Data dependent modeling of Task-relatedness in Multi-Task Learning

M. Tech. Project Report

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology

by

Lokesh Rajwani
Roll No: 10305066

under the guidance of

Prof. J. Saketha Nath

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai

Dissertation Approval Certificate

Department of Computer Science and Engineering

Indian Institute of Technology Bombay

The dissertation entitled **"Data dependent modeling of Task-relatedness in Multi-Task Learning"**, submitted by **Lokesh Rajwani** (Roll No: **10305066**) is approved for the degree of **Master of Technology** in **Computer Science and Engineering** from **Indian Institute of Technology Bombay**.

[illegible]

Prof. J. Saketha Nath
Dept CSE, IIT Bombay
Supervisor

[illegible]

Prof. Sunita Sarawagi
Dept CSE, IIT Bombay
Examiner-I

[illegible]

Prof. Ganesh Ramakrishnan
Dept CSE, IIT Bombay
Examiner-II

[illegible]

Prof. J. Adinarayana
Dept CSRE, IIT Bombay
Chairperson

Place: IIT Bombay, Mumbai
Date: 21st June, 2012

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/-fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

[illegible]

Lokesh Rajwani
Roll No. : 10305066

Date: 21st **June, 2012**

Acknowledgments

I thank and express my utmost gratitude to **Prof. J. Saketha Nath**, Department of Computer Science and Engineering, IIT Bombay, who has guided me and helped me throughout this project. Without his deep insight into this domain and his valuable time for this project, it would not have been possible for me to move ahead properly. He rectified my basic mistakes and explained me the things in as easy way as possible. Without him and his efforts, my understanding would have been incomplete towards the topic. He has been a constant source of inspiration for me throughout for the achievement of this task. He has been remarkable in his attempt to keep me motivated in this project and has always tried to improve me with proper feedback. I would also like to thank Pratik Jawanpuria for his valuable discussions. He provided me with the various datasets which I could use in my experiments later.

Lokesh Rajwani
Roll No. : 10305066

Abstract

Multi-Task Learning methods are nothing but extension to single-task learning methods. When it comes to learn several related tasks, Multi-Task Learning predominantly had high advantages and hence the tasks are learnt together. Many researchers gave various forms of problem formulations and the techniques to solve it. The learning techniques could be leveraged at one more step - by grouping the multiple tasks in various clusters. This Clustered Multi-Task Learning forms a new approach in which the tasks are adapted to different clusters according to the the degree of relatedness amongst them. Each cluster can then represent one MTL problem in its own. The challenge in this approach is that the clusters are unknown before-hand, and hence they are also to be learned during the process. We worked upon the Recommender System, a real world problem and an application to these multi-task approaches. We noticed that if the clusters are found correct, then we can witness some gains over some recent approaches to multi-task learning methods.

Contents

Chapter 1

Introduction

Traditionally, machine learning methods focussed upon learning each task individually (Single Task Learning (STL)). Researchers then realized that there is some information gain (and hence better learning/generalization) by learning several *related* tasks, not individually, but learning them together parallely. This approach is called Multi-Task Learning (MTL). In MTL, tasks are jointly trained and some common features shared among them is exploited. Rich Caruana in his work [Car97] motivates for MTL and provides several insights that how MTL can lead to better results. He briefed several mechanisms as to how MTL works, like

- Tasks may share common hidden features among them
- Uncorrelation among tasks can appear as noise to other tasks, and noise helps in better generalization of each task.
- Each task having few training samples, but together as a whole they can have large training data, which will again help in better generalization

With the notion of MTL, the researchers explored several methods and developed different kinds of problem formulations. Some of the approaches to solve MTL problems are worked out in [EP04, AEP07, EMP05, PW10, ZYX10]. MTL was indeed found beneficial in terms of accuracy and computational gains, but it had one condition: *the tasks must be related.* Un-related tasks may produce worse results, in which case, STL would outperform better. So then recently, a notion of grouping related tasks into several clusters arises. Its underlying claim is that, its better to group related tasks (from many, possibly uncorrelated tasks), and learn these sub-groups in a Multi-Task learning fashion.

The clusters formed by grouping some tasks may again involve several variations and hence can pose different problems for each different variant. For example, each cluster can be regarded mutually exclusive (i.e. one task will go to only one cluster), and within each cluster/group the tasks share common features amongst them, independently of the tasks in other groups. In some sense, the problem reduces back to Multi-Task Learning within each cluster. Z. Kang et al. propose this kind of grouping tasks in their work [KGS11]. Another variant is described in Bach's et al. work [JBV09], in which they again form clusters of the tasks, but also account for the between-cluster variance and within-cluster variance in their formulation. In more complex scenarios, one can think of tasks getting shared in more than one clusters.

1.1 Motivation

Better generalization and higher accuracies over predicted data is the key target for any real world application of machine learning. Over the years, researchers have worked upon the problems and invented several algorithms to achieve better results. Multi-Task Learning methods have been developed to cater with many tasks together. Since they assume that *all* tasks are *equally* related, this may not be possible in real, some may be more related and some may have little correlation. So the need is to enhance such methods to more complex scenarios, where some less related tasks can be taken care of. One approach which researchers come up is - grouping such tasks according to degree of correlation. But one cannot know before-hand, how much is the task related. Will it hamper in learning other tasks, so that we can separate such tasks out into other groups. All this brings to a challenge of discovering the underlying correlation structure amongst the tasks. Can more better methods be framed to solve this problem, and yet achieve good efficiency, is what this project searches for.

1.2 Problem Statement

During this project overall we aimed at these key things:

- Identifying the current state-of-the-art method, and comparing with the new approaches evolved recently.
- Applying the studied methods on real world datasets and search for any gains by trying all approaches of MTL.
- Can some hybrid mix of two or more approaches lead to better results.

Chapter 2

Stage 1 Review

We studied various papers as mentioned below to get familiar with the problem of Multi-Task Learning. Various experiments were carried out and we compared the results of various literature works. We extended our study to Clustered Multi-Task Learning, and we hoped that continuing in this direction can give better results in comparison to standard Multi-Task learning framework. The following sections gives a brief detail about the papers related to the Mutli-Task methods studied during Stage 1.

2.1 Multi-Task Feature Learning

Argyriou et al. gave a problem formulation in their paper Multi-Task Feature Learning (MTFL) [AEP07], in which tasks are related in such a way that they all share a commom underlying representation. They generalized the 1-norm regularization problem to multiple task case, where 1-norm regularization will help to learn a low-dimensional representation which is shared across multiple tasks.

They posed their problem as minimizing the following error function

$$\mathcal{E}(A, U) = \sum_{t=1}^{T} \sum_{i=1}^{m} L(y_{ti}, \langle a_t, U^\top x_{ti} \rangle) + \lambda \|A\|_{2,1}^2 \tag{2.1}$$

where L is the loss function,
Matrix U consists of orthonormal vectors u_i, which form the basis for weights w_t for each task t, and Matrix A plays the role of feature selector, so as to learn low-dimensional representation by taking $(2, 1)$-norm of it.

The above Error function is to be minimized w.r.t both U and A, but that led to a non-convex problem. In the paper [AEP07], they gave equivalent convex optimization formulation, and provided Alternate Minimization algorithm to solve that.

2.2 Task Relationships in Multi-Task Learning

Zhang et al. in their paper [ZY10], came up with a formulation for MTL similar to that of Argyriou's, but they modelled MTL with relationship between tasks, rather than common feature representation. Comparing this method which stresses upon 'task relationship', with the Argyriou's method described above, which stresses upon 'common feature representation', yet the two formulations given by respective authors are similar. The reason being is that in both the cases, the regularization term boils down to the trace norm of weight matrix W. Hence this paper too follows the Alternate Minimization Algorithm to solve its problem.

2.3 Learning Multiple Tasks with a Sparse Matrix-Normal Penalty

This paper [ZS10] could be stated as a mix of first two approaches. The authors, Zhang and Schneider, formulates the Multi-Task Learning approach by including the regularization term being Matrix-Variate Normal with sparse inverse covariances. Matrix-variate normal incorporates *a*) row covariance $\mathbf{\Omega}$ to describe similarity among tasks *b*) column covariance $\mathbf{\Sigma}$ to represent a shared feature structure in multi-task learning. The total loss $\mathcal{L}$ to be optimized is:

$$\mathcal{L} = \sum_{t=1}^{m} \sum_{i=1}^{n_t} L(y_i^{(t)}, x_i^{(t)}, \mathbf{W}(t,:)) + \lambda tr\left(\mathbf{\Omega}^{-1}\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{W}^{\top}\right) \tag{2.2}$$

Here again alternatively problem is solved by first estimating $\mathbf{W}$ by keeping $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ fixed, and then estimating $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ by keeping $\mathbf{W}$ fixed, until convergence.

2.4 Clustered Multi-Task Learning

In this paper [JBV09], Bach et al. came up with the new approach towards MTL involving clustering of tasks. They group different tasks into different clusters such that the weight vectors of tasks within a group are similar to each other. The clustering of tasks also prevents *outlier* tasks from affecting other related tasks. Clustering effect is encapsulated in the regularization term, and the problem to be solved is in the form:

$$\min_{W \in \mathcal{R}^{d \times m}} \ell(W) + \lambda \Omega(W) \tag{2.3}$$

Here W comprises of the weight vectors of each task, and $\Omega(W)$ is designed from prior knowledge to constrain some sharing information between the tasks. The constraints could be

- a global penalty $\Omega_{mean}(W)$

- A measure of between-cluster variance, $\Omega_{between}(W)$
- A measure of within-cluster variance, $\Omega_{within}(W)$

or any combination of above three, hence,

$$\Omega(W) = \varepsilon_M \Omega_{mean}(W) + \varepsilon_B \Omega_{between}(W) + \varepsilon_W \Omega_{within}(W) \tag{2.4}$$

On varying the parameters $\varepsilon_M, \varepsilon_B, \varepsilon_W$, we can get different types of penalties. For eg, $\varepsilon_W > \varepsilon_B > \varepsilon_M$ promotes compact clusters (within cluster variance is more penalizable than between them).

2.5 Results from Stage 1

Argyriou's formulation [AEP07] is one of the finest and proper formulation for the general Multi-Task Learning notion (without grouping the tasks into clusters). So we took up this as the base evaluation criteria to check the performance of the Clustered Multi-Task Learning [JBV09] approach. We took three real world datasets: School, Sarcos and Park datasets and compared the perfomance on these datasets by the two approaches in an unbiased way. On comparison we found that, the results after taking clustering into account was only marginally better than Argyriou's approach in two datasets. In Park dataset, the result worsen. We wanted to hunt for more better ways of Multi-Class Clustering approaches, since clustering looked promising with the slight better results over Argyriou's method in 2 out of 3 datasets. Hence in stage 2, we worked on an application of Multi-Task Learning, namely, Multi-task learning for recommender systems, which we discuss in next chapter.

Chapter 3

Multi Task Learning for Recommender Systems

Collaborative filtering predicts a users interest (i.e., rating) on an unseen item based on the historical profiles of that user in relation to the historical profiles of the other users. Conventionally ratings were predicted on new items as an aggregate of ratings from similar users or similar items [SKKR01], independent of the relationship that existed among the users' preferences. Since then various recommendation models were built to incorporate the relationship. One popular technique was Matrix Factorization by Koren et al. [KBV09] which performed very well for large-scale recommendation problems.

This paper Multi-task Learning for Recommender Systems [NK10], captures relationships among users. It first identifies a set of related users for the active user based on his/her rating patterns, and then it utilizes rating information from all the related users together to build a multi-task regression model for predicting the active users ratings on unseen items. In this way, the rating prediction models of a set of related users act as parallel learning tasks.

3.1 Multi Task Learning for Collaborative Filtering

Here we present a brief review of the paper [NK10]. The paper explains how the users are selected which are related to the active user (user under consideration). Then each user's problem is formulated as a regression problem which involves items rated by its related users as well.

3.1.1 Notations

Users and Items are denoted by u and i respectively.
Active user is denoted by u_*.
The rating of user u_i on item i_j is denoted by $r_{i,j}$.
The set of items rated by user u_i is $\mathcal{I}_i$

The set of users that have rated i_j is $\mathcal{U}_j$.
The average rating of user u_i on items $\mathcal{I}_i$ is denoted by $\bar{r}_{i,.}$.
The average rating of user u_k over all the items that he/she rated (i.e. $\mathcal{I}_i$) is $\bar{r}_{k,.}$.
$\mathcal{U}$ and $\mathcal{I}$ denote set of **all** users and items in the system.
For a certain active user u, the identified set of related users is denoted by $\mathcal{N}_*$.
The set of items rated by a set of users $\mathcal{N}$ is denoted by $\mathcal{I}_{\mathcal{N}}$.

3.1.2 Selection of Most Similar Users

For a given user u_*, m most related users $\mathcal{N}_* = \{u_{*_1}, u_{*_2}, \ldots, u_{*_m}\}$ are selected whose historical ratings on co-rated items are the most similar to user u_*. The rating similarity between two users is computed, using modified version of Pearson correlation coefficient, as

$$\mathrm{sim}_u(u_i, u_j) = p_u(u_i, u_j) \cdot \frac{\sum\limits_{i_k \in \mathcal{I}_c} (r_{i,k} - \bar{r}_{i,.})(r_{j,k} - \bar{r}_{j,.})}{\sqrt{\sum\limits_{i_k \in \mathcal{I}_c} (r_{i,k} - \bar{r}_{i,.})^2} \sqrt{\sum\limits_{i_k \in \mathcal{I}_c} (r_{j,k} - \bar{r}_{j,.})^2}} \tag{3.1}$$

where $\mathcal{I}_c$ is the set of items co-rated by users u_i and u_j ($\mathcal{I}_c = \mathcal{I}_i \cap \mathcal{I}_j$) and p_u is a penalty factor defined as $p_u(u_i, u_j) = \min(|\mathcal{I}_c|, C)/C$ i.e, it penalizes when the number of co-rated items is smaller than C (a predefined small constant).

3.1.3 Selection of Most Covering Users

The above method, USM$_{\text{sim}}$, selects the most *similar* users, but it does not say anything about what all items the selected users has rated. If a user u is selected depending upon the same rating pattern as of active user u_*, and user u hasn't rated any such items which is also yet unseen by user u_*, then u is not at all informative. The goal is to select such users who not only are similar but are informative as well, i.e. users u should also have rated new items (new in the sense that those items are not rated by u_*).

So, the author proposes the method, USM$_{\text{cvg}}$, which selects users that collectively maximize the coverage of the unrated items for u, in addition to the selection of users that rated in a consistent way with u. This method's algorithm is shown below. Here the key point is that from the list identified by USM$_{\text{sim}}$, (i.e. $(u_{*_1}, u_{*_2}, \ldots, u_{*_{2m}})$), this approach selects the user who have rated maximum number of items that are not rated by both of the active user and the users that are already selected by USM$_{\text{cvg}}$. In case that fewer users are selected by USM$_{\text{cvg}}$ than required, USM$_{\text{cvg}}$ chooses the most similar users from the rest of the user list.

Algorithm 1: USM$_{\text{cvg}}$

```
Data: $(u_{*_1}, u_{*_2}, \ldots, u_{*_{2m}}), m, \mathcal{I}_*$
Result: $\mathcal{N}_*$
begin
    $\mathcal{N}_* = u_{*_1}$
    $i = n = 1$
    while $i < 2m$ do
        for $j = 1$ to $WINDOWS\_SIZE$ do
            $k = i + j$
            if $k > 2m$ then
                goto L
            else
                count the number of items that have been rated by $u_{*_k}$ but not
                rated by $u_*$ or $\forall u \in \mathcal{N}_*$
        look for $u_{*_{kmax}}$ with maximum non-zero count in this window
        if such user $u_{*_{kmax}}$ exists then
            $\mathcal{N}_* = \mathcal{N}_* \cup \{u_{*_{kmax}}\}$
            $n = n + 1$
            $i = k_{max}$
        else
            $i = i + WINDOWS\_SIZE$
        if $n \geq m$ then
            break
    L: if $n < m$ then
        select $m - n$ most similar users that are not selected into $\mathcal{N}_*$
end
```

3.1.4 Kernel Functions for Users and Items

Like user-similarity function (3.1), item-similariy between i_i and i_j is defined based on adjusted cosine similarity to determine the similarity between items as done in [SKKR01],

$$\text{sim}_i(i_i, i_j) = p_i(i_i, i_j) \cdot \frac{\sum\limits_{u_k \in \mathcal{U}_c} (r_{k,i} - \bar{r}_{k,.})(r_{k,j} - \bar{r}_{k,.})}{\sqrt{\sum\limits_{u_k \in \mathcal{U}_c} (r_{k,i} - \bar{r}_{k,.})^2} \sqrt{\sum\limits_{u_k \in \mathcal{U}_c} (r_{k,j} - \bar{r}_{k,.})^2}} \tag{3.2}$$

The user-similarity function (3.1) is used to construct $\mathcal{K}_u$, the kernel on users. And, the item-similarity function (3.2) is used to construct $\mathcal{K}_i$, the kernel on items. To make them as valid positive semi-definite Kernels, the least eigen value is subtracted from all the diagonal elements of this symmetric similarity matrices. The multi-task Kernel function $\mathcal{K}_{mt}$ can now be defined as

$$\mathcal{K}_{mt}((u_i, i_j), (u_{i'}, i_{j'})) = \mathcal{K}_u(u_i, u_{i'}) \times \mathcal{K}_i(i_j, i_{j'}) \tag{3.3}$$

Using this Kernel the multi-task model of user u is learned using error-insensitive Support Vector Regression (ϵ-SVR) [SS04]. The input to the model are tuples of the form $((u_i, i_j), r_{i,j})$, where,
$u_i \in u_* \cup \mathcal{N}_*, i_j \in \mathcal{I}_i$ and $r_{i,j}$ is the target rating that u_i has given to i_j.
The ratings are predicted as real numbers (since its a regression problem) and it can be approximated to nearest integers.

Chapter 4

Our approach towards a Hybrid method

We know clustering helps in better learning, but what we don't know is how to get correct clusters of tasks so that it helps in performance-boost. So we approached for a mix of the two methods – Multi-Task Learning for Recommender Systems to obtain the clusters [NK10] and Multi-Task Feature Learning [AEP07] to train each cluster independent of the tasks in other clusters.

Here we describe our approach in brief.
From the MTL for Recommender System method, we obtained m related users for each active user. We build a $N \times N$ matrix M (where N is the total no. of users/tasks) whose element m_{ij} corresponds to the no. of common users in the m most covered users set of the active users i and j. This matrix M will be symmetric and from this matrix M, we obtain n adjacency matrices, each giving us different clusters, such that all users are covered and no task belongs to two or more than two clusters. Here we sketch out the algorithm for obtaining n adjacency matrices, each adjacency matrix will further give us various clusters, and its upto the user to set appropriate n. As such there is no relation between n and the no. of clusters obtained in the end, because primarily it depends on the dataset being used. So one needs to cross-validate over this parameter n.

Since if users having z common related users in their covering user set, then users having $z - 1$ common related users can also be treated in the same group. The idea is, we cannot strictly impose grouping just because there is a difference of only 1 user in common. So first we find out a more coarse matrix M', where we can distinguish such groups easily. Each group is given a no. and that group contains all those users, which have some minimum threshold of common users amongst them. Now again these thresholds may vary, and here we choose parameter n to determine various thresholds. Finding groups on the basis of n thresholds and thus getting a coarse_matrix is shown in Algorithm 2.

Algorithm 2: Finding Coarse Matrix M'

```
begin
    Data: Matrix $M$
    Result: Matrix $M'$
    Let $l = \min(M)$ and $h = max(M)$
    Choose $n$ and divide the range $(h - l)$ into $n$ parts.
    if $m_{ij}$ lies in part $p$ then
        set $m'_{ij} = p$
end
```

After calculating the coarse matrix M', we can now strictly partition the users into n different adjacency matrices, such that,

$$A_1 + A_2 + \ldots + A_n = \mathbf{1}_{N \times N}$$

where, each A_i is an adjacency matrix of size $N \times N$ corresponding to group i,
N is the total no. of users/tasks, and,
$\mathbf{1}$ is a matrix of all ones, i.e. each entry (i, j) is 1.

Depending upon the no. m'_{ij}, the corresponding users i and j will go into that group and hence the a_{ij} and a_{ji} elements of the respective adjacency matrix is set to one. All adjacency matrices are symmetric and they disjointly partition the users. Now from each adjacency matrix A, some set of clusters are obtained as shown in Algorithm 3. In following algorithm, the list *clusters* of size N (N is the total no. of users) is computed, such that, user i belongs to cluster no. given by $clusters[i]$. Maximum no. of clusters thus obtained is $\max(clusters[i])$. Here, we traverse each adjacency matrix one-by-one and for each matrix, we iterate through each user which are not clustered yet. For each such user, we find who all are related to it according to the current adjacency matrix. Classify all these users into one single cluster. Iterate over each user and each adjacency matrix until all users are clustered.

Algorithm 3: Finding *clusters* from matrix M'

```
begin
    Data: Matrix $M'$
    Result: array $clusters[]$ of size $N$
    Initialize: all $clusters[]$ elements to 0, $Cno = 1$
    $n = \max(M')$
    for $k = n$ to 1 step $-1$ do
        create Adjacency Matrix $A$, such that,
        if $a_{ij} == k$ then
            $a_{ij} = 1$
        else
            $a_{ij} = 0$
        list_i = all those users $i$ such that $clusters[i] == 0$
        for each user $i$ in list_i do
            if user $clusters[i]! = 0$ then
                This user has already been clustered
                Continue for next iteration
            list_j = all those users $j$ related to user $i$ (obtained from matrix $A$)
                such that, $clusters[j] == 0$ and $j! = i$
            if list_j is not empty then
                set $clusters[i] = Cno$
            for each user $j$ in list_j do
                set $clusters[j] = Cno$
                $Cno+ = 1$
end
```

Once we get the clusters, we apply Argyriou's method over each cluster as a single MTL problem. Thereby training all the tasks in that cluster. Once all clusters have been trained by Argyriou's method of multi-task feature learning, then we get all tasks trained depending only on other tasks in its cluster. In the next chapter, we discuss the experiments performed and the results obtained.

Chapter 5

Experiments

We evaluated the performance of our methods on MovieLens dataset [1]. It contains 100,000 ratings from 943 users on 1,682 movies. Each user rates more than 20 movies. The rating scores range from 1 to 5 as integers.

We tried following variations, and measured its performance against Argyriou's method (Multi-task Feature Learning) and Bach's method (Clusteres Multi-Task Learning). Since Argyriou has performed well enough, and hence its performance forms our base criteria to be compared with. The methods which we tried on the MovieLens dataset are:

(i) Multi-Task Feature Learning method by Argyriou et al. [AEP07]

(ii) Clustered Multi-Task Learning by Francis Bach et al. [JBV09]

(iii) The Recommender Systems method given by paper [NK10]

(iv) Obtaining the clusters from method (iii), and then supplying each cluster independently to method (i)

In the MovieLens dataset, each user we considered as a separate task and the task is to predict the rating of items. We divided the dataset into 80:20 ratio for training and testing respectively. We used 5-fold cross-validation, and in each fold the training data constitues the 4 parts of the active user's training data plus all 5 parts of other related users' training data. The remaining 5th part of the active user forms the testing data for that fold. Here we summarize our findings and we report the mean square errors for the respective methods below.

5.1 Recommender System versus Argyriou's method

We compared the two papers on the MovieLens dataset. The Multi-Task Learning for Recommender System paper [NK10] computes the Kernel function only on the

[1] http://www.grouplens.org/node/73

basis of ratings given. We modified that and also included the user features and item features, and finally the precomputed Kernel was supplied to (ϵ-SVR) using LibSVM tool [CL11]. Since each user is a separate task, and the recommender system paper builds a multi-task model explicitly designed for each user, so we experimented this method on first 50 users. The errors were calculated for each user/task separately and compared them on both the methods. Following is the brief result obtained.

Table 5.1: Recommender System v/s Argyriou

	Test Error	
	MTL Recommender	**MTFL (Argyriou)**
Avg. of First 10 tasks	1.115	1.16
Avg. of First 30 tasks	1.0365	1.038
Avg. of First 50 tasks	1.25	1.06
Avg. of tasks 31-50	1.58	1.09

It can be seen that MTL for Recommender Systems outperformed on initial few tasks as compared to Argyriou's method. Since in Argyriou's method *all* tasks are taken simulataneously as related tasks in Multi-Task Learning approach, but in former method only some tasks (which are related to the active user only) are taken into account. This shows that there may be some tasks which are *less* related and can hamper overall learning of tasks. But then, the tasks from 31-50 cluster show that Argyriou has performed well. This may happen because the tasks may not have been properly clustered. Some non-related task to the cluster may act as a noise and boost-up the generalization overall which happens in Argyriou, but fails to occur in case of MTL for Recommender Systems. Thus we conclude that though clustering is required for witnessing some gains, but improper clusters may harm the learning of tasks in that cluster. This point was also highlighted in Stage 1 where clustering method by Bach's paper [JBV09] results good on 2 out of 3 datasets.

5.2 Our approach - Mixing the two formulations

So we approached for another method, where we aimed to create adjacency matrix over all tasks. For each task we selected some m most related users as given by MTL for Recommender Systems approach. We associated those users in the adjacency matrix with their active user. The procedure was repeated for all users, where each user became active user one-by-one. On applying some threshold, we distinguished the users into various groups such that the users in the same group have approximately same related users - Algorithm 2. Finally clusters of users/tasks were obtained from each group - Algorithm 3. Each cluster independently was then applied on Argyriou's method. The test errors were computed for each task each cluster separately and we compared the results of this method with the overall Argyriou's method (without clustering). We also considered the boundary case of each task independently being learned by again applying Argyriou's method, once for each user. Since there are many tasks (users), so we summarize our results for the first 500 tasks in the steps of 100 tasks below.

Table 5.2: Net results of Clustering and applying Argyriou method

	Test Error		
	Argyriou on Clusters from MTL Recommender	**Argyriou (overall)**	**Argyriou (each task independently)**
Avg. of tasks 1-100	1.032	1.028	1.14
Avg. of tasks 101-200	1.15	1.114	1.216
Avg. of tasks 201-300	**1.049**	1.055	1.097
Avg. of tasks 301-400	1.083	1.06	1.11
Avg. of tasks 401-500	**1.134**	1.135	1.187
Avg. of all tasks	1.087	1.0673	1.135

The bold figures above show the improvement, though marginally better, over Argyriou's method, but overall the performance is not as expected. But the clustering surely gives certain advantages provided the tasks are correctly clustered so that it won't harm the performance overall.

Chapter 6

Related Work

For Multi-Task Learning there exist many other formulations, such as, probabilistic interpretation of the general multi-task feature selection problem [ZYX10]. Also since it is noted that the basic regularization term is what that couples the task relatedness, so methods have been developed to tune such regularization terms and develop more efficient learning algorithms. For example, [JY09] propose an extended gradient algorithm and shows that by exploiting the special structure of the trace norm, the optimal convergence rate for general non-smooth problems can be improved. Kernel methods have also been employed in Multi-Task Learning approaches to generalize the linear multi-task learning methods to the non-linear case. Micchelli and Pontil described reproducing kernel Hilbert spaces of vector-valued functions and discussed their use in multitask learning [MP04].

In case of clustered multi-task learning approaches, Z.Kang et al. have studied the problem of clustering task in the distinct setting where relatedness is modeled as learning shared features among the tasks. [KGS11] They simultaneously determine "with whom" each task should share features, while also optimizing the model parameters for all tasks per group.

Chapter 7

Conclusion

In the Experiment section above, we showed that the hybrid system of MTLRecommender + Argyriou's method, performs better than Argyriou's method alone in some cases. Though we have chose the best parameter n for it, but ideally we should cross-validate over parameter n also. Clustering can be indeed helpful and since the overall result isn't good enough, hence more sophistacated methods to figure out clusters needs to be employed.

We also compared the results of this hybrid approach with the Bach's paper, Clustered Multi-Task Learning. We clearly outperform from Bach's method and can say that Bach's method doesn't finds appropriate clusters. The reason being that the method itself asks for input to no. of clusters required from the user, instead of searching the space of all possible clusters and choosing the best amongst them. If the latter case happens (searching best possible clustering structure over entire span), then it would require to evaluate the method over 2^T clustering structures, where T is the no. of tasks. This would be highly inefficient and thus this area needs some more research for exploring better clustering methods since we can see some promising results of clustering.

Bibliography

[AEP07] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems 19*. MIT Press, 2007.

[Car97] Rich Caruana. Multitask learning. *Mach. Learn.*, 28:41–75, July 1997.

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[EMP05] Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

[EP04] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 109–117, New York, NY, USA, 2004. ACM.

[JBV09] L. Jacob, F. Bach, and J.-P. Vert. Clustered multi-task learning: A convex formulation. In *Advances in Neural Information Processing Systems 21*, pages 745–752. MIT Press, 2009.

[JY09] Shuiwang Ji and Jieping Ye. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 457–464, New York, NY, USA, 2009. ACM.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[KGS11] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 521–528, New York, NY, USA, June 2011. ACM.

[MP04] Charles A. Micchelli and Massimiliano Pontil. Kernels for multi-task learning. In *Proceedings of NIPS 2004*, 2004.

[NK10] Xia Ning and George Karypis. Multi-task learning for recommender system. *Journal of Machine Learning Research - Proceedings Track*, 13:269–284, 2010.

[PW10] Shibin Parameswaran and Kilian Weinberger. Large margin multi-task metric learning. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1867–1875. 2010.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[SS04] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, August 2004.

[ZS10] Yi Zhang and Jeff Schneider. Learning multiple tasks with a sparse matrix-normal penalty. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2550–2558. 2010.

[ZY10] Yu Zhang and Dit-Yan Yeung. A convex formulation for learning task relationships in multi-task learning. In *Proceedings of the 26th International Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 733–742. 2010.

[ZYX10] Yu Zhang, Dit-Yan Yeung, and Qian Xu. Probabilistic multi-task feature selection. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2559–2567. 2010.