Task Relationship Modeling in Lifelong Multitask Learning

By

Meenakshi Mishra

Submitted to the Department of Electrical Engineering & Computer Science and the
Graduate Faculty of the University of Kansas
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Committee members

Dr. Jun Huan, Chairperson

Dr. Arvin Agah

Dr. Swapan Chakrabarti

Dr. Rongqing Hui

Dr. Zhuo Wang

Date defended: July 21, 2015

The Dissertation Committee for Meenakshi Mishra certifies
that this is the approved version of the following dissertation :

Task Relationship Modeling in Lifelong Multitask Learning

Dr. Jun Huan, Chairperson

Date approved: July 21, 2015

Abstract

Multitask Learning is a learning framework which explores the concept of sharing training information among multiple related tasks to improve the generalization error of each task. The benefits of multitask learning have been shown both empirically and theoretically. There are a number of fields that benefit from multitask learning such as toxicology, image annotation, compressive sensing etc. However, majority of multitask learning algorithms make a very important key assumption that all the tasks are related to each other in a similar fashion in multitask learning. The users often do not have the knowledge of which tasks are related and train all tasks together. This results in sharing of training information even among the unrelated tasks. Training unrelated tasks together can cause a negative transfer and deteriorate the performance of multitask learning. For example, consider the case of predicting in vivo toxicity of chemicals at various endpoints from the chemical structure. Toxicity at all the endpoints are not related. Since, biological networks are highly complex, it is also not possible to predetermine which endpoints are related. Training all the endpoints together may cause a negative effect on the overall performance. Therefore, it is important to establish the task relationship models in multitask learning.

Multitask learning with task relationship modeling may be explored in three different settings, namely, static learning, online fixed task learning and most recent lifelong learning. The multitask learning algorithms in static setting have been present for more than a decade and there is a lot of literature in this field. However, utilization of task relationships in multitask learning framework has been studied in detail for past several years only. The literature which uses feature selection

with task relationship modeling is even further limited.

For the cases of online and lifelong learning, task relationship modeling becomes a challenge. In online learning, the knowledge of all the tasks is present before starting the training of the algorithms, and the samples arrive in online fashion. However, in case of lifelong multitask learning, the tasks also arrive in an online fashion. Therefore, modeling the task relationship is even a further challenge in lifelong multitask learning framework as compared to online multitask learning.

The main contribution of this thesis is to propose a framework for modeling task relationships in lifelong multitask learning. The initial algorithms are preliminary studies which focus on static setting and learn the clusters of related tasks with feature selection. These algorithms enforce that all the tasks which are related select a common set of features. The later part of the thesis shifts gear to lifelong multitask learning setting. Here, we propose learning functions to represent the relationship between tasks. Learning functions is faster and computationally less expensive as opposed to the traditional manner of learning fixed sized matrices for depicting the task relationship models.

Acknowledgements

I would like to take this opportunity to thank my advisor, Dr. Jun Huan for his support and guidance. It is due to his suggestions and help given by him, that this dissertation was possible.

I express my gratitude towards Dr. Arvin Agah, Dr. Swapan Chakrabarti, Dr. Rongqing Hui, and Dr. Zhuo Wang for serving on my Ph.D. committee and providing me with valuable suggestions.

I also thank my friends and lab mates for their help throughout this project.

Lastly, and most importantly I express my forever gratitude to my parents Mr. and Mrs. Shiva Kant Mishra, my husband Viraj Singh and my brother Akhilesh Mishra. I thank them for their unconditional love, emotional support and for everything they have done for me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As human beings, we need to learn multiple tasks during our lifespan. The knowledge of one task almost always helps us in training for other related tasks. For example, a basketball player learns his major skills from playing basketball. However, the training he has in running, weight training etc also helps him in learning basketball as well. Or a person who has knowledge about regularized linear regression can more easily be trained to acquire knowledge about multitask learning as opposed to a person who comes from a completely different background like fine arts or medicine. Thus, it is very common in our day to day lives to apply the training we have from other tasks, into learning the current task. This process of using past learning experience to learn the current task makes the training both easier and faster. Same analogy can be applied to machine learning as well. There are many tasks that might be related, and using information from training of one task should help in training of other related tasks (142).

The concept of multitask learning in machine learning and data mining has been around for more than a decade but the freshness of this topic is preserved. Multitask learning has shown to improve the generalization performance both empirically and theoretically. However, the challenges encountered in multitask learning are still not near their end. For example, given a set of tasks, our brain can easily identify the correlated tasks and easily utilize the knowledge acquired by learning

one task into other. However for machines, it is not so easy to identify the related tasks. The machine will just learn all the tasks together and risk learning from tasks which are not related. This scenario may be deteriorating for the performance and overall generalization error and is commonly known as negative transfer (73) (79) . Similar challenges make multitask learning an active research topic even today.

1.1 Motivation

The sources of motivation for this thesis in the domain of multitask learning lies in both the application and the theoretical side. There are numerous number of applications that may benefit from multitask learning such as computational prediction of toxicity, bioinformatics, object recognition etc. and each application comes with its unique set of requirements which may suggest directions to improve upon the multitask learning framework. Motivations for multitask learning may also be sought from theoretical limitations of the existing algorithms as well. Listed here are some of the application and theoretical motivation for the current work.

1.1.1 Toxicology

Toxicity for long has been predicted using traditional methods such as animal testing. A fixed dose of chemicals is fed to the experimental animal such as mouse, rat or rabbit, and the chemical's effect is observed on the animals over a predefined period. Often these testing methods tend to be time consuming and expensive because a number of the animals need to be fed varying amount of dosage for varying time to observe the effects. Not only do such methods involve a lot of cruelty to animals, but these methods are slow and do not meet the demands of present day when thousands of new chemicals are synthesized every year. To combat this problem, there is a need to use alternative methods of testing toxicity of chemicals. One way would be to use computational means to predict toxicity. For computational prediction of toxicity, we need sufficient data which might be hard to get. Also, we know that toxicity prediction at some endpoints across some species

are related. Thus, toxicity prediction can find a lot of help from sharing information during the training process, or in other words using multitask learning.

1.1.2 Bioinformatics

Over the last decade or so, there has also been a tremendous development in biological technologies. Mapping the entire genome of any species used to be considered a daunting task, but now it can be achieved in a couple of days. There is tremendous data relating to gene expressions, protein expressions etc and hence, computational aid to process these data are in big demand. This plethora of information can be used for human benefits as well. For example, the risk of cancer in each patient might be predicted using gene expression data, or the gene that plays an active role in various forms of cancer might be identified. The challenge here, however, is that the availability of labeled data is very limited. Thus, for any task, the number of samples become limited. One way to combat this problem is again to group the related tasks together and use information from related tasks as part of the training process as well (138) (113) (111) (149). For example, the prediction of some forms of cancers might be related in the body. Or similar groups of genes might increase the risk of multiple cancers. Thus, in these scenarios, multitask learning is useful.

1.1.3 Object Recognition

Object recognition is a wide topic in both artificial intelligence and machine learning. Object recognition does not only mean the recognition of physical objects such as table, chair etc from pictures, it can also mean hand written digit recognition, or recognition of animal species based on the sound recording. Basically, where ever any kind of identification task is present, machine learning is extremely helpful. Often potentially related tasks can be spotted in object recognition as well. For example, in the task of identification of animal species based on their sound recording, there can be several species whose sound characteristics are similar and aid in training as well. In such a scenario, multitask learning will be favorable (151) (19) (93).

1.1.4 Theoretical Motivation

As can be seen in all the above examples, it is rather difficult to point out which tasks are related. For example, in the case of toxicology, are all toxicity prediction task related or the tasks that predict toxicity at similar endpoints in different species related, or the tasks that predict toxicity in the same species related? Similarly, in the example given in bioinformatics, how do we know which kind of cancers should be learnt together to get the maximum performance. Same argument holds for object recognition and any other application of multitask learning. We cannot group all the tasks together, as that can risk in deterioration of performance. We have to find the related tasks.

Another question to ask ourselves would be, how the task relatedness is defined? In machine learning when we talk about task relatedness, we usually talk about closeness of the parameters that are trained for each task. This proximity can either be measured in terms of physical distance between the parameters, such as euclidean space or in underlying latent space. Would these metric be appropriate for what we humans define as related tasks and force such information be shared?

The third daunting question that comes up after reading all the related material, is that, if all the information we have about the tasks is being appropriately represented to be fully utilized for training? For example, if we know the task structure beforehand, do we fully utilize that structure in training. As an instance, suppose that the tasks have hierarchical structure. There has been considerable work done to utilize this kind of structure and train the higher order nodes from samples of the leaf node. However, when using feature selection, how do we frame the relationships between leaf nodes and their ancestors? Will it be more appropriate to enforce leaf nodes to select a subset of features, or is it appropriate to enforce all tasks to select a common set of features?

Another major question that is frequently encountered is, are all the tasks known beforehand. There are multiple applications where new tasks may be added during the training process. For instance, in the case of toxicity prediction, the data which is available for training may only consist

of labels for prediction of toxicity related to some endpoints. As the training progresses, more and more data becomes available for other endpoints as well. Similarly, in the case of image annotation, the number of objects that need to be identified by the learning agent may increase as the learning agent trains on increasing number of picture. In each of the cases, new tasks are encountered during the training process. In such cases, is it possible to use the existing knowledge learnt from previous tasks to learn the new tasks as well and use the knowledge from the new task to update the previously learnt tasks as well? Or would it be beneficial to retrain the entire learning agent each time a new task is encountered?

Thus, there are still many open-ended questions that remains to be explored in multitask learning which make the topic of multitask learning an interesting area to work on. This thesis aims at delving further into these areas and find some of these answers. We study task relationship modeling and its representation in the case of lifelong multitask learning where the tasks arrive in an online fashion.

1.2 Contributions

The key contribution of this thesis is the idea of using functions to represent the task relationship models in lifelong multitask learning and to learn both the task relationships and task parameters using supervised learning. Using functions to represent task relationships particularly benefits lifelong multitask learning because the functions learnt may be used to transfer knowledge from existing knowledge pool to the new incoming knowledge, and incorporate the new knowledge into the existing knowledge pool. Using functions also help us formulate the problem of task relationship modeling in a supervised framework. In this thesis, we first present all our preliminary studies and contributions, and lastly, present our algorithm for learning task relationships in lifelong setting.

As part of our preliminary work, we studied the task relationship modeling in static multitask learning first. Our main contribution in the static multitask learning is to develop an algorithm that first identifies the related tasks together based on the features shared among tasks and then attempt to train only the related tasks together. There has been some work done in the realm of training heterogenous tasks together as well. However, this algorithm uses a common set of features selected as a source of information shared amongst the related tasks, instead of the relationships that might exist amongst the parameters of the model. It is more intuitive that the tasks which are related might depend on the same factors. In this framework, the prior is designed using a mixture of gaussians over the probabilities of the features to be selected and an spike and slab prior to select the features. The mixture of gaussian prior clusters together all the tasks with similar probabilities of similar features to be selected. Then, the tasks within each group are trained together to further induce selection of similar features across tasks within the same group. Although, this algorithm does produce good results on toxicity dataset, there is a major drawback when using this algorithm. This algorithm requires that the number of tasks be large so that the clustering process is aided with larger number of samples.

It is difficult to find scenarios where the number of tasks might be large. Moreover, there are numerous applications where the number of tasks are small but still many are unrelated to others. Thus, the grouping of the tasks may still be required despite the number of tasks being relatively small. The second algorithm presented in this thesis is another multitask learning algorithm in static setting which also groups the tasks together and assumes that the similar tasks depend on similar features. In this algorithm, I use the bayesian equivalent of the L_0 norm, spike and slab prior, for feature selection. For grouping the tasks, I use categorical distribution rather than mixture of gaussian distribution, which has a similar form with spike and slab prior and thus makes approximation of the posterior easier. Therefore, this algorithm is efficient for even a small number of task, and is more stable and converges faster.

The previous two algorithms are related to static multitask learning setting. The next two algorithms we developed are in the realm of lifelong multitask learning. In these algorithms, we propose the idea of using linear functions to partition the task space to cluster the tasks and learn the task relationships. More specifically, we use a supervised learning approach to partition the task space recursively using linear partition function to identify the related tasks. The advantage of learning a series of linear partition functions is that when a new task arrives, these functions easily transfer knowledge from the previously learnt tasks and the new task easily finds and learns from the set of tasks which are related. Another advantage of learning linear functions is that learning real valued parameters for a straight line are easier than learning binary value indicators or fixed sized matrices to indicate group membership of the tasks in terms of optimization solutions.

The first of these algorithms provide a framework to partition the task space hierarchically, and provide a global framework to use the learnt hierarchical structure to train the tasks in an lifelong learning setting. In the second algorithm, we provide a single framework to learn both the task partitions and the task parameters. In this algorithm, we also assume that similar tasks depend on similar features and provide a method for feature selection again in lifelong learning.

1.3 Organization of this proposal

This thesis initially covers a brief literature survey in the field of multitask learning. Following which we present the algorithms we developed for static multitask learning setting. Both of these algorithms are for learning task clustering with feature selection using L_0 norm. The first algorithm uses mixture of gaussian prior while the second algorithm uses categorical distribution for task clustering. Then, the gear of this thesis is shifted towards lifelong learning. A brief survey of the existing input space partitioning methods which learn a local model in each region is provided. We then use the input space partitioning methods in multitask learning to group the related tasks in lifelong learning setting and provide the description of two algorithms that we developed in this

setting. The thesis is concluded with the future work on which I would like to work on to extend the ideas presented in this thesis.

Chapter 2

Literature Survey on Multitask Learning

Multi-task learning has been long known to improve the generalization performance significantly (20) (142). When there is not enough number of samples to train individual task, it is intuitive to train the related tasks together. In this case information from each task can be used to influence the training of other tasks. The benefits of multitask learning have been shown both theoretically (95) (135) and in practice for real world applications (112) (148) (56) (84) (24).

There are multiple applications of multitask learning which range from prediction of protein-chemical interactions, image annotations, handwritten digit recognition, document and text categorization to robotics and natural language processing. It is fairly clear how these applications are formalized as multitask learning. One of the many interesting applications of multitask learning is in compressive sensing (112)(71) (147) (153). Although machine learning and compressive sensing may seem to be completely different realms, the fundamental principle behind finding the solution for both the fields is same. In both machine learning and compressive sensing, one aims to find the solution of underdetermined problem where the number of variables are greater than the number of equations. In machine learning, multitask learning helps in transferring knowledge between the related tasks and constrain the underdetermined problem. Therefore, if there are multiple related signals or images that are compressed, these images or signals can be uncompressed

together using multitask learning framework. Multitask learning also finds its way in many industrial application such as recommendation systems. The vast and diverse applications of multitask learning make the topic very interesting and popular among researchers.

There are multiple ways in which the broad literature of multitask learning may be organized. For example, the literature can be classified into methods that use feature selection versus methods that don't, or bayesian versus regularized method, or parametric versus non-parametric approaches etc. Here, our discussion will focus on static multitask literature versus online fixed task and lifelong multitask learning. Let us first focus on the generalized formulation of multitask learning, and then we will focus on the vast literature on static multitask learning followed by the work done in online fixed task multitask learning and lifelong multitask learning.

2.1 Generalized Formulation of Multitask Learning

The earlier days of multitask learning were highlighted by ways to modify neural networks to incorporate information from multiple tasks (20), (9). Collobert and Weston (29) apply deep neural networks to learn features common to all tasks in Natural Language Processing using deep neural networks. With time, the multitask learning algorithms have taken a shift towards regularization based approaches.

Most multitask learning algorithms now follow a common structure. Consider the case of learning a group of tasks from the dataset Z, which consists of the input matrix X_t and output vectors y_t, for all $t = 1, 2, ...T$. Here, T are the total number of tasks to be trained. Then, our objective is to find $f_t(X_t)$, such as to minimize the expected loss function between y_t and $f_t(X_t)$ where f_t belongs to a common set S for all t.

$$\min_{f_t \in S} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}(l(f_t(X_t), y_t) \tag{2.1}$$

Solving the above equation would be the same as learning the tasks independently, which may not be always beneficial. For example, if the number of samples are very small compared to the number of features for each of the tasks, it is not possible to learn the tasks independently. In these cases, additional information may be learnt from related tasks. The most common approach to share some information amongst the tasks is to include a regularization term in the above equation which is a constraint provided to connect the tasks together. In bayesian approach, a common prior is usually used to connect the tasks together, which can usually be proven to be same as the regularization function. Thus, a generalized model for multitask learning can be given as

$$\min_{f_t \in S} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}(l(f_t(X_t), y_t) + \lambda \Theta(f) \tag{2.2}$$

The relationship between tasks can be expressed using different approaches by changing the $\Theta(f)$. The value of $\Theta(f)$ is usually selected based on the prior information available about the relationship between the tasks, additional bias that needs to be imposed on the model and also the compatibility of $\Theta(f)$ with the choice of set S and with the loss function used. The choice of $\Theta(f)$ is the major variance among various multitask learning algorithms. There are multiple ways in which the existing multitask learning algorithms might be studied. One such way is to divide them into algorithms that utilize task relationship models versus those which do not in static, online and lifelong learning settings.

2.2 Static Multitask Learning

Literature related to static multitask learning or batch multitask learning has been present for more than a decade. In static multitask learning, it is assumed that all the samples related to all the task are provided beforehand. There is a vast literature present for static multitask learning and it is not possible to review the entire literature in this thesis. However, we provide a summary of the important directions that the literature related to static multitask learning has been in. Mainly, the literature started with the assumption that all the tasks are related with each other and did not

assume any particular task relationship model. Later, researchers started to loosen this assumption and started using and learning task relationship models to prevent negative transfer of incorporation. Here, we first discuss the multitask learning algorithms which do not use task relationship models followed by a brief survey of various methods to incorporate task relationships in multitask learning.

2.2.1 Multitask Learning Algorithms without task relationship model

There are many algorithms which do not make any assumption about the structure of task relationships. These algorithms either incorporate feature learning, or do not incorporate feature learning.

2.2.1.1 Without Feature Learning

There are numerous implementations of multitask learning algorithms which make use of all the features. One kind of such algorithm assumes that the tasks are related by the proximity of the parameters chosen for each task. These kinds of algorithm impose that the parameters remain close together by using an additional regularization term consisting of norm of the weights like the L_2 norm of the difference between the parameters. Let us assume that the set S belongs to a family of linear functions given by the dot product of the feature vector $X_{t,i}$ and the parameter vector w_t. Then, the problem of training multiple tasks reduces to

$$\min_{w_t} \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N_t} l(< w_t, X_{t,i} >, y_t) + \lambda \sum_{t=1}^{T} ||w_t - \frac{1}{T} \sum_{w=1}^{T} w_s||^2 \tag{2.3}$$

Often in the above formulation, L_2 regularization over each w_t is also added for further minimizing the model variance in case of underdetermined problems. Thus, the model parameters are given by

$$\min_{w_t} \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N_t} l(< w_t, X_{t,i} >, y_t) + \lambda_1 \sum_{t=1}^{T} ||w_t - \frac{1}{T} \sum_{t=1}^{T} w_s||^2 + \lambda_2 \sum_{t=1}^{T} \|w_t\|_2 \tag{2.4}$$

in (43), the authors show that the above formulation is equivalent to assuming that $w_t = w_o + v_t$ where w_t are the task parameters of individual task. Here, w_t is decomposed into two parts, w_o

which is common for all the tasks and a task specific vector v_t. In this case, if we consider hinge loss, the above formulation is same as

$$\min_{w_o,v_t} \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N_t} l(< w_t, X_{t,i} >, y_t) + \rho_1 \sum_{t=1}^{T} \|v_t\|^2 + \rho_2 \|w_0\|^2 \tag{2.5}$$

Alternatively, the problem can be formulated as following.

$$\min_{w_t} \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N_t} l(< w_t, X_{t,i} >, y_t) + \lambda tr(W^T \Omega W) \tag{2.6}$$

The formulation mentioned above is formulated as an regularized approach. However, using a common gaussian prior over all the parameters will give the same results. As we can see, the weight vectors are forced to remain close to the average of the weight vectors. Examples of these kind of algorithms can be seen in (43) (9) (42) (161) (88) (74) (106) (161). Using this approach, often there might be some outlier tasks that might cause the performance of all the tasks to go down. Therefore, Shipeng Yu et al (162) used t-processes for developing a robust multitask learning algorithm. T-distribution has a heavier tail than gaussian making it more forgiving for outliers than gaussian distribution.

Other implementation of multitask learning are by using the assumption that each task is a linear combination of common set of basis functions that need to be learned (20) (31) (118).

There are multiple areas in machine learning which are closely linked with multitask learning. For example, multi-label learning problems can easily be formulated as multitask learning where each task is assigned for predicting each label. However, multi-label learning problems only encompass classification tasks where as multitask learning can have both classification and regression tasks. Similarly, multi-class learning problems may be framed as both multi-label problems and multitask learning problem. Thus, it is often considered that multi-label and multi-class problems are a subset of multitask learning problems. Similarly, multitask learning can be viewed as a spe-

cial case of transfer learning where knowledge is transferred in the same domain across different tasks.

In (33), the author, Hal Daume III pointed out the relationship between domain transfer and multitask learning. He developed a hierarchical bayesian framework which is similar for both domain transfer and multitask learning. The only difference being that the parameters are shared in the case of domain transfer and the covariance matrix of the features is shared in the case of multitask learning. There has also been some work done on learning multiple tasks across different domains as in (55). Han et al. use a latent probit model to learn the domain transformations as well as the multiple tasks.

2.2.1.2 With Feature Learning

The most common formulation used for feature learning in multitask learning algorithms is the use of L_1 norm across the parameters of each task to ensure a common set of features are selected for the related tasks (145) (90). This formulation may be similar to group lasso where L_2 norm is used to keep the parameters close together while L_1 norm is used for feature selection(94). (89) propose the use of coordinate descent to optimize such formulation. (169) and (63) observe a general settings of using $L_{1,\infty}$ and connect these models with probabilistic models. Then, they use jefferys prior to deal with outlier tasks as well. Other methods to encourage sparsity is the use of L_0 norm or the spike and slab prior as in the case of (83). Some of these algorithms are formulated as regularization methods (5) (159) , while others use matrix-variate approach (4). Both the approaches are often interchangeable.

Other approaches of multitask learning algorithms with feature learning are algorithms that learn a set of basis vectors for all the tasks as in (165) (141). Alternatively, the common latent structure can also be found using spectral decomposition methods as (7). Or, (85) use a hierarchical bayesian approach to model the feature correlation by using gamma distribution as a prior to the

precision matrix of the prior of each task.

Basically, multitask learning algorithms assume that all the tasks are related to each other in the same way. However, for most applications, we may not have a domain expert's opinion which can help us decide which tasks are related and which are not. Thus, there is a requirement to learn the task relationships, and the above mentioned algorithms do not deal with task relationship learning.

2.2.2 Multitask Learning Algorithms with task relationship model

The other group of multitask learning algorithms are algorithms which relax the assumption that all tasks are related to one another in similar way. These algorithms either utilize the known task relationships that exist among the task, or they try to learn the structure of the task relationships.

2.2.2.1 Incorporation of known task relationships

In most multi-task learning algorithms, it is often assumed that all the tasks are related with each other. However, this is usually not the case, as there might only be some tasks that are closely related. Training with tasks that are not related with each other can further deteriorate the performance (9). There are some applications where we know the task relationships beforehand and this prior knowledge can be used in the model. For example, the task relationships may either be given as an undirected graphs or as a hierarchical tree (directed acyclic graph). If the graph structure is known, this structure can easily be encoded in form of a positive semidefinite matrix which can easily be incorporated in equation 2.6.

The current hierarchical multitask algorithms either bound the label structure in the regularization term to ensure the leaf node parameters are close to their ancestors (18) (23) (35), or use a cascading approach where they train individual classifier, and then try to incorporate the structure information (34) (40) (76) (160) (91). (102) aim at collecting expert traces to acquire the task network and use that for training. However, these algorithms also do not use feature selection, and the work done which incorporates feature learning with incorporating known task relationships is

limited.

2.2.2.2 Identification of Task Relationship

As discussed in section 2.1, most multitask learning algorithms assume that all the tasks are related either physically or in latent space in the same way, which may not be true resulting in performance degradation. If the task structure is known, then that structure can be applied. However, there are many cases where the task structure is not known. A common scenario for such a problem arises in toxicity endpoint predictions. Suppose, we have a dataset consisting of chemical structures and the endpoints for which these chemicals are toxic. Here, the objective is to predict the toxicity of the chemical at each endpoint. This problem can be formulated as a multitask problem where predicting toxicity at each endpoint can be treated as one task. In this case, it can be beneficial to assume that not all tasks are related to each other. If we knew which tasks are related, we could select only those tasks manually for traditional multitask learning and benefit from it. However, we may or may not have this prior knowledge.

A similar scenario can arise in the prediction of occurrence of different types of cancers given the microarray data. Again, there are cancers that might result from mutations in similar genes while others can be caused by mutation in different genes. We know that some of these tasks can benefit when they are trained together, but training all the tasks together might not be beneficial. However, similar to the case of toxicity prediction, we do not have a priori knowledge of which tasks will benefit the most from training together.

Jalali et al, (69) deal with this problem by not enforcing that all parameters selected should be shared across all tasks. That is, only the parameters that help when shared across all tasks are shared. Rest are trained individually, so as to keep the model from having a worse generalization performance than single task learning. They achieve this by breaking their parameter matrix W as sum of two matrices S and B. Their regularizer function is a weighted some of $L_{1,1}$ norm of S

and $L_{1,2}$ norm of B. Thus, part S is sparsified individually, while S undergoes a group sparsification resulting selection of vectors that are shared across all tasks. Similarly, in (156), the authors dirichlet prior to encode higher correlation among some tasks as opposed to others. (15), (167) learn an intertask dependency matrix to learn task relationships in a gaussian process framework. However, these papers only attempts at loosening the definition of the task relatedness, but it does not identifies the groups of related tasks and binds them together. There is still a single group of tasks in these papers.

Recently, there has also been some development in the identification of the tasks that will benefit from training with each other, and only training those tasks together. A common approach for this is clustering of the tasks based on the closeness of the physical parameter space (68). Jacob et al. utilized the covariance of the tasks to create the clusters. Another approach is to use the similarity of the latent space of the parameters of each task to determine the closely related tasks (171). (70) develop a convex formulation using $L_{p,q}$ norm regularizer to group the related tasks together. (170) group the tasks after assuming that each feature can have different cluster structure. They assume that the parameters that need to be learnt in training process can be split into $W = U + V$. There are other algorithms that use this decomposition, but Zhong et al have a regularization term on U forcing the value of U to be close for each feature, while a regularizer in V penalizes the difference of each W from U. This formulation is used to cluster the feature together in cases where feature selection is not required. Another way to identify the task relationships are by representing the parameters of each task as the linear combination of a set of basis vectors where some basis may be common between the tasks while are individually characterize the specific task (79). The above-mentioned algorithms do not use feature selection. There is limited literature that deals with identification of related tasks and feature learning simultaneously. This thesis will briefly deal with this area of multitask learning as well. However, the main topic in this thesis is lifelong multitask learning, which can be viewed as an extension of online multitask learning. Therefore, let us go over the literature related to online multitask learning.

2.3 Online Fixed Task Multitask Learning

Online multitask learning has gained recent attention of the researchers because it is a useful framework when data is being streamed at high speed, as well as when the quantity of data is so large that it is not possible to load the entire data in the memory. With the rapid rise in the interest in big data, online methods in machine learning in general have gained a lot of popularity. In the realm of multitask learning, the major assumption made is that the number of tasks are finite and fixed beforehand. The major advances in online multitask learning algorithms are similar to that of static multitask learning. Although the literature in online multitask learning is not vast, there have been studies in incorporating feature selection and task structure relationships in online multitask learning as well.

Online multitask learning was initially proposed by Dekel et al (36). Dekel et al. assume that the data arrives one sample at a time. For each sample that arrives, the algorithm first makes the prediction for each of the t tasks, and then obtains labels for all the tasks and learns from them. The authors provided the general framework for the online multitask learning and proposed several algorithms based on the same framework. The authors state that the online multitask learning algorithms update the weights according to the equation

$$w_{t,i+1} = w_{t,i} + \tau_{t,i} x_{t,i} y_{t,i} \tag{2.7}$$

where $w_{t,i+1}$ is the weight update of t^{th} task at i^{th} iteration. The weights are updated according to the error estimated on incoming sample $x_{t,i}$ and its label $y_{t,i}$. Different choices of $\tau_{t,i}$ result in different update mechanisms and different algorithms. Most of the online multitask learning algorithms fit in this framework because the formulation indicates that the weight vector is updated according to some function times the incoming sample. Usually, the use of stochastic gradient descent or minimization of regret bounds leads to this framework easily.

Online multitask learning can also be divided into two categories broadly, algorithms which do not use task relationship models versus the algorithms that do consider the task relationship models. There are multiple models which do not consider task relationships (36), (21), (155), (86). (21) is a perceptron based algorithm which adapts to the conditions of access of limited memory, whereas in (155), the authors develop an ensemble based model where a series of forecasts need to be predicted and the true values are not available for all the data points. In this paper, the authors assume that the parameters for each task w_t are given by $w_t = w_0 + v_t$, where w_0 is the global model for all the tasks and v_t are the task specific parameters. Formulation used in (87) is similar to that of (43) in static multitask learning. Li et al in (86) build an online multitask learning model for regression problems. This algorithm is unique in the sense it regularizes the mahalabonis distance between each update of the weight vectors and the difference in error at each iteration for regression problems. The authors assume that for each update of the tasks, the error structures hold significant information pertaining to the tasks as well. (92) propose an interesting aspect of online multitask learning problems where the next course of action is predicted given the current actions and users responses related to the current action. The set from which the next course of action needs to be predicted is also constraint. In (158), (157), (2) and (12) the authors do feature selection in online multitask learning as well.

There are many algorithms which model the task relationship in online multitask learning setting. These algorithms often used a fixed sized matrix which is assumed to be known a priori to represent the task relationships. The examples of these algorithms are (140), (139), (22). Both (140) and (139) use a similar online multitask learning formulation and use it for different applications, namely spelling correction of a query and tracing activities of different people. Cavallanti et al in (22) develop a perceptron based algorithm using fixed task relationship matrix. There are multiple algorithms which aim at learning the task relationship matrix as well such as (125). In this algorithm, the authors penalize the updates in both the task relationship matrix and the task parameters.

There are other ways to encode the task structure as well. For example, (3) and (124) assume all tasks to be related in a lower dimensional space and learn a set basis and their coefficients to learn each task. When the coefficients are sparse, the resulting algorithm assumes the task have overlapping cluster structure. Similarly, in (1), the authors assume that the predictions for each task is given by a small set of experts. (130) build a coactive model which gets its feedback from human beings to learn a structured output.

As can be seen, there is not much existing literature for online multitask methods. The next category of multitask learning algorithms is lifelong multitask learning. This topic is fairly new and the literature related to lifelong multitask learning is extremely limited.

2.4 Lifelong Multitask Learning

Lifelong multitask learning is the process of learning unseen tasks based on the tasks that have previously been learnt. In simple words, lifelong multitask learning is online multitask learning in terms of tasks. The early work of lifelong multitask learning was proposed by (142; 19; 134; 131; 133) and uses neural networks for their algorithm design. Recently, lifelong learning has been studied in detail by Eaton et al in (41). In this study, Eaton et al proposed an algorithm, named ELLA, which uses the knowledge gained from the existing task to learn from the new oncoming task. The authors follow a model similar to (79) and learn a common set of basis for all the tasks and learn a sparse set of coefficients for each task. The sparsity in the coefficients of the basis ensures that all the tasks are not forced to be related in the same way and can deviate from each other based on the basis vectors the tasks depends on. The authors assume that the data arrives in batches with a set of samples from each task incrementally. The authors also extend their formulation for active task selection (122), (123) and reinforcement learning (3). The authors also introduce a faster solution of ELLA by using spectral value decomposition of the basis vectors

which improves the speed while compensating on the accuracy of ELLA (124).

Another study in lifelong multitask learning was done by Pentina et al in (109). The paper deals with the study the Bayesian PAC bounds in lifelong multitask setting. The authors propose that in these settings, the prior also needs to be learnt along with the posterior of each task.

In this thesis, we wish to focus our attention on multitask models that learn the task relationships. The simplest kind of structure that can be learnt is task clustering. Therefore, in this thesis, we develop a couple of algorithms which learn the task clustering as well. These algorithms ensure that the tasks in same clusters depend on the same features. Then, we shift our focus to lifelong multitask learning framework and we develop couple of algorithms to cluster the tasks in lifelong multitask learning setting.

Chapter 3

Preliminary Study: Multitask Learning with Feature Selection for Groups of Related Tasks

Multi-task learning has been long known to improve the generalization performance significantly (20) (142). When there is not enough number of samples to train individual task, it is intuitive to train the related tasks together. In this case, information from each task can be used to influence the training of other tasks. The benefits of multitask learning have been shown both theoretically (95) and in practice for real world applications (112) (148). However, the key assumption here is that all the tasks that are trained simultaneously are related, which is not always true. In many cases, sharing all the tasks together can worsen the generalization error. In these conditions, it might be beneficial to group only the relevant tasks together.

Also, most of the current multi-task learning algorithms assume that the function parameters for each task are close to each other by using either regularization methods (5) (159) or multivariate-matrix penalty (4) (161) methods (which are often reduced to regularization methods). But, when the number of samples are very less than the number of features, these methods do not perform as

well as their counterpart bayesian approaches. But, one problem of traditional bayesian approach is that feature selection in bayesian framework is a difficult task. And, in most cases where the number of features are very large, it is quite unlikely that the given task will depend on all the features. Thus, it is helpful to incorporate some kind of feature selection framework. Extending this argument for the case of multitask learning, it can be the case that the multiple tasks are only related as they depend on the same set of features which need to be selected

Bayesian MTL is a new direction of MTL research. Daniel Hernandez-Lobato et al (61) proposed a Bayesian multitask learning algorithm which utilizes sparsity encouraging priori, coupled with the assumption that all the tasks use the same subset of features. The authors propose the use of spike and slab prior to incorporate feature selection in their algorithm. Then, the authors train the functional parameters of each individual task based on the common subset of features that are selected for all the tasks. The authors use expectation propagation to solve for the resulting posterior distribution.

We study a new extension of Bayesian Multi Task Learning. In our work, we assume that not all tasks are related to each other. Instead, there are a certain number of groups that can be formed by grouping the related tasks. Thus, the primary objective of the current work is to simultaneously recognize the closely related tasks and train only these tasks together. We achieve this by using a mixture of gaussian prior over the probabilities of selecting each feature. This prior helps us to select the closely related tasks group individually. Like (61), we also assume that the related tasks in the same group share the same subset of features. However, we relax this assumption by allowing small variations of the features selected within the same group of tasks as well.

3.1 Background

Before providing a description of the algorithms developed, I would first like to review some background information needed. In this shatter, we will first describe the notation used in rest of the document, then in Section 3.1.2, we will describe the expectation propagation algorithm as developed by Minka(97) (96). In section 3.1.3, we will discuss about the use of expectation propagation in classification with feature selection . Lastly, in section 3.1.4, we will talk about extending the classification algorithm to multi-task framework.

3.1.1 Notation Used

We denote the real numbers by $\mathbb{R}$. All the matrices are denoted by bold face capital letter, and all the vectors are denoted by either the arrow symbol on top of the letter or by lower case bold letters. The scalars are denoted by lower small case letter. The symbol $X_{:,i}$ denotes the i^{th} column of matrix X, and the symbol $X_{i,:}$ denotes the i^{th} row of matrix X. The given dataset Z consists of the input matrices X^k, where $k \in \{1...t\}$ given t number of tasks. Also, $X^k \in \mathbb{R}^{n_k \times d}$ where n_k is the number of samples in each task and d is the number of dimensions of the input. It is not necessary in this algorithm for all tasks to have same number of samples, however, for simplicity purposes, we will just assume each task has n samples. We can easily extend this algorithm to the case when the number of samples are not the same. But, we do assume that the initial input space for each of the task is the same. The given dataset Z also consists of the output vector $y^k \in \{-1, 1\}$ which describes the output for each of the n samples for the k^{th} task.

3.1.2 Expectation Propagation

Before we begin to describe our algorithm, I would like to begin with the case of single task learning, and first discuss the use of expectation propagation for the estimation of the posterior

distribution by approximating the posterior distribution as a Gaussian. This work was primarily done by Thomas Minka (97) (96).

Given, the dataset Z consisting of input matrix $X \in \mathbb{R}^{n \times d}$ and the output vector $\vec{y} \in \{1, -1\}$. Here, we are will describe this method only for linear classification models, even though it is easy to extend expectation propagation into kernel space. Thus, let us assume that the prediction for unknown input vector $\vec{x}$ can be given by the model $y = sign(< \vec{w}, \vec{x} >)$, where $\vec{w}$ is the d dimensional parameter vector that is needed to be estimated. In order to estimate w, $P(w|Z)$ needs to be estimated first.

$$P(w|Z) = \frac{P(Z|w)P(w)}{P(Z)} \tag{3.1}$$

$P(Z|w)$ is the likelihood function of the dataset and can be written as the product of loss functions for each sample, and $P(w)$ is the prior over w. For traditional expectation propagation based classification model, the prior is set to be a standard normal distribution given by $\mathcal{N}(w|0, I)$, where I is the identity matrix. $P(Z)$ is just a normalizing constant. Now,

$$P(w|Z) = \prod_{i=1}^{n} l(w, X_{i,:}, y_i) \ \mathcal{N}(w|0, I) \tag{3.2}$$

Here, $l(w, X_{i,:}, y)$ is the loss function that is used for a given model. Let us assume the use of cumulative Gaussian $\phi(yX_{i,:}w)$ for loss function, as we use this loss function in the method we developed. Also, since through out the algorithm, X is always multiplied by y, we will omit the occurrence of y in the future, and assume that X is always multiplied by y. Estimation of exact distribution of equation 3.2 is highly computationally expensive and can be practically infeasible. Thus, expectation propagation can be used to approximate equation 3.2 as a normal distribution. Let the posterior distribution be denoted by Q. Thus,

$$Q = \mathcal{N}(w|\mu, \Sigma)$$

The objective now becomes to estimate the parameters μ and Σ. In order to approximate equation 3.2 in form of Q, Minka first considers equation 3.2 to be formed by product of $n+1$ terms t_i where $i \in \{1...n+1\}$. Each term t_i can be approximated by a Gaussian $\tilde{t}_i = \mathcal{N}(w|m_i, s_i)$. Thus,

$$Q = \prod_{i=1}^{n+1} \tilde{t}_i = \mathcal{N}(w|\mu, \Sigma)$$

The whole idea of expectation propagation is to remove each $\tilde{t}_i$ term one at a time and replace it with the exact t_i term, and approximate the new posterior as a gaussian. Then, $\tilde{t}_i$ is re-estimated. This entire process is repeated until convergence. The summary of this algorithm is represented by the following equations.

$$\begin{aligned} Q^{\setminus i} &= Q_{old}/\tilde{t}_i \\ Q_{new} &\approx Q^{\setminus i} \times t_i \\ \tilde{t}_i &= Q_{new}/Q_{old} \end{aligned} \tag{3.3}$$

The first step to solve is to initialize the posterior Q by standard normal distribution, and the term approximations $\tilde{t}_i$ with infinite variance Gaussian. The reason for initial choice of Q to be standard normal is that this is the prior which is considered for this algorithm, and the reason for $\tilde{t}_i$ to be flat is to have a non-informative starting point for the term approximations.

The next step is to divide the $\tilde{t}_i$ from the posterior Q. Since, both Q and $\tilde{t}_i$ are normal distribution, dividing $\tilde{t}_i$ from Q will be another Gaussian. Thus,

$$Q^{\setminus i} = \mathcal{N}(w|\mu^{\setminus i}, \Sigma^{\setminus i})$$

where,

$$\Sigma^{\setminus i} = (\Sigma_{old}^{-1} - {s_i}^{-1})^{-1}$$
$$\mu^{\setminus i} = \Sigma^{\setminus i}(\mu_{old}^T \Sigma_{old}^{-1} - {m_i}^T {s_i}^{-1}) \tag{3.4}$$

Now, it is required to estimate the new posterior by estimating $\hat{p} = t_i Q^{\setminus i}$ as a Gaussian. Thus, the Kullback-Leiber divergence between Q_{new} and $t_i Q^{\setminus i}$ is minimized to estimate the parameters μ_{new} and Σ_{new}. Since, Q comprises of product of members from exponential families, Kullback-Leiber divergence can be minimized by simply equating the expected values of the two distribution.

$$E_{Q_{new}}(w) = E_{\hat{p}}(w)$$
$$E_{Q_{new}}(ww^T) = E_{\hat{p}}(w^T w) \tag{3.5}$$

If

$$K(\mu^{\setminus i}, \Sigma^{\setminus i}) = \int_w t_i Q^{\setminus i} dw \tag{3.6}$$

Then, it can be shown that

$$E_{\hat{p}}(w) = \mu^{\setminus i} + \Sigma^{\setminus i} \bigtriangledown_{\mu^{\setminus i}} logK(\mu^{\setminus i}, \Sigma^{\setminus i})$$
$$E_{\hat{p}}(ww^T) - E_{\hat{p}}(w)E_{\hat{p}}(w)^T = \Sigma^{\setminus i} - \Sigma^{\setminus i}(\bigtriangledown_{\mu^{\setminus i}} \bigtriangledown_{\mu^{\setminus i}}^T$$
$$-2 \bigtriangledown_{\Sigma^{\setminus i}} logK(\mu^{\setminus i}, \Sigma^{\setminus i}))\Sigma^{\setminus i} \tag{3.7}$$

Solving 3.7, the values of μ_{new} and Σ_{new} are estimated as

$$\mu^{new} = \mu_{old} + \Sigma_{old} \alpha_i X_{i,:}^T$$
$$\Sigma^{new} = \Sigma_{old} - (\Sigma_{old} X_{i,:}^T)(\frac{\alpha_i X_{i,:} \mu^{new}}{X_{i,:} \Sigma_{old} X_{i,:}^T})(\Sigma_{old} X_{i,:}^T)^T \tag{3.8}$$

where,

$$\alpha_i = \frac{1}{\sqrt{X_{i,:}\Sigma_{old}X_{i,:}^T}} \frac{\mathcal{N}(k;0,1)}{\phi(k)}$$
$$k = \frac{\mu_{old}^T X_{i,:}^T}{\sqrt{X_{i,:}\Sigma_{old}X_{i,:}^T}}$$

Last, the term approximations need to be updated again, which can be done by just dividing Q_{new} by Q_{old}. Thus,

$$\tilde{t_i} = \mathcal{N}(w|m_i, s_i)$$

where,

$$s_i = (\Sigma_{new}^{-1} - \Sigma_{old}^{-1})^{-1}$$
$$m_i = s_i(\mu_{new}^T \Sigma_{new}^{-1} - \mu_{old}^T \Sigma_{old}^{-1}) \tag{3.9}$$

This process has to be repeated for each sample $\forall i = 1...n$, and then repeated till convergence in parameter is acquired. It is not required to deal with the $n+1^{th}$ term, the prior, because the posterior can be simply initialized as the prior. Since, the prior is fixed and has same form as the approximate posterior, it does not need to be updated and approximated.

Also, it is not entirely necessary to consider the full covariance matrix. In our observations, we noticed that we do not lose much by considering each feature to be independent on other and consider the covariance matrix to be only diagonal. But by doing that, we achieve much more stability in our algorithm because instead of dealing with matrix inverses, we can just find the reciprocal of the diagonal terms. The algorithm is also a little faster. In this case, the posterior becomes

$$Q = \prod_{j=1}^{d} \mathcal{N}(w_j|\mu_j, \sigma_j)$$

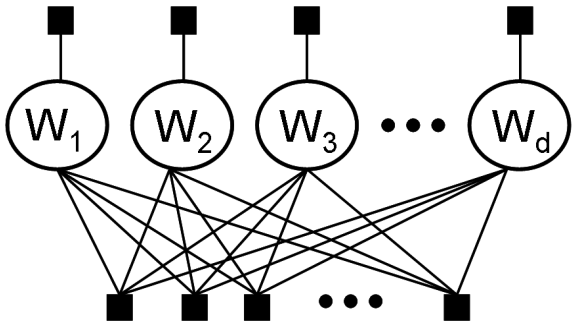


Figure 3.1: Graphical representation of Classification Problem

The other equations remain basically the same, except all the matrix multiplication reduce to element wise multiplication of the vectors and taking inverse of covariance matrix is equivalent to taking reciprocal of each diagonal element. A graphical representation of classification, when assuming each parameter to be independent of other is shown in Figure 3.1.

3.1.3 Classification with Feature Selection

There are a multiple ways, feature selection can be induced in a given algorithm. Using a laplace prior is one way, but often leads to stability issues during optimization. Another way is to use spike and slab prior. Daniel Hernandez-Lobato used spike and slab prior in the expectation propagation framework described in section 3.1.2 to induce feature selection (59) (62). So first, we will review the work of Daniel Hernadez-Lobato in using spike and slab prior in the expectation framework to do feature selection in single task learning (59) (62). The spike and slab prior can be represented as

$$P(w|\gamma) = \prod_{j=1}^{d} \mathcal{N}(w_j|0,\sigma_1^2)^{\gamma_j}\mathcal{N}(w_j|0,\sigma_0^2)^{1-\gamma_j} \tag{3.10}$$

where, γ_j is an induced variable which can only take the values 0 and 1. The value of σ_1 is a larger value (Hernandez-Lobato and we used 1), and the value of σ_0 zero. If the value of γ_j is one, that feature is multiplied by a gaussian with variance 1 as a prior, making this feature have higher probability to be selected. If the value of γ_j is 0, this feature is multiplied by a gaussian with zero variance, making the chances of selecting this feature very feeble. The prior over gamma

is a bernoulli distribution with expected value of 0.5, so that each feature has an equal chances of getting selected to start with.

$$P(\gamma) = \prod_{j=1}^{d} \rho_0^{\gamma_j} (1-\rho_0)^{1-\gamma_j} \tag{3.11}$$

The value of ρ_0 is 0.5.

Now, it is required to infer

$$\begin{aligned} P(\gamma, w|Z) &= \frac{P(Z|w)P(w|\gamma)P(\gamma)}{P(Z)} \\ &\propto (\prod_{i=1}^{n} l(w, X_{i,:}, y_i))(\prod_{j=1}^{d} P(w_j|\gamma_j))P(\gamma) \qquad (3.12) \\ &= \prod_{i=1}^{n+d+1} t_i(w,\gamma) \approx \prod_{i=1}^{n+d+1} \tilde{t_i}(w,\gamma) \qquad (3.13) \end{aligned}$$

Hernandez-Lobato approximated the posterior as product of bernoulli and gaussian distribution. Thus, Q is given by

$$Q(w,\gamma) = \prod_{j=1}^{d} \rho_j^{\gamma_j} (1-\rho_j)^{1-\gamma_j} \mathcal{N}(w_j|\mu_j, \sigma_j) \tag{3.14}$$

Usually, the term-approximation is desired to have the same form as the posterior. Thus,

$$\tilde{t_i}(w,\gamma) = \prod_{j=1}^{d} p_{ij}^{\gamma_j} (1-p_{ij})^{1-\gamma_j} \mathcal{N}(w_j|m_{ij}, s_{ij}) \tag{3.15}$$

However, it can be shown that while approximating the loss function terms, there is no effect on ρ's. Thus, the term approximations used for $i = 1...n$ are just $\mathcal{N}(w_j|m_{ij}, s_{ij})$. For the terms $i = n+1...n+d$, the term approximations are given by equation 3.15.

Next step is to evaluate equation 3.3 for all the terms from $i = 1...n+d$. The $n+d+1^{th}$ term is the prior, and thus, it suffices to set the initial posterior at this value. Thus, the first step is to initialize the ρ_j to 0.5, μ_j to 0, σ_j to 1, m_{ij} to 0, s_{ij} to infinity and p_{ij} to 0.5. Next, divide Q with

$\tilde{t}_i$. For $i = 1...n$, we get

$$\begin{aligned}\sigma_j^{\setminus i} &= (\frac{1}{\sigma_{j_{old}}} - \frac{1}{s_{ij}})^{-1} \\ \mu_j^{\setminus i} &= \sigma_j^{\setminus i}(\frac{\mu_{j_{old}}}{\sigma_{j_{old}}} - \frac{m_{ij}}{s_{ij}})\end{aligned} \tag{3.16}$$

For terms $\tilde{t}_i$ for $i = n+1...n+d$, μ_j and σ_j can be updated as above. The value of ρ_j becomes

$$\rho_j^{\setminus i} = \frac{\rho_{j_{old}}/p_{ij}}{\rho_{j_{old}}/p_{ij} + (1-\rho_{j_{old}})/(1-p_{ij})} \tag{3.17}$$

The next step is to multiply t_i and $Q^{\setminus i}$ and approximate the result in the form of equation 3.14 by minimizing KL divergence. Since, the distributions belong to the exponential family of distributions, it suffices equate the expected values of w, ww^T and γ. Thus, using equation 3.5 along with

$$E_{Q_{new}}(\gamma) = E_{\hat{p}}(\gamma) \tag{3.18}$$

the new parameters for the posterior can be estimated. For this, first it is required to compute the value of K from equation 3.6. For $i = 1...n$, the value of K is same as in expectation propagation, that is $\phi(k)$. For $i = n+1...n+d$,

$$\begin{aligned}K_i =& \sum_{\gamma} \int_w t_i(w,\gamma) Q^{\setminus i}(w,\gamma) dw \\ =& \sum_{\gamma} \int_w \mathcal{N}(w_i|0,\sigma_1^2)^{\gamma_i} \mathcal{N}(w_i|0,\sigma_0^2)^{1-\gamma_i} \\ & \prod_{j=1}^{d} (\rho_j^{\setminus i})^{\gamma_j} (1-\rho_j^{\setminus i})^{1-\gamma_j} \mathcal{N}(w_j|\mu_j^{\setminus i}, \sigma_j^{\setminus i}) dw \\ =& \rho_i^{\setminus i} G_1 + (1-\rho_i^{\setminus i}) G_0\end{aligned} \tag{3.19}$$

where,

$$G_0 = \mathcal{N}(0|\mu_i^{\setminus i}, \sigma_i^{\setminus i} + \sigma_0^2)$$
$$G_1 = \mathcal{N}(0|\mu_i^{\setminus i}, \sigma_i^{\setminus i} + \sigma_1^2)$$

Now, for $i = 1...n$, identity 3.7 is used to solve, and come up with the same solution as in case of simple expectation propagation. For $i = n+1...n+d$, one more additional identity,

$$E_{\hat{p}}(\gamma) = \frac{\partial logK}{\partial p} p(1-p) + p \tag{3.20}$$

in addition to the ones mentioned in equation 3.7 are used to estimate the new parameters of the posterior. And, the new parameters of the posterior are

$$\begin{aligned}
\mu_{new} &= \mu^{\setminus i} + \delta_i c_1 \sigma_i^{\setminus i} \\
\sigma_{new} &= \sigma^{\setminus i} - \delta_i c_3 (\sigma_i^{\setminus i})^2 \\
\rho_{new} &= \rho^{\setminus i} + \delta_i \frac{G_1 - G_0}{K_i} \rho_i^{\setminus i} (1 - \rho_i^{\setminus i})
\end{aligned} \tag{3.21}$$

where, δ_i is a vector of dimensionality d. Only its i^th component is equal to 1, and rest of the terms are zero. Also,

$$\begin{aligned}
c_1 &= \frac{1}{K_i}(\rho_i^{\setminus i} G_1 \frac{-\mu_i^{\setminus i}}{\sigma_i^{\setminus i} + \sigma_1^2} + (1 - \rho_i^{\setminus i}) G_0 \frac{-\mu_i^{\setminus i}}{\sigma_i^{\setminus i} + \sigma_0^2}) \\
c_2 &= \frac{1}{2K_i}(\rho_i^{\setminus i} G_1 (\frac{(\mu_i^{\setminus i})^2}{(\sigma_i^{\setminus i} + \sigma_1^2)^2} - \frac{1}{\sigma_i^{\setminus i} + \sigma_1^2}) \\
&\quad + (1 - \rho_i^{\setminus i}) G_0 \frac{(\mu_i^{\setminus i})^2}{(\sigma_i^{\setminus i} + \sigma_0^2)^2} - \frac{1}{\sigma_i^{\setminus i} + \sigma_0^2}) \\
c_3 &= c_1^2 - 2c_2
\end{aligned}$$

The last step is to update the $\tilde{t}_i$ terms. Since, it is the division of Q_{new} by Q_{old}, the resulting

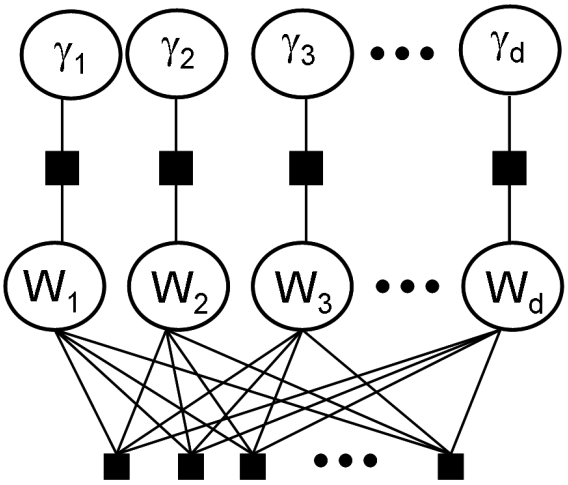


Figure 3.2: Graphical representation of Sparsified Classification Problem using spike and slab prior

equations are similar to equation 3.16 and 3.17. A graphical representation of this method is given in Figure 3.2. Here, we can see that the variable γ_j influences the values of w_j.

3.1.4 Multitask Learning with Feature Selection

Transition from Single task learning with feature selection to multitask learning with feature selection is easy and was implemented by Daniel Hernandez-Lobato et al. (61). The true posterior distribution in this case is given by

$$\begin{aligned} P(\gamma, w|Z) \propto & (\prod_{k=1}^{t}\prod_{i=1}^{n} l(w^k, X^k_{i,:}, y^k_i)) \\ & (\prod_{j=1}^{d} \mathcal{N}(w_j|0,\sigma_1^2)^{\gamma_j}\mathcal{N}(w_j|0,\sigma_0^2)^{1-\gamma_j})P(\gamma) \\ & = \prod_{i=1}^{nk+d+1} t_i(w,\gamma) \approx \prod_{i=1}^{nk+d+1} \tilde{t}_i(w,\gamma) \end{aligned} \tag{3.22}$$

This posterior can be approximated as

$$Q(w,\gamma) = (\prod_{k=1}^{t}\prod_{j=1}^{d} \mathcal{N}(w^k_j|\mu^k_j,\sigma^k_j)) \prod_{j=1}^{d} \rho_j^{\gamma_j}(1-\rho_j)^{1-\gamma_j} \tag{3.23}$$

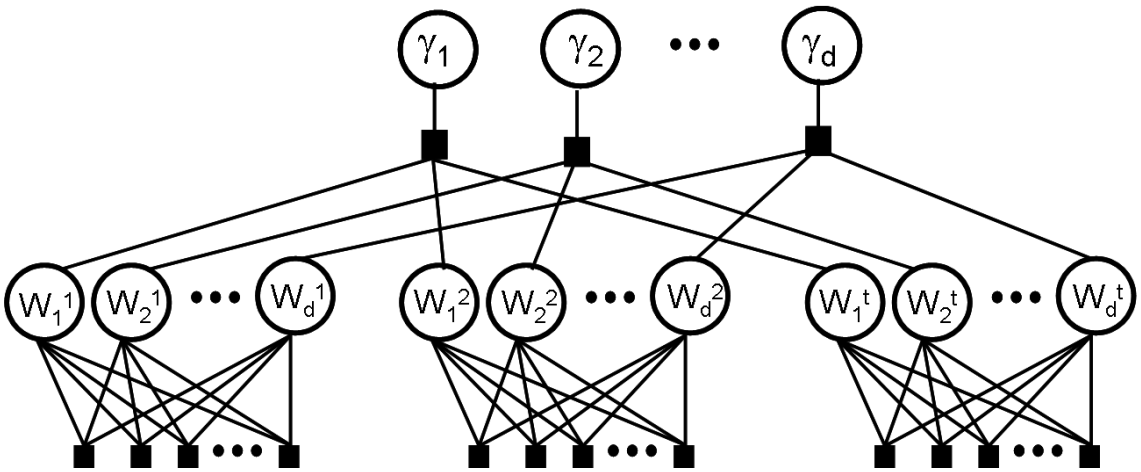


Figure 3.3: Graphical representation of multitask learning with sparsity enforced across features

The application of expectation algorithm on multitask learning is similar to the case of single task learning case as described in section 3.1.3. The graphical representation of this method is shown in Figure 3.3. As can be seen in the figure also, there is not much difference in the single task and multitask learning feature selection. The only difference is γ_j has effect on w_j for all the tasks.

This framework however makes the assumption that all the tasks are related to each other. In the next section I will talk about the algorithm I developed to cluster the tasks together, and use a common set of selected features among the related tasks.

3.2 Methodology

In this paper, we designed an algorithm, which is able to simultaneously group multiple similar tasks and train each group of tasks together. We impose that the tasks in each groups are related by the features they select for performing each task. We use the expectation propagation as a tool to estimate the posterior distribution given the dataset.

3.2.1 Clustering of tasks

In the previous section, we talked about multitask learning with feature selection. The main assumption that was there was that the features selected for all tasks are the same. The measure

for tying tasks together is also selecting a common set of features for all the tasks. However, in real world scenario, it is quite frequent to come across multitask learning algorithms where only certain groups of tasks will share certain features. Our approach in this algorithm is to induce the grouping of tasks that select same features. We essentially use a similar idea to multitask learning with feature selection in the sense that we use spike and slab prior for feature selection and use expectation propagation to estimate the posterior. However, we use mixture of Gaussian prior to encourage grouping the probabilities associated with choosing each feature. In other words, we put the prior on ρ's associated with each task. Assume, we want to classify the tasks into *nclusters* number of clusters. Then,

$$P(\rho) = \sum_{c=1}^{nclusters} \delta_c \mathcal{N}(\rho|\alpha_c, \beta_c) \tag{3.24}$$

This prior is a sum of Gaussian which will encourage the ρ's of different tasks to stay together. However, we notice that this distribution is not a part of exponential family. However, if we add another variable z which follows a categorical distribution (that is, the value of z is 1 for only one element, and for all other elements it is zero), then,

$$\begin{aligned}
P(\rho|z) &= \prod_{c=1}^{nclusters} \mathcal{N}(\rho|\alpha_c, \beta_c)^{z_c} \\
P(z) &= \prod_{c=1}^{nclusters} \delta_c^{z_c} \\
P(\rho, z) &= \prod_{c=1}^{nclusters} \delta_c^{z_c} \mathcal{N}(\rho|\alpha_c, \beta_c)^{z_c}
\end{aligned}$$

$P(\rho, z)$ is from an exponential family. Hence, we approximate the posterior by

$$\begin{aligned}
Q(w, \gamma, z) =& (\prod_{k=1}^{t} \prod_{j=1}^{d} \mathcal{N}(w_j^k|\mu_j^k, \sigma_j^k) \prod_{j=1}^{d} \rho_j^{\gamma_j} (1-\rho_j)^{1-\gamma_j}) \\
& \prod_{k=1}^{t} \prod_{c=1}^{ncluster} \delta_c^{z_c} \mathcal{N}(\rho|\alpha_c^k, \Sigma_c)^{z_c}
\end{aligned} \tag{3.25}$$

The true posterior will then be

$$P(\gamma, w|Z) \propto \prod_{k=1}^{t}(\prod_{i=1}^{n} l(w^k, X_{i,:}^k, y_i^k) \prod_{j=1}^{d} \mathcal{N}(w_j|0,\sigma_1^2)^{\gamma_j}\mathcal{N}(w_j|0,\sigma_0^2)^{1-\gamma_j}) (\prod_{k=1}^{t}\prod_{c=1}^{ncluster} \mathcal{N}(\rho|\rho^k,\lambda_c^k I))(\prod_{c=1}^{ncluster}\sum_{k=1}^{t} r_c^k) \tag{3.26}$$

Here,

$$r_c^k = \frac{\delta_c \mathcal{N}(\rho^k|\alpha_c,\Sigma_c)}{\sum_{a=1}^{ncluster}\delta_a \mathcal{N}(\rho^k|\alpha_a,\Sigma_a)}$$

and

$$\lambda_c^k = \frac{1}{r_c^k}$$

The idea here is to impose that each task in each group of related tasks should choose a set of features which are similar to other tasks within the same group. In other words, for each task there is a zone of closeness around the space of the feature selected. The feature selected for the similar tasks should be within that space and that space is enforced by an overlying Gaussian prior for that group. Thus, the number of Gaussians we choose in the mixture of Gaussian model is same as the number of groups of task we assume our model to have.

The graphical representation of our method is shown in Figure 3.4. It can be clearly seen from the figure that γ_j^t constraints each of w_j^t. However, the values of γ_j^t are forced to cluster together into groups by enforcing another prior which is a mixture of gaussian structure.

In order to approximate true posterior as the approximate posterior, we can make use of the localization property of expectation propagation. Thus, we can just approximate the loss function terms as gaussian as in section 3.1.2. Now, in equation 3.25, we notice that ρ is a variable and not a parameter. Hence, we cannot proceed to update the spike and slab prior using the method

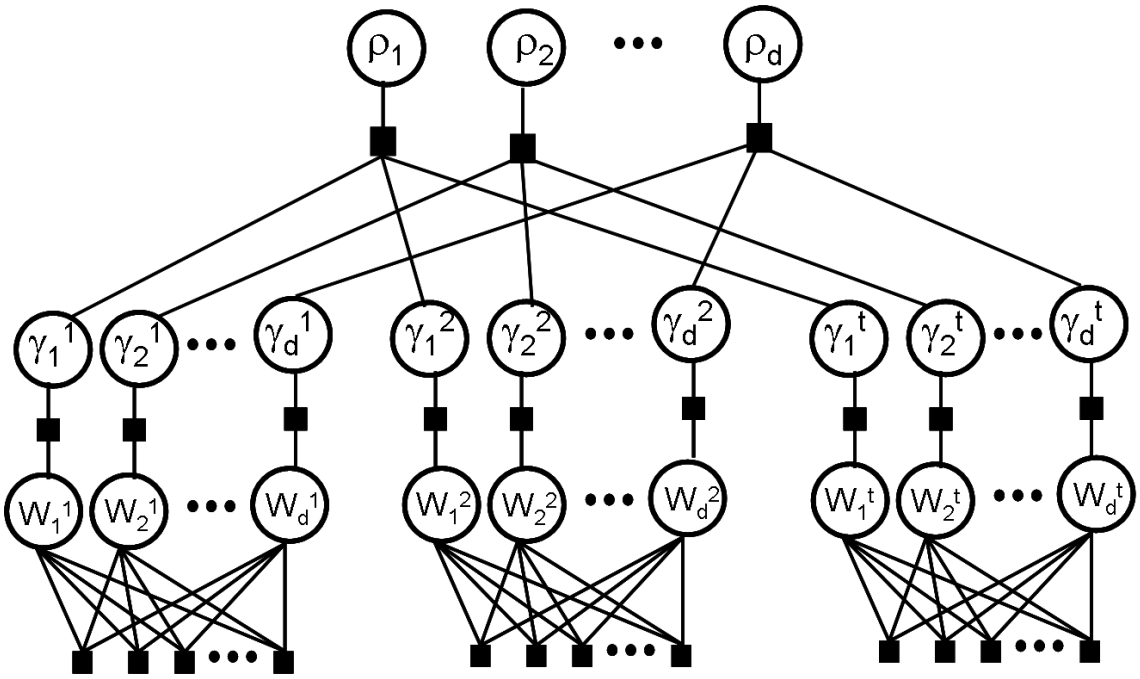


Figure 3.4: Graphical representation of multitask learning with sparsity enforced across features

referred to in sections 3.1.3 and 3.1.4. One way to update $Q^{\setminus i}t_i$ will be to calculate K as integration of $Q^{\setminus i}t_i$ over all the variables, and then use expectation identities for minimizing KL-divergence to approximate $Q^{\setminus i}t_i$ in form of equation 3.25.

For computational ease, instead, we divided the prior into three parts and make use of the hierarchical structure of the prior. We use this approach for this paper. Thus, first, we estimate the values of w and ρ as in the case of single task learning (3.1.3). Next, based on the values of ρ's estimated, we estimate the distribution $P(\rho, z)$ which is the mixture of gaussian model. The values of α_c are randomly chosen to be same as any of the ρ's value, and the values of Σ_c are a d dimensional vector of ones. The δ_c is initialed as $1/ncluster$. The value of r_c^k is then computed for each $c = 1...ncluster$ and $k = 1...t$. Now, for each $k = 1..t$, the term $t_k = \prod_{k=1}^{t} \prod_{c=1}^{ncluster} \mathcal{N}(\rho|\rho^k, \lambda_c^k I)(\prod_{c=1}^{ncluster} \sum_{k=1}^{t} r_c^k)$ is multiplied with $P(\rho, z)$ and the values of α_c, Σ_c and δ_c are re-estimated. The new values of α_c,

and Σ_c are then estimated by simply multiplying the gaussians together. Thus,

$$\Sigma_c = (\Sigma_{c_{old}}^{-1} + \lambda \times 1^{-1})^{-1}$$
$$\alpha_c = \Sigma_c(\alpha_{c_{old}}\Sigma_{c_{old}}^{-1} + \rho_k\lambda \times 1^{-1})^{-1}$$

Here, x^{-1} represent the element wise reciprocal of the vector x, and 1 is just a vecor of ones. δ_c is updated to be the same as r_c^k. Now, the clusters of the tasks have been obtained with means α_c. We now estimate the value of z^k. z^k is equal to 1 for the cluster for which r_c^k is maximum and zero for other values of c. Thus, z^k is 1 for the group to which ρ_k belongs to. We now assign each ρ to be equal to the mean of the group it belongs to. Remainder of the procedure for updating ρ and w is similar to the method referred to in section 3.1.4.

3.3 Results

In this paper, we developed an algorithm for training multiple tasks when the number of features and tasks are large in comparison to the number of samples. We grouped the related tasks together and transferred information only amongst the related tasks. We use spike and slab and mixture of gaussians as prior and use expectation propagation for the inference of the posterior. We compare our method on two datasets, toxcast and mnist. We show that when the number of tasks are large, then our algorithm performs better than simple multitask learning. The algorithms we use for comparing are:

MTLC: This is our method described in this paper.

MTL: This method is the simple multi-task learning with feature selection method.

EP STL: This method is single task learning using expectation propagation. We assume that the covariance matrix is diagonal.

EP STLF: This is single task learning with feature selection.

Kang: This is the method as described in (73). This method is very similar to our method as it finds the groups of relevant tasks together and then trains the relevant tasks together.

We thank Kang and Hernandez for providing us their code for comparison.

3.3.1 Toxicity Dataset

The information gap between the new chemicals developed and the toxicity profiles we have for these chemicals is increasing drastically. The time required for testing a chemical can be a couple of years, but the number of new chemicals being developed each year is around 1000. The number of chemicals untested for their toxicity profile is increasing every year, and for this purpose the Environmental Protection Agency (EPA) thought about developing new methods for toxicity testing of chemicals which is faster and cheaper than the traditional methods. Thus, the EPA collected a dataset comprising of results of certain in vitro assays on each of the chemicals and their entire in vivo toxicity profiles on several species of animals.

The toxicity dataset compromises of 245 unique chemicals. Most of these chemicals are either currently or previously used pesticides, so this ensures that we have the full toxicity profiles of these chemicals. We obtained the structural properties of these chemicals using a software called Dragon version 5.5. The chemicals were also screened using various different assay technologies and the results of those assays were also provided by the EPA. The physical-chemical properties of these chemicals were also accounted for in this dataset. EPA used several softwares to compute the physical-chemical properties of these chemicals.

We concatenated all these descriptors to form the feature set for the data. There were a total

Table 3.1: Comparison of performance of our algorithm with others on toxicity dataset.

Method	Mean Error %	Variance
MTLC	**4.42**	**0.13**
MTL	13.33	0.13
EP STL	5.32	2.92
EP STLF	4.73	0.4
Kang	NA	

of 988 features in the data. The in vivo toxicity results were also compiled by the EPA Toxicity Reference Database. This dataset consists of in vivo toxicity predictions on multiple end-points on rat, rabbit and mouse. There were a total of 181 tasks. We chose the number of groups for my algorithm to be three, because the toxicity was tested on three different species. We divided the entire dataset into two parts, training and testing. Eighty percent of the dataset was used for training and remaining twenty percent for testing. This whole process was repeated 100 times and the mean of the error is recorded here in Table 3.1.

3.3.2 MNIST dataset

MNIST dataset consists of the images of a set of handwritten set of digits from 0 to 9. Each of the images was normalized to fit in 20 $\times$ 20 pixel box, then these images were centered in 28 $\times$ 28 pixels image. For this, first the center of mass of the pixels was computed, and then the pictures were aligned so that the center of mass falls in the center of 28 $\times$ 28 pixels. This database consisted of 60,000 samples. We formed a vector from the 28 $\times$ 28 image giving us a feature vector of 784 dimensional length. We randomly chose 600 samples for training and tested on remaining samples. We repeated this process 100 times. The number of groups were again chosen to be 3 for being consistent. The results are reported in Table 3.2.

Table 3.2: Comparison of performance of our algorithm with others on MNIST dataset.

Method	Mean Error %	Variance
MTLC	7.47	0.12
MTL	**6.81**	**0.12**
EP STL	10.22	0.018
EP STLF	7.67	0.006
Kang	25.3	0.0007

3.3.3 Discussion

As can be seen from Tables 3.1 and 3.2 that when the number of tasks is small, as in MNIST dataset, we did not benefit much from grouping only the relevant tasks together. And thus, simple multi-task learning, which is equivalent to having just one group performed the best. However, when we had a large number of tasks, as in toxicity dataset, the negative transfer of information significantly reduced the performance of multitask learning with feature selection.

To elaborate this point further, we varied the number of tasks in the toxicity datasets between 5 and 70 and plot the error produced. The tasks were picked randomly, and the dataset was also randomly divided into 80% training and 20% testing. This process was repeated five times and the reported values are the mean of the errors. The results are plotted in Figure 3.5. As can be seen, as the number of tasks increases, our method performs significantly better than traditional multitask learning.

We also compare our method against the algorithm developed by Kang et al. We observe that this algorithm takes a very long time to train as compared to ours. For toxicity dataset, training one set of training samples took more than 18 hours and resulted in complex parameters, whereas, our algorithm just took less than 5 minutes. For MNIST dataset, this method does not perform well

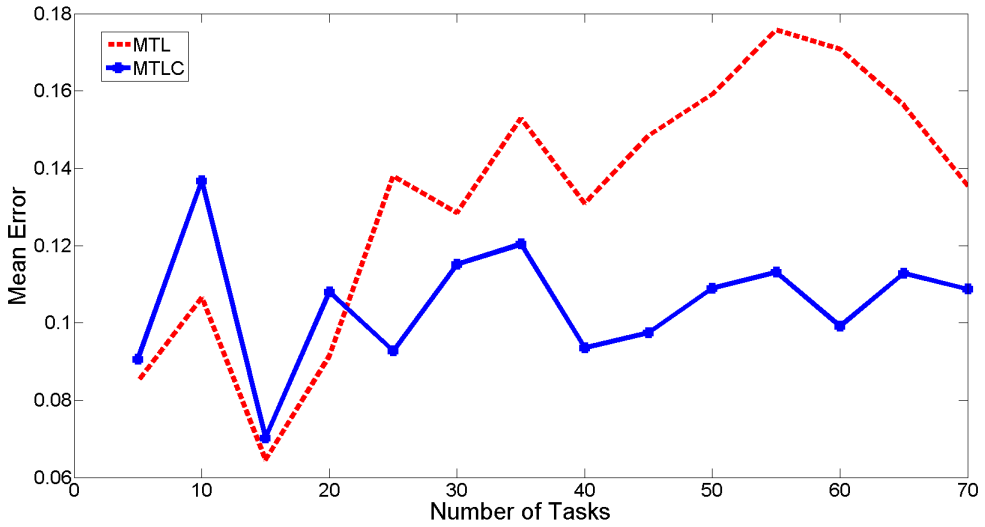


Figure 3.5: The error produced by conventional multi-task with feature selection and our method when the number of tasks is increased

because of the high number of features.

3.4 Conclusion

In the current work, we showed that conventional multitask learning does not always improve the performance. More specifically, if the tasks are not related to each other, the performance of multitask learning deteriorates. Thus, we developed an algorithm that will group the relevant task together and tie the relevant tasks by selecting the common features required to train each tasks. We successfully show that our algorithm does improve the performance when the number of tasks and features is large.

Chapter 4

Identification and Training of Task Clusters in Multitask Feature Learning with L0 Norm using Bayesian Framework

Multitask feature learning is the name given to the process of learning of both the features and multiple tasks function simultaneously (5). It has been an active research topic for several years (6) (103) (90) (169) and finds natural applications in diverse application domains including image recognition, toxicity prediction of chemicals, and gene expression data analysis among others. The major advantage of multitask feature learning is that, it improves the generalization error when the number of features are much larger than the number of samples in a given dataset (103). Multitask feature learning improves the generalization error by sharing the training information between the tasks and also reducing the variance of the parametric model by penalizing the effective number of features (13) which results in an improved performance.

There are two key problems associated with multitask feature learning, first, the use of loosely defined approximations of L_0 norm for feature learning, and second, the assumption that all the tasks in multitask feature learning are related to each other. L_0 norm is the ideal choice for gen-

erating sparsity as it directly penalizes the number of non-zero values. However, the optimization problem using L_0 norm becomes difficult to solve owing to the non-convex nature of L_0 norm. Instead, the most popular choice for feature learning remains to be L_1 norm (5) because it is the nearest convex approximation (51) of the ideal L_0 norm. Hence, L_1 norm is only used because of the ease of optimization, and results in a suboptimal performance (51). The convergence rate using the convex approximation is also suboptimal (166) when compared with L_0 norm.

The other aspect of multitask feature learning is the assumption that all the tasks are related with each other. It has been shown that training unrelated tasks with each other can actually degrade the performance (54). In the recent years, there has been developments in the area of identification, clustering and training of related tasks. For example, Jacob et al (68) and Evgeniou (43) utilized the assumption that the related tasks lie close together in the physical space or share a common prior and group the tasks together. On the other hand, following authors (79), (107), (54) assume that the tasks are related in an underlying latent space to identify the related tasks and train them. However, all of above mentioned algorithms in literature do not use feature learning.

In this paper, we have developed a novel bayesian multitask feature learning framework for the identification and training of related tasks using an equivalent of the ideal L_0 norm which encourages sparsity in the feature space. This approach is simpler and more intuitive than existing clustered multitask feature learning algorithms. In this approach, we formulate a new prior that combines the spike and slab prior, which is an equivalent of L_0 norm, with a categorical distribution. This prior identifies the related tasks together and selects a common set of features that are used by the related tasks. We also show the process of inferring the tasks parameters using the combination of discontinuous priors mentioned above, which is otherwise difficult to obtain. The proposed algorithm in this paper is referred to as "MTLC0" from here onwards. At last, we have tested our algorithm on several real world datasets, such as toxicity prediction, digit recognition and object recognition from descriptive words. Experimental studies show that MTLC0 not only

has better performance than other algorithms, but the time required for inference is also lower than the leading algorithms. The complexity of MTLC0 varies only linearly with the number of tasks, samples and features.

The subsequent discussion is organized as follows. We first present a brief literature review in section 4.1, thereafter, we introduce MTLC0 algorithm which is outlined in section 4.2. In the methodology section, we describe the notation used, present the proposed algorithm and describe the process of inferencing the task parameters. Lastly, in section 4.3, we compare the performance of MTLC0 algorithm with other state of the art algorithms on both synthetic and real world data.

4.1 Related Work

Multitask Learning has been known to improve generalization performance (20). There are multiple approaches for modeling the relationships between tasks in multitask learning. Some algorithms impose a regularization penalty over the task parameters (43) (45), while other algorithms impose a common prior over the task parameters (9) (4) (161). Below, we first focus on feature learning in multitask learning formulation and then the problem of grouping related tasks together.

Feature Learning in multitask formulation is a widely studied topic (90) (6) (5). The most common way to achieve feature learning in multitask formulation is to impose L_1 regularization over the task parameters. L_1 norm is a loosely defined approximation of the ideal L_0 norm which is used for encouraging sparsity (163). Basically, L_0 norm penalizes the number of non-zero terms. Any norm in the interval (0,2) is an approximation to L_0 norm that might be used for feature learning. Norms in the interval (0,1) are not convex, while norms in the interval [1,2) are convex. Thus, the closest convex approximation for L_0 norm is L_1 norm. L_1 norm also has a number of properties related to proving the optimality of solutions, and there are a wide array of tools available to solve L_1 norm. This makes L_1 norm as the most popular choice for feature learning (100). But, using

the L_1 norm results in a performance degradation when compared to L_0 norm (69). To combat this issue, some algorithms propose the use of a combination of L_1 and L_∞ norm (69) (145) (163), which is still a loose approximation of ideal L_0 norm.

The major problem in using the L_0 norm for feature selection is that it is computationally difficult to solve. One way to solve for L_0 norm is to do a subset selection (166). But the complexity of this problem quickly grows with the number of features (66). The equivalent of L_0 norm in bayesian framework is the spike and slab prior (66) (60). Basically, spike and slab prior is a discrete mixture of a point mass and a continuous distribution known as spike and slab. In (61), Hernandez et al. proposed a multitask learning algorithm which made use of spike and slab prior to promote feature learning. However, the assumption made in this algorithm was that all the tasks are related to each other.

Multitask learning with the identification of related tasks is also a popular topic recently. It has been known that training unrelated tasks can actually have a negative impact on the performance rather than improving it. Thus, there have been studies where the researchers have tried to group the related tasks together. In (27), the authors assumed that the task relationships are known a priori. This assumption is more than often not valid, as it is often difficult to identify the related tasks. In (68) and (170), authors assume that the task parameters of related tasks lie close together in the physical space. Thus, the clusters are formed using L_2 norm of the distance of the task parameter with the mean of the parameters of that cluster. Algorithms such as (79), (171), (107) and (54) assume that the tasks are related in underlying latent space and group the tasks using this information. The above mentioned algorithms, however, do not consider feature learning in their framework. In (70), the authors used $L_{p,q}$ norm where both p and q belong to the interval $(1,2)$ for feature learning, and group the tasks together as well. In (73), Kang et al. used feature learning with multitask learning to group the related tasks. However, Kang et al. used L_1 norm to enforce feature learning, which is the closest approximation to L_0 norm proposed here. Also, the authors

use a convex approximation of integer programming to identify the groups of tasks. In (98), the authors had designed an algorithm that coupled spike and slab capped with a mixture of gaussian prior for grouping of related tasks. The key limitation of this work is that the algorithm needs a larger number of tasks to be able to show an improvement in performance over conventional multitask learning without grouping. For a smaller set of tasks, this algorithm often becomes unstable and often does not converge.

In this paper, we develop a novel way for identification of related tasks in multitask learning framework with feature selection. Our algorithm, MTLC0, uses spike and slab prior to identify the related tasks and encourage the same choice of features to be selected within each group of related tasks. We compare MTLC0 majorly with (73), because this algorithm not only groups the tasks but also uses the closest convex approximation of L_0 norm for feature learning. In addition, MTLC0 is also compared against (98) and (61). It is empirically proven that the performance of MTLC0 is superior to other state of the art methods. We also show that our algorithm performs much faster than similar algorithms in the same domain.

4.2 Methodology

In this paper, we develop an algorithm which identifies the tasks which are related to each other, and selects a common set of features for each of the group of related tasks. Thus, only the tasks which are related to each other share information among themselves. With this framework, we hope to improve the generalization performance of multitask learning algorithms. In this section, we will first talk about the notation used, then present the problem formulation and finally present the algorithm that we proposed.

4.2.1 Notation Used

Consider a given dataset Z consisting of input matrix X_t and output vector y_t, for all $t = 1, 2, ...T$, where T is the total number of tasks. Let there be d number of features in the input vector for each sample, and let there be N number of samples. We denote all the matrices by bold capital letters, and all the vectors with bold small case letters. Thus, $x_{t,i}$ denotes the feature vector of i^{th} sample of t^{th} task. We consider a linear classifier where the boundary of the classifier for each task is denoted by $y = X_t w_t$, where w_t is the parameter vector for task t, and W is the entire parameter matrix formed by putting together all the parameter vectors for each of the tasks in individual columns.

4.2.2 Problem Formulation

Linear classifiers are the most popular classifiers. There are several reasons for that. When the dimensional space is very large compared to number of samples, it is always possible to separate the training samples perfectly using a linear classifier. Since, linear classifiers are the simplest classifiers, we do reduce the model variance considerably in choosing a linear classifier without jeopardizing the model bias. Also, it is very easy to change linear classifier into nonlinear classifier by simply projecting the input space to a higher dimensional space such as done in using kernels for classification. Thus, we consider a linear classifier for each of the task, and we assume that the decision for each of the task t can be given by

$$y_t = sign(x_t w_t) \tag{4.1}$$

We have not considered the bias term as bias term can be easily incorporated by simply extending the input vector by a constant.

Further, we assume that the parameters for the above classifier can be obtained from obtaining the mean of a gaussian distribution. Thus, in order to infer the parameters w_t, a bayesian approach

was taken. Utilizing the bayes rule, we get

$$P(w|Z) = \frac{P(Z|w)P(w)}{P(Z)} \tag{4.2}$$

$P(Z|w)$ is the likelihood function of the model, which is a product of loss functions for classification. Here, we consider the loss function of the model to be a cumulative gaussian function which ranges from 0 to 1. Then, the likelihood function is given by

$$P(Z|w) = \prod_{t=1}^{T}\prod_{i=1}^{N} l(x_{t,i,:}w_t y_i) \tag{4.3}$$

where $l(x_{t,i,:}w_t y_i)$ is the loss function used. We can use any other loss function such as 0-1 loss function, or the sigmoid loss function in this case. Both sigmoid and cumulative gaussian function are similar and their advantage over 0-1 loss function is their continuity. Despite being discontinuous, we can easily modify the below algorithm to use 0-1 loss function as well. However, we simply chose cumulative gaussian loss function over others.

$P(w)$ is the prior chosen for the parameters. The choice of standard normal distribution as the prior is basically the same problem formulation as using the L2 norm regularization. In this case, the parameters are penalized in the direction the variation of the data is least. Since, we want to incorporate feature selection in our formulation, using a standard normal distribution for prior is not beneficial. Priors such as Laplace distribution, which is similar to L1 regularization, is not ideal either in the sense that these priors assign a zero probability to the case that some of the parameters are zero (60) (83). ARD prior is also used to encode sparsity, but this prior does not encode the uncertainty of selecting each feature in the posterior. In this paper, spike and slab prior was used because it is considered as the golden standard for the sparse prior from bayesian perspective (83) and it is equivalent of ideal L_0 norm used for feature learning. The reason spike and slab prior is not used often is that because of its discontinuity, it is much harder to infer.

The spike and slab prior is given by

$$P(w_j|\gamma_j) = \mathbb{N}(w_j|0,\sigma_1^2)^{\gamma_j}\mathbb{N}(w_j|\mu_j,\sigma_0^2)^{1-\gamma_j} \tag{4.4}$$

where $\sigma_1 = 1$ and $\sigma_0 = 0$. Thus, if a feature j is selected, the value of γ_j is one and a standard normal prior is associated with that feature. If a feature is not selected, this implies that $\gamma_j = 0$, and a spike is associated as the prior of that feature. This indicates that there is a very feeble probability of that feature to be selected.

The above mentioned spike and slab prior can also be approximated as a product of bernoulli distribution (61),

$$\prod_{j=1}^{d} \rho_j^{\gamma_j}(1-\rho_j)^{(1-\gamma_j)} \tag{4.5}$$

where γ_j is a bernoulli variable and only accepts the value of 0 and 1. Thus, a feature j is selected if γ_j is 1, else feature j is rejected.

In order for simple multitask learning as in (61), we can keep the same γ_j for all tasks, enforcing the choice of features which are selected and rejected are same across all tasks. However, identification of relevant tasks is also a key problem, and we need to ensure that similar features are shared across tasks belonging to the same group. One way to group the tasks together is to keep different $\gamma_{j,t}$ for each feature of each task, and then cluster the $\gamma_{:,t}$ together so that groups of tasks are identified as in (98). However, this is a much slower approach where the rigidness of the coupling of the tasks is not tight enough. Hence, it does not necessarily improve the performance of the generalization error over multitask learning and needs a large number of tasks to converge.

In this paper, the grouping of the tasks is considered using a categorical distribution which is also an approximation of L_0 norm. Thus, we propose using the same values for γ_j for the tasks in

a same group. The group with which a task belongs to is decided using a categorical distribution. The prior in this case is given by

$$P(w|\gamma,\delta) = \prod_{k=1}^{n_c} \delta_{k,t}^{z_{k,t}} \left(\prod_{j=1}^{d} (\mathbb{N}(w_j|0,\sigma_1^2)^{\gamma_{j,k}} \mathbb{N}(w_j|\mu_j,\sigma_0^2)^{(1-\gamma_{j,k})} \right)^{z_{k,t}} \tag{4.6}$$

Here, $\delta_{k,t}$ is the probability of task t to be associated with group k. For any given task t, the sum of $\delta_{:,t}$ is constrained to be equal to one. Also, $z_{k,t}$ is a categorical variable, which can take the value of one for only one k, and is zero for all other values of k. Thus, the posterior of the parameters is formulated as

$$P(w|Z,\gamma) = \prod_{t=1}^{T} \left(\prod_{i=1}^{N} l(x_{t,i,:} w_t y_i) \right) \prod_{k=1}^{n_c} \delta_{k,t}^{z_{k,t}} \left(\prod_{j=1}^{d} (\mathbb{N}(w_j|0,\sigma_1^2)^{\gamma_{j,k}} \mathbb{N}(w_j|\mu_j,\sigma_0^2)^{(1-\gamma_{j,k})} \right)^{z_{k,t}} \tag{4.7}$$

It is difficult to infer this framework of the posterior because of the discontinuous probabilities involved at both feature selection level and group selection level. Thus, we approximate the above posterior using expectation propagation.

4.2.3 Expectation Propagation

Expectation Propagation is an approximation technique formulated by Thomas Minka in 2001. It is inspired from kalman and adaptive filters. According to expectation propagation, the posterior Q is considered to be a product of n terms, t_i, for $i = 1,2,...n$. Suppose we need to approximate Q with $\tilde{Q}$. Then, all the terms t_i get approximated by $\tilde{t_i}$.

The way expectation propagation works is that the term $\tilde{t_i}$ is removed one at a time and replaced with the exact t_i term. The resulting product is approximated as $\tilde{Q}$. Then, $\tilde{t_i}$ is re-estimated. This entire process is repeated until convergence. The summary of this algorithm is represented by the following equations.

$$\begin{aligned} Q^{\setminus i} &= Q_{old}/\tilde{t_i} \\ Q_{new} &\approx Q^{\setminus i} \times t_i \\ \tilde{t_i} &= Q_{new}/Q_{old} \end{aligned} \tag{4.8}$$

For our case, the true posterior distribution is given by Equation 4.7. One way to approximate this posterior is to approximate it as a product of gaussian, bernoulli and categorical distribution. Thus,

$$\begin{aligned} P(w|Z,\gamma) = &\prod_{t=1}^{T}\left(\prod_{i=1}^{N} l(x_{t,i,:}w_t y_i)\right) \\ &\prod_{k=1}^{n_c} \delta_{k,t}^{z_{k,t}} \left(\prod_{j=1}^{d} \rho_{k,j}^{\gamma_{j,k}}(1-\rho_{k,j})^{(1-\gamma_{j,k})}\right)^{z_{k,t}} \end{aligned} \tag{4.9}$$

We consider breaking the problem into two parts. The first part consists of the product of loss functions which needs to be approximated as the gaussian. The second part is approximating the spike and slab prior as the product of bernoulli distribution as given in equation 4.5. The second part is the step which not only encourages sparsity, but is responsible for grouping the tasks as well. Basically, the second part involves inferring the values of $\delta_{k,t}$ and $\rho_{k,j}$.

The approximation of the first term is done just as in (96). The first step to solve is to initialize the posterior Q by standard normal distribution, and the term approximations $\tilde{t_i}$ with infinite

variance Gaussian. The reason for initial choice of Q to be standard normal is that this is the initial prior which is considered for classification without feature selection in (96), and the reason for $\tilde{t_i}$ to be flat is to have a non-informative starting point for the term approximations.

The next step is to divide the $\tilde{t_i}$ from the posterior Q. Since, both Q and $\tilde{t_i}$ are normal distribution, dividing $\tilde{t_i}$ from Q will be another Gaussian. Thus,

$$Q^{\setminus i} = \mathcal{N}(w|\mu^{\setminus i}, \Sigma^{\setminus i})$$

where,

$$\begin{aligned} \Sigma^{\setminus i} &= (\Sigma_{old}^{-1} - {s_i}^{-1})^{-1} \\ \mu^{\setminus i} &= \Sigma^{\setminus i}(\mu_{old}^{T}\Sigma_{old}^{-1} - {m_i}^{T}{s_i}^{-1}) \end{aligned} \tag{4.10}$$

Now, it is required to estimate the new posterior by estimating $\hat{p} = t_i Q^{\setminus i}$ as a Gaussian. Thus, the Kullback-Leiber divergence between Q_{new} and $t_i Q^{\setminus i}$ is minimized to estimate the parameters μ_{new} and Σ_{new}. Since, Q comprises of product of members from exponential families, Kullback-Leiber divergence can be minimized by simply equating the expected values of the two distribution.

$$\begin{aligned} E_{Q_{new}}(w) &= E_{\hat{p}}(w) \\ E_{Q_{new}}(ww^T) &= E_{\hat{p}}(w^T w) \end{aligned} \tag{4.11}$$

If

$$K(\mu^{\setminus i}, \Sigma^{\setminus i}) = \int_w t_i Q^{\setminus i} dw \tag{4.12}$$

Then, it can be shown that

$$
\begin{aligned}
&E_{\hat{p}}(w) = \mu^{\setminus i} + \Sigma^{\setminus i} \bigtriangledown_{\mu^{\setminus i}} logK(\mu^{\setminus i}, \Sigma^{\setminus i}) \\
&E_{\hat{p}}(ww^T) - E_{\hat{p}}(w)E_{\hat{p}}(w)^T = \Sigma^{\setminus i} - \Sigma^{\setminus i}\big(\bigtriangledown_{\mu^{\setminus i}} \bigtriangledown_{\mu^{\setminus i}}^T \\
&- 2 \bigtriangledown_{\Sigma^{\setminus i}} logK(\mu^{\setminus i}, \Sigma^{\setminus i})\big)\Sigma^{\setminus i}
\end{aligned} \tag{4.13}
$$

Solving 4.13, the values of μ_{new} and Σ_{new} are estimated as

$$
\begin{aligned}
&\mu_{new} = \mu_{old} + \Sigma_{old}\alpha_i X_{i,:}^T \\
&\Sigma_{new} = \Sigma_{old} - (\Sigma_{old}X_{i,:}^T)\left(\frac{\alpha_i X_{i,:}\mu_{new}}{X_{i,:}\Sigma_{old}X_{i,:}^T}\right)(\Sigma_{old}X_{i,:}^T)^T
\end{aligned} \tag{4.14}
$$

where,

$$
\begin{aligned}
\alpha_i &= \frac{1}{\sqrt{X_{i,:}\Sigma_{old}X_{i,:}^T}} \frac{\mathcal{N}(k;0,1)}{\phi(k)} \\
k &= \frac{\mu_{old}^T X_{i,:}^T}{\sqrt{X_{i,:}\Sigma_{old}X_{i,:}^T}}
\end{aligned}
$$

Last, the term approximations need to be updated again, which can be done by just dividing Q_{new} by Q_{old}. Thus,

$$
\tilde{t_i} = \mathcal{N}(w|m_i, s_i)
$$

where,

$$
\begin{aligned}
&s_i = (\Sigma_{new}^{-1} - \Sigma_{old}^{-1})^{-1} \\
&m_i = s_i(\mu_{new}^T\Sigma_{new}^{-1} - \mu_{old}^T\Sigma_{old}^{-1})
\end{aligned} \tag{4.15}
$$

A small note we want to make here is that we make an assumption that Σ matrices in the above

case are diagonal. This assumption reduces the complexity of the algorithm, and according to our observations, does not effect the performance significantly. In many cases, we have actually observed an improvement in performance. This assumption also helps in improving the stability of the algorithm.

The next step is the approximation of ρ terms. For this, we work according to the guidelines of (61). However, in this algorithm, we need to learn the clusters of the related tasks, and ensure that similar features are shared across tasks belonging to the same group. Thus, we divided this part of inference into two sub-parts. The first part focuses largely on inferring ρ, which is the probability of each feature to be selected within a group. The second sub-part focuses on inferring δ, which is the probability of each task to belong to a certain group. Thus, first we need to initialize the ρ_j to 0.5, μ_j to 0, σ_j to 1, m_{ij} to 0, s_{ij} to infinity and p_{ij} to 0.5. Next, divide Q with $\tilde{t}_i$. For $i = 1...n$, we get

$$\begin{aligned} \sigma_j^{\setminus i} &= \left(\frac{1}{\sigma_{j_{old}}} - \frac{1}{s_{ij}}\right)^{-1} \\ \mu_j^{\setminus i} &= \sigma_j^{\setminus i}\left(\frac{\mu_{j_{old}}}{\sigma_{j_{old}}} - \frac{m_{ij}}{s_{ij}}\right) \end{aligned} \tag{4.16}$$

For terms $\tilde{t}_i$ for $i = n+1...n+d$, μ_j and σ_j can be updated as above. The value of ρ_j becomes

$$\rho_j^{\setminus i} = \frac{\rho_{j_{old}}/p_{ij}}{\rho_{j_{old}}/p_{ij} + (1-\rho_{j_{old}})/(1-p_{ij})} \tag{4.17}$$

The next step is to multiply t_i and $Q^{\setminus i}$ and approximate the result in the form of equation 4.7 by minimizing KL divergence. Since, the distributions belong to the exponential family of distributions, it suffices equate the expected values of w, ww^T and γ. Thus, using equation 4.11 along with

$$E_{Q_{new}}(\gamma) = E_{\hat{p}}(\gamma) \tag{4.18}$$

the new parameters for the posterior can be estimated. For this, first it is required to compute the

value of K from equation 4.12. For $i = 1...n$, the value of K is same as in expectation propagation, that is $\phi(k)$. For $i = n+1...n+d$,

$$\begin{aligned}
K_i^t &= \sum_{\gamma,z} \int_w t_i(w,\gamma) Q^{\setminus i}(w,\gamma) dw \\
&= \sum_{\gamma,z} \int_w \mathcal{N}(w_i^t|0,\sigma_1^2)^{\gamma_i} \mathcal{N}(w_i^t|0,\sigma_0^2)^{1-\gamma_i} \\
&\quad \prod_{k=1}^{nc} \prod_{j=1}^{d} \left(\delta_{k,t} (\rho_{j,k}^{\setminus i})^{\gamma_{j,k}} (1-\rho_{j,k}^{\setminus i}\right)^{1-\gamma_{j,k}})^{z_{k,t}} \\
&\quad \mathcal{N}(w_j|\mu_j^{\setminus i}, \sigma_j^{\setminus i}) dw \\
&= \sum_{k=1}^{nc} \left[\delta_{k,t} (\rho_{j,t}^{\setminus i} G_1 + (1-\rho_{j,t}^{\setminus i}) G_0) \right]^{z_{k,t}}
\end{aligned} \tag{4.19}$$

where,

$$\begin{aligned}
G_0 &= \mathcal{N}(0|\mu_i^{\setminus i}, \sigma_i^{\setminus i} + \sigma_0^2) \\
G_1 &= \mathcal{N}(0|\mu_i^{\setminus i}, \sigma_i^{\setminus i} + \sigma_1^2)
\end{aligned}$$

Now, for $i = 1...n$, identity 4.13 is used to solve, and come up with the same solution as in case of simple expectation propagation. For $i = n+1...n+d$, one more additional identity,

$$E_{\hat{p}}(\gamma) = \frac{\partial log K}{\partial p} p(1-p) + p \tag{4.20}$$

in addition to the ones mentioned in equation 4.13 are used to estimate the new parameters of the posterior. And, the new parameters of the posterior are

$$\begin{aligned}
\mu_{new} &= \mu^{\setminus i} + d_i c_1 \sigma_i^{\setminus i} \\
\sigma_{new} &= \sigma^{\setminus i} - d_i c_3 (\sigma_i^{\setminus i})^2 \\
\rho_{new} &= \rho^{\setminus i} + d_i \frac{G_1 - G_0}{K_i} \rho_i^{\setminus i} (1-\rho_i^{\setminus i})
\end{aligned} \tag{4.21}$$

where, d_i is a vector of dimensionality d. Only its i^{th} component is equal to 1, and rest of the terms are zero. Also,

$$
\begin{aligned}
c_1 &= \frac{1}{K_i}\sum_{k=1}^{nc}\delta_{k,t}^{\backslash i}\Big(\rho_{i,k}^{\backslash i}G_1\frac{-\mu_i^{\backslash i}}{\sigma_i^{\backslash i}+\sigma_1^2}+(1-\rho_{i,k}^{\backslash i})G_0\frac{-\mu_i^{\backslash i}}{\sigma_i^{\backslash i}+\sigma_0^2}\Big)\\
c_2 &= \frac{1}{2K_i}\sum_{k=1}^{nc}\delta_{k,t}^{\backslash i}\Big[\rho_{i,k}^{\backslash i}G_1\Big(\frac{(\mu_i^{\backslash i})^2}{(\sigma_i^{\backslash i}+\sigma_1^2)^2}-\frac{1}{\sigma_i^{\backslash i}+\sigma_1^2}\Big)\\
&\quad +(1-\rho_{i,k}^{\backslash i})G_0\Big(\frac{(\mu_i^{\backslash i})^2}{(\sigma_i^{\backslash i}+\sigma_0^2)^2}-\frac{1}{\sigma_i^{\backslash i}+\sigma_0^2}\Big)\Big]\\
c_3 &= c_1^2-2c_2
\end{aligned}
$$

The last step is to update the $\tilde{t}_i$ terms. Since, it is the division of Q_{new} by Q_{old}, the resulting equations are similar to equation 4.16 and 4.17.

$$
\begin{aligned}
s_{ij} &= \left(\frac{1}{\sigma_{j_{new}}}-\frac{1}{\sigma_j^{\backslash i}}\right)^{-1}\\
m_{ij} &= s_{ij}\left(\frac{\mu_{j_{new}}}{\sigma_{j_{new}}}-\frac{\mu_j^{\backslash i}}{\sigma_j^{\backslash i}}\right)\\
p_{ij} &= \frac{\rho_{j_{new}}/\rho_j^{\backslash i}}{\rho_{j_{new}}/\rho_j^{\backslash i}+(1-\rho_{j_{new}})/(1-\rho_j^{\backslash i})}
\end{aligned}
\tag{4.22}
$$

The next sub-part involves the inference of the δ terms to identify the group of tasks as well. For approximating the last part, the first step again is to divide Q with $\tilde{t}_i$. So, delta is updated as

$$
\delta_{k,t}^{\backslash i} = \frac{\delta_{k,t_{old}}/d_{k,t}}{\sum_{m=1}^{n_c}\delta_{m,t_{old}}/d_{m,t}} \tag{4.23}
$$

The next step is to approximate $Q^{\backslash i}t_i$ to the form of Q^i by minimizing the KL divergence between the two distributions. Since, the posterior is a member of exponential family, KL divergence

is minimized by equating the expectations of both the distributions. Thus, the value of $\delta_{k,t}$ would be the same as the expectation of $Q^{\setminus i}t_i$.

Consider a distribution of the form $p(x) = \frac{1}{K}t(x)f(x|\delta)$ where $f(x|\delta)$ is a categorical distribution and $t(x)$ is an arbitrary function of x, where $K = \sum_x t(x)f(x|\delta)$, then the expected value of δ can be easily derived as

$$E_{\delta_j}(x_j) = \frac{\partial logK}{\partial \delta_j}\delta_j \tag{4.24}$$

Thus, the value of K can be obtained from equation 4.19. Plugging it in above equation, the value of $\delta_{k,t}$ is given by

$$\delta_k^t = \frac{1}{K}\delta_{k,t}^{\setminus i}\left(\rho_{j,t}^{\setminus i}G_1 + (1-\rho_{j,t}^{\setminus i})G_0\right) \tag{4.25}$$

Finally, the $\tilde{t}_i$ terms are updated using equation similar to 4.23.

$$d_{k,t} = \frac{\delta_{k,t_{new}}/\delta_{k,t}^{\setminus i}}{\sum_{m=1}^{n_c}\delta_{m,t_{new}}/\delta_{m,t}^{\setminus i}} \tag{4.26}$$

For the above algorithm to work, we need to first perturb the initial values of $\rho's$ and $\delta's$. So, we first solve equations 4.14 and 4.21 for all samples for all tasks only once. Then we use k-means to perform a quick estimation of ρ_k, and initialize δ_k accordingly. Then, the algorithm is implemented further until convergence to estimate the correct values of $W's$, $\rho's$ and $\delta's$.

4.2.4 Summary of the Algorithm

A summary of the above mentioned algorithm is mentioned in the Algorithm 1. Basically, we first need to initialize all the parameters of the algorithm, and then solve the above-mentioned equations iteratively until convergence is reached. The final class of a given sample can be predicted by using

Algorithm 1 Summary of MTLC0 algorithm

Require: Number of groups nc, input data and output labels
Initialize all μ as 0, σ as 1, ρ as 0.5 and δ as $1/nc$.
Initialize the term parameters m's as 0, s's as ∞, p's as 0.5, d's as $1/nc$
Solve 4.14 and 4.21 once for all $i = 1..n$ and $t = 1..T$, and cluster the rho's into nc groups for inital estimate of ρ and δ
repeat

for all t=1..T **do**

for all i=1..n **do**

Divide term approximations from Q using equations 4.10.
Find updated μ and Σ using equation 4.14
Update term approximations using equation 4.15
end for
end for
for all t=1..T **do**

for all j=1..d **do**

Divide term approximations from Q using equations 4.16 and 4.17.
Find updated μ, σ, ρ and δ using equations 4.21 and 4.25 for all values of k=1...nc
Update the term approximations using equations 4.22 and 4.26 for all values of k=1..nc
end for
end for
until covergence
return The mean of the weights μ

the equation $y = sign(x_t^T \mu_t)$.

4.3 Experimental Study

In this paper, we presented an algorithm that will identify the groups of related tasks and select a common set of features for each group of task. First, we show the performance of our dataset on synthetic data, and then we present the results on several real world. At the end, we show how our algorithm compares in the training time compared to other state of the art data. We compare our

results against the following algorithms:

MTLF: This algorithm was formulated by Hernandez et al. (61). It is similar to our algorithm as it uses spike and slab prior for the feature selection process. But this algorithm also makes the assumption that all the tasks are related to each other and does not group the tasks.

Kang: This algorithm learns the groups of related tasks along with feature learning and then trains the model (73). This algorithm is the closest to MTLC0, as it uses $L-1$ norm for feature learning, which is the closest convex approximation to L_0 norm, and groups the related tasks together.

MNEP: This algorithm (98) also uses spike and slab prior for feature selection as well, and uses mixture of gaussian prior for grouping the tasks. As can be seen from the results, it has high convergence problems when the number of tasks are small.

cCMTL: Convex relaxation of Clustered Multitask Learning (171) uses k-means clustering by minimizing sum of squared error to discover the task relationships. This method does not do feature selection.

CASO: Convex relaxation of Alternate Structure Optimization (26) learns a common shared structure for all the task in lower dimensional subspace.

JMTFL: JMTFL (70) is another algorithm for clustered multitask learning with feature selection. However, we could not compare with this method as this algorithm produced out of memory error for our datasets.

MTLC0: Proposed algorithm

We would like to thank Jawanpuria, Chen, Zhou, Kang and Hernandez for providing us with their

code.

4.3.1 Synthetic Dataset

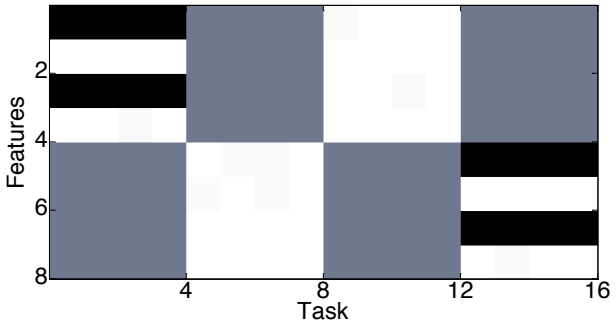
We designed a synthetic dataset to check the working of our algorithm. The dataset consisted of sixteen tasks, thirty samples and eight features. The thirty samples for each task were drawn from normal distribution with zero mean and 0.1 variance. The labels of the tasks were set equal to $y_i^t = sign(\beta_t x_i^t)$. For tasks 1 to 4, the parameters β were drawn from a gaussian of mean $[-1,1,-1,1,0,0,0,0]$. For task 5 to 8, parameters were drawn from gaussian distribution of mean $[0,0,0,0,1,1,1,1]$, and for tasks, 9 to 12, the parameters were drawn from normal distribution of mean $[1,1,1,1,0,0,0,0]$. For the last four tasks, the parameters were drawn from normal distribution of mean $[0,0,0,0,-1,1,-1,1]$. The variance was kept at a value of 0.01 for each case. The true value of parameters are represented in Figure 4.1a. The number of groups in this case are 4. This dataset was constructed 100 times and divided into 60% for training and 40% for testing and tested on the algorithms mentioned above. The evaluation metric we use is the fraction of incorrectly classified samples. The results are summarized in Table 4.1.

The image of the parameters for one randomly picked instance from the training process is also plotted in Figure 4.1. It can be seen that the parameters estimated by MTLC0 much closely represents the ground truth as compared to Kang et al's method in terms of the groups of non-zero and zero values formed. This difference, even though may seem to be small, results in significant improvement in the average classification error as shown in Table 4.1.

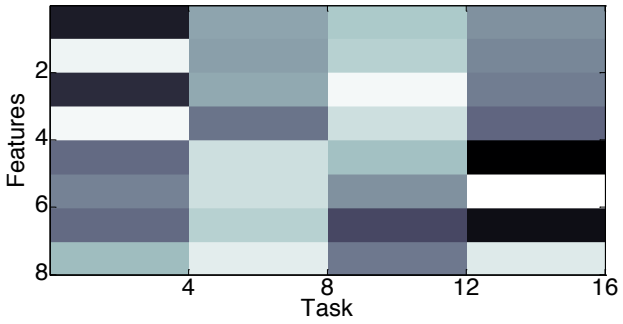
We also vary the number of groups to 2, 4, 6 and 8 and evaluate the performance of each of the algorithms. These results are also summarized in Table 4.1. It can be observed that MTLC0 has a superior performance in all cases.

Table 4.1: Performance on Synthetic Dataset for varying number of groups. Here, fraction of misclassified samples are reported. G stands for the number of groups used

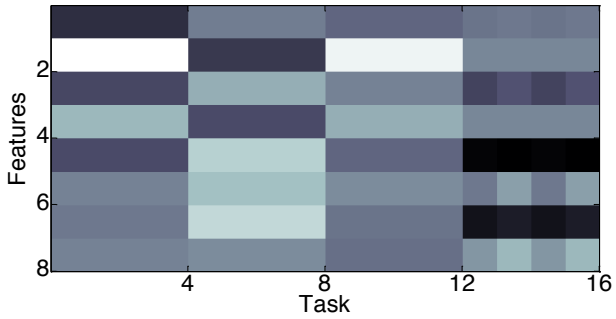
	Error			
Algorithm	G=2	G=4	G=6	G=8
MTLF	0.1890	0.1890	0.1890	0.1890
Kang	0.2150	0.2170	0.2162	0.2147
MNEP	0.2651	0.2676	0.2662	0.2658
MLC0	**0.1769**	**0.1781**	**0.1788**	**0.1762**



(a) Ground Truth



(b) MTLC0



(c) Kang et al

Figure 4.1: The figure displays the value of the normalized weights estimated for the sythetic data. (a) is the ground truth, (b) is the result from current MTLC0 method and (c) is the result from Kang et al's method. The number of groups was set to 4. In these figures, -1 and 1 are represented by black and white colors respectively. All the in-between values are represented by shades of gray.

4.3.2 Datasets Used

We considered several real world datasets from different realms of life. We use three datasets from Kang's et al paper called small-MNIST, USPS, and Animal dataset. We also downloaded the dataset from the MNIST website and used the raw data without any preprocessing. Additionally, we evaluated our algorithm on a toxicity dataset as well. The number of groups were kept to 3 for all the datasets.

MNIST dataset

This dataset was directly downloaded from the MNIST webpage. The dataset consisted of 28×28 pixels images of handwritten digits. The images obtained was centralized to fit the inner 20×20 pixels of each image. The feature vector was obtained by simply concatenating all the pixels into a vector. The labels consisted of the digits from zero to nine. Ten binary tasks were created from the labels, such that each task classified one digit against the rest. There were a large number of samples for this dataset. We only used 1% of the dataset for training and the other 99% for testing. The experiment was repeated 100 times. All the algorithms were trained and tested on the same set of training and testing samples each time.

small-MNIST dataset

Small-MNIST is a preprocessed MNIST dataset obtained from (73). This is a classic handwritten digit recognition dataset. The images were preprocessed using PCA, and the dimensionality of the images were reduced to 64 as described in (73). Only 2,000 samples were provided for the ten tasks as discussed above.

USPS dataset

This is another digit recognition dataset (64). The preprocessing on these images was similar to smallMNIST. After PCA, only 87 features were kept (73). The ten tasks were created by binarizing the ten labels, and only 2000 total samples were provided.

Animal dataset

This dataset classifies the images of animals using their text attributes. The dataset is originally from (81). Along with the images, a SIFT bag of word descriptors were provided in the original dataset. PCA was used to reduce dimensionality to 202, and 2000 samples were picked. We obtained this pre-processed dataset from (73).

For the above datasets obtained from Kang et al, the dataset was randomly divided into 70% training and 30% testing. Each of the mentioned algorithms were trained on the training and tested on the remaining portion of the dataset. The entire experiment was repeated 100 times. All the algorithms were trained on the same sets of training and testing samples each time. The summary of the datasets can be obtained from table 4.2, and the performance on these datasets is summarized on Tables 4.3 and 4.4.

Toxicity dataset

The number of new chemicals that are being created each year is large, and it is becoming more and more difficult to obtain the full toxicological profile of each of these chemicals. One way to combat this problem is to be able to rule out some of these chemicals for traditional toxicity testing by the use of computational and in vitro methods. For this purpose, the environmental protection agency (EPA) provided a dataset that maps in vitro tests performed on different chemical with the toxicological effect of those chemicals on various endpoints on rats, mouse and rabbit. We picked 8 endpoints related to rat and mouse livers for this test. We divided the dataset into 70% training and 30% testing. The experiment was repeated 100 times. The results of toxicity dataset are also reported in Tables 4.3 and 4.4.

Since the tasks were built by binarizing a multi class problem, the datasets are heavily biased. For example, in the digit recognition datasets, the task of identifying each digit only has one tenth

Table 4.2: Dataset description

Dataset	# of tasks	# of features	# of samples
small-MNIST	10	87	2,000
USPS	10	64	2,000
Animal	20	202	2,000
MNIST	10	784	60,000
Toxicity	8	988	245

Table 4.3: Performance comparison of our algorithm, MTLC0, with other algorithms. The evaluation criteria reported below is the fraction of misclassified instances. The bold values represent the significantly best performances with p-values less than 0.01.

Algorithm	MTLF	Kang	MNEP	cCMTL	CASO	MTLC0
small-MNIST	**0.0354**	0.0414	0.9519	0.1292	0.1976	**0.0356**
USPS	0.0295	0.0395	0.8942	0.1276	0.1577	**0.0289**
Animal	0.3221	0.0541	1.0000	0.0886	0.0981	**0.0509**
MNIST	0.6736	0.2405	0.1124	0.1035	0.0649	**0.0410**
Toxicity	0.6143	**0.3586**	0.4633	0.4254	0.4123	0.4173

of positive samples and the rest are negative. Hence, we evaluate the performance by measuring the error, which is the fraction of misclassified samples, and the F1 score. F1 score is evaluated by

$$F1_{score} = \frac{2 \times prec \times recall}{prec + recall}$$

where,

$$prec = \frac{tp}{tp + fp}$$
$$recall = \frac{tp}{tp + fn}$$

Here, prec is the precision, tp is true positive, fp is false positive and fn is false negative. Sometimes, the classifiers classified all the classes as negative. Since, the dataset is biased, the error is still low in this case, however, the value of precision and hence the value of f1-score is not a number. In such cases, the f1-score used was zero. Also, if the algorithm failed to converge, the parameters became not a number. In these cases also, the error was 1 and the F1 score was zero. A

Table 4.4: Performance comparison of our algorithm, MTLC0, with other algorithms. The following table reports the F1-score on the tested algorithms. The bold values represent the significantly best performances with p-values less than 0.01.

Algorithm	MTLF	Kang	MNEP	cCMTL	CASO	MTLC0
small-MNIST	**0.8173**	0.7543	0.0392	0.6235	0.4282	0.8141
USPS	0.8512	0.7702	0.0917	0.6185	0.6178	**0.8519**
Animal	0.0025	0.0409	0	0.1263	0.1483	**0.0980**
MNIST	0.2783	0.3397	0.4195	0.5499	0.6991	**0.7961**
Toxicity	0.2690	0.3421	0.2640	0.4786	0.4620	**0.4427**

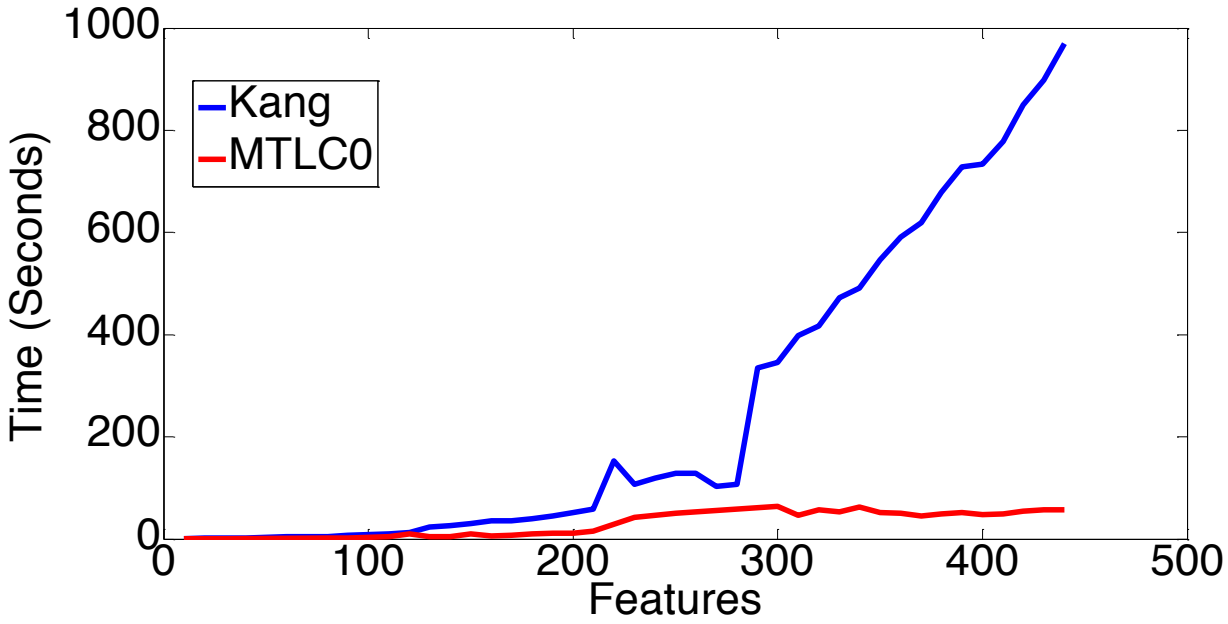
note that should be made here is that the high values of error and low values of f1-score in MNEP algorithm are actually true and are a result of the convergence problems of this algorithm on some of the datasets.

As can be seen from Tables 4.1, 4.3 and 4.4, our algorithm performs significantly better than other state of the art methods. The F1-score for smallMNIST data and the error on toxicity data are the only places where MTLC0 did not give the best performance. However, MTLC0 does show the best performance for the error and F1-score of small-MNIST and toxicity data respectively. For all the other datasets, our algorithm does perform the best results. Next, we show that the time taken by our algorithm is significantly lower than other algorithms.

4.3.3 Computational Complexity

The time requirement of MTLC0 is significantly lower than Kang et al's algorithm. For example, when we kept the number of features to a constant value of 100, and set the number of samples equal to 50, 100 and 150, the average time over 10 runs for MTLC0 algorithm was 2.8666, 2.7874 and 2.8229 seconds respectively. However, for Kang's algorithm, the average time for same number of features and samples was 8.4807, 8.3499 and 8.3634 seconds. Moreover, the complexity of our algorithm is linear with respect to number of tasks, samples and features. However, the

Figure 4.2: Average time required for MTLC0 and Kang's algorithm with varying number of features



complexity of Kang's algorithm is proportional to square of the number of features. To illustrate this difference, we run a simple experiment to show the difference of time behavior between our method and Kang's method with varying number of features. We increment the number of features by steps of 10 and keep the number of samples to be constant as 100, and measure the time for execution of each algorithm ten times. Figure 4.2 shows the average time taken to run these algorithms for varying number of features. All the above experiments are performed on the same computing facility. It is pretty clear that the rate of increase in time with the number of samples for Kang's method is much higher than our method .

4.4 Conclusion

In the current work, we present a new multitask feature learning algorithm for identification and training of groups of related tasks together. Present algorithm uses a modified spike and slab prior, which is a bayesian equivalent of L_0 norm for both grouping of tasks and feature selection. We

applied our algorithm on several real world data sets and showed that our algorithm performs significantly better than leading state of the art algorithms. We also showed that the time requirement for our algorithm is lower than other state of the art algorithms.

The next question that we need to answer is if we can model the task relationships in lifelong learning framework as well. The major challenge in lifelong learning is that we do not know which tasks are going to be faced by the learner during its lifetime. Therefore, the task relationship model needs to be capable of incremental learning. For this purpose, we propose learning functions to divide the task space or the model space. The tasks which fall into the same region are said to be in the same group and regularized together. Partitioning task space has not been explored earlier. However, this divide and rule strategy has been extensively used in the input space. Therefore, let us first review the existing literature which considers partitioning the input space. In the later chapters, we present our algorithms for lifelong learning.

Chapter 5

Literature Survey on Learning local models by Partitioning Input Space

There are a lot of algorithms for supervised learning that divide the input space into regions and build a local model for each region. Learning local models for each region is a powerful technique to learn complex functions using simple representations. The most popular of such algorithms are decision trees. However, there have been other algorithms that build more subtle boundaries for the local learners such as nearest neighbor classifiers and adaboost. In this chapter, let us go over the literature that deal with the divide and conquer strategies in machine learning history.

5.1 Decision Trees

The decision trees are one of the most popular supervised learning methods primarily used for classification. The great popularity of the use of decision trees comes from their simplistic design, ease of training, intuitiveness and scalability. Decision trees have been studied in multiple disciplines such as machine learning, data mining, statistics, decision theory etc (119). Many times multiple decision trees are bagged together to create an ensemble of decision trees as in random forest.

The most straight forward description of a decision tree algorithms is that they are a kind of clas-

sification method who have a structure of a tree consisting of nodes and leaves. At each node, an attribute is chosen and the training instances are partitioned according to the value of that attribute. At the leaves of the trees, all the instances in a particular leaf are assigned a single label. Decision trees can be univariate or multivariate based on the number of attributes used for splitting at each nodes.

5.1.1 Univariate Splitting Criteria

The splitting criteria of the decision trees are based on a number of different metrics but can be broadly classified into two main measures (108), (119) based on information theory and gini index.

5.1.1.1 Information Theory

Concepts and measures from information theory have been widely used in establishing the splitting criteria for decision trees (114), (116), (82), (72). According to information theory, entropy is defined as the measure of uncertainty about the source of information. It can take any value $x_i \; \forall i \in \{1,2...k\}$, and $P(x_i)$ is the probability of observing the value x_i. In terms of a decision tree, let T be a set of samples X, and the samples X may belong to a class $C_i \; \forall i \in \{1,2...k\}$. The probability of a sample to belong in class C_i is given by $P(C_i) = \frac{freq(C_i,X)}{|T|}$, where $freq(C_i,X)$ is the number of samples in class C_i, and $|T|$ is the number of samples in dataset T. Then, the entropy would be defined by

$$H(X) = -\sum_{i=1}^{k} P(C_i) log P(C_i) \tag{5.1}$$

The change in the entropy after a split is performed is called information gain and is a widely used metric for choosing the best attribute for splitting. Let a sample X consist of attributes $x_1, x_2 ... x_d$, and let the value of attribute x_a, for any value of a between 1 and d, belong to set v, then the

information gain is defined as

$$I = H(X) - \sum_{j \in v} \frac{|T_j|}{|T|} H(X_j) \tag{5.2}$$

Here, X_j are all the samples where the value of attribute x_a is j, and $|T_j|$ is the number of such samples. The attribute which maximizes the value of I is chosen for the split.

There are some variations of the information gain that have also been explored. The criteria mentioned above are biased towards attributes with larger domain. Therefore, normalized information gain is also proposed commonly known as gain ratio (115). Likelihood ratio chi squared criteria is used to measure the statistical significance of the information gain. Jun et al (72) proposed using the base of the logarithm as the number of successors that node will have.

5.1.1.2 Gini Index

Gini index is another popular metric used for deciding the input criteria (137), (Breiman et al.), (50). Using the same notation from section 5.1.1.1, the gini index for a given dataset is defined as

$$gini(X) = 1 - \sum_{i=1}^{k} P(C_i) \tag{5.3}$$

Here, $P(C_i)$ is the ratio of the frequency of class C_i occurring to the total number of samples. If an attribute, x_a is chosen for splitting, then the gini gain is given by

$$G = gini(X) - \sum_{j \in v} \frac{|T_j|}{|T|} gini(X_j) \tag{5.4}$$

The notation remains same as section 5.1.1.1. The attribute which minimizes the gini index is chosen as the attribute for splitting. Gini index has issues with handling large number of classes. In these cases, binary criteria such as twoing criteria is used (Breiman et al.). When the classification task are binary, both gini index and twoing criteria are the same. A similar measure is

Distinct Class based Splitting Measure (DCSM) (25) which creates a partition by giving weight to the number of distinct classes in a partition as well.

There are other measures for the splitting criteria as well. Kolmogorov-Smirnov (48), (120), (146), Orthogonality Criteria (44), misclassification rates and Liklihood Ratio Chi-Squared Statistics (8) are just to name a few. However, impurity measures such as information gain and gini index continue to be the most popular choice for univariate splitting criteria in decision trees.

5.1.2 Multivariate Splitting Criteria

The decision tree model had this major limitation that only one variable was chosen for splitting at each node. Thus, all the partitions were axis-parallel. Later, researchers started studying the combination of attributes to create the partition at each node. Although this makes the decision tree model more flexible, the increase in the complexity of training process is increased significantly as well.

There are multiple ways in which multivariate splits can be determined. One of the methods is to use feature construction method where attributes are added iteratively at each node. There are multiple approaches to add the new features. For example, linear discriminant function may be used (58), or projecting the data on first principal component (48), (49), or identification of promising direction to partition data by building multiple trees and utilizing the centroids of promising pairs of leaf nodes (67).

One popular algorithm in this area is CART-LC (Classification and Regression Trees with Linear Combinations) (Breiman et al.). For each nodes, the weights of the linear combination of the attributes are computed at each node using locally optimum values. The authors also consider backward deletion strategies to eliminate unnecessary features at each node and increase the interpretability.

OC1 is another popular algorithm which considers multivariate partitions at each node (101). It is very similar to CART-LC, except that the authors realize that CART-LC algorithm is too deterministic and are more likely to get stuck in local minima. They perturb their algorithm by using multiple restarts for the optimization problem and randomly perturb the hyperplanes generated for partitioning in random directions to escape the local minima.

There are multiple other strategies used for oblique trees. Many authors have considered using different metric than the impurity criteria for optimizing the splits such as maximizing separability of the subsets (126), or maximizing the margin between separating hyperplanes and the data (11). Other authors have considered evolutionary algorithms (77), (78) and (16) for finding the optimal decision tree.

All the decision trees described above are mainly for classification. As we observe, after partitioning the input space, decision trees assign a single label to each leaf. Thus, each local region in a decision tree is assigned a most simplistic model, which is a uniform label to all the samples in the leaf.

5.2 Regression Trees

Regression trees are an extension of decision tree for regression. Just like in classification trees, regression trees also assign a single value to each leaf node. Since the true outcome of the samples is continuous, the mean of all the samples that reach the leaf is the value that is assigned to the leaf usually. In most cases, the decision trees described above can be easily extended to form regression trees. The most popular regression tree which is used is CART (Breiman et al.), (137). Usually in regression trees, the node splitting criteria is minimization of mean squared error rather

than the impurity criteria described in decision trees. Although the recursive partitioning of space captures the structure of the data which other global linear regressor models fail to capture, the major limitations of these trees is the introduction of bias because of dividing the output space into bins.

5.3 Hybrid Trees

Hybrid trees are attempts to connect worlds of decision trees and global regression models. A hybrid tree is basically a decision tree in which another model such as perceptron is used at the leaves. Hybrid trees are better suited for regression problem than regression trees, however, these can be used for classification as well. These trees use local model for each region created in the true sense. Quinlan et al proposed these trees under the name model trees (117). There are multiple examples of hybrid trees in literature. For example, (144) uses local kernel model at each leaf. In (39), authors use nearest neighbor model with the single attribute decision tree. (152) and (65) also integrate nearest neighbor and trees for regression and classification purposes.

5.4 Local Models

There are other approaches which aim at building local models in the input region. These methods do not draw a clear partition line in the input space. However, these models do assign a local model to each sample depending on the position of the sample in the input space.

One example of these kinds of models are nearest neighbor (30; 32; 121). For each test case, the nearest neighbor finds the samples from the training data which lies closest to the test case, and assign the output value based on the selected training samples. Many times a local model is also assigned to each region (57; 75). As we can see here, the input region is not formally partitioned, however, very small regions are formed around each training samples which dictate the value assigned to any sample which fall in these regions.

The nearest neighbor approach uses distance to measure the nearest neighbors (53; 80). Localized kernel models use kernel functions such as radial basis function and find the nearest samples using the metric defined by the kernel function.

Another example of the local models is adaboost (47; 38). Adaboost actually builds a series of classifiers where each successive classifier weighs the misclassified samples from previous classifiers more heavily. Because of the multiple classifiers, local regions are formed around the training samples and the samples falling into these regions get the same class. In the earlier days of machine learning, researchers also looked at exemplar based classification where series of hyper-rectangles were created around samples to explicitly mark regions with same labels.

5.5 Global Formulations

All the methods mentioned above either do not explicitly create partition, or creating partition and local models are two separate processes. That is, first the ideal partitions are developed, and then local models are trained for each region. Recently, the researchers have explored global formulations of creating both the partitions and develop the local models as part of single formulation (104; 150). Thus, the partitions are created in a manner which provide maximum benefit to the performance of local models.

In our research, we use the global formulations to partition the model space instead of the input space. Our procedure is reported in the following chapters.

Chapter 6

Lifelong Multitask Learning using Local Partition Models

Lifelong multitask learning is a learning mechanism having the ability to entertain new tasks which may be encountered during the training process. Lifelong multitask learning has the potential of finding applications in many autonomous machines that have the capability to learn and predict any unforeseen tasks during its lifetime (41; 122). The examples of such applications include but are not limited to stock price prediction where new companies keep entering the market, online image annotation where the machine may encounter a different element in image which it has not seen before, or protein-chemical interactions where new chemicals are constantly being produced.

Lifelong multitask learning was studied initially by Eaton and Ruvolo (41). They also developed another algorithm for incorporating active task selection in life long learning framework (122; 123). In both of these methods, the authors learn a projection of the data on a lower dimensional space. The primary drawback of these approaches is that the entire basis matrix needs to be updated each time a new batch of samples are streamed.

We propose using the underlying task structural relationships to transfer knowledge among the

related tasks in a lifelong learning setting. The most popular method of insuring that knowledge is transferred only between the related tasks is to learn the task clusters, and the most popular method of learning these task clusters is the use of k-means clustering. However, learning task clusters in an online setting is difficult, because each time a new task comes in, it is assigned to one of the existing clusters even if the task is an outlier with respect to the existing tasks. Thus, the outlier task shares knowledge with the unrelated tasks. One way to overcome this limitation is to use hierarchical task relationship structure, where if a new outlier task arrives at a leaf node, the node has the flexibility to split without effecting any of the existing clusters.

If we know the underlying hierarchical structure of the task relationship model, each time we encounter a new task, we can utilize this structure to update only those parameters which are relevant to the nodes a given task belongs to. However, using task relationships in lifelong multitask learning is a very challenging problem. It is not possible to know the structural relationship between the tasks beforehand. Any unknown task can arrive at any point during the training process. The same reason implies that learning this structure of the tasks is also challenging. The task structure is dynamically changing and growing with each new oncoming task. With a dynamic structure, the traditional methods of modeling structure using a fixed positive semidefinite matrix is highly expensive. To learn the hierarchical structure in a lifelong setting, we propose learning functions in the task space based on which task relationships may be established. More specifically, we propose learning a series of linear partition functions which hierarchically partition the task space. Learning linear functions to partition the task space instead of group membership results in a convex formulation which is easy to solve. For each region thus created, we learn a local model in the task space. These local models allow the transfer of already learnt information to the new incoming task. For each batch of new incoming samples, only the models relevant to the region the given task falls into needs to be updated. Thus, using this framework, the number of linear models which need to be updated at a given time is as low as the depth of the tree structure which is usually logarithmic with respect to the number of clusters. Thus, this formulation both

saves time and improves efficiency.

In this paper, our major contribution is to propose a solution for finding multi-level partitioning functions to divide the task space in an online learning framework. We also present a convex optimization framework for learning local models in task space based on given partition functions. It is observed that the optimization function for learning local models in the task space has the same form as matrix factorization problem and using this observation we present an online solution to find the local models for each partition. The major advantage of learning this method is its applicability in lifelong multitask learning as this algorithm handles new upcoming tasks in an efficient manner. From this point onwards, we refer to our algorithm as “LiLeLopam" (LIfelong LEarning with LOcal PArtition Models).

The rest of the paper is organized as follows. First, in section 6.1, we discuss about the related work. Next we present the algorithmic design of our framework, followed by the empirical results. At the end, we conclude by presenting a summary statement.

6.1 Related Work

There are two major aspects of this paper, lifelong multitask learning and task relationship modeling. The first part of the study, lifelong multitask learning is a relatively new field and the related work for lifelong multitask learning is described in the introduction section. Another topic which is closely related to lifelong multitask learning is online multitask learning. The second aspect of the study is task relationship modeling. In this section, we will briefly go over some of the literature in online multitask learning and task relationship modeling which bears closest resemblance with our work.

There has been a number of recent publications that focus on online multitask learning (110; 92;

105). In (36; 37) , the authors present a framework for training multiple tasks in parallel using a shared regularization function in an online setting. In (157), the authors include feature selection with online multitask learning as well. (110) develops a kernel based method whereas (22) develops a perceptron based method for online multitask learning. The current online learning algorithms only consider the scenario when all the tasks are given from the starting point. However, this may not be always the case and the learning agent may be required to adapt itself to the newly encountered tasks.

Multitask learning with task relationship modeling has also been explored in multiple publications. The structural relationships in tasks has usually been explored in the form of clusters (73; 79; 99), hierarchical structures (52; 18; 35) or directed and undirected graphical structures (28; 46). The structures may be given or need to be learnt. These algorithms usually use a fixed sized structure representation to learn the task relationship model. Most commonly used choice of structure representation is defined by a fixed size matrix (7; 168). In, (125), the authors developed an algorithm for learning the task relationship modeling in an online setting. They also use fixed sized TxT dimensional matrix to represent the task relationships. The major drawback of these methods is that the structure representation of the tasks is not flexible to grow in size. Thus, it is difficult to introduce new tasks amidst the training process.

Incorporation of the task relationship model is especially important in lifelong learning setting because there is no previous knowledge about the kind of tasks that arrive during the training process. Training unrelated tasks together can degrade the performance further (79). In (79), the authors use a set of basis vectors and their coefficients to represent the task relationship in a sub dimensional space. The authors assumed that the coefficients are sparse, which results in the modeling of overlapping clusters in the task. This is a reasonably efficient way to model the task relationships which does not grow with the number of tasks, and thus, was utilized by (41) to model the task relationships in lifelong setting. However, a major inefficiency in this method is that all the basis

vectors need to be updated when new samples arrive.

In this paper, we aim to propose an efficient method for lifelong multitask learning which learns and utilizes the task relationship model during the training process. In order to achieve this, we learn a series of functions which hierarchically partition the task space and use the knowledge of these functions to regularize the related tasks together. Therefore, this method can also be viewed as an efficient method for multi-level task clustering. To the best of our knowledge, learning series of partition functions and region specific models in lifelong multitask setting has not been explored earlier and we are the first people to propose a solution for this problem.

6.2 Methodology

In a lifelong multitask learning setting, we assume that the lifelong learning system comes across a sequence of samples which may belong to any task t. The incoming task t can either belong to the pool of the tasks that the system has already seen, or can belong to a new task. Thus, the essential challenge in designing lifelong learning systems is that these systems have to be online in terms of both the samples and the tasks. In this section, we present an outline of our proposed methodology to learn the task relationships in a lifelong multitask learning setting.

6.2.1 Notation Used

We assume that the dataset Z is composed of $x_{i,t}$ and $y_{i,t}$, where $i = 1 \dots n_t$, n_t is the number of samples in task t, and $t = 1 \dots T$. Here, T is the total number of tasks. Since, in this paper, we are studying the case of lifelong multitask learning, the values of T and n_t are not known before hand and keeps on increasing with each set of arriving samples.

In the dataset Z, each entry of $x_{i,t}$ is a d dimensional vector of real numbers. The value $y_{i,t}$ is

a scalar which belongs to real number, $\Re$, in the case of regression problems and belongs to $\{0,1\}$ in the case of classification problems. The objective of this algorithm is to find a vector w_t such that $y_{i,t} = x_{i,t}w_t$ for linear regression problems and $y_{i,t} = \sigma(x_{i,t}w_t)$ for classification problems where $\sigma(z)$ represents the logistic function.

Throughout the text, we represent the matrices by capital letters, the vectors by bold faced lower case letters and scalar by normal font lower case letters. Thus, suppose a matrix is given by X. Its i^{th} row will be represented by $x_{i,:}$, j^{th} column will be represented by $x_{:,j}$, and the single value at i^{th} row and j^{th} column will be given by $x_{i,j}$.

6.2.2 Background

In this current study, we learn the partition of the task space and use these partitions to regulate the information transferred from previously learnt tasks to the new task. We find our algorithm bears close resemblance to an algorithm recently developed by Oiwa and Fujimaki (104). Therefore, before we describe our algorithm, we would like to give a brief overview of Partition-wise Linear Models. We also give a brief overview of regularized multitask learning before we proceed to the description of our model.

6.2.2.1 Partition-wise Linear Models

A recent study by Oiwa and Fujimaki, Partition-wise Linear Models (104), is related to partitioning the input space to learn localized linear regression model. Since our approach is similar to Oiwa's approach, but in the task space, we would like to briefly introduce Oiwa's algorithm here. From here onwards, let us refer to this algorithm as PWLM.

In PWLM, the authors aim to divide the input space into partitions and associate each partition

with a linear model. Thus, the output predicted for each sample is a combination of the predictions produced by the linear models associated with all the partitions the sample falls in. In this paper, the authors only assign one linear model to each partition line and that model applies to only one side of the partition. Let us consider an example in which the input sample $x = [x_1, x_2]$ lies in a two dimensional space and we need to map it to a output real number, y. Suppose, the first partition A_1 marks the region $x_1 > 1$. Thus, the linear model a_1 will be applied to all the samples for which $x_1 > 1$. Now, let us assume that the second partition a_2 is for $x_1 > 0$ and the third partition a_3 corresponds to $x_2 < 2$. Then the linear models applied to sample $x = [3,4]$ will be a_1 and a_2, as $x_1 = 3$ is greater than both 1 and 0, but $x_2 = 4$ is not less than 2. However, all three models will be applied to $x = [3,0]$ as in this sample, x_1 is greater than both 1 and 0, and x_2 is less than 2.

A mathematical formulation for the above problem is depicted as

$$\tilde{y_i} = g(x_i) = x_i a_0 + \sum_{p=1}^{P} f_p(x_i) x_i a_p \tag{6.1}$$

Here, $a_0, a_1 ... a_P$ are the linear models associated with each of the P partitions, $f_p(x_i)$ consists of values either 1 and 0 indicating which partitions are active for each sample x_i. Thus, in order to obtain the values of partition models a_0, a_1 ... a_P, the following equation needs to be solved

$$\min_{a, f(x)} \sum_{i=1}^{n} l(y_i, g(x_i)) \tag{6.2}$$

where l is any loss function and the total number of samples are n. For regression problems, squared loss function may be chosen. The authors also add sparsity constraint on each linear model associated with each partition and bound the maximum number of partitions a sample can fall in. Thus, the above model can be obtained by solving the following optimization problem.

$$\min_{a,f(x)} \sum_{i=1}^{n} l(y_i, g(x_i)) \; s.t. \sum_{p \in 1 \ldots P} I(a_p \neq 0) \leq \mu_p$$

$$and \; \|a_p\|_0 \leq \mu_0 \; \forall p \tag{6.3}$$

The authors assume that the partitions are given beforehand. For their experiment studies, the authors divide the input space into grids at the quartiles of the data across each dimension and get the values of $f_p(x)$ for their formulation. Then the authors approximate the above function with its closest convex approximate and present a solution for the above problem.

In the current study, our formulation may seem similar to the PWLM approach at first. However, we would like to mention that our algorithm is significantly different from PWLM because we are partitioning the task space as opposed to the input space as done in PWLM. We also do not use the grid formation approach for creating the partitions by using quantiles of the data in the task space because this is an online algorithm and the entire data is not available to us. Even though the convex formulation of our method may seem similar to PWLM once the partitions are given, we also cannot use the solution proposed in PWLM as we are interested in online learning implementation of the problem.

6.2.2.2 Regularized Multitask Learning

Multitask learning framework can be regularized in multiple ways. Here, we mainly use the method as described by Evgeniou and Pontil (43). In this paper, the authors assume that the weight vector w_t for each task t is composed of two parts w_0 and v_t. The weight vector w_0 is common for all the tasks, whereas, the vector v_t is different for each task. Thus,

$$w_t = w_0 + v_t \tag{6.4}$$

To find the values of w_0 and v_t, the following problem is solved

$$\min_{w_0,v_t} \sum_{t=1}^{T} \sum_{i=1}^{n} l(y_{i,t}, x_{i,t} w_t) + \lambda_1 \frac{1}{T} \sum_{t=1}^{T} \|v_t\|^2 + \lambda_2 \|w_0\|^2 \tag{6.5}$$

where l is the loss function used and λ_1 and λ_2 are regularization constant used. In this paper, the authors also show that the above problem is equivalent to solving for

$$\begin{aligned} \min_{w_0,v_t} \sum_{t=1}^{T} \sum_{i=1}^{n} l(y_{i,t}, x_{i,t} w_t) + \rho_1 \sum_{t=1}^{T} \|w_t\|^2 \\ + \rho_2 \sum_{t=1}^{T} \left\| w_t - \frac{1}{T} \sum_{s=1}^{T} \|w_s\|^2 \right\|^2 \end{aligned} \tag{6.6}$$

in the case of hinge loss. Therefore, (i) if we partition the task space into regions, and (ii) assume that the parameters associated with each task is equal to the sum of a local model associated with the region and a task specific model, we ascertain that the task parameters lies close to the mean of the task parameters in that region. Our goal in this paper is to learn the local models associated with each region in task space in a lifelong learning setting. With this note, we conclude the description of the background information and proceed to the description of the formulation relevant to the current paper.

6.2.3 Basic Approach

Our basic approach for modeling the task relationship model is to build a partition tree in the task space. The main objective of this partition tree in the task space would be to learn regions in which the tasks might benefit by transferring information learnt among each other. Then we use the developed partition tree to learn local linear models in the task space. More specifically, for each partition created, we learn two local models, one for each side of the partition. We assume that the task parameters are given by the sum of all the local models a task is associated with and a task

specific model.

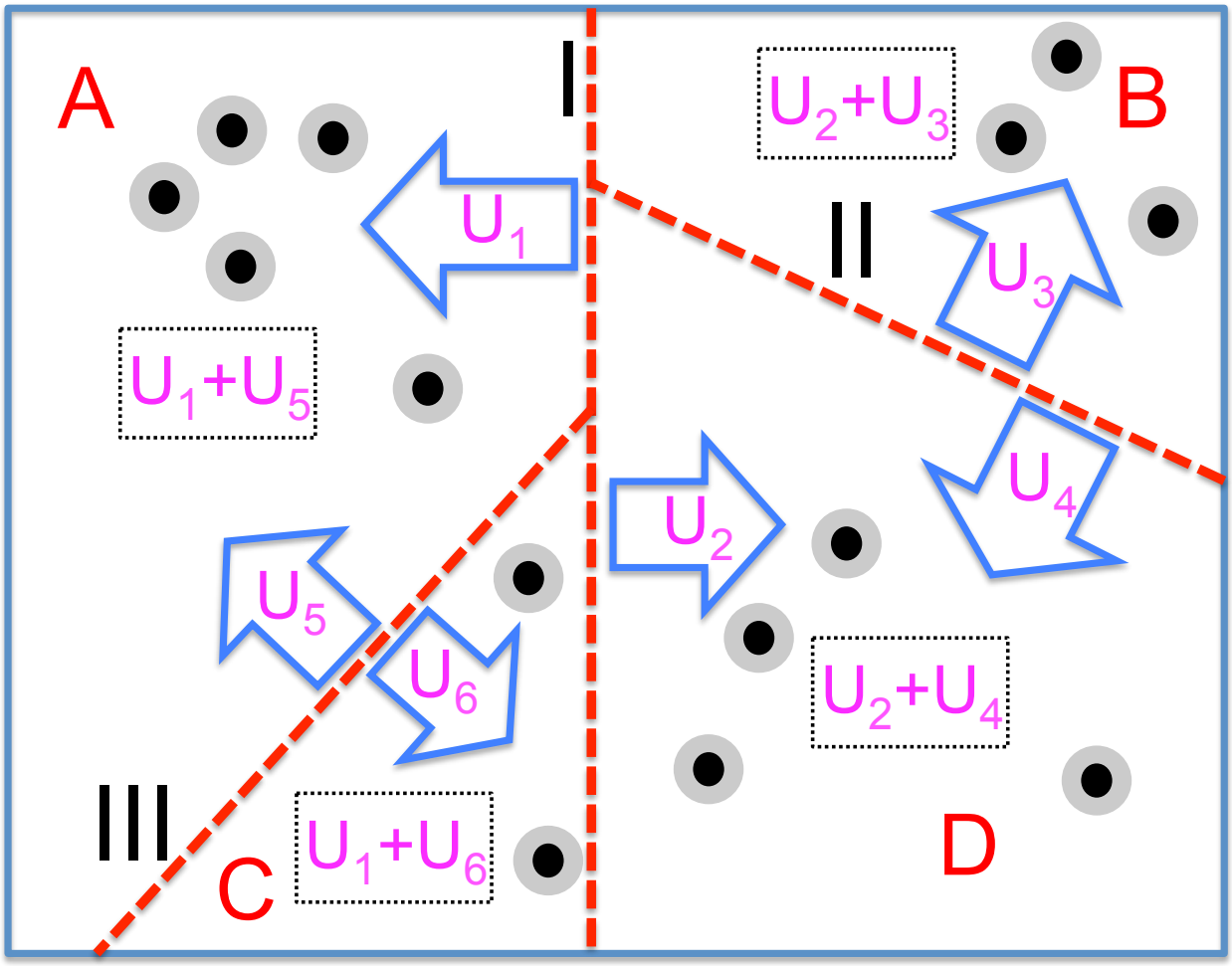
A cartoon of our problem formulation is represented in Figure 6.1. In the picture, each task is depicted by black dots with the gray circle around it representing the uncertainty associated with the location of each task. The partitions are denoted as dashed red lines. The roman numeral shows the partition number, and the capital alphabets A, B, C and D label each region. As can be seen by the picture, partition I is associated with two models U_1 and U_2. Model U_1 is the local model for the left hand side of the partition, while U_2 is for the right hand side of the partition. The two regions created by partition I are further divided by partitions II and III. The two models associated with partition II are U_3 and U_4, and with partition III are U_5 and U_6. Therefore, we obtain four regions A, B, C and D. Given this scenario, the model for any task in region in A will be given by $U_1 + U_5 + v_t$ as this task lies on left side of partition I and top of partition III. Similarly, the task model for any task in region D will be depicted as $U_2 + U_4 + v_t$. We now present a formal description of our problem.

In this paper, we mainly focus on lifelong multitask learning for both linear classification and regression tasks. Assuming a parametric linear model, let the vector w_t be the parameter vector associated with each task. Thus, we need to estimate the vector w_t, such that $y_t = x_{i,t} \cdot w_t + \varepsilon$ for any given task t. Here ε is the white noise. The estimate of each w_t can be obtained by minimizing the loss function plus the penalizing term.

$$w_t = \min_{w_t} \sum_{i=1}^{n} l(x_{t,i}, w_t, y_{t,i}) + ||w_t||^2 \tag{6.7}$$

Here, $l(x_{t,i}, w_t, y_{t,i})$ represents the loss function and can be chosen in accordance with the task at hand. For linear regression, the choice of the loss function is usually squared loss while for classification, it is a sigmoid function.

Figure 6.1: Task Partition Model. The partitions are depicted as dashed red lines. Each partition is associated with two local linear models, one for each side of the partition. The task model is given as the sum of all the local models associated with the task and the task specific model.



One way to transfer information among tasks in multitask learning problem is to regularize the task parameters together. However, when regularizing all the tasks together, we assume that all the tasks are related to each other. This assumption in lifelong learning scenario may not hold. For this reason, we propose partitioning the task space and learn the local models in each region. Let us initially assume that the partition rules in the task space are given to us beforehand. Later, we will present a way to divide the task space into regions by creating hierarchical partitioning.

Let the rules whether a task t belongs to a given partition p be given by function $f_p(\theta_t)$, where θ_t are parameters used to characterize task t by just the observed samples. Thus, f_p is a function which maps a real number vector of dimensionality d to a binary number which can take the value of either zero or one, $f_p : \Re^d \rightarrow \{0,1\}$. Then, let us assume that the task parameters for task t are given by

$$w_t = \sum_{p=0...P} f_p(\theta_t) u_p + v_t \tag{6.8}$$

Here, P are the maximum number of partitions into which the task space is divided, and u_p represents the linear model associated with the region p. v_t is the task parameters associated with specific task t. The value of $p = 0$ represents a global partition which applies to the entire task space. Thus, the value of $f_0(\theta_t)$ is 1 for all values of θ_t and u_0 is a global model which applies to all the tasks.

The log likelihood function for this multitask learning scenario is given by

$$
\begin{aligned}
L(y_t, g(x_t)) &= \sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} l(y_{i,t}, w_t x_{i,t}) \\
&= \sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} l\left(y_{i,t}, v_t x_{i,t} + \sum_{p=0\ldots P} f_p(\theta_t) u_p x_{i,t} \right)
\end{aligned} \tag{6.9}
$$

$l(y_{i,t}, w_t x_{i,t})$ is the loss function used. We further go ahead and consider that the local linear models are sparse. Thus, we would need to minimize

$$
\begin{aligned}
E = & \min_{v_t, u_p \ \forall t, p} \sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} l\left(y_{i,t}, v_t x_{i,t} + \sum_{p=0\ldots P} f_p(\theta_t) u_p x_{i,t} \right) \\
& s.t. \ \|v_t\|_0 \leq \mu \ and \ \sum_{d=1}^{D} \|u_{d,p}\|_0 \leq \nu \ \forall p
\end{aligned} \tag{6.10}
$$

Therefore, we consider a sparse model to be associated with each region and the task specific parameters are also sparse. The above model is highly non-convex, and thus, it would be very hard to find the global optimal solution for the above problem. Therefore, we consider the use of convex relaxation of the above problem and use L_1 norm and L_∞ norm instead of L_0 norm. Using convex relaxation of the above formulation and Lagrange multipliers, we get

$$
\begin{aligned}
E = & \min_{v_t, u_p \ \forall t, p} \sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} l\left(y_{i,t}, v_t x_{i,t} + \sum_{p=0\ldots P} f_p(\theta_t) u_p x_{i,t} \right) \\
& + \lambda_1 \sum_{t=1}^{T} \|v_t\|_1 + \lambda_2 \sum_{p=1}^{P} \sum_{d=1}^{D} \|u_{d,p}\|_\infty \\
= & \min_{v_t, u_p \ \forall t, p} L(y_t, g(x_t)) + \lambda_1 \sum_{t=1}^{T} \|v_t\|_1 \\
& + \lambda_2 \sum_{p=1}^{P} \sum_{d=1}^{D} \|u_{d,p}\|_\infty
\end{aligned} \tag{6.11}
$$

It is quite interesting to see that the above model becomes very similar to the PWLM model even though the PWLM model partitions the input space and build a local model for all the samples

that fall in a given region. Here, we are building a local model for all the tasks falling in the same region. However, there are two major problem for using the PWLM solution to solve the above problem. First, the solution of the above problem will depend on all the previous training data. Secondly, the solution provided by PWLM to the above problem is for batch processing in terms of both samples and tasks and cannot be used in the lifelong learning scenario.

To deal with the first problem, we use the similar formulation as used in (41). We approximate the equation 6.11 by replacing the term $L(y_t, g(x_t))$ by the second order taylor expansion of $\frac{1}{n_t}\sum_{i=1}^{n_t} l(y_{i,t}, g(x_{i,t};\theta))$ around $\theta = \theta_t$. The value of θ_t is obtained by minimizing single task learning problem without any regularization.
Thus, $\theta_t = \min_\theta \frac{1}{n_t}\sum_{i=1}^{n_t} l(y_{i,t}, g(x_{i,t};\theta))$. Therefore, equation 6.11 can be written as

$$\begin{aligned} E = \min_{v_t, u_p\ \forall t,p} & \sum_{t=1}^{T} \frac{1}{n_t} \| \theta_t - \left(\sum_{p=0\ldots P} f_p(\theta_t) u_p + v_t \right) \|_{D_t} \\ & + \lambda_1 \sum_{t=1}^{T} \|v_t\|_1 + \lambda_2 \sum_{p=1}^{P} \sum_{d=1}^{D} \|u_{d,p}\|_\infty \end{aligned} \tag{6.12}$$

Here,

$$D_t = \frac{1}{2} \nabla^2_{\theta,\theta} \frac{1}{n_t} \sum_{i=1}^{n_t} l(y_{i,t}, g(x_{i,t};\theta))|_{\theta=\theta_t}$$

$$\|V\|_A = V^T A V$$

The constant terms have been removed from the expansion term as they will not effect the minimization. Also, note that for now, we consider the values of $f_p(\theta_t)$ provided to us. Now, there are two problems ahead of us, (i) solving the above equations for given values of the partition function and (ii) developing a method to find the values of the partition function.

Let us first consider the problem of finding the solution for the above optimization equation. Suppose, we have a given matrix F consisting of values of the function $f_p(\theta_t)$. Thus, the value of

each element of F, $f_{i,j}$ is the value of $f_p(\theta_t)$ for i^{th} partition and j^{th} task. Also, let U be a matrix with each column consisting of the values of u_p. Let Θ be matrix consisting of the values of θ_t as each column and V be a matrix of v_t. In this case, the above equation can be written as

$$E = \|\Theta - UF - V\|_{D_t} + \lambda_1 \sum_{t=1}^{T} \|v_t\|_1 + \lambda_2 \sum_{p=1}^{P} \sum_{d=1}^{D} \|u_{d,p}\|_\infty \tag{6.13}$$

It is interesting to note that the above problem is same as matrix factorization problem where we need to estimate the basis vectors U and the error vector V for a given set of coefficients F. Therefore, we need to solve for columns of matrix U and the vectors v_t. Recently, Shen, Xu et al. (127) published a study on online solution to maximum norm problem. We base our solution for the above problem on this study.

Now, returning to our formulation of equation 6.12, we use alternating optimization method to solve for v_t and u_p $\forall p$. With each coming task t, we will first update the vector v_t, and then we update the vectors u_p. In order to find the value of vector v_t, we first need to obtain the value of θ_t and D_t. These values depend on whether the problem at hand is a regression problem or a classification problem. For regression problems, these values can be computed using

$$\begin{aligned} \theta_t &= (X_t^T X_t)^{-1} X_t^T y_t \\ D_t &= X_t^T X_t \end{aligned} \tag{6.14}$$

For logistic regression, there are many algorithms available to solve for θ_t. The value of D_t is given by

$$D_t = \sum_{i=1}^{n_t} \sigma(x_{i,t}\theta_t)(1-\sigma(x_{i,t}\theta_t))x_{i,t}^T x_{i,t} \tag{6.15}$$

Both θ_t and D_t can be updated in an online fashion if the samples arrive for an existing task. Having obtained these values, the value of v_t is updated using

$$v_t = \min_{v_t} \frac{1}{n_t} \left\| \theta_t - \left(\sum_{p=0...P} f_p(\theta_t) u_p + v_t \right) \right\|_{D_t} + \|v_t\|_1 \tag{6.16}$$

where all the u_p and $f_p(\theta_t)$ are fixed. Any optimization technique may be used for this step. Then, the values of matrix U are updated using block coordinate descent algorithm. For this, first step is to gather two accumulation matrices A and B as

$$\begin{aligned} A_k &= A_{k-1} + f_{:}(\theta_t) f_{:}(\theta_t)^T \\ B_k &= B_{k-1} + (\theta_t - v_t) f_{:}(\theta_t) \end{aligned} \tag{6.17}$$

where $f_{:}$ is a p dimensional vector consisting of the values of function $f_p(\theta_t)$ for each partition for the current task t, and k is the update number.

Let G be a matrix consisting of subgradient of $G = \frac{1}{2}\delta\|U\|_{1,\infty}$. The subgradient is computed according to (127). Then, the p^{th} column of U, u_p, is given by

$$u_p = u_p - \frac{1}{a_{p,p}}(Ua_p - b_p + \lambda_2 g_p) \tag{6.18}$$

It is worth noting here, that the vector $f_{:}$ will be sparse with the maximum number of non-zero elements equal to the depth of the hierarchical partition tree created. Thus, the updates that are needed to be made in matrix U will only be necessary for the columns for which $f_{:}$ is non-zero.

The summary of the algorithm is provided in Algorithm 2.

Algorithm 2 LileLopam: Lifelong Learning with Local Partition Models

Require: Parameters λ_1, λ_2 and p
  $U \leftarrow 0_{d,p}$, $A \leftarrow 0_{p,p}$, $B \leftarrow 0_{d,p}$
  while MoreDataArriving() **do**
    if isNewTask() **then**
      $t \leftarrow t+1$
    else
      $A_k = A_k - f_{:}(\theta_t) f_{:}(\theta_t)^T$
      $B_k = B_k - (\theta_t - v_t) f_{:}(\theta_t)$
    end if
    Update the value of θ_t and D using Equations 6.14 or 6.15
    $f_{:}(\theta_t)$ = UpdatePartitionTree()
    Update task specific model v_t using Equation 6.16
    Update A and B matrices using Equations 6.17
    Update Local Models for respective regions u_p using Equation 6.18
  end while
  return Partition Functions $f_p(\theta_t)$, Local Models for each region U, Task specific model v_t

Let us now present a solution using which the task space may be partitioned in an hierarchical way. We first present the process of how to form the partition function at a given node. Then, we show how we create the partition tree and obtain values of all the $f_p(\theta_t)$.

6.2.4 Creating Each Partition in Task Space

Here, we focus on the problem of partitioning a given region into two parts using a linear separation. Let us assume that the equation of the linear separation line is given as $f^T\theta$, where we need to estimate the linear coefficients f. We estimate the linear partition function such that the two groups that the tasks are divided into minimize the penalized log likelihood function. Therefore, in order to find the linear separator among the tasks, we minimize the following function

$$
\begin{aligned}
Q = \min_{f,\hat{\theta}} & \sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} l(\hat{\theta}_t, x_{i,t}, y_{i,t}) \\
& + \sum_{t=1}^{T} \left\| \sigma(f^T \hat{\theta}_t)\hat{\theta}_t - \frac{1}{n_R} \sum_{s=1}^{T} \sigma(f^T \hat{\theta}_s)\hat{\theta}_s \right\|_2 \\
& + \sum_{t=1}^{T} \left\| (1-\sigma(f^T \hat{\theta}_t))\hat{\theta}_t - \frac{1}{n_L} \sum_{s=1}^{T} (1-\sigma(f^T \hat{\theta}_s))\hat{\theta}_s \right\|_2
\end{aligned} \tag{6.19}
$$

Here, $\sigma(f^T \hat{\theta}_t)$ is the logistic function. which divides the tasks into two parts. Thus, the tasks are assigned a label of 0 or 1 and are regularized to be close to the mean of all the tasks bearing the same labels. Let us assume that the mean of tasks having the label 0 is given by M_0 and the mean of the tasks having the label 1 be M_1. Then, the above equation may be written as

$$
\begin{aligned}
Q = \min_{f,\hat{\theta}} & \sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} l(\hat{\theta}_t, x_{i,t}, y_{i,t}) + \sum_{t=1}^{T} \left\| \sigma(f^T \hat{\theta}_t)\hat{\theta}_t - M_0 \right\|_2 \\
& + \sum_{t=1}^{T} \left\| (1-\sigma(f^T \hat{\theta}_t))\hat{\theta}_t - M_1 \right\|_2
\end{aligned} \tag{6.20}
$$

Applying the same trick as in equation 6.12 and use taylor series expansion around θ_t, we get

$$
\begin{aligned}
Q = \min_{f,\hat{\theta}_t} & \sum_{t=1}^{T} \|\hat{\theta}_t - \theta_t\|_{D_t} + \sum_{t=1}^{T} \left\| \sigma(f^T \hat{\theta}_t)\hat{\theta}_t - M_0 \right\|_2 \\
& + \sum_{t=1}^{T} \left\| (1-\sigma(f^T \hat{\theta}_t))\hat{\theta}_t - M_1 \right\|_2
\end{aligned} \tag{6.21}
$$

We assume that the optimum value of $\hat{\theta}_t$ will be near θ_t for each task t. Then, the first term approaches the value of zero and we are left with the last two terms. We use the online gradient descent algorithm to find the optimum value of f. The values of M_0 and M_1 are updated accordingly. To first find the update functions for f, we need to find the gradient of equation 6.21 first.

The gradient of the above equation is given as

$$\begin{aligned}\nabla_f Q = &\sum_{t=1}^{T} 2\left((\sigma(f^T\theta_t)\theta_t - M_0)^T\theta_t\right)\theta_t \frac{exp(-f^T\theta_t)}{(1+exp(-f^T\theta_t)} \\ &-2\left((1-\sigma(f^T\theta_t))\theta_t - M_1)^T\theta_t\right)\theta_t \frac{exp(-f^T\theta_t)}{(1+exp(-f^T\theta_t)}\end{aligned} \tag{6.22}$$

Let, $\sigma(f^T\theta)$ be denoted by z. Then, the above equation can be written as

$$\begin{aligned}\nabla_f Q &= \sum_{t=1}^{T} 2(z\theta - M_0)\theta^T\theta z(1-z) \\ &\quad -2((1-z)\theta - M_1)\theta^T\theta z(1-z) \\ &= \sum_{t=1}^{T} 2((2z\theta_t - M_0 + M_1 - \theta_t)^T\theta_t)\theta_t z(1-z)\end{aligned} \tag{6.23}$$

Therefore, we update the linear function coefficients f at k^{th} step as

$$\begin{aligned}f &= f - \alpha\nabla_f Q \\ &= f - \alpha\sum_{t=1}^{T}((2z\theta_t - M_0 + M_1 - \theta_t)^T\theta_t)\theta_t z(1-z)\end{aligned} \tag{6.24}$$

where α is a predefined step function. Thus, for each new incoming task, the partition function may be updated as

$$f_k = f_{k-1} - \alpha((2z\theta_t - M_0 + M_1 - \theta_t)^T\theta_t)\theta_t z(1-z) \tag{6.25}$$

Now, the question remains what values should we use for M_0 and M_1. The values of M_0 and M_1 are also updated according to the updates of f_k.

In order to update the values of M_0 and M_1, we start with the value of M_0 have the value of first task θ_1 and M_1 have the value of second task M_2. The value of f can be initially set as $(M_0 - M_1)/2$. Then, after the arrival of each task, and making each update of f, the values of M_0 and M_1 can be updated accordingly. More specifically, if a new task θ_t gets assigned a label 0 at a given partition, then

$$M_0^{new} = \frac{n_0 M_0^{old} + \theta_t}{n_0 + 1} \tag{6.26}$$

where n_0 is the number of tasks assigned the label 0. Similarly, the values of M_1 may be updated. If a task changes labels in between the process, then that task may be removed from one mean and added to the other mean.

This algorithm is summarized in Algorithm 3. We now go on to explain how we create the partition tree using this partition scheme.

Algorithm 3 UpdateAPartition

if isUpdatedTaskinCurrentRegion() **then**
$f = f + \alpha((2z\theta_t^{old} - M_0 + M_1 - \theta_t^{old})^T \theta_t^{old})\theta_t^{old} z(1-z)$
end if
Update function f using Equation 6.25
Update the values of M_0 and M_1
Update task membership indicator vector $f_p(\theta_t)$

6.2.5 Creation of Partition Tree

The idea of a partition tree is creating an hierarchical way of dividing the task space. We hope with this process, we can hopefully identify the task hierarchical relationships and incorporate this relationship in our lifelong multitask learning model.

In order to create the partition tree, we create a series of partition functions. As the samples

related to the new tasks arrive, the first partition function f_1 is updated using equation 6.25. Using the updated function, it is decided whether a label 0 or a label 1 is assigned to the new task. At this point, we want to recall that our local partition model assumes only one model per partition. Therefore, based on function f_1, we create another function f_2 which is equal to $-f_1$. Therefore, the first local model will be common for all the tasks labeled 1 and the second local model will be common for all the tasks labeled 0.

Next, based on the label assigned to the new task, the next level function is updated. If the assigned label is 0, then say only function f_3 will need to be updated, else function f_5 will need to be updated and so on. We again assume that the function f_4 takes the value of $-f_3$ and f_6 takes the value of $-f_5$ for the same reason. We continue this process till a maximum of $2P$ partition functions is reached. We assume that the value of P is user defined. Thus, a maximum of $log_2(P)$ partition functions need to be updated for each incoming task.

It is possible that during the updates of partition function, some of the previous learnt tasks near the margin may change membership. However, there is no need to update the location of these tasks at this moment as the current local models are trained to include these tasks currently. Therefore, the region marked by these partite functions are allowed to have some crossovers in the vicinity of the the margin. When new samples belonging to these tasks will arrive, the task memberships and the local models for these tasks are updated accordingly.

Initially, when the number of tasks are very less than P, then it is not possible to create P partition functions. In that case, we just create the number of partitions that are possible. The summary of the algorithm is given in Algorithm 4.

Algorithm 4 UpdatePartitionTree

Require: Updated value of θ_t and old value of θ_t^{old} if it exists
  $l \leftarrow 1$
  while l < P **do**
    if NumberOfTasks(Region l) = 0 **then**
      $M_0^l \leftarrow \theta_t$
      return
    end if
    if NumberOfTasks(Region l) = 1 **then**
      $M_1^l \leftarrow \theta_t$
      $f_l \leftarrow (M_0 - M_1)/2$
      return
    end if
    UpdateAPartition(θ_t, f_l, θ_t^{old})
    $l \leftarrow$ NextChildNode
  end while

6.3 Experimental Study

In this paper, we develop a lifelong learning algorithm to learn the task structure as a series of functions and we assume a local model for tasks at each level of hierarchy. Here, we describe the performance of our algorithm and show that our algorithm achieves better results than other state of the art methods on many real world datasets.

In this paper, we compare our method to the following methods:

Batch MTL: We compare our model with model developed by Kumar et al (79). In this method, the authors learn the linear basis and the task specific coefficients. The framework of this problem is similar to matrix factorization problem except there is no need to compute the error coefficient. Since, the formulation of our problem has the same format as matrix factorization problem, we use this batch multitask learning method to compare our results. The code was provided by the authors, we would like to thank them.

ELLA: This is the lifelong multitask learning algorithm developed by Paul Ruvolo and Eric Eaton.

Paul Ruvolo and Eric Eaton assume that the tasks are related in the sub dimensional space and represent each task parameters as the sum of the product of a set of basis vectors and the coefficient of the basis vectors. The set of basis vectors are common across all tasks and the coefficients are unique for each task. Because the coefficients are sparse, they sometimes capture the overlapping cluster structure of the tasks. We thank the authors for providing us their code.

TREE: This is a baseline method. Here, we use online hierarchical clustering to learn the tree structure of the task and use the mean at each node as the common model of the region. Therefore, we skip the learning of local model in each region.

STL: We compare our model with single task learning. Here, we use bayesian online linear regression and classification method on each task to evaluate the error.

LileLopam: The method described in this paper.

6.3.1 Dataset Used

We test the performance of our model on a number of real world datasets as described below.

SMALL-MNIST dataset: This dataset is downloaded from Kang's webpage (73). It is a preprocessed version of the MNIST dataset for handwritten digit recognition. In this dataset, the principle components of the the images were computed and only the top 64 components were used to describe each image. Therefore, the dimensionality of each sample was reduced to 64. Only 2000 samples were provided. If we combine all the tasks, we get 20,000 samples. The labels consisted of values ranging from 0 to 9. Again, the binarization of the tasks was used to get 10 tasks, where each task represents the identification of each digit.

USPS dataset: This dataset is again downloaded from Kang's webpage (73) and is another digit

Table 6.1: Summary of each of the dataset including the dimensionality.

Dataset	# of tasks	# of features	# of samples
SMALL-MNIST	10	87	20,000
USPS	10	64	20,000
LANDMINE	29	9	14,820
STOCK Market	25	16	24,452

recognition dataset. The preprocessing used in this dataset is similar to the process used in SMALL-MNIST dataset. However, in this dataset, the authors used 87 features. Again, the ten tasks were created by assigning each task the job of identifying a single digit. There were 2000 samples in this dataset as well. Again, combining all the tasks, we get a total of 20,000 samples.

Stock Market dataset: This is a dataset which we compiled ourselves. We randomly pick 25 different companies. These companies are from various domains such as financial market, technology, software, oil and gas industries and pharmacy. We collect their weekly stock market prices from December 1994 to December 2014. The target was to predict the current stock market price based on the price history in the past sixteen weeks. All the information was collected from Yahoo's financial services website. There are twenty-five tasks in this dataset and 24,452 samples. The number of dimension is 16.

Landmine Detection: In this dataset, the goal is to detect if a land mine is present at a given location based on the radar images. The original dataset is from (156). However, we obtained the dataset from (41). It was included in the code we obtained from the authors. There were nine features which were extracted from the radar images using four moment-based features, three correlation-based features, one energy-ratio feature and one spatial variance feature. The problem is modeled as a binary classification problem with twenty nine tasks, nine features. There are a total of 14,820 samples. The summary of the dataset is provided in Table 6.1.

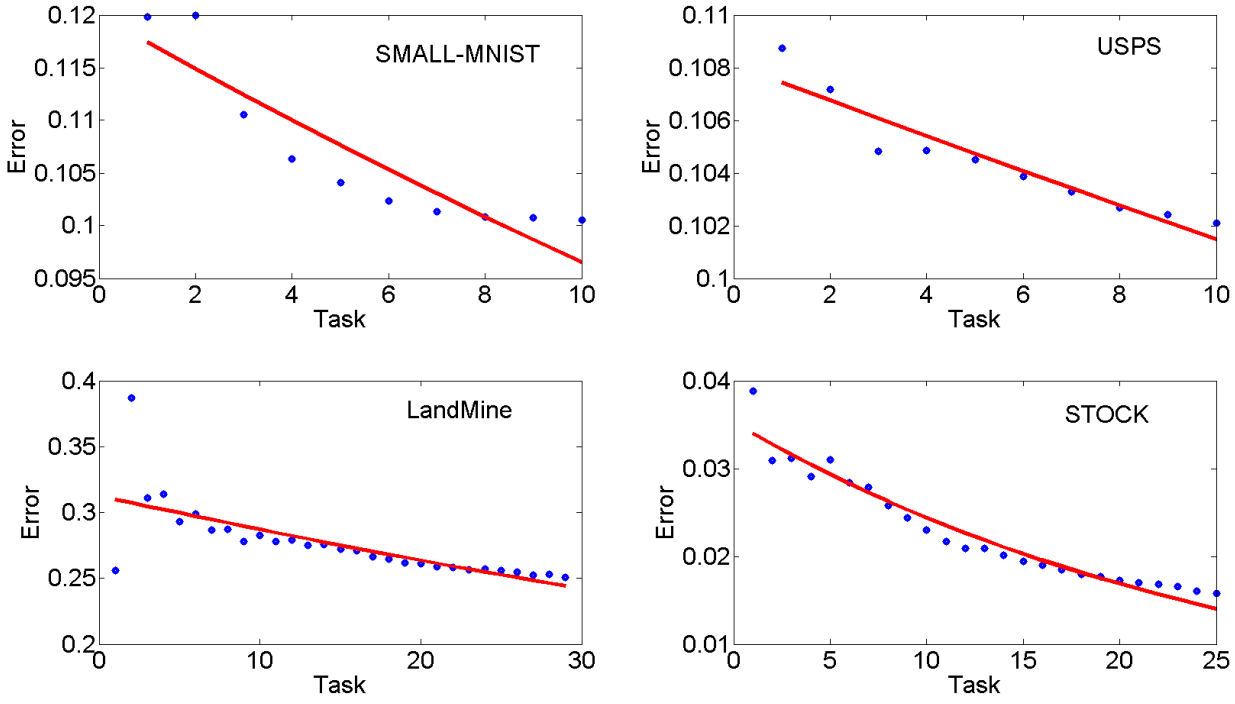
Table 6.2: The performance comparison of our method with other methods. The mean accuracies are reported here. LileLopam produced accuracies closest to the Batch MTL.

Dataset	Batch MTL	STL	TREE	ELLA	LileLopam
SMALL-MNIST	**0.9141**	0.5608	0.5608	0.8174	**0.9000**
	± 0.0031	± 0.0292	± 0.0297	± 0.0651	± 0.001
USPS	**0.9060**	0.4976	0.4999	0.7881	**0.8656**
	± 0.0038	± 0.0190	± 0.0207	± 0.0906	± 0.0098
LANDMINE	**0.8107**	0.5535	0.5658	0.6515	**0.7739**
	± 0.0525	± 0.0173	± 0.0181	± 0.0313	± 0.0321
STOCK Market	**0.9418**	0.9356	0.9372	0.9399	**0.9410**
	± 0.0018	± 0.0019	± 0.0018	± 0.0022	± 0.0018

Table 6.3: The average time taken in seconds to train plus predict from Batch MTL model and the speed up achieved for other models when compared with Batch MTL along with their standard deviations.

Dataset	**Batch MTL** (sec)	**ELLA** (Speedup)	**TREE** (Speedup)	**LilePopam** (Speedup)
SMALL-MNIST	108.8608 ± 4.7735	62.75 ± 11.32	16.71 ± 0.81	**128.23 ± 10.18**
USPS	98.4760 ± 2.0013	82.28 ± 16.04	58.98 ± 3.79	**193.55 ± 15.97**
LANDMINE	74.5201 ± 0.8796	178.40 ± 15.78	27.29 ± 5.21	**245.63 ± 29.48**
STOCK Market	22.3230 ± 10.5685	96.26 ± 47.43	0.1591 ± 0.0766	**246.18 ± 117.73**

Figure 6.2: The decrease in the overall error as a function of position in task sequence. The red line indicates best exponential fitting curve.



6.3.2 Model Evaluation Procedure

For each of the tested models we use five fold cross validation to find the optimum parameters for each of the datasets. If there are a multiple parameters which need to be provided, we use a greedy approach to find the optimum parameters. We fix all the parameters and change the value of one parameter at a time and find the value of that specific parameter which gives the lowest cross validation error. Having found the optimum parameters, we fix these parameters for the specific model and dataset tuple for the remaining portion of the experiment study.

For Batch MTL and ELLA, we choose the value of the number of basis from a pool of {2, 4, 6, 8, 10}. The value of λ and μ were chosen from {exp(-12), exp(-8), exp(-4), exp(0), exp(4)}. We picked these pool of parameters so that they are around the default values provided by the authors in their code. The maximum number of iterations used in Batch MTL was 100. In ELLA, there were two additional number of parameters to be optimized, the ridge term for single task learner and μ_1, the L_2 regularization penalty for single task specific component. The value of ridge term were also chosen from {exp(-12), exp(-8), exp(-4), exp(0), exp(4)}. The value of infinity was added to the pool for selecting μ_1 according to the author's guidelines.

In the TREE method, bayesian linear regression and classification methods were used as single class learner as described in (14). The single task learners were clustered hierarchically and the mean of all the tasks at the ancestors node were applied as the prior for each of the tasks. There were three parameters which needed to be optimized, the variance of noise in the data α, the prior variance term β and the decay term which is multiplied to the prior as we move further up in the hierarchical tree. The values of α and β were chosen from {0.001, 0.01, 0.1 0, 1, 10, 100}, and the value of decay factor was chosen from {1, 2, 5, 10, 20}. The same values of α and β were used for online single task learning (STL).

In the current method, LileLopam, three parameters needed to be optimized, number of parti-

tion, λ_1 and λ_2. The number of partitions was picked from $\{3, 5, 7, 9, 10, 12\}$, and the values of λ_1 and λ_2 were chosen from $\{0.001, 0.01, 0.1, 0, 0.1, 10, 100\}$.

Using the parameters obtained from the above method, we continue to test the performance of the models. For testing the performance, we randomly select around fifty percent of the samples from each task from each dataset for training. Thus, 100 samples were randomly from each task from stock data, 1000 samples from each task for SMALL-MNIST and USPS data, and 300 samples from each task were used from the landline data for training. The remaining samples were used for testing. We randomly sample the dataset 100 times and train each of the model. The data division into train and test was consistent across all the models. The order of the tasks were also randomized each time. The values reported are the average accuracies. For classification tasks, the accuracy is obtained using the fraction of the correct predictions made. For regression, we define the accuracy as $1 - nrmse$, where *nrmse* is the normalized root mean squared error. Thus, the higher number represents better performance for all the models. We also report the standard deviation of the accuracies. The summary of the results is provided in Table 6.2. It can be seen from Table 6.2 that our method performs better than ELLA and other online methods and has the performance closest to the batch multitask learning algorithm.

We also measure the total time needed to train and predict from each of the models for each of the 100 splits of the data. These experiments were conducted on the same Intel(R) Xeon(TM) 3.2 GHz Linux Machines. The average time required is also reported with the standard deviation. We do not compare time with online STL as this method does not have a good performance. The summary of the time taken are reported in Table 6.3. As can be observed in Table 6.3, the current method is faster than ELLA and achieves a speed of orders of magnitude over the batch MTL method.

In this algorithm, we do not update the task member function until new samples arrive. To show

that this does not result in negative transfer of information, we randomize the tasks 100 times and randomly separate the data into training and testing. We measure the overall error of the tasks introduced on the test data after each new task is introduced to the model. We plot these errors as function of the position of the tasks. As can be seen in Figure 6.2, the errors decrease after introduction of each new task.

6.4 Conclusion

In this paper, we proposed an algorithm to learn the task structure relationships using functions rather than fixed sized matrices. The current algorithm learns the partition functions for dividing the task space into hierarchical partitioning using linear functions. These partitions were used to learn local models for each region along with the task specific models. We formulated the problem of learning local models with given partitions to have the same mathematical formation as matrix factorization problem, and devise an online solution for the problem. It was observed, that the current algorithm has a better performance and speed than the current state of the art lifelong multitask learning methods. In the next algorithms, we go a step further and propose that the task relationship modeling part of lifelong learning can be formulated as a supervised learning problem.

Chapter 7

Learning Task Grouping using Supervised Task Space Partitioning in Lifelong Multitask Learning

Lifelong multitask learning is a multitask learning framework where a learning agent transfers the knowledge from the already learnt tasks to the new tasks which it encounters for learning during its lifetime. Lifelong multitask learning can be viewed as a subset of online multitask learning problem where the framework is online in terms of the tasks. The inspiration for lifelong multitask learning comes from the natural process used by human beings for learning (142), as they have a natural tendency to transfer the knowledge from previously learnt tasks to new tasks. The lifelong multitask learning setting is of great utility for many real world problems where one needs to make predictions related to new tasks regularly. Examples of such applications may arise in online image annotation where new images are constantly being added by the users and new objects that need to be identified are repeatedly added to the list, or in the autonomous robots sent on exploratory mission in outer space or under water to explore and identify a new surrounding.

Lifelong multitask learning has been primarily studied by Eaton et al (41), where the authors

develop an algorithm which can handle new tasks as they are faced by the learning agent during its lifetime of training. The authors assume that all the tasks are related in a sub-dimensional space which can be represented by a set of basis vectors. The authors aim to learn these basis vectors and the coefficients of these basis vectors for each task. This work has been extended by the authors in multiple ways as in (3; 124; 122; 123). These methods only suffer from one inefficiency, and that is entire basis vector needs to updated when a new task arrives. Also, this method cannot do feature selection while learning the tasks.

In this paper, we wish to focus on learning the structure of the tasks in lifelong multitask learning along with feature selection. Learning the task structure is especially hard in lifelong learning setting because there is no previous knowledge of the nature of new tasks that may be arriving. Thus, the task learning mechanism needs to be dynamic and able to evolve with changing nature of the task sets. For this purpose, we propose learning partition functions rather than task structure matrix for learning the task relationships. In this paper, we learn the task clusters by imposing a series of partitions on the task space in a supervised manner. We also assume that the similar tasks depend on the similar features. Therefore, we incorporate feature selection in our model as well.

Our main contribution in this paper is to present a global formulation for learning both partition models in task space and the task models using novel supervised learning formulation. This formulation partitions the task space such that the similar tasks remain in the same region using supervised learning and enforces similar tasks to depend on similar features. Learning both the task parameters and relationships is done in a supervised manner. We present the solution of this formulation using dual averaging technique for regularized stochastic gradient methods to solve for the sparsity constraints.

7.1 Related Work

This paper is essentially a framework for multitask learning in a lifelong learning framework. The core idea of the present approach is to learn both the task relationships and task models using supervised learning in a lifelong learning framework.

The literature related to lifelong learning is very limited. The idea of lifelong learning has been known since 1996 when Thrun (142; 143) introduced the concept first in the machine learning community. Since then multiple authors have contributed to lifelong multitask learning in supervised setting (133; 132; 136). These papers are about methods to incorporate previously learnt neural network so that the knowledge learnt may be used for future tasks and do not deal with identifying the task relationships. Recently, Eaton et al (41) published a study on lifelong multitask learning which assumes that the tasks are related in a sub dimensional latent space. This study is based on (79). The authors learn the basis of the latent space in which the tasks are related along with the coefficients of the basis for each tasks. Since the authors assume sparse coefficients, an implicit task relationship is assumed in their algorithm, but the explicit task relationships may not be identified using this algorithm. Even though learning task relationships is a new concept in lifelong learning, it has been much explored in multitask learning algorithms.

In the traditional multitask learning algorithms, where all the tasks were initially assumed to be related to each other (43; 9; 4; 5), researchers observed that training unrelated tasks together may lead to negative transfer of information, which results in degradation of generalization error (73; 79). To avoid this problem, researchers investigated learning task relationships such that the information is only transferred among the related tasks. The most common approach for solving this problem is to learn a positive semi-definite matrix to represent the task relationships (167; 45). This positive semidefinite matrix is often the task covariance matrix or task kernel matrix (167), or Laplacian matrix or indicator matrix mapping each task to other related tasks (45). Although task relationships are learnt using these fixed sized matrices, there are inherent problems in extending

this method to lifelong learning. When a new task arrives, the knowledge learnt regarding existing task relationships is not directly transferrable to the new task. Also, each time a new task arrives, a new row and column needs to be added to task relationship matrix which is computationally expensive. Another method for learning task relationships is to use k nearest neighbors to identify the task clusters (68; 171; 170). These methods use an unsupervised step to learn the task relationships. Some authors assume that the tasks are related in sub dimensional latent space, and learn the relationships in that latent space (79; 107; 54). This method does not learn the implicit task relationships in the physical space. Kang et al learn an indicator matrix whose element in i^{th} row and j^{th} column takes the value 1 if the task in row i belongs to the group in column j otherwise 0. This algorithm does an explicit task relationship modeling, and does feature selection as well while learning the task parameters and thus bears closest resemblance to our algorithm. However, the information learnt regarding task relationships is non-transferable to the new tasks. All of the above algorithms learn the task relationship using an unsupervised step.

In this paper, we propose using supervised learning to learn explicit task relationship modeling. More specifically, we learn a global formulation for learning partition functions in a supervised fashion to group the tasks together and then learn the task models. Learning functions in the task space which divide the tasks into clusters are easier to learn in terms of optimization, and they provide an easy transfer of information from previously learnt task to the new task. Using a global formulation to learn the task clusters and task model ensures maximum coupling between the two learning assignments ensuring higher accuracy.

7.2 Methodology

In this paper, we aim to learn a model for lifelong multitask learning scenario which learns the cluster of tasks as well as does feature selection. Before we give a description of our model, we would first like to describe the notation used, followed by a brief overview of our work. Proceeding

which, we provide the details of our algorithm.

7.2.1 Notation Used

Throughout this text, we represent the matrices by capital letter, the vectors by bold faced lowercase letters and scalar values by lowercase letters. Given a matrix X, its j^{th} column is represented by $x_{*,j}$. For the same matrix X, the element at the i^{th} row and j^{th} column is $x_{i,j}$.

The dataset being provided is represented by D, which consists of the input features $x_{i,t}$ and the output label $y_{i,t}$, where i ranges from $1...N_t$ and t ranges from $1...T$. N_t represents the number of samples in task t and T represents the total number of tasks. In a lifelong multitask setting, the values of N_t and T are undetermined.

The given dataset D consists of 2-tuple elements, where each entry $x_{i,t}$ belongs to a d dimensional real valued vector, $x_{i,t} \in \mathbb{R}^d$. The values of outputs depend on the tasks. For regression problems, the values of $y_{i,t}$ is a real number, $y_{i,t} \in \mathbb{R}$. For classification problems, $y_{i,t}$ is either 1 or 0, $y_{i,t} \in \{0,1\}$. We assume linear models for prediction throughout. Thus, the objective of the algorithm would be to learn the vector w_t, where $\hat{y}_{i,t} = x_{i,t} w_t$ for linear regression and $\hat{y}_{i,t} = \sigma(x_{i,t} w_t)$ for classification problem. σ represents any monotonic function.

Given the above framework, the average empirical risk for all the tasks in this case may be given as

$$\begin{aligned} R &= \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i=1}^{N_t} l(w_t, x_{i,t}, y_{i,t}) \\ &= \frac{1}{T}\sum_{t=1}^{T} l(w_t, D) \end{aligned} \tag{7.1}$$

Here, l is the loss function used. Usually, l belongs to the sigmoid family of functions for classification problems and squared loss function for regression problems. If we minimize the above equation to solve for w_t, then the tasks are not regularized and the objective is same as learning each task separately. Therefore, for multitask learning problems, a regularization term is added to the empirical risk to ensure that the tasks lie close together.

$$R = \frac{1}{T}\sum_{t=1}^{T} l(w_t, D) + \lambda F(W) \tag{7.2}$$

where λ is a regularization constant, and F is a positive monotonic function of W. The common choices of F are L_1 norm, L_2 norm and trace norm.

7.2.2 Supervised Partitioning of the Task Space

In most multitask learning formulations, all the tasks are regularized together under the assumption that they are related similarly. There are multiple applications where all the tasks are not related to each other and it is beneficial to identify the related tasks. The existing multitask learning approaches alternate between unsupervised step and supervised step to learn the task relationships and task models. We introduce a new method to learn both the task relationships and task models in a supervised manner which may also be used in lifelong multitask setting.

In this paper, we propose learning partition functions to divide the task space into regions. As a result, the tasks in the same region are effectively clustered together. There are multiple benefits of using partition functions instead of indicator matrices to learn task relationships. Functions are easier to update in an online setting as opposed to indicator matrices. Therefore, learning functions results in more adaptive algorithm to lifelong learning setting. Second benefit of using function is that it is easier to transfer knowledge of previously learnt tasks to the new incoming tasks. The learnt function can easily provide rules for clustering the new incoming task. Third benefit lies in

the fact that learning a discrete valued binary matrix using optimization techniques is much harder than learning a real valued continuous function. Therefore, we present this novel idea of learning partition functions in a supervised manner to cluster the task.

Let us first describe our algorithm for including a single partition function, and then we later extend the idea for creating multiple partition functions for creating r groups of tasks. Let us assume a function $g(X,y)$ which divides the task space or the model space into two regions, region 1 and region 2. Also, we assume that the task parameters, w_t can be given as either u_1+v_t or u_2+v_t depending on if the given task belongs to the region 1 or 2. Here, u_1 is the common model for all the tasks which lie in region 1, and u_2 is the model common for all the tasks in region 2. v_t are task specific models.

In this case, the empirical loss function can be written as

$$\begin{aligned} R = \min \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}(g(\theta_t(X_t,y_t))=0)\frac{1}{N_t}\sum_{i=1}^{N_t} l(u_1,v_t,x_{i,t},y_{i,t}) \\ + \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}(g(\theta_t(X_t,y_t))=1)\frac{1}{N_t}\sum_{i=1}^{N_t} l(u_2,v_t,x_{i,t},y_{i,t}) \end{aligned} \tag{7.3}$$

Here, $\mathbb{I}$ is the indicator function and θ_t is a function that maps the input data X_t to the output y_t for a given task t, and thus, can be seen as the representative of the task t. Henceforth, we will represent $g(\theta_t(X_t,y_t))$ as simply g. The function $l(u_1,v_t,x_{i,t},y_{i,t})$ is the loss function and measures the mean squared error between $x_{i,t}(u_1+v_t)$ and $y_{i,t}$. For classification problems, the loss function may belong to either zero-one loss function or sigmoidal family of functions.

Also, let us assume

$$l_t^1 = \frac{1}{N_t}\sum_{i=1}^{N_t} l(u_1, v_t, x_{i,t}, y_{i,t})$$

and

$$l_t^2 = \frac{1}{N_t}\sum_{i=1}^{N_t} l(u_2, v_t, x_{i,t}, y_{i,t})$$

Thus, the empirical risk described in equation 7.3 is

$$R = \min \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}(g=0) l_t^1 + \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}(g=1) l_t^2 \tag{7.4}$$

Now, let us consider the case where there are more than one partitions of the task space. Since, each partition function divides the task space into two regions, we adapt the framework of using a series of partition functions which recursively partition the task space hierarchically. In other words, we establish a tree structure in the task space with a partition function g^k at each node k of the tree. The function g^k divides the region at that node of the task space into two more regions. There is no partition function at the leaf nodes. Thus, each leaf node is associated with a region of the task space which is not partitioned any further. For each leaf node r, there exists a common task specific model u_r which is common for all the tasks that belong in the region associated with leaf node r. As mentioned before, there also exists task specific model v_t, thus, the task model is given by $w_t = u_r + v_t$, assuming task t belongs to region or leaf node r.

Let the left child of k^{th} node of the partition tree be k_L and right node be k_R. For any node k, the value of empirical loss for a task t would then be

$$L_t^k = \begin{cases} \mathbb{I}(g_k = 0)L_t^{k_L} + \mathbb{I}(g_k = 1)L_t^{k_R} & \text{if } k \notin \text{leaf node} \\ l_t^k & \text{if } k \in \text{leaf node} \end{cases} \tag{7.5}$$

This formulation of the loss function is recursive. Thus, the loss function of the entire tree may be given by the loss function of the root node. The loss function of the root node is the sum of the total loss of its left child and right child. The left child loss and right child loss may further be broken down in the same way till we reach the leaf node. Thus, the loss function of the root node is same as the loss functions of the sum of leaf nodes. Therefore, if the root node is 1, the cumulative empirical risk is

$$R = \frac{1}{T}\sum_{t=1}^{T} L_t^1 \tag{7.6}$$

Additionally, we also impose the models to be sparse. That is, the common model in region r, u_r should only select some features relevant to all the tasks in the region. Similarly, the task specific models, v_t are assumed to be sparse too. Sparsity assumption is especially useful for tasks with large dimensionality. The empirical loss function is regularized using L_1 norm on regional and task specific models. Thus, the empirical loss function may be written as

$$R = \frac{1}{T}\sum_{t=1}^{T} L_t^1 + \lambda_1 \frac{1}{T}\sum_{t=1}^{T} \|v_t\|_1 + \lambda_2 \sum_{r=1}^{p_l} \|u_r\|_1 \tag{7.7}$$

assuming there are p_l leaf nodes. We need to solve for the values of u_r, v_t and g_k. Henceforth, we will refer to all g_k as partition functions, u_r as region specific models and v_t as task specific model. We use alternative minimization to solve for each of the values. First, let us discuss how we can solve for a single partition function. Later, we will show how the same framework is extended

for multiple partition functions.

7.2.2.1 Finding Single Partition Function

To find the solution for g for a single partition case, let us assume that the values of u_1, u_2 and v_t is fixed. Then, the risk function is

$$R = \min \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}(g = 0) l_t^1 + \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}(g = 1) l_t^2 \tag{7.8}$$

The regularization terms need not be included as the values of u_1, u_2 and v_t are fixed. For the same reason, the values of l_t^1 and l_t^2 can be easily computed.

Theorem 1. *For a fixed values of vectors u_1 and u_2, the solution for finding the optimal partition function g which minimizes Equation 7.8 is same as finding the optimal classifier which minimizes the following empirical loss*

$$\frac{1}{T} \sum_{t=1}^{T} |l_t^1 - l_t^2| \mathbb{I}(g = (l_t^1 < l_t^2)) \tag{7.9}$$

Proof. Let us define a set S consisting of all the indices t, where $l_t^1 < l_t^2$. Also let us define operator $>$, such that for $a > b$, the result is 1 if a is greater than b and 0 otherwise. We can rewrite equation 7.8 as.

$$
\begin{aligned}
R =& \frac{1}{T}\Big[\sum_{t\in S}\mathbb{I}(g=0)l_t^1 + \sum_{t\notin S}\mathbb{I}(g=0)l_t^1 \\
&+ \sum_{t\in S}\mathbb{I}(g=1)l_t^2 + \sum_{t\notin S}\mathbb{I}(g=1)l_t^2\Big] \\
=& \frac{1}{T}\Big[\sum_{t\in S}\mathbb{I}(g=(l_t^1>l_t^2))l_t^1 + \sum_{t\notin S}\mathbb{I}(g=(l_t^1<l_t^2))l_t^1 \\
&+ \sum_{t\in S}\mathbb{I}(g=(l_t^1<l_t^2))l_t^2 + \sum_{t\notin S}\mathbb{I}(g=(l_t^1>l_t^2))l_t^2\Big]
\end{aligned}
$$

Since, for all $t \in S$, the value of $l_t^1 > l_t^2$ is 0. Similar argument holds for other terms. Rearranging

terms,

$$
\begin{aligned}
R =& \frac{1}{T}\Big[\sum_{t\in S}\mathbb{I}(g=(l_t^1>l_t^2))l_t^1+\sum_{t\notin S}\mathbb{I}(g=(l_t^1>l_t^2))l_t^2 \\
&+\sum_{t\in S}\mathbb{I}(g=(l_t^1<l_t^2))l_t^2+\sum_{t\notin S}\mathbb{I}(g=(l_t^1<l_t^2))l_t^1\Big] \\
=& \frac{1}{T}\Big[\sum_{t=1}^{T}\mathbb{I}(g=(l_t^1>l_t^2))\min(l_t^1,l_t^2) \\
&+\sum_{t=1}^{T}\mathbb{I}(g=(l_t^1<l_t^2))\max(l_t^1,l_t^2)\Big] \\
=& \frac{1}{T}\Big[\sum_{t=1}^{T}\mathbb{I}(g\neq(l_t^1<l_t^2))\min(l_t^1,l_t^2) \\
&+\sum_{t=1}^{T}\mathbb{I}(g=(l_t^1<l_t^2))\max(l_t^1,l_t^2)\Big] \\
=& \frac{1}{T}\Big[\sum_{t=1}^{T}\mathbb{I}(g=(l_t^1<l_t^2))\max(l_t^1,l_t^2) \\
&+\sum_{t=1}^{T}\left[1-\mathbb{I}(g=(l_t^1<l_t^2))\right]\min(l_t^1,l_t^2)\Big] \\
=& \frac{1}{T}\Big[\sum_{t=1}^{T}\mathbb{I}(g=(l_t^1<l_t^2))\max(l_t^1,l_t^2) \\
&-\sum_{t=1}^{T}\mathbb{I}(g=(l_t^1<l_t^2))\min(l_t^1,l_t^2)+\sum_{t=1}^{T}\min(l_t^1,l_t^2)\Big]
\end{aligned}
\tag{7.10}
$$

Here, the last term is a constant and does not depend on function g. Minimization of R with respect to g will not be effected by the third term, and hence, can be removed. Thus, empirical risk may be given by

$$R = \frac{1}{T}\sum_{t=1}^{T}\mathbb{I}(g = (l_t^1 < l_t^2))(\max(l_t^1, l_t^2) - \min(l_t^1, l_t^2))$$
$$= \frac{1}{T}\sum_{t=1}^{T}|l_t^1 - l_t^2|\mathbb{I}(g = (l_t^1 < l_t^2)) \tag{7.11}$$

□

From theorem 1, it can be deduced that the optimal partition of the task space is obtained when each task is given the label 0, if model in partition 0 gives a lower error, and 1, if model in partition 1 gives a lower error. The empirical risk is then the weighted error of classifying these tasks, with weights for each error given as $|l_t^1 - l_t^2|$. The partition function in the task space, g, can be obtained by using any cost sensitive classifier. Here, we assume that the classifier g is a linear classifier. The value of θ_t used here is the single task learners obtained from solving logistic regression or linear regression problem. The details for θ_t are provided in section 7.2.2.3.

To compute the weights of linear classifier g, we use the cost sensitive classifier developed by Zhang and Garcia (164). For each incoming task t, the classifier g may be updated as

$$g_t = g_{t-1} + \tau_t \text{label } \theta_t \tag{7.12}$$

Here, the value of label is the label given to task t according to previous paragraph. The value of τ_t is defined as

$$\tau_t' = \frac{1 - \text{label } g_{t-1}^T \theta_t}{\theta_t^T \theta_t}$$
$$\tau_t = \begin{cases} 0 & \text{if } \tau_t' \leq 0 \\ \tau_t' & \text{if } 0 < \tau_t' \leq \alpha c_t \\ \alpha c_t & \text{if } \tau_t' > \alpha c_t \end{cases} \tag{7.13}$$

Here, α is a constant determining the size of step, and c_t is the weight associated with each task. Here, the value of c_t is $|l_t^1 - l_t^2|$.

7.2.2.2 Multiregion-Partitioned Task Space

The above theorem can be easily extended to a multi-region partitioned task space as well. For a given node k in the partition tree and a given task t, the empirical loss and the global empirical risk is given in equations 7.5 and 7.7.

In order to solve for each of the g_k for fixed task and model specific models in equation 7.7, we use alternate minimization. Thus, for each task, we first fix all the u_r and g_k for k in all nodes of the partition tree. Then, we pick one partition function g_k at a time and solve for that. Therefore, for each node k not in leaf node, the values of g_k is given by minimizing

$$g_k = \min \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}(g_k = 0) L_t^{k_L} + \mathbb{I}(g_k = 1) L_t^{k_R} \tag{7.14}$$

Minimizing the above equation is same as minimizing

$$g_k = \min -\frac{1}{T} \sum_{t=1}^{T} |L_t^{k_L} - L_t^{k_R}| \mathbb{I}(g_k = (L_t^{k_L} > L_t^{k_R}) \tag{7.15}$$

This is a direct extension of Theorem 1. It is also easy to infer from theorem 1 that the value of $L_t^{k_L}$ is the smallest value of $l_t^{r_{kL}}$ where r_{kL} represents all the leaf nodes under the left child of node k. Similarly, $L_t^{k_R}$ is the minimum value of $l_t^{r_{kR}}$. Since, all other partition functions and region models are fixed, the value of $L_t^{k_L}$ and $L_t^{k_R}$ may easily be computed. Thus, the value of each partition function may be computed using any online cost sensitive classifier. After updating the partition functions, we solve for each u_r and v_t as described in the next section.

7.2.2.3 Predicting Regional and Task Specific Models

Given the value of the partition function, we know the region a given task belongs to. Let this region be depicted by r, and the indices of all tasks in this region belong to set S. The equation 7.7 then reduces to

$$\begin{aligned} R = & \frac{1}{T}\sum_{t\in S}\frac{1}{N_t}\sum_{i=1}^{N_t} L(x_{i,t}(u_r+v_t), y_{i,t}) \\ & + \lambda_1 \frac{1}{T}\sum_{t\in S}\|v_t\|_1 + \lambda_2\|u_r\|_1 \end{aligned} \tag{7.16}$$

The major inefficiency in the above equation is that it depends on all the samples from previous data owing to the inner summation in the equation. In order to overcome this efficiency, we use the trick used in (41). We expand the inner summation term, $\frac{1}{N_t}\sum_{i=1}^{N_t} L(x_{i,t}(u_r+v_t), y_{i,t})$, around $\theta_t = \min \frac{1}{N_t}\sum_{i=1}^{N_t} L(f(x_{i,t},\theta), y_{i,t})$. In other words, θ_t is the single task learner of task t. Using second order Taylor series expansion and ignoring all the constant terms, we get

$$R = \frac{1}{T}\sum_{t\in S}\|\theta_t - (u_r+v_t)\|_{D_t} + \lambda_1\frac{1}{T}\sum_{t\in S}\|v_t\|_1 + \lambda_2\|u_r\|_1 \tag{7.17}$$

where,

$$\begin{aligned} D_t &= \nabla_{\theta,\theta}\frac{1}{N_t}\sum_{i=1}^{N_t} L(f(x_{i,t},\theta), y_{i,t})|_{\theta=\theta_t} \\ \theta_t &= \min\frac{1}{N_t}\sum_{i=1}^{N_t} L(f(x_{i,t},\theta), y_{i,t}) \end{aligned}$$

and, $\|K\|_D$ is defined as $K^T DK$.

We can obtain the value of θ_t using linear or logistic regression. The value of θ_t may be eas-

ily updated online as well.

In order to find v_t and u_r, we first fix u_r and solve for v_t. The solution for v_t can be obtained by solving the following equation

$$R = \|\theta_t - (u_r + v_t)\|_{D_t} + \lambda_1 \|v_t\|_1 \tag{7.18}$$

This equation can be solved using any off the shelf algorithm for solving L_1 norm regularization.

Then, we keep the values of v_t fixed, and find the value of u_r. Therefore, we minimize the following equation

$$R = \frac{1}{T} \sum_{t \in S} \|\theta_t - (u_r + v_t)\|_{D_t} + \lambda_2 \|u_r\|_1 \tag{7.19}$$

The above equation depends on all the tasks in the given region. In this case, using simple stochastic gradient algorithm to find the values of u_r in an online fashion may be used. However, it is very difficult to find the sparse solution in online framework because it is hard to find summation of different values which add to zero. For this purpose, we base our solution to find u_r on Lin Xiao's regularized dual averaging method (154).

The regularized dual averaging method updates the parameters by the average sub gradient of the function. Here, the value of the sub gradient d_t is

$$d_t = -2(\theta_t - v_t - u_r) tr(D_t) \tag{7.20}$$

where $tr(D_t)$ is the trace of matrix D_t. Then, the average sub gradient is computed as

$$\bar{d_t} = \frac{t-1}{t}\bar{d}_{t-1} + \frac{1}{t}d_t \tag{7.21}$$

Thus, the value of u_r is updated as

$$u_{r,j} = \begin{cases} 0, & \text{if } \|\bar{d}_{t,j}\| \leq \lambda_t^{RDA} \\ -\frac{\sqrt{t}}{\gamma}\left(\bar{d}_{t,j} - \lambda_t^{RDA} sign(\bar{d}_{t,j})\right) & \text{otherwise} \end{cases} \tag{7.22}$$

The value of $\lambda_t^{RDA} = \lambda_2 + \frac{\rho}{\sqrt{t}}$ and γ and ρ are regularizing parameters, $u_{r,j}$ is the j^{th} element of vector u_r.

7.2.2.4 Online Partitioning of Task Space

There is one more inefficiency that needs to be discussed. Since this algorithm is online algorithm, the number of partitions or the size of the partition tree is not fixed. Infact, we start out with the entire task space and partition the task space recursively as we encounter more and more tasks. There are two conditions that still need to be discussed. When do we decide to split a given region into two parts, and when do we stop growing the partition tree. The answer to the second question is to choose a maximum depth of the tree and do not grow the partition tree beyond the maximum depth. To evaluate the criteria for deciding when to split a region, we use a common intuition. All the tasks in a given region r share the same model u_r. Thus, the loss function of the models in the same region will be in similar range. If a new task comes in, and the resulting loss l_t^r of new task t is significantly different in value than losses of other tasks, then new task t probably belongs to a separate group. Hence, the region must be split.

Algorithm 5 Supervised Clustering in Lifelong Multitask Learning

Require: Data $\{X_t, y_t$, λ_1, λ_2, Maximum depth of partition tree p, ρ
  Initialize values of u, v and g as 0
  for all incoming task t **do**
    Get the next batch of data for task t
    Update value of θ_t using any single task regression or classification
    if Task t belongs to existing task **then**
      s = number of task in region r where task t belongs
      $\bar{d}_{t-1} = s\bar{d}_t - \frac{1}{t-1}d_t$
      $u_{r,j} = -\frac{\sqrt{s-1}}{\gamma}\left(\bar{d}_{t-1,j} - \lambda_t^{RDA} sign(\bar{d}_{t-1,j})\right)$ for all $\|\bar{d}_{t,j}\| \leq \lambda_t^{RDA}$
    end if
    Update the partition functions as in Algorithm 6
    Update the values of v by minimizing equation 7.18
    Update the values of u given the partition functions using equation 7.22
  end for

In order to detect the significant difference in values of losses, we use theory from change point detection. Change point detection estimation is a problem in machine learning which is associated with detecting change of events in a given signal. More specifically, we implement Shewart Control Chart (128; 10) to decide if change point has occurred and given region needs to be split. Here, we assume that the losses of the tasks in a given region are the samples belonging to gaussian distribution with mean m and variance σ. The values of m and σ can easily be estimated. Let us assume that the new task t belongs to a gaussian distribution with mean l_t^r and variance σ. Then the value of decision function for Shewhart Control Chart is given by $df = (m - l_t^r)^2/(2x\sigma)$. Refer to (10) for derivation of decision function in Shewhart Control Chart for two gaussian functions. If df is greater than a constant h, then the region is split.

The summary of the entire algorithm is described in Algorithm 5.

7.3 Experimental Studies

In this paper, we developed a lifelong multitask learning algorithm which learns groups of tasks and selects a common set of features for each group of tasks. The description of the algorithm

Algorithm 6 Update the partition functions

Require: θ_t, t, h, p
  if t is the first task **then**
    Assign task t to first node of tree G which is the root
    Set the root node as the leaf node as well
  else
    g = partition function at root node of tree G
    while leaf node is not reached **do**
      Update partition g according to equation 7.13
      Set value of g as partition function of the next child of tree G
    end while
    if leaf node reached **then**
      m = mean of loss of all tasks in current region
      σ = variance of loss of all tasks in current region
      l = loss of task t
      if $(m-l)^2/(2*\sigma) > h$ and maximum depth of tree p is not reached **then**
        Split node
        Place all tasks in left child
        Place the task t in right child
        Compute the partition function g for this node
      end if
    end if
  end if

has been provided in the previous section. Now, we demonstrate the performance of our algorithm with respect to other state of the art similar algorithms. We show that current algorithm performs better than or equivalent to other methods.

The list of the different methods to which we compare our algorithm is provided below:

BatchMTL: This algorithm is a multitask learning algorithm developed by Kang et al (73). Kang et al also developed an algorithm for clustering the tasks in multitask learning framework and using feature selection to enforce that similar tasks depend on the similar features. BatchMTL is the closest multitask learning algorithm to our framework, except that BatchMTL is implemented in batch setting where entire data is available. Therefore, we use Kang et al's method to compare the performance of our algorithm and thank Kang for providing his code.

Table 7.1: Summary of each of the dataset

Dataset	# of tasks	# of features	# of samples
MNIST	10	87	20,000
USPS	10	64	20,000
LANDMINE	29	9	14,820
STOCK Market	25	16	24,452

ELLA: Efficient Lifelong Learning Algorithm, or ELLA, is a lifelong multitask learning algorithm (41). In this algorithm, the authors assume that all the tasks are related in a sub dimensional space, and learn a set of basis functions representing the sub dimensional space in which the tasks are related. For each task, the coefficients for the basis function are also learnt. Since the coefficients are assumed to be sparse, sometimes an overlapping clustering effect on the tasks is also observed. We pick this algorithm for comparison because it is based in lifelong setting. We thank the authors for providing us with their code.

TREE: This algorithm is a baseline method. Here, we build the clustering structure of the task using hierarchical clustering and enforce that the tasks at each node remain close to each other by using a gaussian prior around the mean of the tasks belonging to that node.

STL: The single task learning algorithm which we use to compare our algorithm with bayesian online regression and classification methods as described in (14).

Current Algorithm: This is the algorithm described in current paper. In this paper, we propose a global formulation for simultaneously partitioning and learning the tasks. In this way, both the task partitions and task parameters are learnt in a supervised setting.

7.3.1 Dataset Used

In order to evaluate the performance of our dataset, we use several real world datasets. The description of the dataset is provided below.

MNIST Dataset: The MNIST dataset is a collection of handwritten digits. We have downloaded this dataset from Kang's webpage (73). This dataset is basically a subset of the original MNIST dataset which has already been preprocessed. In this dataset, principal component analysis was used to extract the top 64 features of each image. The data has been provided for only 2,000 image, and each image needs to be classified into one of the digits between 0 and 9. We binarize the tasks, and thus, each task description includes identification of a single digit out of the ten digits. Thus, there are 10 tasks, and all of the 2,000 images serve as the samples for each task. Therefore, there are effectively 20,000 samples belonging to the total of 10 tasks. The dimensionality of each of the samples is 64.

USPS Dataset: This dataset is also downloaded for Kang's webpage (73). It is also a handwritten digit dataset which has been preprocessed using a similar procedure as the MNIST dataset. In USPS dataset, top 87 features were used. The task was again to identify the digit between 0 and 9 in the image. Again, we binarized the task, and thus, each task was to identify if an image contains a particular digit or not. The number of samples were 2,000 for each of the 10 task, making the effective number of samples to be 20,000, and the number of features in each sample was 87.

Stock Market Dataset: Stock market dataset is a compilation of the stock market prices which we compiled from Yahoo's financial services website. We selected 25 different companies from various market domains such as oil and gas industries, computers and electronics industries, pharmaceutical industries and finance. The weekly stock market prices from the period of December 1994 to December 2014 were obtained. Each task consisted of predicting the current stock market prices of each of the company based on the stock market prices of past sixteen weeks. There were

a total of 25 tasks and 24,452 samples, and each sample having 16 features.

Landmine Detection Dataset: This dataset is about predicting the presence of a mine based on the radar images of the location. Originally, this dataset is from (156), however, we obtain the data from (41). The dataset was available with the software for ELLA. The radar images were preprocessed and 9 features were extracted from the images, out of which 4 features were moment-based, 3 correlation-based, one energy-ratio feature and one spatial variance feature. The problem formulated as binary classification having 29 tasks, nine features and 14,820 samples.

Table 7.1 presents a summary of all the datasets.

7.3.2 Evaluation Protocol

All the models selected consisted of some hyper-parameters which need to be fixed by the user. The hyper-parameters were chosen using five fold crossvalidation on around fifty percent of the dataset. We also use a greedy search procedure if the algorithm has multiple hyper-parameters which need to be selected. Thus, we first fix all the hyper parameters to a constant value, and vary one at a time. We pick the value of the hyper parameter which minimizes the cross validation error. The values of the hyper parameters thus selected are kept constant across all the experiments.

For Kang et al's method, the hyper parameters which needed to be selected were the number of groups to use, the regularization coefficient for the task parameters and task correlation matrix. The value of the number of groups were chosen from a pool of $\{2,3,4,5\}$ and the values of regularization parameters were picked from $\{10^{-3},10^{-2},10^{-1},1,10,100\}$.

In ELLA, there were five different parameters which needed to be selected, namely, the number of basis selected, the regularization parameter for the basis, λ_1, regularization parameter for sparsity constraint of the basis coefficients in each task, μ, the ridge term for single task learner,

λ_2, and the L_2 regularization component for the single task specific components, μ_2. The values of the number of basis function were chosen from a pool of $\{2,4,6,8,10\}$. The values of λ_1, μ and λ_2 were chosen from the pool of {exp(-12), exp(-8), exp(-4), exp(0), exp(4)}. The value of μ_2 was also selected from the same pool except the value of infinity was added to the pool according the author's guidelines.

In the TREE algorithm, the online bayesian methods for classification and regression were used as single task learners. The single task parameters were clustered using online top-down hierarchical clustering, and the mean of the tasks at each node was applied as the prior of each task. The same online single task learners were used for predicting online STL algorithm. There were two parameters used for online single task learners, the variance of the noise in the data, α, and the variance of the zero mean gaussian prior, β. The values of α and β were chosen from the pool of {0.001, 0.01, 0.1, 0, 1, 10, 100}. The gaussian prior was again used on each node of the hierarchical structure to regularize the related tasks together. The mean of all the tasks belonging to the node was used as the mean of the prior. The variance of the prior is chosen as the variance of the prior of its child node times the decay factor. The decay factor was picked from the pool of {1, 2, 5, 10, 20}.

Finally, in the current algorithm, there were five parameters which needed to be selected, namely, the maximum depth of the tree, p, regularization constant of task specific parameters, λ_1, regularization constant for regional classifier, λ_2, the sparsity constraint, ρ and parameter dictating the step size, γ. The number of reject classifiers was picked from {2, 3, 4, 5}, and the values of λ_1, γ and ρ were picked from {0.001, 0.01, 0.1, 1, 0, 10, 100}, and λ_2 from { 10^{-6}, 10^{-5},10^{-4},10^{-3} }. For deciding when to split the nodes, we need to choose the value of constant h. The value of h was picked from a pool of $\{0.05, 0.1, 0.15, 0.2\}$.

7.3.2.1 Performance Comparison

For all the algorithms tested, datasets were randomly divided into training and testing. All the algorithms were trained on the training dataset and evaluated on the testing dataset. For the MNIST and USPS dataset, 1000 samples from each tasks were picked for training and remaining for testing. Similarly, for Landmine Dataset, 300 samples and for Stock Market Dataset, 100 samples from each task were kept for training. The splits were consistent across all the models. The hyper parameters were fixed to the values that minimized the cross-validation error as found previously. The division of the dataset into training and testing was repeated 100 times. The samples from each task were fed into the algorithm one at a time and their performance on the test samples was measured. The order of the tasks were randomized between each run. The test samples were only used for testing the model and did not contribute to the training. The results reported are on test dataset.

The results recorded are the average accuracy obtained from each randomization. The accuracy for the regression task is defined as $1 - nmse$, where $nmse$ is the normalized mean squared error. For the classification tasks, we define the accuracy as the ratio of the correctly classified samples to the total number of samples. The results are reported in Table 7.2.

As can be seen in the Table 7.2, the performance of our algorithm is superior to all other algorithms, except for stock market dataset, where there is not significant difference in the performance of our algorithm versus others.

We also compare our algorithm with a batch multitask learning algorithm. We use Kang's method (73), Batch MTL, as the batch multitask method as this algorithm groups the tasks explicitly into groups and does feature selection as well. The comparison of our algorithm with Batch MTL is provided in Table 7.3. Usually, online algorithms do not have as good performance as the batch algorithms and batch algorithms are often viewed as the maximum performance that may be

Table 7.2: The performance comparison of our method with other methods. The mean accuracies are reported here. The Current method almost always produced best accuracies. For Stock Dataset, even though ELLA has the best performance the improvement is not statistically significant.

Dataset	STL	TREE	ELLA	Current
MNIST	0.5561 ± 0.0294	0.5691 ± 0.0286	0.8179 ± 0.0651	$\mathbf{0.8960 \pm 0.0189}$
USPS	0.5217 ± 0.0379	0.5269 ± 0.0399	0.8016 ± 0.0786	$\mathbf{0.8998 \pm 0.0001}$
LANDMINE	0.5535 ± 0.0173	0.5658 ± 0.0181	0.6515 ± 0.0313	$\mathbf{0.7308 \pm 0.0107}$
STOCK	0.9356 ± 0.0019	0.9372 ± 0.0018	$\mathbf{0.9399 \pm 0.0022}$	0.9388 ± 0.0026

Table 7.3: The performance comparison of our method with Batch Method. The mean accuracies are reported here. The performance of batch algorithms are often seen as the maximum performance which an online algorithm may reach. Here, the current method has a performance almost equivalent to the Batch Multitask Learning.

Dataset	Batch MTL	Current
MNIST	0.9000 ± 0.0001	0.8960 ± 0.0189
USPS	0.9000 ± 0.0001	0.8998 ± 0.0001
LANDMINE	0.7181 ± 0.0103	0.7308 ± 0.0107
STOCK	0.9249 ± 0.0101	0.9388 ± 0.0026

achieved by any online algorithm. However, our algorithm performs almost as good as the Batch MTL. In Landmine and Stock Market data, it performs even better. The reason may be that both Landmine and Stock Market data have few number of features, and hence, do not benefit by the feature selection.

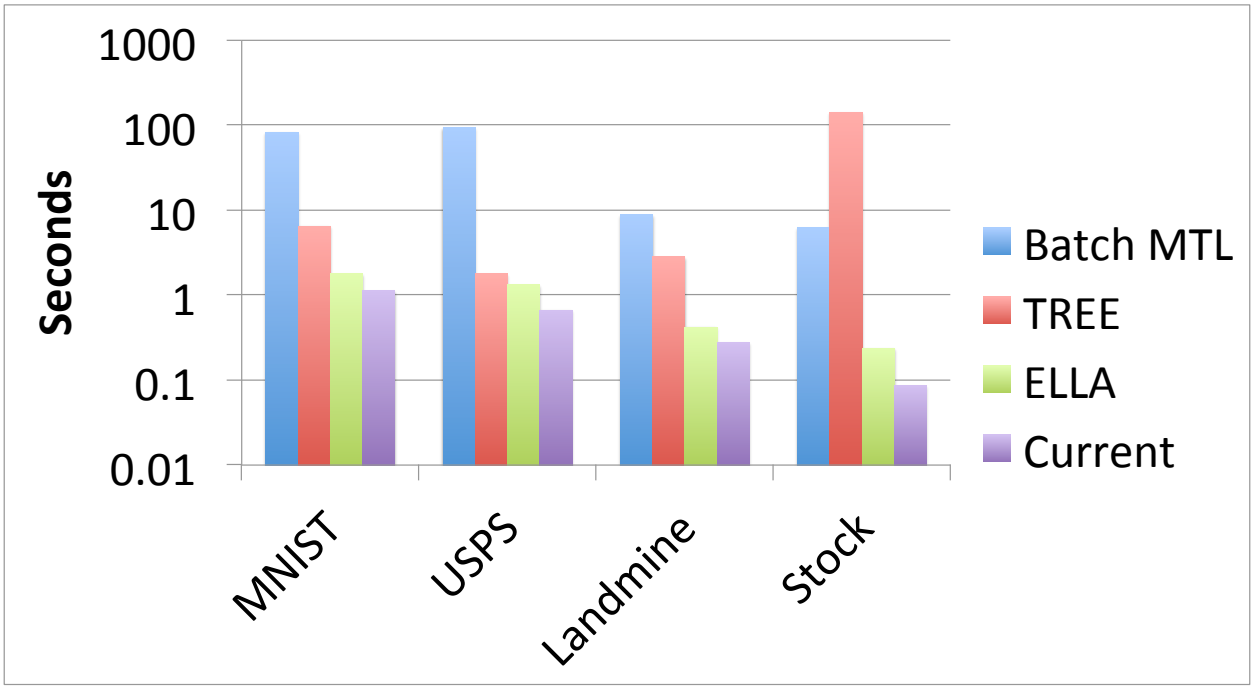
7.3.2.2 Time comparison

The time taken for training and testing each of the model was also recorded. We present the average time taken for each of the 100 randomizations along with the standard deviation. All the experiments were conducted on Intel(R) Xeon(TM) 3.2 GHz Linux Machine. The time is measured in seconds. The results are reported in Figure 7.1. Thus, it can be observed that the current algorithm has significantly faster performance than other methods.

7.3.2.3 Comparison with online k-mean partitions

In existing literature pertaining to multitask learning algorithms, unsupervised step is used to learn the task relationships. Most common formulation for learning task relationship is the use of k-

Figure 7.1: The average time taken in seconds to train plus test each algorithm. The vertical axis is the time in seconds on logarithmic scale and the horizontal axis represents the dataset. The current algorithm performs orders of magnitude faster than the batch MTL and significantly faster than other online algorithms.



means to cluster the tasks together. An important contribution of this paper is to show that supervised learning may also be used to learn task relationships. In order to show that supervised learning is better for partitioning task space, we use two methods to partition the task space in the current framework. The first method is using k-means algorithm to cluster the tasks into two groups at each node of the partition tree being developed in this algorithm. We use the online k-mean algorithm presented by Shindler (129). This algorithm is both fast as it requires only one pass of the dataset and has shown a good performance in unsupervised learning. The second method is to use our supervised learning approach to partition the task space as described in section 7.2.2.1. All other parameters were kept exactly the same in both the methods. The data was divided into training and testing and the performance was recorded 100 times. The same division of the data as used in previous sections were used here as well. The performance of using k-means versus the current algorithm is shown in table 7.4. The time comparison of the two methods is shown in table 7.5. As can be seen, the current method is faster and has better performance than using k-means

Table 7.4: The performance comparison of our method versus using k-means to group the tasks together. Results reported are average accuracies. As can be seen, our algorithm has a better performance than using just k-means.

Dataset	k-Means	Current
MNIST	0.8863 ± 0.0245	$\mathbf{0.8960 \pm 0.0189}$
USPS	0.8972 ± 0.0128	$\mathbf{0.8998 \pm 0.0001}$
LANDMINE	0.7301 ± 0.0110	$\mathbf{0.7308 \pm 0.0107}$
STOCK	0.9317 ± 0.0027	$\mathbf{0.9388 \pm 0.0026}$

Table 7.5: The time comparison of our method versus using k-means to group the tasks together. Results reported are average time in seconds. As can be seen, our algorithm is slightly faster than using just k-means.

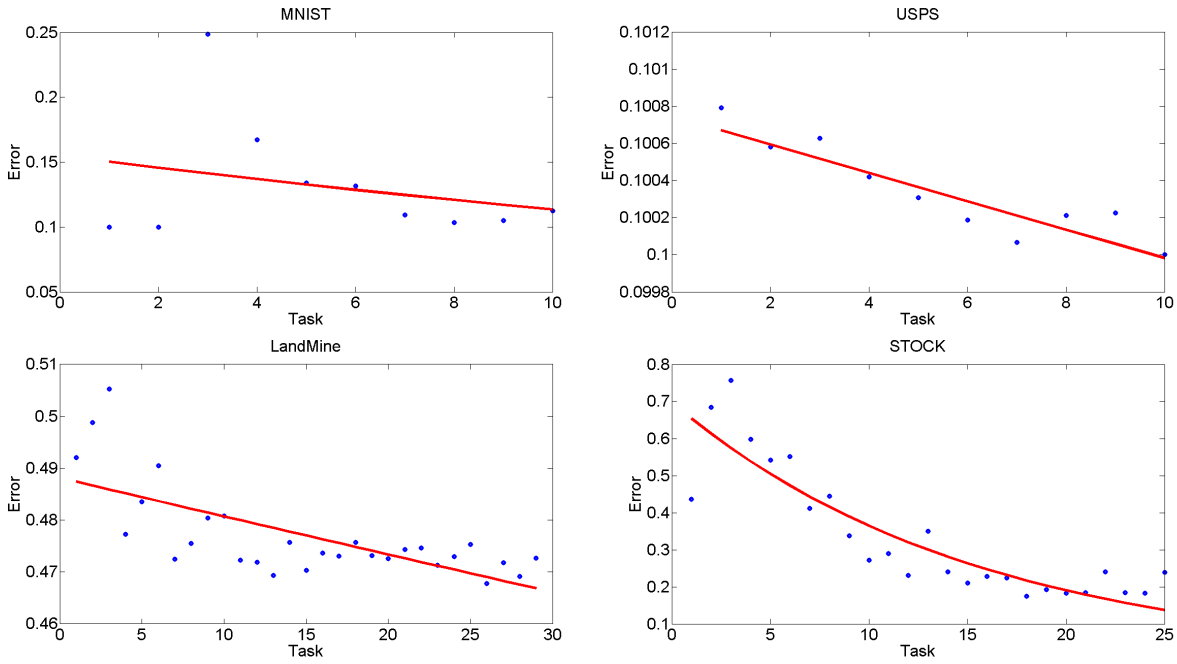
Dataset	k-Means (sec)	Current (sec)
MNIST	1.1583 ± 0.0530	$\mathbf{1.1326 \pm 0.0573}$
USPS	0.6819 ± 0.0437	$\mathbf{0.6637 \pm 0.0543}$
LANDMINE	0.4766 ± 0.0206	$\mathbf{0.2768 \pm 0.0320}$
STOCK	0.1184 ± 0.0170	$\mathbf{0.0865 \pm 0.0256}$

to partition the tasks. It is observed that the k-means algorithm takes significantly more time for datasets with larger number of tasks such as landmine and stock dataset.

7.3.2.4 Validation against Negative Transfer of Information

The last experiment which we conducted was to measure the error as a function of task. For this experiment, we plot the average error on existing tasks over 100 runs, each time a new task is made available. The tasks were randomized in each of the 100 runs. The errors were plotted against the position of the task in the sequence in which it was encountered. We also plot the best exponential fit curve. The figure is shown in Figure 7.2. As can be seen in the figure, the overall trend of the error is decreasing in all the tasks. This shows that negative transfer does not occur in general as new tasks arrive and the new tasks arriving improve the overall performance.

Figure 7.2: The decrease in the overall error as a function of position in task sequence. The red line indicates best exponential fitting curve.



7.4 Conclusion

In this paper, we develop a novel algorithm for lifelong multitask learning in which we use partition functions to cluster the related tasks and learn the task parameters. We present a global formulation which partitions the task space and learns the task models using supervised learning. We also assume that the related tasks depend on similar set of features, and thus, regularize the similar tasks together using L_1 norm to implement feature selection.

The current algorithm is implemented and its performance is compared against leading lifelong learning algorithm. We observe that the current algorithm performs better than the leading algorithms in terms of both speed and accuracy.

Chapter 8

Conclusion and Future Work

This thesis is dedicated to the study of task relationship modeling in multitask learning. There are a lot of implementation of multitask learning algorithms where the algorithms assume that all tasks are related in the same way. These algorithms may or may not use feature learning. However, when shifting gears to move to the domain of utilization of known task relationship structure or trying to learn the task relationship structure, the implementations that utilize feature learning become very limited. If we move to domain of online fixed task multitask and lifelong multitask learning, the amount of existing literature reduces significantly. When considering task relationship models and feature learning, the literature is even further scarce. Thus, the focus of this thesis is to develop multitask feature learning algorithms in both static and lifelong learning settings, which learn the task relationships, and apply them in a real world dataset.

The first two algorithms we explored were related to static multitask learning with task relationship modeling. In both the algorithms, we delved into the simplest kind of task relationships, which is identification of groups of related tasks. Most of the existing multitask learning algorithms assume that all the related tasks are close to each other in either the physical space or an alternative sub-dimensional latent space. Also, we often observe that the tasks do not depend on all the features, and feature selection improves the generalization error of the task. Usually, the similar tasks tend

to depend on the similar sets of features. Therefore, in these two learning algorithms we identify groups of similar tasks and enforce similar tasks to depend on similar features. For feature selection, bayesian equivalent of the L_0 norm, spike and slab prior, is used as L_0 norm is the ideal feature selection method and improves the efficiency. For task clustering, we explore two different priors, namely, mixture of gaussian priors and categorical distribution. For the first algorithm, we assume that the probability of selecting a feature ρ_d in each task lie close to each other, and enforce a gaussian prior over ρ_d in each group. Therefore, mixture of gaussian prior describes the distribution of tasks across the groups. We empirically observe that using the mixture of gaussian prior gives a good performance in large number of tasks, but has convergence issues when the number of tasks are small. The reason may be the large number of parameters that need to be optimized in this method. Therefore, in the second algorithm, categorical distribution was used to group the tasks together. We make a stronger assumption here that the probabilities of selecting each feature in a group is same. And the probability of a task belonging to a group is defined using categorical distribution. We empirically show this algorithm out performs leading state of the art methods used for identification of groups of related tasks in multitask learning.

The next two algorithms explore the setting of learning task relationship model in lifelong multitask learning. Task relationship modeling often takes the form of estimating a fixed sized matrix to represent the task relationships. With varying number of tasks, it is not computationally efficient to use fixed sized task correlation matrices. Therefore, we propose the use of partition models to divide the task space. The partition model makes it easier to transfer the learnt information to the new incoming tasks. We develop two different algorithms for identification of groups of tasks in lifelong multitask learning. In the first approach, the task space is partitioned hierarchically based on the proximity of the tasks. For each node in the partition tree, a local model is learnt for all the tasks which belong to that node. The task parameters are given by the sum of all the models corresponding to the partition the task falls in and the task specific parameters. In this algorithm, task partition and model learning takes place in two independent steps. In the next algorithm, we

propose a global formulation for task partitioning and learning local models common to the tasks in a given group. Both, the learning of local models and the learning of partition functions is formalized as a supervised learning problem. Further, this algorithm executes feature selection as well. We empirically show that our algorithms perform better than the state of the art algorithms in terms of both accuracy and computational performance.

This thesis is actually the beginning of a new topic of structure identification in lifelong multitask learning framework. The domain of lifelong multitask learning is a potentially unexplored area, and there are multiple directions in which we might proceed. In this thesis, we only explore the identification of groups of tasks. There are multiple applications where the tasks are arranged either hierarchically or have a graphical structure. Task clusters are merely a simplification of these structures. Therefore, one direction may be to learn the complete task relationship structures rather than their simplification in form of clusters.

Another direction which might be explored is the use of non-parametric methods for lifelong multitask learning. The parametric methods usually pre-define the size of the model to be used. For example, the maximum number of partitions or clusters in our model are set before the training process starts. However, allowing the model to grow in lifelong learning becomes specially important because the total number of tasks a learning agent acquires during its lifetime is unknown. Therefore, incorporation of non-parametric algorithms with lifelong multitask learning can provide one solution.

The application domain of lifelong multitask learning is relatively unexplored as well. A wide range of applications may benefit from lifelong multitask learning. Some of the examples of such applications are image annotations, predicting protein-chemical interaction, robotics and document classification. In all these applications, the number of tasks may increase as new data is added to the learning agent. It will be interesting to explore and apply lifelong learning in real

world applications so that humanity may benefit from lifelong multitask learning.

References

[1] Abernethy, J., Bartlett, P., & Rakhlin, A. (2007). Multitask learning with expert advice. In *Learning Theory* (pp. 484–498). Springer.

[2] Agarwal, A., Rakhlin, A., & Bartlett, P. (2008). Matrix regularization techniques for online multitask learning. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-138*.

[3] Ammar, H. B., Eaton, E., Ruvolo, P., & Taylor, M. (2014). Online multi-task learning for policy gradient methods. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (pp. 1206–1214).

[4] Archembeau, C., Guo, S., & Zoeter, O. (2011). Sparse bayesian multi-task learning. In *Advances in Neural Information Processing Systems*, volume 1 (pp.$\tilde{4}$1).

[5] Argyriou, A., Evgeniou, T., & Pontil, M. (2007a). Multi-task feature learning. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19 (pp.$\tilde{4}$1).: The MIT Press.

[6] Argyriou, A., Evgeniou, T., & Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, 73(3), 243–272.

[7] Argyriou, A., Pontil, M., Ying, Y., & Micchelli, C. A. (2007b). A spectral regularization framework for multi-task structure learning. In *Advances in Neural Information Processing Systems* (pp. 25–32).

[8] Attneave, F. (1959). Applications of information theory to psychology: A summary of basic concepts, methods, and results.

[9] Bakker, B. & Heskes, T. (2003). Task clustering and gating for bayesian multitask learning. *The Journal of Machine Learning Research*, 4, 83–99.

[10] Basseville, M., Nikiforov, I. V., et al. (1993). *Detection of abrupt changes: theory and application*, volume 104. Prentice Hall Englewood Cliffs.

[11] Bennett, K. P., Cristianini, N., Shawe-Taylor, J., & Wu, D. (2000). Enlarging the margins in perceptron decision trees. *Machine Learning*, 41(3), 295–313.

[12] Binh, N. D. (2011). Online multiple tasks one-shot learning of object categories and vision. In *Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia* (pp. 131–138).: ACM.

[13] Bishop, C. M. et al. (2006a). *Pattern recognition and machine learning*, volume 1. springer New York.

[14] Bishop, C. M. et al. (2006b). *Pattern recognition and machine learning*, volume 4. springer New York.

[15] Bonilla, E. V., Chai, K. M. A., & Williams, C. K. (2007). Multi-task gaussian process prediction. In *Nips*, volume 20 (pp. 153–160).

[16] Bot, M. C. & Langdon, W. B. (2000). Improving induction of linear classification trees with genetic programming. *space*, 10(1.25), 2.

[Breiman et al.] Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. Classification and regression trees.

[18] Cai, L. & Hofmann, T. (2004). Hierarchical document categorization with support vector machines. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management* (pp. 78–87).: ACM.

[19] Caruana, R. (1996). Algorithms and applications for multitask learning. In *ICML* (pp. 87–95).

[20] Caruana, R. (1998). *Multitask learning*. Springer.

[21] Cavallanti, G. & Cesa-Bianchi, N. (2012). Memory constraint online multitask classification. *arXiv preprint arXiv:1210.0473*.

[22] Cavallanti, G., Cesa-Bianchi, N., & Gentile, C. (2010). Linear algorithms for online multitask classification. *The Journal of Machine Learning Research*, 11, 2901–2934.

[23] Cesa-Bianchi, N., Gentile, C., Zaniboni, L., et al. (2006). Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research*, 7(1).

[24] Chai, K. M. A., Williams, C. K., Klanke, S., & Vijayakumar, S. (2008). Multi-task gaussian process learning of robot inverse dynamics. In *NIPS* (pp. 265–272).

[25] Chandra, B., Kothari, R., & Paul, P. (2010). A new node splitting measure for decision tree construction. *Pattern Recognition*, 43(8), 2725–2731.

[26] Chen, J., Tang, L., Liu, J., & Ye, J. (2009). A convex formulation for learning shared structures from multiple tasks. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 137–144).: ACM.

[27] Chen, X., Kim, S., Lin, Q., Carbonell, J. G., & Xing, E. P. (2010). Graph-structured multi-task regression and an efficient optimization method for general fused lasso. *arXiv preprint arXiv:1005.3579*.

[28] Chen, X., Lin, Q., Kim, S., Carbonell, J. G., Xing, E. P., et al. (2012). Smoothing proximal gradient method for general structured sparse regression. *The Annals of Applied Statistics*, 6(2), 719–752.

[29] Collobert, R. & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning* (pp. 160–167).: ACM.

[30] Cover, T. & Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1), 21–27.

[31] Crammer, K. & Mansour, Y. (2012). Learning multiple tasks using shared hypotheses. In *Advances in Neural Information Processing Systems 25* (pp. 1484–1492).

[32] Dasarathy, B. V. (1991). Nearest neighbor ({NN}) norms:{NN} pattern classification techniques.

[33] Daumé III, H. (2009). Bayesian multitask learning with latent hierarchies. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 135–142).: AUAI Press.

[34] DeCoro, C., Barutcuoglu, Z., & Fiebrink, R. (2007). Bayesian aggregation for hierarchical genre classification. In *ISMIR* (pp. 77–80).

[35] Dekel, O., Keshet, J., & Singer, Y. (2004). Large margin hierarchical classification. In *Proceedings of the twenty-first international conference on Machine learning* (pp.$\tilde{2}$7).: ACM.

[36] Dekel, O., Long, P. M., & Singer, Y. (2006). Online multitask learning. In *Learning Theory* (pp. 453–467). Springer.

[37] Dekel, O., Long, P. M., & Singer, Y. (2007). Online learning of multiple tasks with a shared loss. *Journal of Machine Learning Research*, 8(10).

[38] Domingo, C. & Watanabe, O. (2000). Madaboost: A modification of adaboost. In *COLT* (pp. 180–189).

[39] Domingos, P. (1996). Unifying instance-based and rule-based induction. *Machine Learning*, 24(2), 141–168.

[40] Dumais, S. & Chen, H. (2000). Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 256–263).: ACM.

[41] Eaton, E. & Ruvolo, P. L. (2013). Ella: An efficient lifelong learning algorithm. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)* (pp. 507–515).

[42] Evgeniou, T., Micchelli, C. A., & Pontil, M. (2006). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6(1), 615.

[43] Evgeniou, T. & Pontil, M. (2004). Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 109–117).: ACM.

[44] Fayyad, U. M. & Irani, K. B. (1992). The attribute selection problem in decision tree generation. In *AAAI* (pp. 104–110).

[45] Fei, H. & Huan, J. (2013). Structured feature selection and task relationship inference for multi-task learning. *Knowledge and information systems*, 35(2), 345–364.

[46] Fei, H., Jiang, R., Yang, Y., Luo, B., & Huan, J. (2011). Content based social behavior prediction: a multi-task learning approach. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (pp. 995–1000).: ACM.

[47] Freund, Y. (2001). An adaptive version of the boost by majority algorithm. *Machine learning*, 43(3), 293–318.

[48] Friedman, J. H. (1977). A recursive partitioning decision rule for nonparametric classification. *IEEE Trans. Computers*, 26(4), 404–408.

[49] Gama, J. (1997). Oblique linear tree. In *Advances in Intelligent Data Analysis Reasoning about Data* (pp. 187–198). Springer.

[50] Gelfand, S. B., Ravishankar, C., & Delp, E. J. (1989). An iterative growing and pruning algorithm for classification tree design. In *Systems, Man and Cybernetics, 1989. Conference Proceedings., IEEE International Conference on* (pp. 818–823).: IEEE.

[51] Gong, P., Ye, J., & Zhang, C. (2012). Multi-stage multi-task feature learning. In *NIPS* (pp. 1997–2005).

[52] Gopal, S. & Yang, Y. (2013). Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 257–265).: ACM.

[53] Gu, Q. & Han, J. (2013). Clustered support vector machines. In *proceedings of the sixteenth international conference on artificial intelligence and statistics* (pp. 307–315).

[54] Gupta, S., Phung, D., & Venkatesh, S. (2013). Factorial multi-task learning: A bayesian nonparametric approach. In *Proceedings of International Conference on Machine Learning (ICML)*.

[55] Han, S., Liao, X., & Carin, L. (2012). Cross-domain multitask learning with latent probit models. *arXiv preprint arXiv:1206.6419*.

[56] Harpale, A. & Yang, Y. (2010). Active learning for multi-task adaptive filtering. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (pp. 431–438).

[57] Hastie, T. & Tibshirani, R. (1996). Discriminant analysis by gaussian mixtures. *Journal of the Royal Statistical Society. Series B (Methodological)*, (pp. 155–176).

[58] Henrichon Jr, E. G. & Fu, K.-S. (1969). A nonparametric partitioning procedure for pattern classification. *Computers, IEEE Transactions on*, 100(7), 614–624.

[59] Hernández-Lobato, D. (2009). *Prediction Based on Averages over Automatically Induced Learners: Ensemble Methods and Bayesian Techniques*. PhD thesis, UNIVERSIDAD AUTONOMA DE MADRID.

[60] Hernández-Lobato, D., Hernández-Lobato, J. M., & Dupont, P. (2013). Generalized spike-and-slab priors for bayesian group feature selection using expectation propagation. *The Journal of Machine Learning Research*, 14(1), 1891–1945.

[61] Hernández-Lobato, D., Hernández-Lobato, J. M., Helleputte, T., & Dupont, P. (2010a). Expectation propagation for bayesian multi-task feature selection. In *Machine Learning and Knowledge Discovery in Databases* (pp. 522–537). Springer.

[62] Hernández-Lobato, D., Hernández-Lobato, J. M., & Suárez, A. (2010b). Expectation propagation for microarray data classification. *Pattern recognition letters*, 31(12), 1618–1626.

[63] Honorio, J. & Samaras, D. (2010). Multi-task learning of gaussian graphical models. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (pp. 447–454).

[64] Hull, J. J. (1994). A database for handwritten text recognition research. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(5), 550–554.

[65] Indurkhya, N. & Weiss, S. M. (1995). Using case data to improve on rule-based function approximation. In *Case-Based Reasoning Research and Development* (pp. 217–228). Springer.

[66] Ishwaran, H. & Rao, J. S. (2005). Spike and slab variable selection: frequentist and bayesian strategies. *Annals of Statistics*, (pp. 730–773).

[67] Iyengar, V. S. (1999). Hot: Heuristics for oblique trees. In *Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on* (pp. 91–98).: IEEE.

[68] Jacob, L., Bach, F., & Vert, J.-P. (2008). Clustered multi-task learning: A convex formulation. *arXiv preprint arXiv:0809.2085*.

[69] Jalali, A., Sanghavi, S., Ruan, C., & Ravikumar, P. K. (2010). A dirty model for multi-task learning. In *Advances in Neural Information Processing Systems* (pp. 964–972).

[70] Jawanpuria, P. & Nath, J. S. (2012). A convex feature learning formulation for latent task structure discovery. *arXiv preprint arXiv:1206.4611*.

[71] Ji, S., Dunson, D., & Carin, L. (2009). Multitask compressive sensing. *Signal Processing, IEEE Transactions on*, 57(1), 92–106.

[72] Jun, B. H., Kim, C. S., Song, H.-Y., & Kim, J. (1997). A new criterion in selection and discretization of attributes for the generation of decision trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(12), 1371–1375.

[73] Kang, Z., Grauman, K., & Sha, F. (2011). Learning with whom to share in multi-task feature learning. In *Proceedings of the 28th International Conference on Machine Learning* (pp. 521–528).

[74] Kato, T., Kashima, H., Sugiyama, M., & Asai, K. (2007). Multi-task learning via conic programming. In *NIPS*.

[75] Kim, T.-K. & Kittler, J. (2005). Locally linear discriminant analysis for multimodally distributed classes for face recognition with a single model image. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(3), 318–327.

[76] Koller, D. & Sahami, M. (1997). Hierarchically classifying documents using very few words.

[77] Koza, J. R. (1991). Concept formation and decision tree induction using the genetic programming paradigm. In *Parallel Problem Solving from Nature* (pp. 124–128). Springer.

[78] Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press.

[79] Kumar, A. & Daume III, H. (2012). Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417*.

[80] Ladicky, L. & Torr, P. (2011). Locally linear support vector machines. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (pp. 985–992).

[81] Lampert, C. H., Nickisch, H., & Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (pp. 951–958).: IEEE.

[82] Last, M. & Maimon, O. (2004). A compact and accurate model for classification. *Knowledge and Data Engineering, IEEE Transactions on*, 16(2), 203–215.

[83] Lázaro-gredilla, M. & Titsias, M. K. (2011). Spike and slab variational inference for multi-task and multiple kernel learning. In *Advances in neural information processing systems* (pp. 2339–2347).

[84] Lee, S., Zhu, J., & Xing, E. P. (2010). Adaptive multi-task lasso: with application to eqtl detection. In *Advances in neural information processing systems* (pp. 1306–1314).

[85] Lee, S.-I., Chatalbashev, V., Vickrey, D., & Koller, D. (2007). Learning a meta-level prior for feature relevance from multiple related tasks. In *Proceedings of the 24th international conference on Machine learning* (pp. 489–496).: ACM.

[86] Li, C., Dong, W., Liu, Q., & Zhang, X. (2014a). Mores: Online incremental multiple-output regression for data streams. *arXiv preprint arXiv:1412.5732*.

[87] Li, G., Hoi, S. C., Chang, K., Liu, W., & Jain, R. (2014b). Collaborative online multitask learning. *Knowledge and Data Engineering, IEEE Transactions on*, 26(8), 1866–1876.

[88] Liao, X. & Carin, L. (2005). Radial basis function network for multi-task learning. In *NIPS*.

[89] Liu, H., Palatucci, M., & Zhang, J. (2009a). Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 649–656).: ACM.

[90] Liu, J., Ji, S., & Ye, J. (2009b). Multi-task feature learning via efficient l 2, 1-norm minimization. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 339–348).: AUAI Press.

[91] Liu, T.-Y., Yang, Y., Wan, H., Zeng, H.-J., Chen, Z., & Ma, W.-Y. (2005). Support vector machines classification with a very large-scale taxonomy. *ACM SIGKDD Explorations Newsletter*, 7(1), 36–43.

[92] Lugosi, G., Papaspiliopoulos, O., & Stoltz, G. (2009). Online multi-task learning with hard constraints. *arXiv preprint arXiv:0902.3526*.

[93] Mairal, J., Ponce, J., Sapiro, G., Zisserman, A., & Bach, F. R. (2009). Supervised dictionary learning. In *Advances in neural information processing systems* (pp. 1033–1040).

[94] Meier, L., Van De Geer, S., & Bühlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1), 53–71.

[95] Micchelli, C. A. & Pontil, M. (2005). On learning vector-valued functions. *Neural Computation*, 17(1), 177–204.

[96] Minka, T. P. (2001a). Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence* (pp. 362–369).: Morgan Kaufmann Publishers Inc.

[97] Minka, T. P. (2001b). *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology.

[98] Mishra, M. & Huan, J. (2013a). Multitask learning with feature selection for groups of related tasks. In *IEEE International Conference on Data Mining*.

[99] Mishra, M. & Huan, J. (2013b). Multitask learning with feature selection for groups of related tasks. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on* (pp. 1157–1162).: IEEE.

[100] Mohamed, S., Heller, K., & Ghahramani, Z. (2011). Bayesian and l1 approaches to sparse unsupervised learning. *arXiv preprint arXiv:1106.1157*.

[101] Murthy, S. K., Kasif, S., & Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2(1), 1–32.

[102] Nejati, N., Langley, P., & Konik, T. (2006). Learning hierarchical task networks by observation. In *Proceedings of the 23rd international conference on Machine learning* (pp. 665–672).: ACM.

[103] Obozinski, G., Taskar, B., & Jordan, M. (2006). Multi-task feature selection. *Statistics Department, UC Berkeley, Tech. Rep*.

[104] Oiwa, H. & Fujimaki, R. (2014). Partition-wise linear models. In *Advances in Neural Information Processing Systems* (pp. 3527–3535).

[105] Ozawa, S., Roy, A., & Roussinov, D. (2009). A multitask learning model for online pattern recognition. *Neural Networks, IEEE Transactions on*, 20(3), 430–445.

[106] Parameswaran, S. & Weinberger, K. Q. (2010). Large margin multi-task metric learning. In *NIPS*, volume 23 (pp. 1867–1875).

[107] Passos, A., Rai, P., Wainer, J., & Daume III, H. (2012). Flexible modeling of latent task structures in multitask learning. *Proceedings of International Conference on Machine Learning (ICML)*.

[108] Patil, D. V. & Bichkar, R. (2012). Issues in optimization of decision tree learning: A survey. *International Journal of Applied Information Systems*, 3(5).

[109] Pentina, A. & Lampert, C. H. (2013). A pac-bayesian bound for lifelong learning. *arXiv preprint arXiv:1311.2838*.

[110] Pillonetto, G., Dinuzzo, F., & De Nicolao, G. (2010). Bayesian online multitask learning of gaussian processes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(2), 193–205.

[111] Puniyani, K., Kim, S., & Xing, E. P. (2010). Multi-population gwa mapping via multi-task regularized regression. *Bioinformatics*, 26(12), i208–i216.

[112] Qi, Y., Liu, D., Dunson, D., & Carin, L. (2008). Multi-task compressive sensing with dirichlet process priors. In *Proceedings of the 25th international conference on Machine learning* (pp. 768–775).: ACM.

[113] Qi, Y., Tastan, O., Carbonell, J. G., Klein-Seetharaman, J., & Weston, J. (2010). Semi-supervised multi-task learning for predicting interactions between hiv-1 and human proteins. *Bioinformatics*, 26(18), i645–i652.

[114] Quinlan, J. R. (1987). Simplifying decision trees. *International journal of man-machine studies*, 27(3), 221–234.

[115] Quinlan, J. R. (1993). *C4. 5: Programs for Machine Learning*. Morgan Kaufmann.

[116] Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

[117] Quinlan, J. R. et al. (1992). Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92 (pp. 343–348).: Singapore.

[118] Rai, P. & Daumé III, H. (2010). Infinite predictor subspace models for multitask learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS). Sardinia, Italy*.

[119] Rokach, L. & Maimon, O. (2005). Top-down induction of decision trees classifiers-a survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 35(4), 476–487.

[120] Rounds, E. (1980). A combined nonparametric approach to feature selection and binary decision tree design. *Pattern Recognition*, 12(5), 313–317.

[121] Roussopoulos, N., Kelley, S., & Vincent, F. (1995). Nearest neighbor queries. In *ACM sigmod record*, volume 24 (pp. 71–79).: ACM.

[122] Ruvolo, P. & Eaton, E. (2013a). Active task selection for lifelong machine learning. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI-13)*.

[123] Ruvolo, P. & Eaton, E. (2013b). Scalable lifelong learning with active task selection. In *AAAI Spring Symposium: Lifelong Machine Learning*.

[124] Ruvolo, P. & Eaton, E. (2014). Online multi-task learning via sparse dictionary optimization. In *Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*.

[125] Saha, A., Rai, P., Venkatasubramanian, S., & Daume, H. (2011). Online learning of multiple tasks and their relationships. In *International Conference on Artificial Intelligence and Statistics* (pp. 643–651).

[126] Shah, S. & Sastry, P. S. (1999). New algorithms for learning and pruning oblique decision trees. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 29(4), 494–505.

[127] Shen, J., Xu, H., & Li, P. (2014). Online optimization for max-norm regularization. In *Advances in Neural Information Processing Systems* (pp. 1718–1726).

[128] Shewhart, W. A. (1931). *Economic control of quality of manufactured product*, volume 509. ASQ Quality Press.

[129] Shindler, M., Wong, A., & Meyerson, A. W. (2011). Fast and accurate k-means for large datasets. In *Advances in neural information processing systems* (pp. 2375–2383).

[130] Shivaswamy, P. & Joachims, T. (2012). Online structured prediction via coactive learning. *arXiv preprint arXiv:1205.4213*.

[131] Silver, D. L. & Mercer, R. E. (2002). The task rehearsal method of life-long learning: Overcoming impoverished data. In *Advances in Artificial Intelligence* (pp. 90–101). Springer.

[132] Silver, D. L. & Poirier, R. (2004). *Sequential consolidation of learned task knowledge*. Springer.

[133] Silver, D. L., Poirier, R., & Currie, D. (2008). Inductive transfer with context-sensitive neural networks. *Machine Learning*, 73(3), 313–336.

[134] Silver, D. L., Yang, Q., & Li, L. (2013). Lifelong machine learning systems: Beyond learning algorithms. In *AAAI Spring Symposium: Lifelong Machine Learning*.

[135] Sollich, P. & Ashton, S. (2012). Learning curves for multi-task gaussian process regression. In *Advances in Neural Information Processing Systems* (pp. 1790–1798).

[136] Solomonoff, R. J. (1989). A system for incremental learning based on algorithmic probability. In *Proceedings of the Sixth Israeli Conference on Artificial Intelligence, Computer Vision and Pattern Recognition* (pp. 515–527).

[137] Steinberg, D. & Colla, P. (2009). Cart: classification and regression trees. *The top ten algorithms in data mining*, 9, 179.

[138] Stevens, R., Goble, C., Baker, P., & Brass, A. (2001). A classification of tasks in bioinformatics. *Bioinformatics*, 17(2), 180–188.

[139] Sun, X., Kashima, H., & Ueda, N. (2013). Large-scale personalized human activity recognition using online multitask learning. *Knowledge and Data Engineering, IEEE Transactions on*, 25(11), 2551–2563.

[140] Sun, X., Shrivastava, A., & Li, P. (2012). Fast multi-task learning for query spelling correction. In *Proceedings of the 21st ACM international conference on Information and knowledge management* (pp. 285–294).: ACM.

[141] Swirszcz, G. & Lozano, A. C. (2012). Multi-level lasso for sparse multi-task regression. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)* (pp. 361–368).

[142] Thrun, S. (1996). Learning to learn: Introduction. In *In Learning To Learn*: Citeseer.

[143] Thrun, S. (1998). Lifelong learning algorithms. In *Learning to learn* (pp. 181–209). Springer.

[144] Torgo, L. (1997). Functional models for regression tree leaves. In *ICML*, volume 97 (pp. 385–393).: Citeseer.

[145] Turlach, B. A., Venables, W. N., & Wright, S. J. (2005). Simultaneous variable selection. *Technometrics*, 47(3), 349–363.

[146] Utgo, P. E. & Clouse, J. (1996). A kolmogorov-smirnoff metric for decision tree induction. *Amherst, Massachusetts: University of Massachusetts, Department of Computer Science*.

[147] Viani, F., Poli, L., Oliveri, G., Robol, F., & Massa, A. (2013). Sparse scatterers imaging through approximated multitask compressive sensing strategies. *Microwave and Optical Technology Letters*, 55(7), 1553–1558.

[148] Wan, J., Zhang, Z., Yan, J., Li, T., Rao, B. D., Fang, S., Kim, S., Risacher, S., Saykin, A. J., & Shen, L. (2012). Sparse bayesian multi-task learning for predicting cognitive outcomes from neuroimaging measures in alzheimer's disease. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (pp. 940–947).: IEEE.

[149] Wang, H., Nie, F., Huang, H., Kim, S., Nho, K., Risacher, S. L., Saykin, A. J., Shen, L., et al. (2012). Identifying quantitative trait loci via group-sparse multitask regression and feature selection: an imaging genetics study of the adni cohort. *Bioinformatics*, 28(2), 229–237.

[150] Wang, J. & Saligrama, V. (2012). Local supervised learning through space partitioning. In *Advances in Neural Information Processing Systems* (pp. 91–99).

[151] Wang, X., Zhang, C., & Zhang, Z. (2009). Boosted multi-task learning for face verification with applications to web image and video search. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (pp. 142–149).: IEEE.

[152] Weiss, S. M. & Indurkhya, N. (1995). Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, (pp. 383–403).

[153] Wu, Q., Zhang, Y. D., Amin, M. G., & Himed, B. (2014). Complex multitask bayesian compressive sensing. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on* (pp. 3375–3379).: IEEE.

[154] Xiao, L. (2009). Dual averaging method for regularized stochastic learning and online optimization. In *Advances in Neural Information Processing Systems* (pp. 2116–2124).

[155] Xu, J., Tan, P.-N., & Luo, L. (2014). Orion: Online regularized multi-task regression and its application to ensemble forecasting. In *Data Mining (ICDM), 2014 IEEE International Conference on* (pp. 1061–1066).: IEEE.

[156] Xue, Y., Liao, X., Carin, L., & Krishnapuram, B. (2007). Multi-task learning for classification with dirichlet process priors. *The Journal of Machine Learning Research*, 8, 35–63.

[157] Yang, H., King, I., & Lyu, M. R. (2010). Online learning for multi-task feature selection. In *Proceedings of the 19th ACM international conference on Information and knowledge management* (pp. 1693–1696).: ACM.

[158] Yang, H., Lyu, M. R., & King, I. (2013). Efficient online learning for multitask feature selection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(2), 6.

[159] Yang, X., Kim, S., & Xing, E. (2009). Heterogeneous multitask learning with joint sparsity constraints. *Advances in neural information processing systems*, 23, 2151–2159.

[160] Yang, Y., Zhang, J., & Kisiel, B. (2003). A scalability analysis of classifiers in text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (pp. 96–103).: ACM.

[161] Yu, K., Tresp, V., & Schwaighofer, A. (2005). Learning gaussian processes from multiple tasks. In *Proceedings of the 22nd international conference on Machine learning* (pp. 1012–1019).: ACM.

[162] Yu, S., Tresp, V., & Yu, K. (2007). Robust multi-task learning with t-processes. In *Proceedings of the 24th international conference on Machine learning* (pp. 1103–1110).: ACM.

[163] Zhang, C.-H. & Huang, J. (2008). The sparsity and bias of the lasso selection in high-dimensional linear regression. *The Annals of Statistics*, (pp. 1567–1594).

[164] Zhang, J. & García, J. (2015). Online classifier adaptation for cost-sensitive learning. *Neural Computing and Applications*, (pp. 1–9).

[165] Zhang, J., Ghahramani, Z., & Yang, Y. (2005). Learning multiple related tasks using latent independent component analysis. In *NIPS*.

[166] Zhang, T. (2008). Multi-stage convex relaxation for learning with sparse regularization. In *NIPS*, volume 8 (pp. 1929–1936).

[167] Zhang, Y. & Schneider, J. G. (2010). Learning multiple tasks with a sparse matrix-normal penalty. In *NIPS* (pp. 2550–2558).

[168] Zhang, Y. & Yeung, D.-Y. (2012). A convex formulation for learning task relationships in multi-task learning. *arXiv preprint arXiv:1203.3536*.

[169] Zhang, Y., Yeung, D.-Y., & Xu, Q. (2010). Probabilistic multi-task feature selection. *Advances in neural information processing systems*, 23, 2559–2567.

[170] Zhong, W. & Kwok, J. (2012). Convex multitask learning with flexible task clusters. *arXiv preprint arXiv:1206.4601*.

[171] Zhou, J., Chen, J., & Ye, J. (2011). Clustered multi-task learning via alternating structure optimization. *Advances in Neural Information Processing Systems*, 25.