems**” submitted by **Shisagnee Banerjee** for the partial fulfillment of the requirements for the degree of *Master of Technology in Computer Science & Engineering* is a record of the bonafide work carried out by her under my guidance and supervision at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.



Dr. Angshul Majumdar

Indraprastha Institute of Information Technology, Delhi

Abstract

In the following three chapters of my thesis , I have applied several methods to solve the coldstart problem in Recommender Systems. The coldstart problem is the situation where a user or item is new to a website and recommendations need to be given to the new user or the new item needs to be recommended. The framework proposed in my work uses user's demographic information and the genre of movies for creating the model. The chapters propose a framework, parallelize it and also improve predictions and allows speedup by using different techniques to meet the given goal (alleviate the coldstart problem).

Acknowledgments

Towards the completion of my Masters degree, I would like to pay my heartfelt tributes to people who contributed in many ways. After expressing gratitude towards God and my loving parents, I would like to thank my advisor Dr. Angshul Majumdar for his support and guidance throughout the journey. His constant guidance and input have helped me prosper towards becoming a confident and improved personality. He made great efforts in supporting me in all possible ways. His advise and guidance not only helped me in my work here but has triggered my interest in this domain for future. This section can not be complete without a vote of thanks to academic department for their help and never ending support .

Contents

1	Introduction	2
2	Addressing Cold-Start Problem using Neural Net	5
2.1	Introduction	5
2.2	Literature Review	6
2.3	Proposed Method	8
2.4	Experimental Results	9
2.4.1	Data Description	9
2.4.2	Evaluation Metric	11
2.4.3	Comparitive Results	12
2.5	Conclusions	14
3	Divide and Conquer Approach Using Sparse Representation Classifier	20
3.1	Abstract	20
3.2	Introduction	20
3.3	Literature Survey	22
3.3.1	Divide and Conquer	22
3.3.2	Sparse Representation Based Classification	23
3.4	Proposed Algorithm	25
3.4.1	Proposed algorithm for Divide and Conquer using SRC	26
3.5	Experimental Setup and Evaluation	28
3.5.1	Description of Dataset	28
3.5.2	Evaluation Measures	29
3.5.3	Results	29
3.6	Conclusion	30
4	Nearest Subspace Classifier	32
4.1	Introduction	32

4.2	Background	32
4.2.1	Nearest Subspace Classifier	32
4.2.2	Dictionary Learning	34
4.3	Proposed Method	34
4.3.1	Algorithm DL-NS	35
4.4	Experiments and Results	36
4.4.1	Description of Data	36
4.4.2	Evaluation Metrics	37
4.4.3	Results	38
4.5	Conclusion	43
5	Conclusion	44

List of Figures

2.1	Genres of Movies	8
2.2	Encoding Occupation Information	9
2.3	Comparison of Precision for 100K Movielens	14
2.4	Comparison of Recall for 100K Movielens	15
2.5	Comparison of Precision for 1M Movielens	15
2.6	Comparison of Recall for 1M Movielens	16
2.7	Comparison of MAEs of all the papers reviewed in with Our Methods(NN-User, NN-Item)	16
2.8	Confusion Matrix for NN-User	17
2.9	Confusion Matrix for NN-Item	18
3.1	Parallel Training	27
4.1	Diagram representing how NS works	35
4.2	Regularisation VS. MAE and RMSE	40
4.3	Precision and Recall VS. No. of Recommendations	40
4.4	Confusion Matrix for MovieLens 100K	41
4.5	Confusion Matrix for MovieLens 1M	42
4.6	Comparison of Methods and their Computational Time	43

List of Tables

2.1	Techniques and Results	6
2.2	Techniques and Results Continued	7
2.3	Organising The Classification Problem	10
3.1	Comparison of Classification Accuracies of SRC, Neural Net and SVM	30
3.2	Comparison of DC-SRC with other popular Divide and conquer methods	30
3.3	Results of DC-SRC	31
4.1	Precision And Recall	38
4.2	DL-NS Results	38
4.3	NS Results for 100K	39
4.4	NS Results for 1M	39

Chapter 1

Introduction

Overview and Research Motivation

Today, with the explosion of e-commerce companies the amount of choices available to any user is humongous. Often it is not possible for a user to browse all possible items listed online as he/she might have done in a shop, the scale is too large. Thus there is a distinct feeling of missing out on what is the best product, or facing frustration at not finding something attuned to ones tastes. If the user does not find the required products they will not purchase / rent this in turn would mean loss of revenue to the e-commerce portal. To ameliorate this problem, e-commerce portals have become proactive; they recommend articles / items to the users based on his/her prior preferences and preferences of other similar users. This is called a recommender system; the de facto approach for recommendation today is based on collaborative filtering. To summarize collaborative ltering, the e-commerce portal collects the users choices on different items these may be ratings (explicit) or browsing/purchase history (implicit). When organised as a matrix (users as columns and items as rows), it becomes a high dimensional albeit a sparse matrix. It is high dimensional because there are typically hundreds of thousands of users and items of the same order.

In a collaborative filtering, the basic idea is to nd items rated highly by similar users and recommend these. The assumption being that, users having similar choices in the past are likely to

have similar choices on new items as well. Broadly speaking there are two approaches in collaborative filtering. The neighbourhood based models, typically find similar users by computing some similarity measure between the users and use the similarity value as linear interpolation weights to compute the expected rating of other similar users. This is called user-based recommendation. Similarly, one can also have item-based recommendation where the similarity between items are computed instead. Both of these fall under the neighbourhood (similarity) based model. The other approach is more abstract and is based on latent factor modelling. The assumption here is that the users and items have some common factors that decide the choice of the user, e.g. movie choices are typically based on director and star cast, similarly book choices are largely based on authors or subjects. Latent factor modelling is abstract, but mathematically powerful and yields more accurate results. However the pressing problem for all recommender systems is the data sparsity, the user-item choice matrix is parsimoniously filled. Typically no individual rates more than one percent of the items. However ameliorating the sparsity problem is not our topic of interest. We look at the more challenging scenario of cold-start. When a new user becomes part of the e-commerce portal, the system has no information about the users prior choices; hence the name cold-start. But it is the responsibility of the e-commerce portal to start suggesting items since they do not want to lose users. Thus in the cold start problem, the challenge is to recommend items to the new user when there is no prior choices available from the user. In almost all e-commerce portals, certain metadata about the user is available from the sign-up process. For example the users age, gender, occupation, locality, etc. are known to the portal. Similar metadata regarding the item is also available, for example the genre information in movies. Such associated information has been used before for improving collaborative filtering [1]. We propose to use the already available metadata to address the cold-start problem. This too has precedence [2], but the approach followed here is different from prior studies. Based on the metadata a binary feature vector is created for each user. Similarly using the item metadata a binary feature vector is created for the item. The rating is treated as a target class and the concatenated user-item binary vector acts as the feature. Based on this model, we can learn any classifier. When there is a new user or an item, the corresponding feature vector is created and fed into the classifier. The resulting class is assumed to be the choice / rating. The decision

regarding recommending / not-recommending is based on the predicted class (choice / rating). Most prior studies (as can be found in [2]) concentrate on the user cold start problem; the item cold-start problem has not received much importance in the past. Electronic Commerce is changing at a rapid pace and we believe that the item cold start problem is equally important. A new item does not have any ratings and hence will not readily interact with the recommendation system. Most e-commerce customers would like to be recommended with the latest items, be it movies, books or clothes. In this work, we provide a unified framework for addressing both the user and item cold-start problems. The following three chapters deal with :

- Chapter 2 ► Framework proposed,
- Chapter 3 ► Parallelization using divide and conquer approach, and
- Chapter 4 ► Removing the class imbalance and reducing computational time

Chapter 2

Addressing Cold-Start Problem using Neural Net

2.1 Introduction

In this work, we propose a unified framework to address the user and item cold-start problem in recommender systems. We make use of the users' and items' metadata and encode them as binary features. The available rating is treated as the class label. The formulation is general, but is particularly suitable for addressing the cold-start problem. We compare our proposed technique with prior works; we found that the proposed method yields considerably better results in terms of Precision and Recall.

2.2 Literature Review

Table 2.1: Techniques and Results

References	Techniques	Datasets	Results
Safoury et al. [20]	Utilizes the training file of each attribute to calculate the frequency of all items rated by users having similar attribute value.	MovieLens 100K	MAE: 0.9441, Precision:0.09(top 30), Recall: 0.244(top 30)
Pereira et al. [18]	Co-clustering users and learning models, simultaneously based on demographic data	MovieLens 100K, Netflix, Jester	NMAE: 0.198 (MovieLens 100K), 0.195(Netflix), 0.2(Jester)
Gantner, et al. [8]	K-NN regression,Attributes Used are : genres(Movielens), Directors, actors, credits (IMDB)	MovieLens (1 Million), IMDB	Precision: 0.3 AUC : 0.7 (directors+genres)
Lika, et al. [15]	Finding neighbours using Demographic data, Each attribute like age, occupation is dealt with separately.	MovieLens	MAE: 0.75 to 1.1 (for different attributes)
Lam, et al. [11]	Parameter estimation using EM Algorithm from triadic aspect model and,the vector aspect model	MovieLens (1Million)	MAE: 0.69 NMAE: 0.45 Sensitivity:0.85 Specificity:0.51
Yu, et al. [23]	Nearest neighbor approach based on similarity of ratings in similar domain	EachMovie	Average MAE of 0.2

Table 2.2: Techniques and Results Continued

References	Techniques	Datasets	Results
Bobadilla, et al. [3]	Neural Networks on similarity of user ratings	Movielens (1 million)	MAE: 1.25 Precision : 0.75 Recall: 0.66
Ahn et al. [1]	Similarity measure on very limited ratings using Impact, Proximity and Popularity as well as co-rated items among users Baseline Recommendation used.	Movielens, Netflix, Jester	MAE : 0.77 (for 25 items, Movielens) MAE : 0.785 (for 25 items, Netflix)
Liu et al. [16]	Proposes a similarity measure composed of Proximity,Significance, Singularity. Assumes new user has rated some prior movies.	MovieLens and Jester	MAE: 0.641(MovieLens) 0.821 (Jester)
Leung et al. [14]	Rating data is converted to the transactional database of association rule mining, fuzzified by fuzzy memberships and then used for prediction.	MovieLens	MAE: 0.636 (MovieLens) 0.899(Jester)
Cuong et al. [6]	Demographic Information of users is used to find neighbours,their ratings are considered representative ratings, else a similar users rating is considered representative.	MovieLens	MAE: 0.701
L.H. Son [21]	Both user demographic data and user ratings are used to calculate final similarity matrix for rating prediction.	MovieLens Jester	MAE: 0.697 (MovieLens), 0.895(Jester)

2.3 Proposed Method

The method proposed in this paper is simple and can be easily extended. The demographic information of users is available to us as a part of the sign-up process into the e-commerce portal; this includes namely gender, age, occupation and location in the form of zip-code. We do not think that zip code is a relevant piece of information. It only says where the person resides and is not likely to determine his/her taste in any fashion. We make use of the other three pieces of information gender, age and occupation.

In this work we are interested in movie recommendation. There are many features for the movies available - movie id, movie title, release date, video release date, IMDb URL, Genre (Action, Adventure, Animation, Childrens, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western). All except the genre information is irrelevant for choosing a movie. We have considered only the genre information here. First we will describe how the movie features have been generated. We generate a binary vector from the genre, the vector contains a 1 if the movie belongs to that genre or 0 otherwise. The vector is shown in Figure ??.

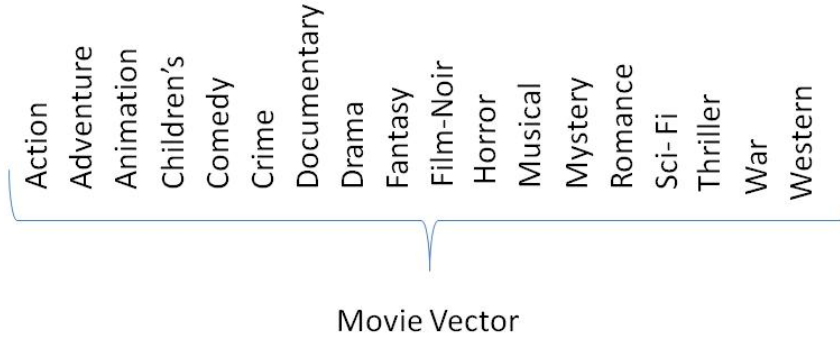


Figure 2.1: Genres of Movies

Say a movie like Shawshank Redemption is tagged as crime and drama in IMDB. The corresponding feature vector will be $[0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0]T$; It has 1s corresponding to crime and drama and 0 everywhere else.

We now discuss about encoding the user demographic information. Encoding the gender information is the simplest. It is a tuple encoded as $[1,0]T$ for male and $[0,1]T$ for female.

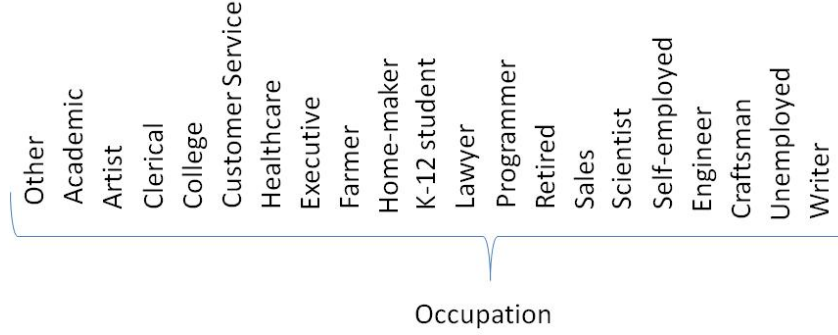


Figure 2.2: Encoding Occupation Information

To encode the occupation information we have an ordered representation of the different occupations as in Figure ?? . For a particular user, one of the occupations is 1 the rest are 0s.

To encode the age information we divide the users into several ranges; more specifically into set into 8 groups (7-14, 14-21, 22-28, 29-36, 37-48, 49-55, 56-65 and 66-73). The groups are divided keeping in mind the relevance of mentality of users according to age. The particular age group, where the individual belongs to is 1 and the rest of them are 0s.

The users are represented by binary vector of size 31 (gender-2, occupation-21 and age-8). The item is expressed as a binary vector of size 19 corresponding to 19 genres.

We model cold-start as a classification problem. The rating is the class. It may be 0 (dislike) or 1 (like) or can be more continuous say on a 5 point or 10 point scale. Each user rates an item; we represent the user-item combination simply as a concatenation of the user vector followed by the item vector. This leads to a feature of size 50 (31+19). The corresponding class is the rating. The user-item feature and the corresponding rating / class is organised in Table 2.3.

2.4 Experimental Results

2.4.1 Data Description

Movielens 100K

Classes: 5 (ratings 1 to 5)

Table 2.3: Organising The Classification Problem

	User1	User2	User3
Gender1	0	1	1
Gender2	1	0	0
Age1	0	0	0
Age2	1	0	1
.	0	1	0
.
Occupation1	0	1	0
Occupation2	1	0	1
.
.
Genre1	0	1	0
Genre2	1	0	1
.
.
Rating	3	5	2

Description: Rating dataset has 100000 ratings by 943 users on 1682 items. Each user has rated at least 20 movies. Users and items are numbered consecutively from 1. The data is randomly ordered. Item information consists of movie id, movie title, release date, video release date, IMDb URL and 19 genres. User information consists of user id, age, gender, occupation, zip code The ratings data is split into 80% and 20% for training and testing.

Movielens 1 Million

Classes: 5 (ratings 1 to 5)

Description: Same as previous dataset except there are 6040 users and 3952 items. The user data has 20 occupations and 7 age groups.

These datasets are available in the website grouplens.org. There are larger datasets on movie recommendations as well, such as the Movielens 10 million and the Eachmovie, but these datasets do not contain the user or item metadata. This information is required for our proposed technique. The datasets we used are the only ones having this information available.

5 fold cross validation was done on these datasets. This is the standard procedure for evaluating these datasets, however the folds defined in the datasets could not be used since they were not

made for the cold-start problem.

In principle any classification algorithm can be used for solving the problem. But we have used a Neural Network for our work. The NN used was a 1-hidden layer network with conguration 10 nodes. The activation function is sigmoid; the learning rate is 1; number of epochs = 30; batchsize = 20 and momentum = 0.9.

2.4.2 Evaluation Metric

To compare the prediction accuracy the standards metrices of precision and recall are used. Precision measures the probability that a recommended item is relevant to the user. We categorize an item in test data as relevant if its rated 4 or higher (out of 5) by the user.

$$Precision = \frac{t_p}{t_p + f_p}$$

where, defines true positive i.e. items that are relevant to the user and are selected by RS; defines false positives i.e. irrelevant items recommended to the user. Precision usually drops with increase in length of recommendation list.

Recall (25) measures the chances that a relevant item will be selected for recommendation.

$$Recall = \frac{t_p}{t_p + f_n}$$

where, defines true positive; defines false negatives i.e. relevant items not recommended to the user. The recall value usually increases with increases in recommendation list length.

Many prior studies on cold-start problems have reported results in terms of mean absolute error (MAE). This metric tells us how much the predicted rating deviates from the actual rating on the test set. This may not be the best possible metric. Imagine a situation where the recommender systems algorithm predicts ratings that is shifted by some constant amount (compared to the actual rating). In such case, the MAE will be poor but the recommendations will be actually

perfect. Precision and Recall are more standard evaluations metrics in information retrieval. We will use them to evaluate different techniques.

2.4.3 Comparative Results

We have compared our proposed technique with two prior studies on pure cold-start; they are the works of Safoury and Salah [20], and Lika et al [15]. Both these studies use a mixture of classification algorithms and similarity based techniques along with user metadata for addressing the cold start problem.

Figures 2.3, 2.5, 2.4 and 2.6 show the results for Movielens 100K and 1M datasets respectively. The algorithms compared against are only for the user cold start problem. We have shown the plots for both user and item cold start.

In the paper by Safoury and Salah, the frequency of every items rated by a certain demographic attribute (like gender , age group or occupation) is calculated first. This means for each item and demographic attribute pair there is a frequency of rating values from 1 to 5. For a new user, first based on a demographic attribute the appropriate category is selected. Then the rating for an item is the frequency weighted average of the rating values for all the other users who belong to the same category as the new user and have rated that item. This is a simple neighbourhood approach and our model significantly outperforms this method in terms of precision and recall. Figures 2.3, 2.4 compare the precision and recall obtained by the three methods used in [20] based on category Gender, Age and Occupation with the results obtained by our method for the 100K Movielens dataset.

In the paper by Pereira et al. [18] the prediction of rating has 3 phases. In the first phase a model based on demographic data is designed. This model is then used to group users into certain categories. Each new user is assigned an estimated category by this model. The users belonging to this category are now taken as the neighbours of this user. In the second phase a distance measure is used to find the similarity between a new user and each of its neighbours. The ratings are generated by calculating the weighted average over the neighbourhood with the similarity measures as the weights to the actual ratings of the users. Figures 2.5 and 2.6 show

the results of precision and recall from both papers discussed along with our results on the 1 Million Movielens dataset.

Some general observations about the results displayed are that as the number of predictions considered is increased the precision value falls while the recall value rises. To make sense of this it is important to go back to how precision and recall is defined in this case. Precision and Recall are calculated via the standard methods [9]. It can be observed from the figures that, precision decreases while recall improves with increasing length of recommendation list. The reason being, Precision in this case is Out of total recommended movies, how many are relevant. So when we recommend fewer movies the percentage of correct recommendation is high. With the increase in number of recommendations, our accuracy decreases not only because of the obvious factor but also because there are very less movies actually rated by user. So while we are recommending 30, there might not exist 30 rated movies for the user, thus rendering them irrelevant. Recall is Out of total relevant movies, how many are recommended by us. So while there might be 100 relevant movies for a user, when we recommend just 10, recall is very less. It increases as more movies are recommended.

In another paper by Le Hoang Son [21] different papers dealing with the particular problem at hand have been reviewed and compared. Figure 2.7 compares the MAE values obtained by those papers with the values obtained by our method. We achieve comparable MAE values to the papers reviewed although it is slightly higher. However our method has considerably faster training time compared to many methods. Also since we do not use a neighbourhood concept to calculate similarity with all neighbours during prediction our prediction time is much less computation intensive than the nearest neighbour methods. Also we do not selectively choose attributes for prediction but rather use all the demographic data available leaving it to the model to learn underlying patterns.

The time taken by user coldstart for 100K dataset 38.5469 seconds for training and an additional 0.0781 seconds for predicting. The time taken by item coldstart for 100K dataset 55.8281 seconds for training and an additional 0.0782 seconds for predicting.

The time taken by user coldstart for 1M dataset 394.5 seconds for training and an additional 0.3594 seconds for predicting. The time taken by item coldstart for 1M dataset 419.5938 seconds

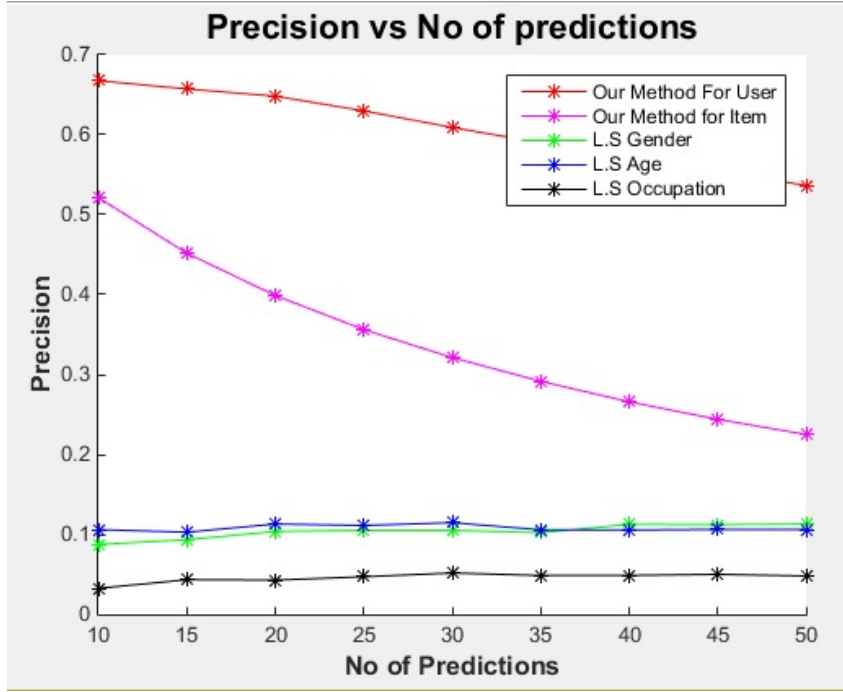


Figure 2.3: Comparison of Precision for 100K Movielens

for training and an additional 0.2343 seconds for predicting. The PC used was Intel CORE i7 2GHz, 8GB RAM.

Even as we have justified the use of metrics like Precsion and Recall, we have also included confusion matrices in the results to give a better visual understanding of classification of the 5 class problem. Follwoing are the Confusion Matrices for the methods NN-User and NN-Item.

2.5 Conclusions

In this chapter we have presented a simple yet efficient method to alleviate the coldstart problem. This can be used for both item and user cold-start problems. Prior studies mostly concentrated on solving the user cold-start problem.

We use the user and item metadata information for addressing the said problem. Information like the users gender, occupation and age are encoded as binary vectors; similarly information about the movies genre is encoded as a binary vector. The concatenation of the two is used

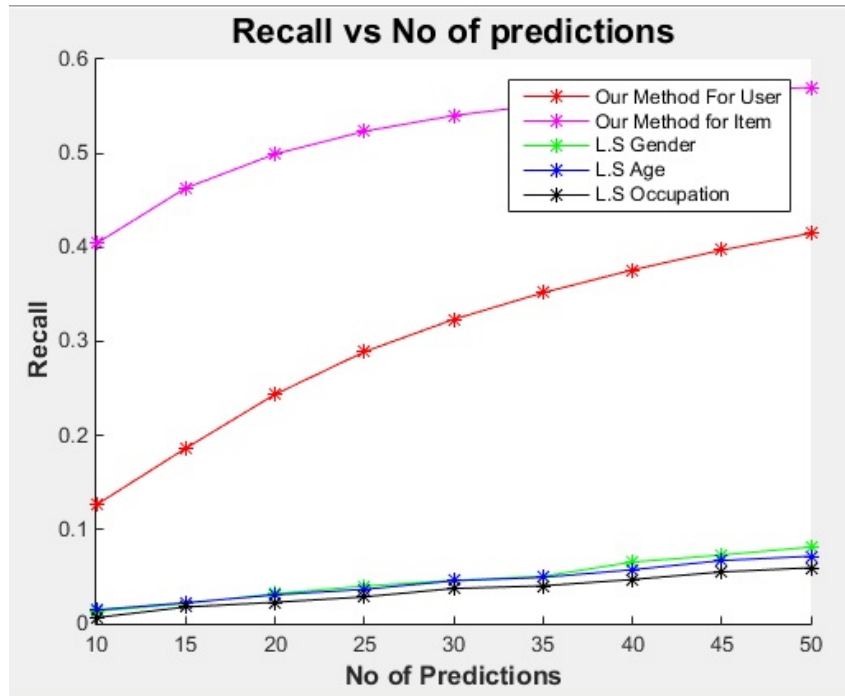


Figure 2.4: Comparison of Recall for 100K Movielens

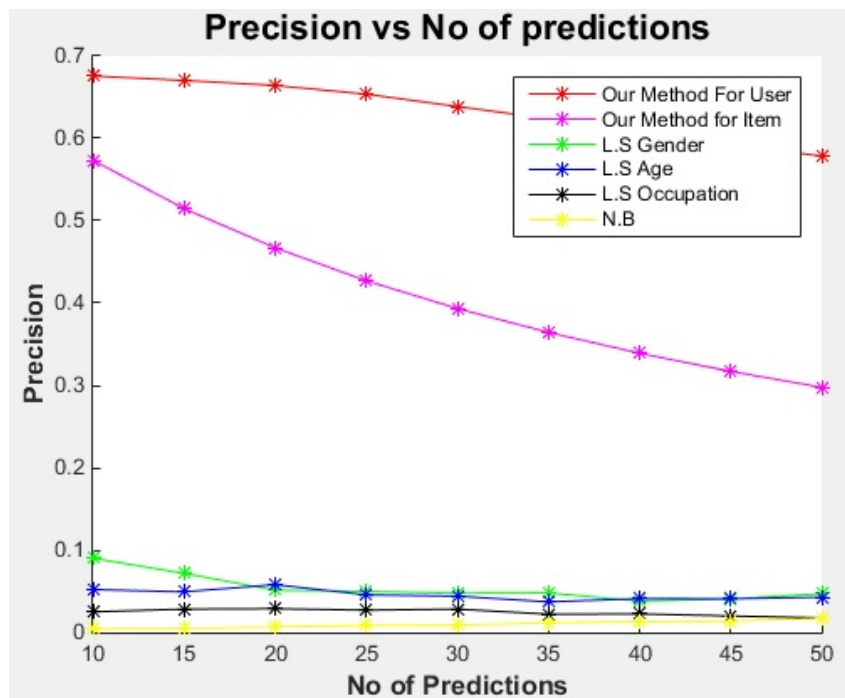


Figure 2.5: Comparison of Precision for 1M Movielens

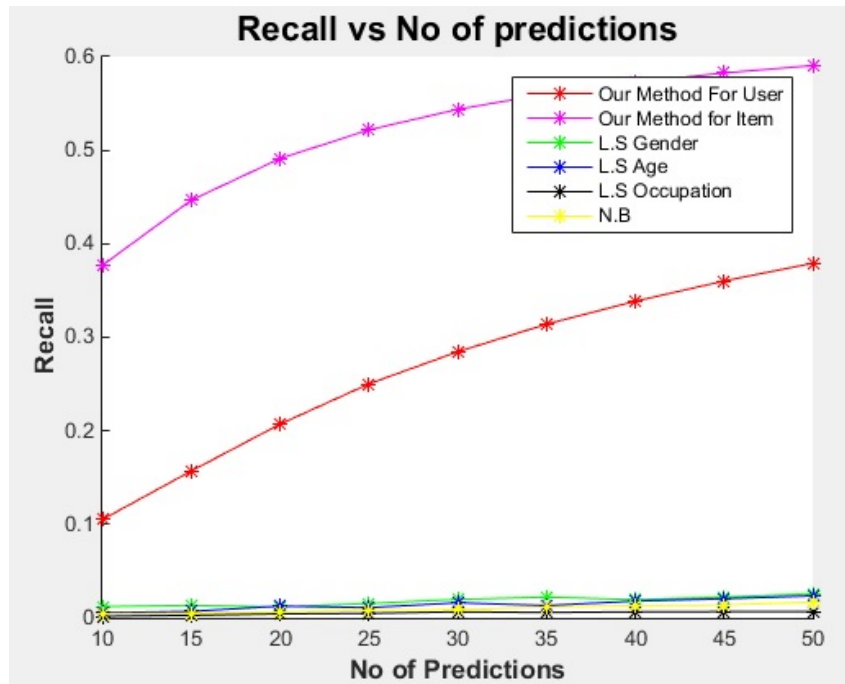


Figure 2.6: Comparison of Recall for 1M Movielens

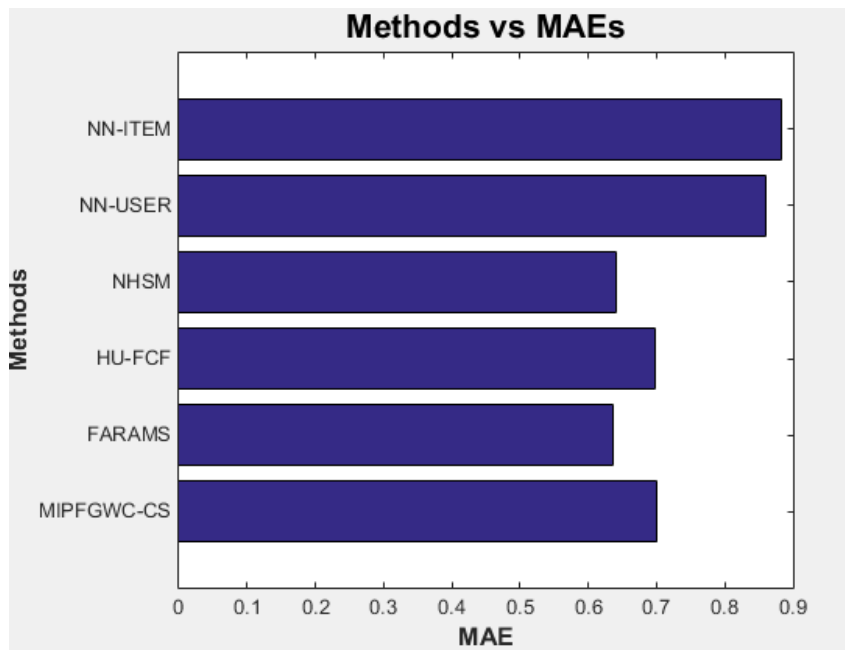


Figure 2.7: Comparison of MAEs of all the papers reviewed in with Our Methods(NN-User, NN-Item)

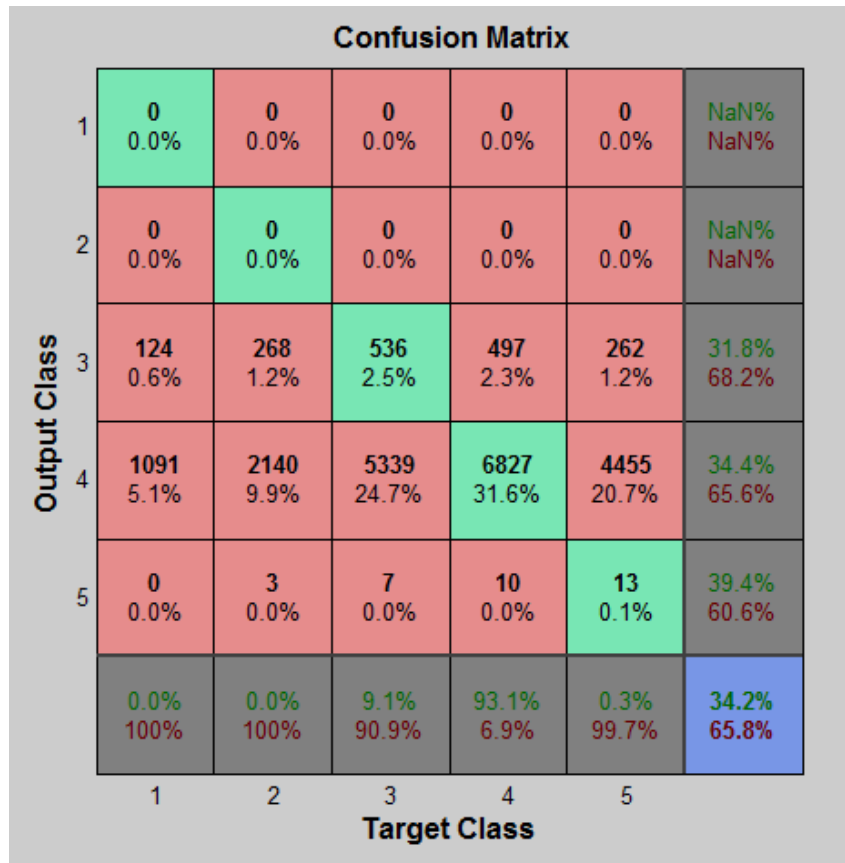


Figure 2.8: Confusion Matrix for NN-User

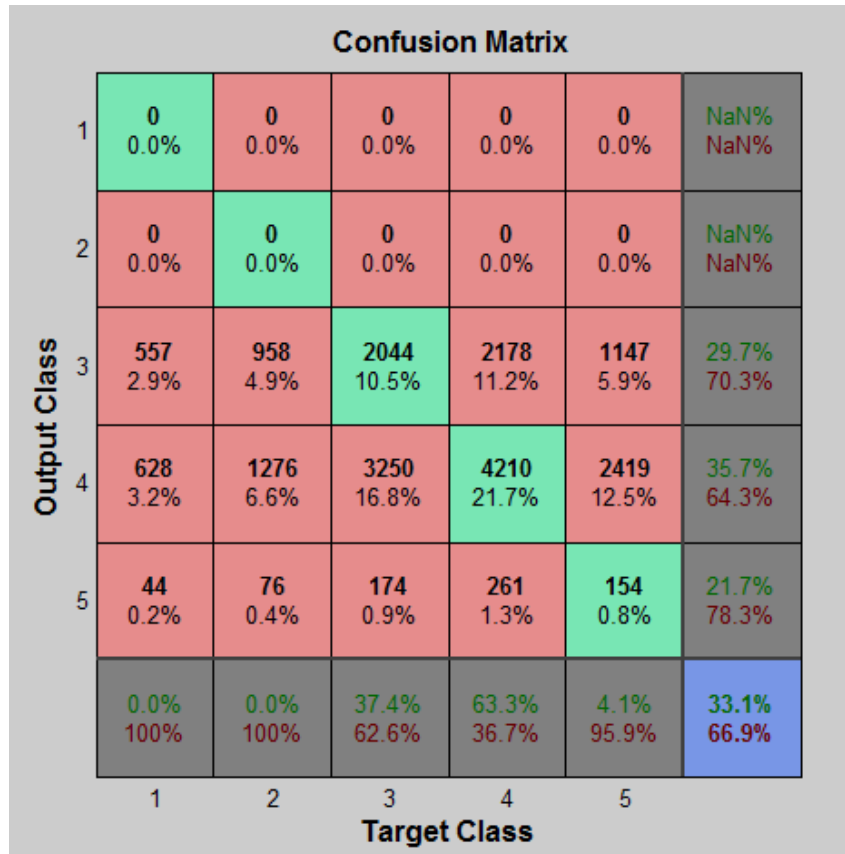


Figure 2.9: Confusion Matrix for NN-Item

as the feature and the corresponding rating is the class label. Recasting the cold-start as a classification problem is the main contribution of the paper. We used a simple single hidden layer neural network as the classifier.

We compared our technique with two existing works; both of them have been published in the last two years. The comparison was carried out on two popular datasets, viz. Movielens 100K and 1M. These are the only datasets known to us where the relevant user and item metadata information is available. Experimental results show that the precision and recall values from our proposed method are considerably superior compared to the techniques compared against. In this work we propose a unified framework for solving the user / item cold start problem. Our method depends on the user / item metadata. We have used users demography and movie genre information; but it can be easily extended to incorporate other relevant information. Although we have shown results on movie datasets, the framework proposed here in is applicable to other kinds of items.

Chapter 3

Divide and Conquer Approach Using Sparse Representation Classifier

3.1 Abstract

In this chapter, we have proposed an algorithm for classification using a Sparse Representation Classifier (SRC) following a divide and conquer approach. The training data is divided into various percentages of partitions and the classifier is applied on them in parallel. The division of data is done using a distributed computing approach which uses distributed memory. SRCs are subsequently used to classify each of the subtasks. The results from all the classifiers are combined to find the appropriate classification decision. Our method exhibits a large decrease in computational time and also surpasses considerable benchmarks in terms of Classification Accuracy.

3.2 Introduction

Machine Learning algorithms have recently made big leaps in producing excellent results on a wide variety of data. However, the explosion of social networking websites as well as ecommerce

websites today produce a massive amount of user data. Along with that increasingly powerful computers and advanced scientific instruments are constantly generating datasets of inconceivable sizes. Along with size the nature of the data is also extremely varied, including images, text, audio as well as user information data, etc. While large datasets have been of immense value in machine learning algorithms like deep learning, the computational costs and the training times involved are significantly high to the point of being infeasible in computers of moderate power. In most classification algorithms when the size of the dataset increases beyond a point, memory bottlenecks cause them to significantly slow down. In some extreme cases, it might become entirely infeasible to train classifiers on such huge datasets due to memory and computation costs. In others, the training times may be of the order of many days which might be a major problem in testing with varying hyper-parameters. One solution to this problem is to train multiple classifiers. The chief concern here is that the discrete learning systems are independent or correlated. If they are correlated, they can be used serially to incorporate the loss of the previous one. This loss is then rectified in the subsequent classifiers. This is the method used in boosting algorithms. These are not of interest to us as they do not eliminate the problem being addressed by us. The second approach is to divide the data into subsets and train independent classifiers on them. These are then recombined by some method like voting or minimum error. The advantage of this divide and conquer approach is that the independent classifiers can be trained in parallel on smaller sets of data rather than on the large original dataset. Distributed computing using distributed memory, helps in materializing the above mentioned concept. The time saved in this is quite huge and it can be computed on a cluster of less powerful machines in a much lesser time than a single extremely high end machine. Using this technique, neural nets and SVMs among others have been proved to work well on a large variety of classification tasks.

Our proposed idea is implementing this approach using Sparse Representation Classifiers. The problem solved by the sparse representation is to search for the most compact representation of data in terms of linear combination of atoms in an over-complete dictionary. The impetus of using SRC stems from the fact that when optimal representation of atoms is adequately sparse, its convex optimization can be easily calculated. Although Sparse Representations of data have

recently been widely used to create compact representation of data in fields like signal processing, its use in such a distributed environment to classify data is a novel approach.. Our work claims its viability by demonstrating that the technique performs well on several standard datasets.

In the next we have presented a survey of recent work in the areas of divide and conquer approaches in machine learning as well as Sparse Representation based classifier. Section 3 describes our novel approach and the technique used to recombine the results of the multiple classifiers. In section 4 the experimental results are provided when our implementation was tested on some standard datasets like MNIST, CIFAR 10 and MovieLens.

3.3 Literature Survey

Detailed surveys of previous work done in the domains of divide and conquer and SRC are separately illustrated in the following section.

3.3.1 Divide and Conquer

The divide and conquer approach has been becoming popular in dealing with large datasets for some time now. It has been used in both machine learning as well as matrix factorization algorithms.

[10], [7], [19], [2] and [18] are some of the papers which have used the divide and conquer approach using machine learning techniques. Neural Nets and SVMs are two of the most commonly used classifiers. [1], [2] and [4] have proven that divide and conquer approach works on these classifiers, showing that our method of partitioning our training data holds. For example in [1], a divide and conquer solver for SVM is proposed with multiple levels. At each level the dataset is partitioned in (No.of clusters)^{level}. The solutions from the lower levels are used in the upper ones. Using small number of clusters, a speed up can be achieved. It is shown to have achieved better accuracies than Lib-SVM [8], C-SVM [9]. [4] also uses a kind of SVM called Rank SVM to give ranking to a web-based search using a similar method as being discussed here.

In [2] the training data is partitioned using fuzzy C-means clustering or pdf estimation using

Gaussian mixtures. Each group is trained using a Multi-layered Perceptron and the results are combined using a weighted sum of outputs.

The nearest neighbor approach is one of the oldest but still extremely effective. [3] gives a variation of it called Pairwise Opposite Class-NNs. They are found by recursively splitting training data into 2 regions. This gives a partitioned training set into respective classes. Though it is computationally cheaper than Class-NNs, they still have the drawbacks on NNs, namely taking a large amount of time and needing the whole training data to be present while testing. There is also some literature on Matrix Factorization methods using the same approach. They follow a similar line of attack as in [7] and [8]. Input matrices are divided into different partitions of sub-matrices. Each sub-matrix is factored in parallel using matrix factorization, with each machine core solving a sub problem, and then all sub-matrix estimates are combined. We have taken a bit of both worlds to make something both simple and computationally cheap.

5] deals with a specific problem using divide and conquer. It implements an iterative divide and conquer approach for solving the partial coldstart problem of collaborative filtering, interleaving clustering and learning based on demographic data.

This works mention is important here since pure coldstart problem in recommender system is one of the sample problems solved by us using our algorithm in this paper as well. However, the data used here is controlled in an artificial environment to a great deal, while we work with the whole data and address pure coldstart as opposed to partial coldstart in this case.

3.3.2 Sparse Representation Based Classification

In this work, Sparse representation based classification [10] assumes that the training samples of a particular class approximately form a linear basis for a new test sample belonging to the same class. Let v_{test} be the test sample belonging to the class k represented as

$$v_{test} = \alpha_{k1}v_{k1} + \alpha_{k2}v_{k2} + + \alpha_{kn}v_{kn} + \epsilon \quad (3.1)$$

where, v_{ki} denotes the i^{th} training sample and ϵ is the approximation error.

In a classification problem, the training samples and their class labels are provided. The task is

to assign the given test sample with the correct class label. This requires finding the coefficients k_i in Equation 3.1. Since the correct class is not known, SRC represents the test sample as a linear combination of all training samples from all classes,

$$v_{test} = V\alpha + \epsilon \quad (3.2)$$

$$\text{where, } V = \begin{bmatrix} \underbrace{v_{1,1} | \dots | v_{1,n}}_{v_1} & \underbrace{v_{2,1} | \dots | v_{2,n}}_{v_2} & \dots & \underbrace{v_{c,1} | \dots | v_{c,n}}_{v_c} \end{bmatrix}$$

$$\text{and } \alpha = \begin{bmatrix} \underbrace{\alpha_{1,1} | \dots | \alpha_{1,n}}_{\alpha_1} & \underbrace{\alpha_{2,1} | \dots | \alpha_{2,n}}_{\alpha_2} & \dots & \underbrace{\alpha_{c,1} | \dots | \alpha_{c,n}}_{\alpha_c} \end{bmatrix}$$

According to SRC, only the training samples from the correct class should form the basis for representing the test sample and the samples from other classes should not contribute. Based on this assumption, it is likely that the vector α is sparse, i.e., it should have non-zero values corresponding to the correct class and zero values for other classes. Thus Equation 3.2 is a linear inverse problem with a sparse solution. In [10], the coefficient α is solved by employing a sparsity promoting l1-norm minimization.

$$\min_{\alpha} \|v_{test} - V\alpha\|_2^2 + \|\lambda\alpha\|_1 \quad (3.3)$$

With the sparse solution of Equation (3.3), Wright et al. [10] proposed the following algorithm to determine the class of the test sample. Solve the optimization problem in Equation (3.1). For each class k repeat the following two steps: Reconstruct a sample for each class by a linear combination of training samples belonging to that class by the equation $v_{recon}(k) = V_k\alpha_k$ Find the error between there constructed sample and the given test sample by $error(v_{test}, k) = \|v_{test} - v_{recon}(k)\|_2$ Once the error for every class is obtained, assign the test sample to the class having the minimum error.

3.4 Proposed Algorithm

We have proposed a method where the training data is divided into a number of partitions since in case of most classifiers training is the most computationally expensive work. The partitions are done by random sampling of data. Both disjoint and non-disjoint training subsets are formed. Noteworthy point is that, non-disjoint subsets give us a slightly better accuracy than disjoint. Each partition is passed through a classifier in parallel. The Classifier used is Sparse Representation Classifier. The datasets were mainly divided into 5 and 10 sets for our experiments. Any number of sets could have been implemented in the given framework. The results from each percentage of splits are compared against each other to give us the best possible split. Figure 2 is given for a clearer illustration of our method using 5 sets as an example. As figure 2 demonstrates, each split upon passing through the classifier yields predictions and residuals. This is due to the fact that these are the two outcomes from our sparse representation classifier. When a new test sample y is given, it needs to be classified into one of the classes. To do so, a sparse representation of it is calculated using equation (4):

$$\hat{x}_1 = \underset{x}{\operatorname{argmin}} \|x\|_1 \text{ subject to } Ax = y \quad (3.4)$$

The non-zero entries of \hat{x}_1 are associated with the columns of training matrix A from a single object class i . The test sample can thus be assigned to that class. However error and noise can make other non-zero entries thus rendering this method flawed. Therefore, the classification of y is done based on the reproduction capability of the coefficients associated with all training samples of each object.

y can be approximated as $\hat{y}_i = A\delta_i(\hat{x}_1)$ by using coefficients of only i^{th} class. It can be classified based on the approximation and assigned to the class which minimizes the residual between y and y_i :

$$\min_i r_i(y) = \|y - A\delta_i(\hat{x}_1)\|_2 \quad (3.5)$$

So, using the above mentioned method for classification we get a prediction vector and a residual matrix. A modal of the predictions are taken to give us the most suitable prediction for

a particular item. The residuals are also used to determine predictions. The class with the least residual for an item is assigned to that item. Accuracies are calculated for both the direct predictions and the ones calculated using residuals. After rigorous experimentation, it is found to give very similar results.

The method used is called fusion where each classifier is assumed to have equal share of expertise in the decision making process.

3.4.1 Proposed algorithm for Divide and Conquer using SRC

The algorithm is expalined in simple words as follows:

- Divide the data into S overlapping or non-overlapping sets using random sampling.
- Run each set S_i into a SRC in parallel.
- Obtain a Predicted set P and Residual R for each set S.
- Take Mode of P_j over all js (cardinality of the test set) to get the Predicted for each test sample.
- R is given for each class; sample t is assigned to class c if R_c is the least.

Figure 3.1 gives a clear graphical representation of the Algorithm.

The following is the formal description of the Proposed Algorithm:

- The training data A is divided into j sets as A_1, A_2, \dots, A_j .
- A sparse representation for each x_{1j} needs to be calculated using $\hat{x}_{1j} = \operatorname{argmin} \|x\|_1$ subject to $A_j x = y$
- Now y is approximated as $\hat{y}_{ij} = A_j \delta_i(\hat{x}_{1j})$ by using coefficients of only i^{th} class.
- We get a residual vector $r_i(y)$ for each class.
- For each data point whichever class has the least residual, that class is assigned to that data

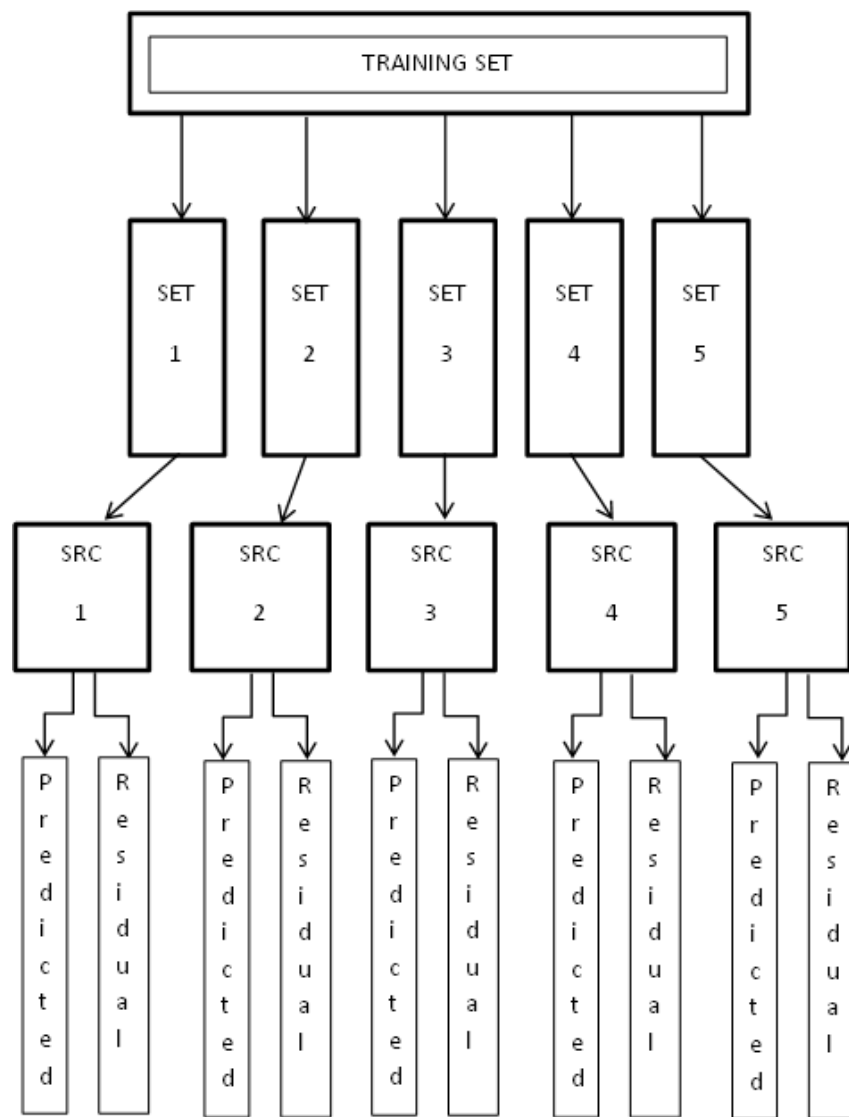


Figure 3.1: Parallel Training

point. And thus we get a residual matrix R_j each consisting of r_i for each set.

► Now, all residual matrices are added. $Res = \sum_1^j R_j$

► Res is sorted in ascending order and again the class having the least residual is assigned to the test data point,

► The Predicted vector received for each set P_j is also used for prediction. A mode of P_j s taken and that class is assigned to the test data point.

3.5 Experimental Setup and Evaluation

We demonstrate the performance of our algorithm using the MNIST [12] , CIFAR 10 [22] (<https://www.cs.toronto.edu/~kriz/cifar.html>), Movielens 100K and 1M dataset [17] (<http://grouplens.org/datasets/movielens/>).

3.5.1 Description of Dataset

The MNIST dataset contains images of handwritten digits of 10 classes. There are 60000 images of 784 features each for training and 10000 images for testing. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Both the Movielens datasets contain ratings on a scale of 1-5. 100K dataset contains ratings by 943 users on 1682 movies and 1M dataset has 1M ratings given by 6040 users on around 3952 movies.

We have split each dataset into 5 and 10 subsets, of overlapping and non-overlapping structure. That is the data is randomly selected and grouped into 5 or 10 sets where in some, the data in two or more sets might be repeated. While in some, the data of each set is distinct. For all test cases, 5-fold cross validation is performed. The simulations are carried out on system with i7-3770S CPU @3.10GHz with 8GB RAM.

3.5.2 Evaluation Measures

The performance of our proposed approach is evaluated in terms of classification accuracy. It is calculated as the percentage of correct classifications among the whole test data. So, if Actual result is A(say) and predicted is P. Then classification accuracy is:

$$Accuracy = \frac{\sum_i A(i) - P(i)}{|A|} * 100 \quad (3.6)$$

3.5.3 Results

Table 3.1 gives the comparison between algorithm using SRC and algorithms using support vector machine and neural net.

Table 3.2 gives the comparison between SRC using the full data together and the proposed Divide and Conquer approach of SRC(DCSRC).It shows that DC-SRC surpasses the divide and conquer methods using Multilayer Perceptrons and Support Vector Machines. It is not a minute increase but quite a notable increase in accuracy that we have achieved here.

The running time of the Full SRC and DCSRC is also noted to show that apart from proposing a viable approach which does not majorly compromise in accuracies, the time to run the classifiers in parallel saves a substantial amount of time. A single classifier on a subset takes $\frac{i^{th}}{x}$ the time x being the number of splits on the original dataset. The time was shown on the MNIST dataset to establish the above point.

SRC on the whole dataset takes 50991 secs(14.16hrs) while running a single set (when the data is divided into 5 sets) takes 10241 secs(2.84 hrs). Table 3.3 shows the results of all types of experiments performed usign DC-SRC for all the datasets. It gives us a clear idea of its strength and weaknesses.

We have also applied our algorithm on SVHN dataset , and gotten an accuracy of 85.15% which is quite impressive for a huge dataset like SVHN.

Table 3.1: Comparison of Classification Accuracies of SRC, Neural Net and SVM

	SRC(all)	Neural Net	SVM
MNIST	98.33	96.39	93.13(t 3)
CIFAR	40.66	15.27	50.11(t 1)
Movielens 100K	32.38	34.83(30 epochs)	33.93(t 1)
MovieLens 1M	35.13	32.42(100 epochs)	34.95(t 1)

Table 3.2: Comparison of DC-SRC with other popular Divide and conquer methods

	DCSRC	Divide and Conquer MultiNet	DC-SVM
MNIST	97.83	86.89	92.38(rbf,C=1,gamma=0.01,clusters=3), 94.76(rbf,C=2,gamma=0.04,clusters=100)
CIFAR	37.50	10	28.88(rbf,C=2,gamma=0.04,clusters=100)
Movielens 100K	33.39	34.01(100 epochs) 32.08(10 epochs) 36.16(50 epochs)	34.22
MovieLens 1M	35.33	34.3	36.06(rbf,C=1,gamma=0.1,clusters=10) 35.48(rbf,C=1,gamma=0.01,clusters=5)

3.6 Conclusion

In this chapter, a method to parallelize classification is proposed. The most used parallel classification algorithms were variations of SVM and Neural Net. Here a divide and conquer variation of Sparse Representation Classifier is presented which serves in not only parallelizing the process but also comes at par and in most cases surpasses the accuracies achieved using SVM and NN. In case of MNIST and CIFAR (two very standard datasets) the algorithm gives an extremely impressive result. We have also tested it on our coldstart problem which uses a far more complicated set due to its sparsity constraint. But the algorithm still achieves results comparable to standard state of the arts in this problem.

Table 3.3: Results of DC-SRC

Datasets	Type and number of sets	Predicted	Residual
MNIST	5 sets	97.7925	97.88375
	5 sets overlap	97.70375	97.7875
	10 sets	97.332	97.444
	10 sets overlap	97.432	97.524
CIFAR 10	5 sets	34.848	37.75
	5 sets overlap	33.578	35.704
	10 sets	35.762	38.108
	10 sets overlap	35.022	36.664
MovieLens 100K	5 sets	32.90	33.39
	5 sets overlap	31.6866	32.97
	10 sets	32.00	33.057
	10 sets overlap	32.088	33.055
MovieLens 1M	5 sets	34.78	35.44
	5 sets overlap	34.28	34.84
	10 sets	34.74	35.24
	10 sets overlap	34.81	35.55

Chapter 4

Nearest Subspace Classifier

4.1 Introduction

In chapter 1, the cold-start problem of recommender systems has been addressed. However there was a huge future scope for it. The time taken by neural nets is quite large with respect to some other classifiers. The distribution of classes is also uneven in our previous classification.

To address the above issues, we have used **Dictionary Learning** on each class distribution which reduces the class imbalance problem and used **Nearest Subspace Classifier** to reduce the classification time. We have also produced results using only Nearest subspace classifier to show the comparison with the above.

4.2 Background

4.2.1 Nearest Subspace Classifier

The use of all the training samples of all classes to represent the test sample goes against the conventional classification methods. These methods typically suggest using residuals computed from one sample at a time or one class at a time to classify the test sample. Nearest subspace classifier [13], [5] does precisely that.

Given a test sample $\mathbf{y} \in \mathbb{R}^M$, the purpose of multi-class classification is to assign \mathbf{y} to one of

the K classes. There are n_i training samples for the i^{th} class and all samples are stacked into a matrix as

$$A_i = [a_{i,1}, \dots, a_{i,n_i}] \in \mathbb{R}^{M \times n_i}$$

, where $a_{i,j} \in \mathbb{R}^M$ is the j^{th} training sample in the i^{th} class. It can be assumed that all samples are normalised. i.e., $\|A_{i,j}\| = 1$ and $\|y\|_2 = 1$.

The Nearest subspace classifier first calculates the distance from the test sample \mathbf{y} to the i^{th} class and measures the projected residual r_i from \mathbf{y} to the orthogonal principal subspace $\mathbf{B}_i \in \mathbb{R}^{M \times k}$ of the training sets \mathbf{A}_i .

The test sample needs to be projected to a orthogonal subspace, as mentioned earlier, the following gives the formulation used based on Tikonov Regularisation. We try to solve the least squares problem with a regularisation term :

$$\|y - Ax\|_2 + \lambda \|x\|_2 \quad (4.1)$$

On solving the above, we get the equation :

$$\underbrace{(A^T A + \lambda I)^{-1} A^T}_{Projector} y = x \quad (4.2)$$

The projected residuals are:

$$r_i = (I - (AA^T)(AA^T + \lambda I)^{-1}) y \quad (4.3)$$

The test sample \mathbf{y} is then assigned to the class with the smallest residual among all, i.e.,

$$i^* = \underset{i}{argmin} r_i$$

It is to be noticed that the subspace for each sample is only an approximation to the true distribution of the test samples. In reality, due to various factors incurred while while collecting

data, the actual distribution of samples could be nonlinear or multi-modal. Using only the distance to the entire subspace ignores information about the distribution of the samples within the subspace, which could be more important for classification.

Thus, in our experiments we have also shown the difference in classification when the distribution of data is taken into consideration using dictionary learning techniques on the original data.

4.2.2 Dictionary Learning

In dictionary learning [4], we want to learn a collection of atomic signals called *atoms* directly from the given data signals so that the data can be accurately or closely represented by a linear combinations of those scarce atoms.

To define it formally, given a training data $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^L$ and positive integers m, s , an $L \times m$ matrix \mathbf{D} and s -sparse vectors $\gamma_1, \dots, \gamma_n \in \mathbb{R}^m$ are to be found, such that $\mathbf{D}\gamma_i$ is close to \mathbf{x}_i for all i . The following is the formulation where l_2 is used to quantify the error.

$$\min_{D, \gamma_1, \dots, \gamma_n} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{D}\gamma_i\|_2^2 \quad (4.4)$$

such that $\|\gamma_i\|_0 \leq s$ for all i .

Here, $\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_m] \in \mathbb{R}^{L \times m}$ is called the dictionary. The columns of the dictionary represent the atoms. The vector $\gamma_i \in \mathbb{R}^m$ contains at most s non-zero entries. The γ_i contains the coefficients needed by the columns of \mathbf{D} to linearly represent \mathbf{x}_i . The dictionary is considered *good* at representing the signals if the total loss is *small*. Furthermore, the fewer columns \mathbf{D} has, the more efficient it is.

4.3 Proposed Method

The formulation of the problem used here is the one described in details in chapter 1. We have built our method on the same formulation of the data. The problem we address remains the same, i.e., User and Item coldstart for recommender systems. The demographic data of users namely their age, gender and occupation (Figure 2.2) and the genre data of items Figure 2.1 is

used to recommend movies to new users and new movies to existing users. The encoding of the information to organise the classification problem is shown in Table 2.3.

Once the data has been encoded in the above mentioned format, we learn a dictionary for each class where the number of atoms for each dictionary is a percentage of the population of that class. So, in our case the dictionary is not only the representation that approximates elements of a signal class, it also shows the contribution of the signal class with respect to its population strength. We call this approach DL-NS. We also only employ Nearest subspace classifier without learning a dictionary previously on it. We call that approach NS.

The figure 4.1 shows how the nearest subspace classifier classifies a sample by assigning it to a

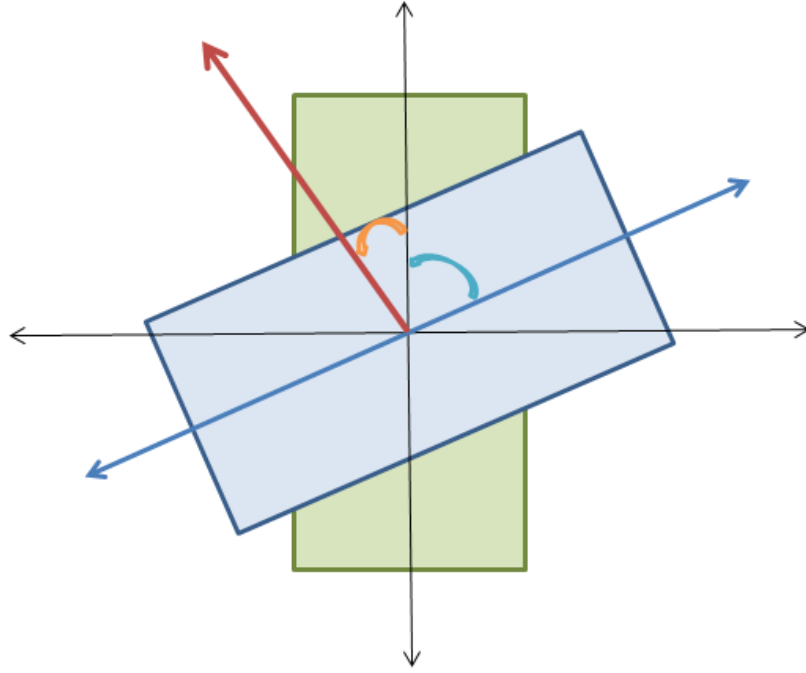


Figure 4.1: Diagram representing how NS works

class with the least projection. Here, the red vector denotes the test sample which lies closer to the green hyperplane than the blue one.

4.3.1 Algorithm DL-NS

- Take a dataset and divide it into train and test sets
- Encode the data as shown in Chapter 1

- Define the number of atoms of a dictionary for a particular class as a percentage of the strength of its class
- Learn a dictionary for each class
- Classify the learnt dictionary using Nearest Subspace classifier

Algorithm NS just skips the 3rd step of the above algorithm.

4.4 Experiments and Results

4.4.1 Description of Data

The datasets used in the following experiments are **MovieLens 100K** and **MovieLens 1M**. Each of them have **5 classes** and they are the ratings of users on items.

MovieLens 100K dataset has *100000 ratings* by *943 users* on *1682 items*. Each user has rated at least 20 movies. Users and items are numbered consecutively from 1. The data is randomly ordered. Item information consists of movie id, movie title, release date, video release date, IMDb URL and 19 genres. User information consists of user id, age, gender, occupation, zip code. The ratings data is split into 80% and 20% for training and testing.

MovieLens 1M dataset has *6040 users* and *3952 items*. The user data has 20 occupations and 7 age groups.

These datasets are available in the website grouplens.org. There are larger datasets on movie recommendations as well, such as the MovieLens 10 million and the Eachmovie, but these datasets do not contain the user or item metadata. However this information is crucial to our work. The datasets used here are the only ones which have this information available.

5 fold cross validation has been performed on these datasets. This is the standard procedure for evaluation, however the folds defined in the datasets could not be used since they were not made for the cold-start problem.

4.4.2 Evaluation Metrics

All known evaluation metrics for recommender systems has been reported by us in this work. This serves in giving a full and well-rounded picture of the scenario based on which proficient further research can be conducted.

The metrics reported are namely:

- **Classification Accuracy** : It is calculated as the percentage of correct classifications among the whole test data. So, if Actual result is A(say) and predicted is P. Then classification accuracy is:

$$Accuracy = \frac{\sum_i A(i) - P(i)}{|A|} * 100 \quad (4.5)$$

- **Mean Absolute Error** : It gives us the deviation in Predicted rating from the actual rating. Let $u(c, s)$ be the true ratings, and $u^p(c, s)$ be the ratings predicted by a recommender system. Let $W = (c, s)$ be a set of *user-item* pairs for which the recommender system made predictions. Then, the mean absolute error, denoted $|\overline{E}|$, is defined as follows:

$$|\overline{E}| = \frac{\sum_{(c,s) \in W} |u^p(c, s) - u(c, s)|}{|W|} \quad (4.6)$$

- **Root Mean Squared Error** : This is a variation of Mean Absolute Error. It is given by the following formula:

$$|\sqrt{\overline{E^2}}| = \sqrt{|\overline{E^2}|} = \sqrt{\frac{\sum_{(c,s) \in W} (u(c, s) - u^p(c, s))^2}{|W|}} \quad (4.7)$$

- **Precision And Recall** : To use these metrics, recommender system must convert its ratings scale into a binary Do not recommend, Recommend scale. Items for which the prediction is to recommend are shown to the user, other items are not shown. The transition mechanism is up to recommender systems. Each item can be either relevant or irrelevant to the user. We get, therefore, the following matrix:

Table 4.1: Precision And Recall

	Recommended	Not Recommended	Total Relevant
Relevant	RR	RN	R = RR+RN
Non-relevant	FP	NN	IR = FP+NN
Total	REC = RR+FP	NREC = RN + NN	N = R+IR = REC+NREC

Precision is the fraction of all recommended items that are relevant.

$$Precision = \frac{RR}{RR + FP} = \frac{RR}{REC} \quad (4.8)$$

Recall is the fraction of all relevant items that were recommended.

$$Precision = \frac{RR}{RR + RN} = \frac{RR}{R} \quad (4.9)$$

4.4.3 Results

Table 4.2 gives a detailed analysis of all the experiments performed using DL-NS algorithm. All the above mentioned evaluation metrics are shown in it.

Tables 4.3 and 4.4 gives a similar detailed report for NS algorithm.

Figure 4.2 shows the change in MAE and RMSE based on the regularization parameters. Figure 4.3 shows the change in Precision and Recall for the number of recommendations.

Table 4.2: DL-NS Results

Number Of Atoms	Accuracy(in %)	MAE	RMSE	Precision (Top 10)	Recall (Top 10)
10	34.5265	0.8828	1.2054	0.6517	0.1269
20	34.6812	0.8927	1.2236	0.6558	0.1265
30	33.8079	0.8944	1.2157	0.6523	0.1256
40	33.8710	0.8857	1.2007	0.6482	0.1222
50	34.6474	0.8778	1.1976	0.6646	0.1286
60	33.9893	0.9006	1.2237	0.6519	0.1254
70	33.8016	0.8956	1.2152	0.6422	0.1202
80	34.2071	0.8866	1.2039	0.6463	0.1248
90	34.2870	0.8877	1.2044	0.6484	0.1288

Table 4.3: NS Results for 100K

Regularization Term	Accuracy(in %)	MAE	RMSE	Precision (Top 10)	Recall (Top 10)
0.1	34.1065	0.9046	1.2310	0.6493	0.1235
0.2	33.5782	0.9042	1.2240	0.6479	0.1289
0.3	34.3599	0.8926	1.2151	0.6469	0.1275
0.4	34.2630	0.8821	1.1987	0.6515	0.1260
0.5	33.9962	0.8895	1.2243	0.6511	0.1273
0.6	35.1538	0.8630	1.1776	0.6605	0.1242
0.7	34.5505	0.8872	1.2114	0.6530	0.1287
0.8	34.5532	0.8740	1.1884	0.6531	0.1276
0.9	34.4056	0.8837	1.2037	0.6514	0.1257

Table 4.4: NS Results for 1M

Regularization Term	Accuracy(in %)	MAE	RMSE	Precision (Top 10)	Recall (Top 10)
0.1	32.0842	0.9322	1.2396	0.6667	0.1038
0.2	34.7842	0.8735	1.1948	0.6450	0.1007
0.3	34.9799	0.8697	1.1926	0.6475	0.1035
0.4	34.9871	0.8644	1.1836	0.6489	0.1033
0.5	34.9824	0.8697	1.1921	0.6470	0.1034
0.6	34.9472	0.8676	1.1885	0.6509	0.1057
0.7	34.6849	0.8786	1.2019	0.6437	0.1031
0.8	35.3511	0.8586	1.1793	0.6531	0.1035
0.9	35.2126	0.8625	1.1835	0.6461	0.1024

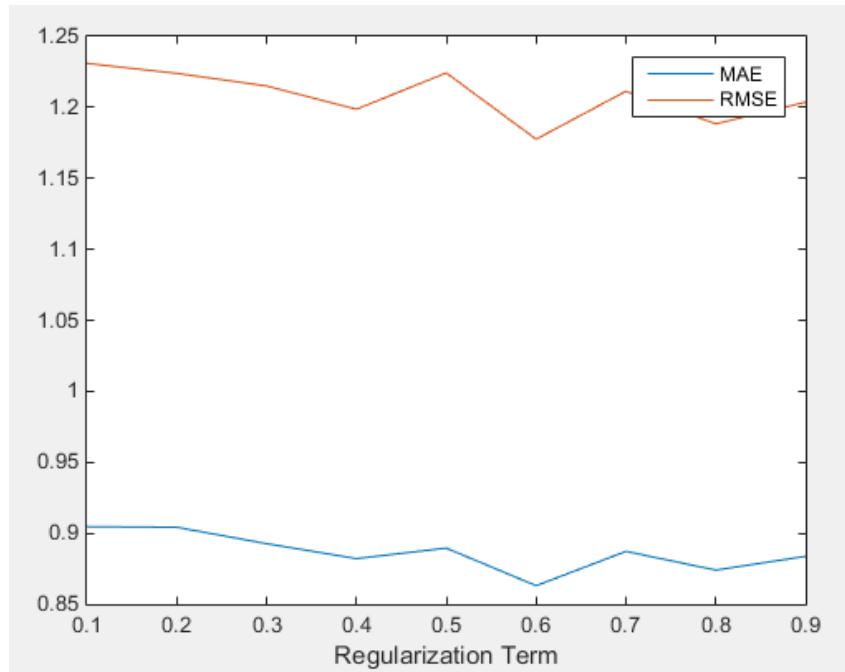


Figure 4.2: Regularisation VS. MAE and RMSE

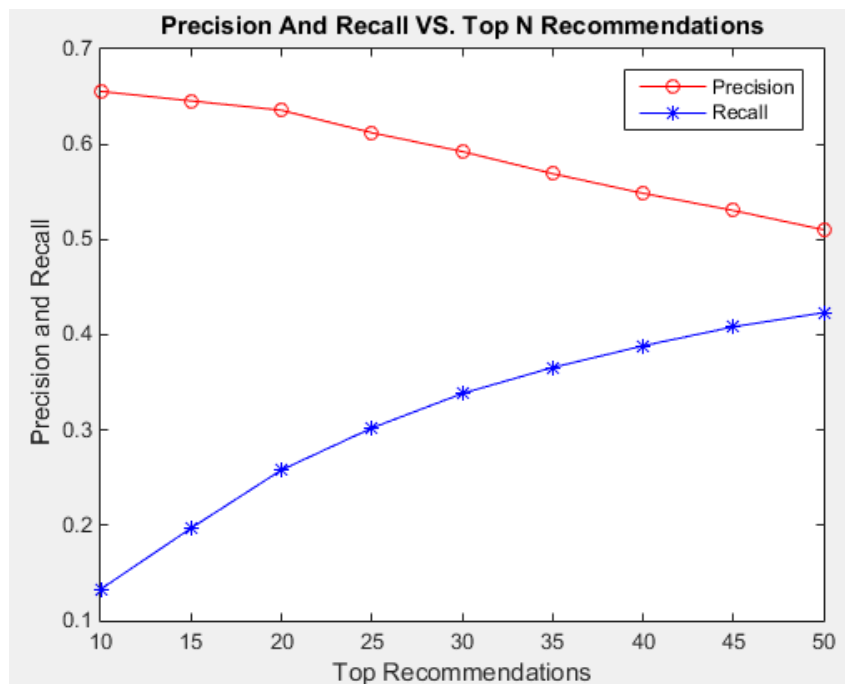


Figure 4.3: Precision and Recall VS. No. of Recommendations

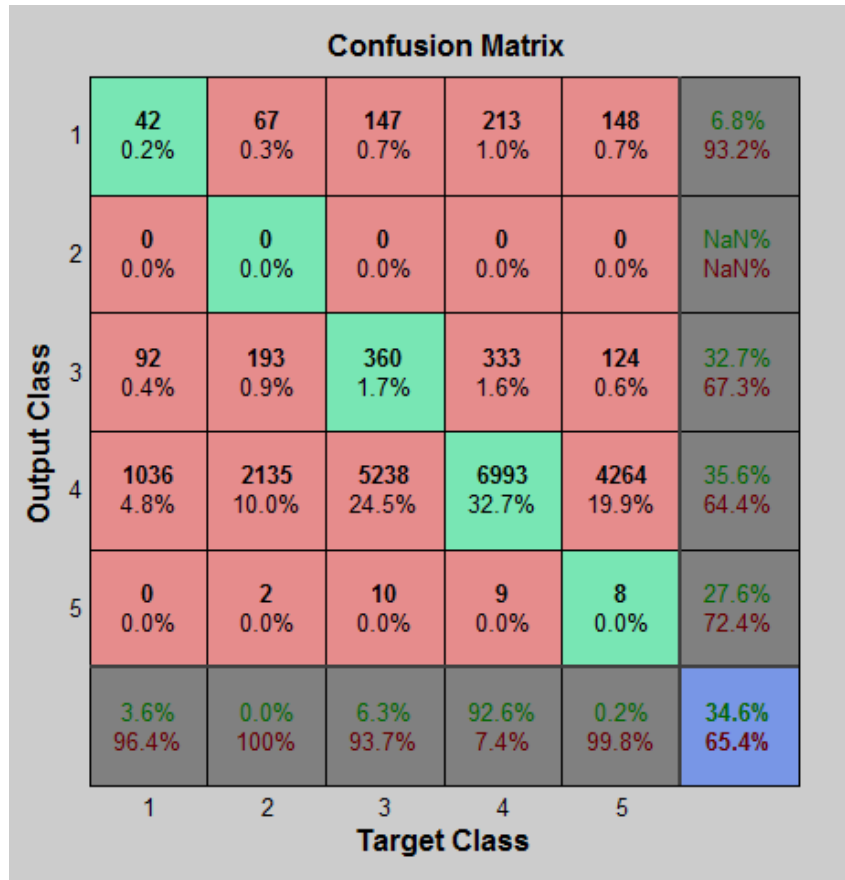


Figure 4.4: Confusion Matrix for MovieLens 100K

For a better visual understanding of the classification accuracies and their distribution in the various classes, the following confusion matrices are shown.

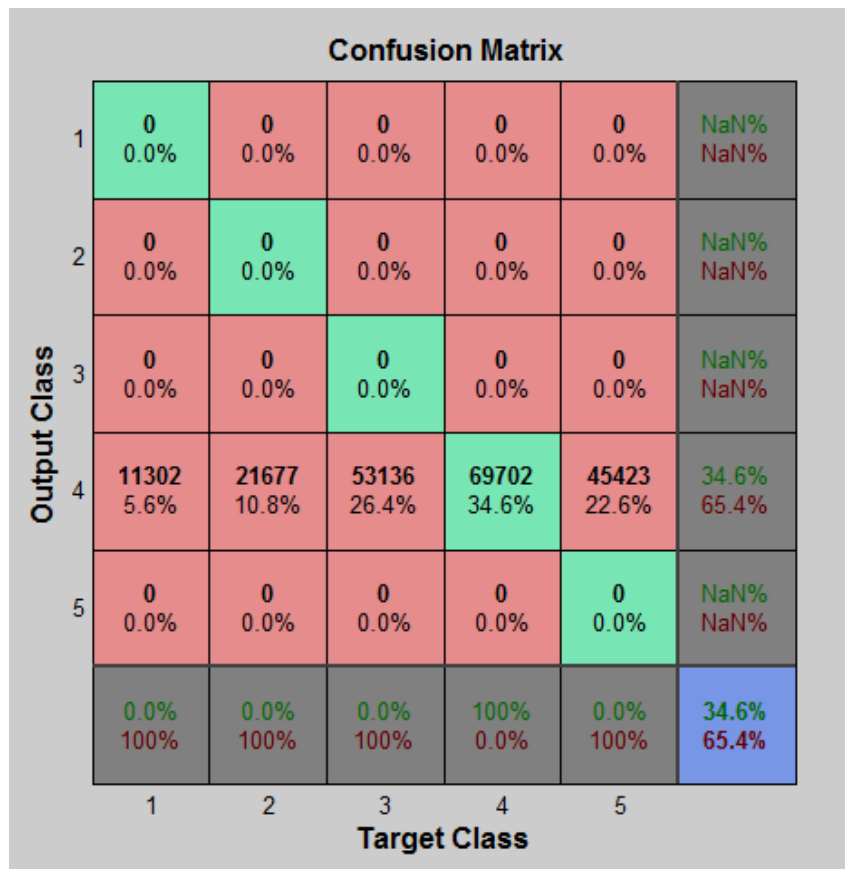


Figure 4.5: Confusion Matrix for MovieLens 1M

The methods DL-NS and NS are both computationally much faster than DC-SRC(3). Whereas results are comparable, DL-NS takes 0.13 times DC-SRC and NS takes less than 0.025 times the time taken by DC-SRC.

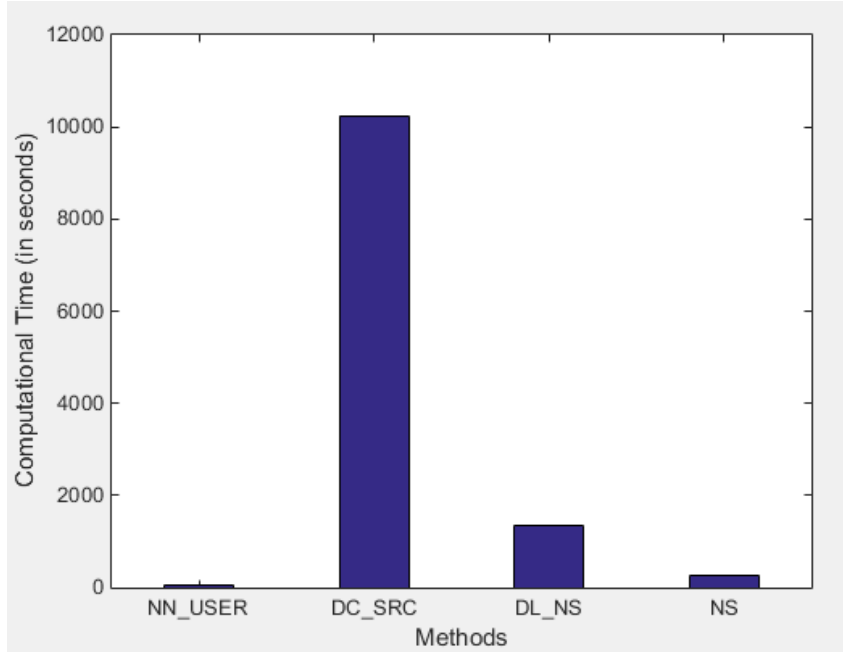


Figure 4.6: Comparison of Methods and their Computational Time

4.5 Conclusion

In this chapter the class imbalance problem of the earlier approaches is addressed. The most important thing solved in this chapter is the reduction in computational time taken by all other algorithms proposed so far. Even if the results achieved are not better than NN-User and DC-SRC proposed in this thesis, it is comparable to them. However, the novelty of this chapter lies in the speedup achieved without compromising on accuracy or other evaluation measures.

Chapter 5

Conclusion

This thesis, comprising of the 3 chapters has been dedicated to the solution of the cold-start problem in recommender systems. The Coldstart problem is a major issue in the e-commerce world thus making it a serious issue faced by every technology-friendly person in daily life as well. Whenever a user first visits a website, he/she hardly ever expects to get the right recommendations, and so for some disheartened folk it might be a not-so-welcome experience. This in turn reduces the business of these sites. However, solving the coldstart problem guarantees better and much more personalised recommendations to all these people and better publicity for new items. While this problem is being tackled by many researchers today, this has been my humble attempt at proposing a simple, usable and progressive framework.

The information of users and items play the basis of the framework. It was a simple idea, rising from the basic fact that the things a person likes are just a reflection of the person they are and the kind of thing it is. When we combine both these factors, we have the preliminary layout for recommendations for coldstart. There is ofcourse a huge future scope to this work. However, the results achieved in this thesis are comparable to the state-of-the-art algorithms at present. It lays a solid groundwork to build upon.

Bibliography

- [1] AHN, H. J. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences* 178, 1 (2008), 37–51.
- [2] BIAN, J., LI, X., LI, F., ZHENG, Z., AND ZHA, H. Ranking specialization for web search: a divide-and-conquer approach by using topical ranksvm. In *Proceedings of the 19th international conference on World wide web* (2010), ACM, pp. 131–140.
- [3] BOBADILLA, J., ORTEGA, F., HERNANDO, A., AND BERNAL, J. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems* 26 (2012), 225–238.
- [4] CHEN, G., AND NEEDELL, D. Compressed sensing and dictionary learning.
- [5] CHI, Y. Nearest subspace classification with missing data. In *Signals, Systems and Computers, 2013 Asilomar Conference on* (2013), IEEE, pp. 1667–1671.
- [6] CUONG, K. M., MINH, N. T. H., VAN CANH, N., ET AL. An application of fuzzy geographically clustering for solving the cold-start problem in recommender systems. In *Soft Computing and Pattern Recognition (SoCPaR), 2013 International Conference of* (2013), IEEE, pp. 44–49.
- [7] FROSYNIOTIS, D., STAFYLOPATIS, A., AND LIKAS, A. A divide-and-conquer method for multi-net classifiers. *Pattern Analysis & Applications* 6, 1 (2003), 32–40.
- [8] GANTNER, Z., DRUMOND, L., FREUDENTHALER, C., RENDLE, S., AND SCHMIDT-THIEME, L. Learning attribute-to-feature mappings for cold-start recommendations. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on* (2010), IEEE, pp. 176–185.
- [9] GUNAWARDANA, A., AND SHANI, G. A survey of accuracy evaluation metrics of recommendation tasks. *The Journal of Machine Learning Research* 10 (2009), 2935–2962.
- [10] HSIEH, C.-J., SI, S., AND DHILLON, I. S. A divide-and-conquer solver for kernel support vector machines. *arXiv preprint arXiv:1311.0914* (2013).

- [11] LAM, X. N., VU, T., LE, T. D., AND DUONG, A. D. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication* (2008), ACM, pp. 208–211.
- [12] LECUN, Y., AND CORTES, C. MNIST handwritten digit database.
- [13] LEE, K.-C., HO, J., AND KRIEGMAN, D. J. Acquiring linear subspaces for face recognition under variable lighting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 27, 5 (2005), 684–698.
- [14] LEUNG, C. W.-K., CHAN, S. C.-F., AND CHUNG, F.-L. An empirical study of a cross-level association rule mining approach to cold-start recommendations. *Knowledge-Based Systems* 21, 7 (2008), 515–529.
- [15] LIKA, B., KOLOMVATSOS, K., AND HADJIEFTHYMIADIS, S. Facing the cold start problem in recommender systems. *Expert Systems with Applications* 41, 4 (2014), 2065–2073.
- [16] LIU, H., HU, Z., MIAN, A., TIAN, H., AND ZHU, X. A new user similarity model to improve the accuracy of collaborative filtering. *Knowledge-Based Systems* 56 (2014), 156–166.
- [17] MILLER, B. N., ALBERT, I., LAM, S. K., KONSTAN, J. A., AND RIEDL, J. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces* (2003), ACM, pp. 263–266.
- [18] PEREIRA, A. L. V., AND HRUSCHKA, E. R. Simultaneous co-clustering and learning to address the cold start problem in recommender systems. *Knowledge-Based Systems* 82 (2015), 11–19.
- [19] RAICHAROEN, T., AND LURSINSAP, C. A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (poc-nn) algorithm. *Pattern recognition letters* 26, 10 (2005), 1554–1567.
- [20] SAFOURY, L., AND SALAH, A. Exploiting user demographic attributes for solving cold-start problem in recommender system. *Lecture Notes on Software Engineering* 1, 3 (2013), 303.
- [21] SON, L. H. Hu-fcf: a hybrid user-based fuzzy collaborative filtering method in recommender systems. *Expert Systems with Applications: An International Journal* 41, 15 (2014), 6861–6870.
- [22] TORRALBA, A., FERGUS, R., AND FREEMAN, W. T. 80 million tiny images: A large data set for nonparametric object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 30, 11 (2008), 1958–1970.

- [23] YU, H.-F., HSIEH, C.-J., DHILLON, I., ET AL. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on* (2012), IEEE, pp. 765–774.