# Classifying protein sequences using regularized multi-task learning

Anveshi Charuvaka and Huzefa Rangwala

**Abstract**—Classification problems in which several learning tasks are organized hierarchically pose a special challenge because the hierarchical structure of the problems needs to be considered. Multi-task learning (MTL) provides a framework for dealing with such inter-related learning tasks. When two different hierarchical sources organize similar information, in principle, this combined knowledge can be exploited to further improve classification performance. We have studied this problem in the context of protein structure classification by integrating the learning process for two hierarchical protein structure classification databased, SCOP and CATH. Our goal is to accurately predict whether a given protein belongs to a particular class in these hierarchies using only the amino acid sequences. We have utilized the recent developments in multi-task learning (MTL) to solve the inter-related classification problems. We have also evaluated how the various relationships between tasks affect the classification performance. Our evaluations show that learning schemes in which both the classification databases are used outperform the schemes which utilize only one of them.

**Index Terms**—Multi-task Learning, Protein Structure Classification.

## 1 Introduction

Structural classification of proteins is an important step in understanding the evolutionary and functional relationships between proteins. SCOP [1] and CATH [2] were two of the earliest efforts in this direction, which classified proteins hierarchically into structurally related groups based on experimentally determined structures. However these gold-standard structural classifications are performed through a laborious and expensive process of first determining the three-dimensional structure of proteins, and curation into a hierarchical database using manual/semi-automated schemes. The rapid growth in the number of sequenced proteins in recent years has created a need for automated methods to accurately classify protein sequences into structural classes. Several computational methods have been proposed to predict the structural classes from sequence information using a range of statistical methods [3], [4], [5], [6], [7]. We develop a novel method that seeks to combine multiple hierarchical classification databases [8].

We have used multi-task learning (MTL) to combine the knowledge present across the two hierarchical protein classification databases. The motivation of our approach stems from the fact that different annotation databases strive to achieve the same objective (i.e., classify protein structures), but differ in terms of curation process, coverage and annotation errors. Further, the use of MTL approach allows us to capture the hierarchical structure that is prevalent across these databases. Within the hierarchies we may have several classes that have few training examples. As such, using MTL for training prediction models for related task assists in improving the generalization performance of individual tasks when the data is scarce [9].

The key contribution in this work is to combine the problem of hierarchical classification in two separate but related class hierarchies. Specifically, we explored different MTL formulations to leverage the combined knowledge present in SCOP and CATH hierarchical databases to improve the classification performance. Each labeled class in SCOP and CATH defines a classification task. The hierarchical structure of the labeled classes explicitly defines the relationships between the classes within each hierarchy. These class relationships can be utilized in the MTL formulations which take task relationships into account. However, the relationships between one class from SCOP and the other from CATH are non-trivial, as there is no direct correspondence between the labeled classes defined by SCOP and CATH.

We performed a comprehensive set of experiments to assess the accuracy of classification when using the MTL approaches in comparison to single task learning (STL). We observed an improvement in the classification performance from the combined use of SCOP and CATH in protein structure classification. We found that using the graph-based structured MTL that explicitly defines the relationships between different tasks and the trace-based MTL formulation that seeks to find a low-dimensional common subspace across the tasks are suitable for our purpose. Specifically, we observed approximately $4-5\%$ improvement for the multiple hierarchical based MTL learning in comparison to the STL approach. Our current study suggests that on the consistent set of protein domains defined by both SCOP and CATH, MTL methods which combined both the hierarchical schemes had a significant advantage over MTL methods which were trained using only one hierarchy.

## 2 BACKGROUND

### 2.1 Multi-task learning

In Multi-task learning (MTL), predictive models for multiple related tasks are learned simultaneously, as opposed to Single-task learning (STL), where a model for each task is learned separately. The main intuition is that the related tasks can help each other learn better models [9], [10]. MTL has been empirically and theoretically [11] [12] shown to improve the performance of the trained models, especially when the training data is scarce. There has been an explosive growth in the research on MTL, see [10], [11], [12], [13], [14], [15], [16], [17], [18] and the references therein.

Given a training set, with $n$ input-output pairs $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, generated from an unknown distribution, the goal is to learn a mapping function $f : \mathcal{X} \rightarrow \mathcal{Y}$ between the input domain $x_i \in \mathcal{X}$ and the output domain $y_i \in \mathcal{Y}$. The objective in standard regularized machine learning methods is to learn a model which minimizes a loss function on the training data while constraining the model complexity with a regularization penalty. The learning objective in general STL regularized learning methods can be represented as,

$$\min_{w} \sum_{i=1}^{n} \underbrace{\mathcal{L}\left(w, x_i, y_i\right)}_{loss} + \lambda \underbrace{\mathcal{R}\left(w\right)}_{regularization} \tag{1}$$

where $w$ is the set of model parameters. The regularization term controls the model complexity to discourage over-fitting. This principle can be extended to MTL, where we have $T$ tasks with training data for each of the $t = 1 \ldots T$ tasks, given by $\{(x_{it}, y_{it}) : i = 1 \ldots n_t\}$. The combined learning objective can be written as,

$$\min_{W} \sum_{t=1}^{T} \sum_{i=1}^{n_t} \underbrace{\mathcal{L}\left(w_t, x_{it}, y_{it}\right)}_{loss} + \lambda \underbrace{\mathcal{R}\left(W\right)}_{regularization} \tag{2}$$

where $n_t$ is the number of training instances for the $t^{th}$ task, $w_t$ denotes the model parameters for the $t^{th}$ task, and $W = \{w_t\}_{t=1}^{T}$ is the combined set of model parameters for all the tasks. Various multi-task learning methods take this general approach to build combined models for many related tasks.

Evegeniou and Pontil [19] proposed a regularized multi-task learning method where the model for each task is constrained to be close to the average of all the tasks. In multi-task feature learning and feature selection methods [20], [13], [14], [21], sparse learning based on lasso [22] type schemes, is performed to select or learn a common set of features across many related tasks. However, a common assumption made by many methods [19], [23], [24] is that all tasks are equally related. This assumption does not hold in all situations.

Therefore, it is sensible to take the task relationships into account in multi-task learning. The MTL formulations proposed in [15], [25] use an externally provided task network. However, these relationships might not be available and may need to be determined from the data. Clustered multi-task learning approaches assume that tasks exhibit a group structure, which is not known a-priori, and learn the clusters of tasks such that tasks within a cluster are more closely related to each other [16], [26], [27]. Another set of approaches, mostly based on Gaussian Process models, learn the task co-variance structure [17], [18] and are able to take advantage of both positive and negative correlations between the tasks.

In this paper we focus on the application of regularized MTL methods to the problem of automated protein structure classification using sequence information only. Preliminary ideas of this work were presented in our conference paper [8].

### 2.2 Protein structure classification

Structural Classification of Proteins (SCOP) [1] and Class, Architecture, Topology, and Homologous Superfamily (CATH) [2] are two prominent schemes for hierarchical classification of protein structures. SCOP follows a predominantly manual process, relying on the expertise of its curators. CATH relies on automated approaches to a greater degree than SCOP and revises its database more frequently. SCOP as well as CATH classify entities known as protein domains, which can be considered as functional units of proteins, into four major levels. In the context of this paper, domains may be viewed as subsequences (not necessarily contiguous) of proteins. A protein may be composed of one or more domains. Thus, each classification system assigns four labels to each of the domains it classifies. The levels defined by SCOP, listed in top to bottom order are Class, Fold, Superfamily and Family. The corresponding levels defined by CATH are Class, Architecture, Topology and Homologous Superfamily.

SCOP and CATH have been extensively compared in several previous studies [28], [29], [30]. Despite their similar purpose, several discrepancies between these two databases have been reported [30], [28]. Inconsistencies primarily arise due to the differences in domain definition and disagreement in the classification of some domains.

Predictive methods developed in recent years have primarily focused on classifying sequences with respect to SCOP. Consequently, remote homology detection and fold recognition, where the objectives are to detect superfamily and fold given the protein sequence, have received considerable attention in the research community. Early predictive methods used sequence similarity as a surrogate for structural similarity and utilized sequence comparison methods such as SW-alignment, BLAST and PSI-BLAST [31], [32]. The later methods took a generative approach to model related sequences. The discriminative methods, primarily Support Vector Machines, which followed generative methods further improved classification performance. Much attention was then given to engineering better kernels [3], [33], [6], [34],

[35], [7] to capture biologically meaningful similarities between sequences.

Protein structure classification is inherently a multi-class problem where the label of a protein domain is defined for classifying a particular level in the hierarchy (fold, superfamily etc.). Most early methods approached this problem by dividing the multi-class problem into several one-vs-rest or one-vs-one problems. If all the levels in the hierarchy are considered then the problem becomes a structured-output classification task [36] with interdependencies between the output labels [37], [38]. These methods are able to utilize the correlations between different labels within the hierarchy.

Our current work takes a slightly different approach. Here the focus is on leveraging combined knowledge across different hierarchical classification systems which enables us to learn better classifiers for all the tasks. To our knowledge this approach toward integrated learning and cross hierarchy transfer for protein structure classification has never been explored before.

## 3 METHODS

Given two hierarchical sources of annotated protein domains (e.g., SCOP and CATH), our primary objective is to train a supervised learning model that can accurately classify new instance into classes within these hierarchies. In this study, we use MTL approaches to take advantage of the fact that the classes within the hierarchy have explicit relationships between them and the two hierarchical source databases have implicit relationships across them due to similar objectives. We assume that each of the classes (nodes) within the hierarchies is associated with a binary classification problem, referred to as a task. Combining these tasks during learning should help improve the prediction performance. We assume that there are $T$ tasks across the two hierarchies with independent training instances given by $\{(x_{it}, y_{it}) : i = 1 \dots n_t\}$. For a task $t$, we seek to learn a linear discriminant function per task given by $sign(W_t^T x_{it} + c_t)$.

| Symbol | Meaning |
|---|---|
| $t$ | index over tasks. $\in \{1, 2, \dots, T\}$ |
| $T$ | number of tasks. |
| $x_i\,(x_{it})$ | input features of $i^{th}$ example (of task $t$) $\in \mathbb{R}^D$. |
| $y_i\,(y_{it})$ | class label of $i^{th}$ example (of task t) $\in \{-1, +1\}$. |
| $\mathcal{L}(\cdot)$ | General Loss function. |
| $\mathcal{R}(\cdot)$ | Regularization function. |
| $W_t$ | $\in \mathbb{R}^D$. $t^{th}$ column of $W$ associated with task $t$. |
| $W^l$ | $\in \mathbb{R}^T$. $l^{th}$ row of $W$ associated with input feature $l$. |
| $c_t$ | bias parameter (of task $t$) $\in \mathbb{R}$ |
| $W$ | $\in R^{D \times T}$. defined as $[W_1, \dots, W_T]$ |
| $f\,(x_{it})$ | $\equiv W_t^T x_{it} + c_t$ |
| $n_t$ | number of training examples of task $t$. |
| $\|W^l\|_2$ | $l_2$-norm of vector $W^l$. |
| $\|W\|_{2,1}$ | $\sum_{t=1}^{T} \|W^l\|_2$ |
| $\|W\|_{2,2}$ | $\sqrt{trace\,(W^T W)}$ |
| $rank(W)$ | The number of linearly independent rows or columns. |
| $\|W\|_*$ | Trace norm of $W$ = sum of singular values of W. |

Table 1: Notation

We use the logistic loss function given by,

$$\mathcal{L}\,(x, y; \theta = (w, c)) = \log\left[1 + \exp^{-y*\left(w^T x + c\right)}\right] \quad (3)$$

The regularized objective is minimized with respect to the classification constraints given by (4).

$$y_{it}\left(W_t^T x_{it} + c_t\right) \geq 0 \qquad \forall t \in \{1, \dots, T\}, \forall i \in n_t \quad (4)$$

The best loss function for binary classification problems is the natural 0-1 loss function. It is, however, non-convex and non-smooth which makes it intractable for optimization problems. Logistic loss is a smooth, convex loss function which is suited for classification. Other loss functions, such as hinge-loss might be better in practice [39] but may require specialized solutions. All notations used in this paper are described in Table 1. Regularization acts upon the weight parameters $W$ only, and the bias parameters $c$ are typically not regularized and hence not included in the joint regularization. In the following section we provide an overview of the MTL methods used in this work.

### 3.1 MTL Methods

#### 3.1.1 Sparse learning through joint feature selection

It is well known that the $l_1$-norm regularization [40] encourages sparsity in STL models. This idea has been extended to multi-task learning [21], [23], [14] to select sparse features across all the tasks. In these models, it is assumed that across all the tasks only a subset of the feature space is critical for classification. The objective function for sparse learning in MTL can be written as,

$$\min_W \sum_{t=1}^{T} \sum_{i=1}^{n_t} \mathcal{L}\,(W_t, x_{it}, y_{it}) + \lambda \|W\|_{2,1} \quad (5)$$

where $\|W\|_{2,1} \equiv \sum_{t=1}^{T} \|W^l\|_2$ and $W^l \in \mathbb{R}^D$ denotes the $l^{th}$ row of $W$ associated with the input feature $l$. The $l_{2,1}$-norm here is essentially the $l_1$-norm of the vector of $l_2$-norm over the weights associated with a particular input dimension. However, the above optimization problem is difficult to solve due to the non-smoothness of $l_{2,1}$-norm. Obozinski and Taskar [21] proposed a solution based on block-wise boosting scheme, and Liu et al. [14] proposed two equivalent formulations for the problem that can be efficiently solved. Argyriou et al. [23] formulated the problem of learning sparse linear features as a convex optimization problem. In our current work we have used the sparse feature selection method described by Argyriou et al. [23]. However, the presence of sparse feature space assumed by this model is too strong for the application studied in this paper and consequently does not perform well. We refer to this method as *Sparse* MTL.

#### 3.1.2 Graph Regularization

As discussed above, many MTL formulations assume uniform correlations between all the tasks. Evgeniou and Pontil [19] extended SVM to MTL by augmenting the SVM objective with an additional term to control the inter-task regularization. In a more general setting, the objective function can be written as,

$$\min_{W} \sum_{t=1}^{T}\sum_{i=1}^{n_t} \mathcal{L}\left(W_t, x_{it}, y_{it}\right) + \lambda_1 \sum_{t=1}^{T} \|W_t\|_2^2 + \lambda_2 \sum_{t=1}^{T} \left\| W_t - \frac{1}{T}\sum_{s=1}^{T} W_s \right\|_2^2 \quad (6)$$

The first regularization term controls the magnitude of the parameters for each task and the second regularization term imposes the constraint that the parameters for each task should be close to the average parameters of all the tasks. This assumption is too strong in many cases where some tasks may be more closely related than others.

When relationships between tasks are known a-priori, inter-task regularization can be controlled such that only the related tasks influence each other. Graph-based MTL formulations [15], [25], [41] are able to exploit such relationships. The relationships between different tasks can be encoded as a set of edges $\mathcal{E} = (e_1, e_2, \ldots, e_m)$, where each edge $e_i \equiv (p, q)$ connects a pair of tasks indexed by $p$ and $q$. The graph regularized MTL utilizes the edge relationships in the following manner,

$$\min_{W} \sum_{t=1}^{T}\sum_{i=1}^{n_t} \mathcal{L}\left(W_t, x_{it}, y_{it}\right) + \lambda_1 \sum_{t=1}^{T} \|W_t\|_2^2 + \lambda_2 \sum_{(p,q)\in\mathcal{E}} \|W_p - W_q\|_2^2 \quad (7)$$

Unlike the previous formulation in (6), the regularization term in (7) minimizes the difference between all pairs of related tasks only and does not bias the learned parameters towards the average of all tasks. Therefore, only the parameter vectors of tasks connected by an edge are forced to be similar to each other.

#### 3.1.3 Rank minimization using Trace norm

The problem of rank minimization arises in many optimization problems in machine learning, image compression and automatic control. In the context of multi-task learning minimizing the rank of $W$ forces it to share a low-dimensional subspace, therefore inducing correlations between the tasks. The objective function with rank minimization as regularization can be written as,

$$\min_{W} \sum_{t=1}^{T}\sum_{i=1}^{n_t} \mathcal{L}\left(W_t, x_{it}, y_{it}\right) + \lambda rank(W) \quad (8)$$

However, the rank minimization problem is known to be NP-hard [42]. Rank minimization objective can be approximated by using the trace norm (nuclear norm) which gives a convex solution [43]. The regularized objective function with trace norm can therefore be written as,

$$\min_{W} \sum_{t=1}^{T}\sum_{i=1}^{n_t} \mathcal{L}\left(W_t, x_{it}, y_{it}\right) + \lambda \|W\|_* \quad (9)$$

This formulation has been extensively studied in the context of multi-task learning [44], [45], [46]. In this paper we have used the solution described by Ji et al. [45].

In all the models described above, an $l_2$-norm is added to the regularization objective however it is only explicitly represented for Graph formulation. The $l_2$-norm essentially controls the magnitude of the weight vectors.

We used the publicly available MALSAR toolkit [47] to implement all methods discussed above. The optimization algorithms within MALSAR are implemented using the accelerated gradient methods [48]. This optimization technique is shown to be faster than traditional gradient descent methods, and involves finding a proximal operator for a non-smooth regularization term [47].

### 3.2 Extracting task relationships

Determination of domains in a protein is a non-trivial process, and SCOP and CATH differ significantly in defining domains for multi-domain proteins. The inconsistencies in these two schemes may lead to errors in the models learnt from these classification systems [28]. To reduce the inconsistencies, we extracted domains which are defined similarly in both the schemes, which we call *consistent domains*. As very few domains definitions agree exactly in both SCOP and CATH, we used an approximate similarity threshold of 0.8 to define similar domains. The domains derived from the same protein sequence are considered to be similar if $\geq 80\%$ of the amino-acids in their sequences overlap.

Identifying whether a sequence belongs to each of the node in SCOP and CATH can be considered a learning task. For our purpose, the tasks are created from superfamily (SCOP), family (SCOP), topology (CATH), and homologous superfamily (CATH). We also indirectly use the fold (SCOP) and architecture (CATH) levels in defining the sibling relationships discussed below.

In order to utilize the relationships between different tasks in graph regularized formulation, we define edges between different nodes in SCOP and CATH. Edges can be categorized based on whether they span only one hierarchy or both the hierarchies. We refer to the edges that connect nodes in SCOP and CATH as *cross links*. The links connecting SCOP nodes to SCOP nodes or CATH nodes to CATH nodes, are referred to as *within links*. The *within links* can be further categorized based on the type of nodes they connect. Below, we list the three types of links studied in this paper.

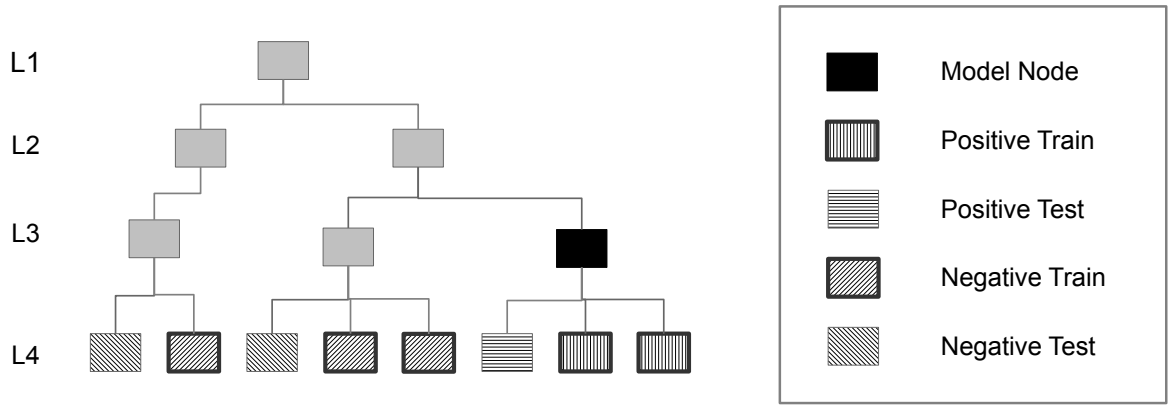



Figure 1: Training and test partitions

Parent Links: These edges join a child node to its parent node *i.e.,* superfamily-family (SCOP), topology-homologous superfamily (CATH). These are explicitly defined by the hierarchical structure of SCOP and CATH.

Sibling Links: These edges join siblings *i.e.,* links between a node and other children of its parent. An edge between a node and itself is not defined. These edges are defined between superfamily-superfamily (SCOP), topology-topology (CATH), family-family (SCOP), and homologous superfamily-homologous superfamily (CATH).

Cross Links: These edges span both the hierarchies and connect nodes at the same level for both the hierarchies *i.e.,* superfamily (SCOP) - topology (CATH) and family (SCOP) - homologous superfamily (CATH). Edges are not defined between superfamily (SCOP - homologous superfamily (CATH) and family (SCOP) - topology (CATH). These edges are derived using *consistent domains.* If a domain is classified by two nodes we add an edge between them. Specifically, we add an edge between a SCOP node $N_s$ and CATH node $N_c$ , if there are are protein domains $p \in N_s$ and $q \in N_c$ such that $p$ and $q$ are similar as described previously.

### 3.3 Dataset

In this study we focus on the problem of building classifiers for superfamily level in SCOP and Topology level in CATH. We used the sequences from SCOP 1.75 and CATH 3.4 with mutual sequence identity of $\leq 95\%$. As described in sec. 3.2 we only use the domains which are defined similarly in both these hierarchies. Table 2 provides summary statistics of the datasets. We use relatively simple string features in this work and note that more sophisticated sequence representations and kernels [3], [4], [49], [34] can be used, however, that is not the primary focus of the current work.

For the purpose of this work protein sequences are represented by varying length sequences of 20 amino acids. The protein sequences were represented using spectrum kernel [5] using contiguous words of length 3, also called 3-mers. Hence, each protein sequence is represented using a feature vector of $20^3 = 8000$ features. For each 3-mer feature the number of times it occurs in a given protein sequence is the feature value for that sequence.

## 4 Experimental Evaluations

### 4.1 Models Evaluated

To test the benefit of combining various tasks, we evaluated different MTL formulations using the following partitions of the datasets.

#### 4.1.1 Single Task Learning (STL)

Each task is trained separately. We performed STL experiments using the elastic net penalty which uses both $l_1$ and $l_2$ norms in regularization.

#### 4.1.2 Single Hierarchy MTL (SHMTL)

Only the tasks belonging to the same hierarchy are learned together. These experiments are conducted for Sparse, Graph and Trace regularization. Graph regularization for SHMTL uses the parent links and sibling links.

#### 4.1.3 Multiple Hierarchy MTL (MHMTL)

We grouped the tasks belonging to both the hierarchies and trained them together. These experiments were

| SCOP | | CATH | |
|---|---|---|---|
| instances | 5587 | instances | 9642 |
| families | 318 | homologous superfamilies | 259 |
| superfamilies | 102 | topologies | 51 |
| folds | 82 | architectures | 17 |
| classes | 4 | classes | 3 |

Table 2: Dataset Statistics

conducted for Sparse, Graph and Trace regularization. Graph regularization for MHMTL uses the cross links between the two hierarchies in addition to the sibling and parent links.

### 4.2 Training and Test set partitioning

We assign an L4 node, i.e. the corresponding instances, to either the training or the test partition. In other words, the instances belonging to a particular L4 node do not appear in both the training and test sets. This strategy is visually represented in Fig. 1 for the node marked as the *Model Node*. The child nodes of the model node are randomly assigned to positive test and positive train. The remaining L4 nodes are randomly assigned to negative train and negative test. This partitioning strategy prevents very similar instances belonging to the same SCOP family or CATH homologous superfamily from appearing in both the training and test partitions. Using this strategy, we generated 30 random training and test partitions of the complete dataset.

### 4.3 Model Selection

To determine the best parameters, we performed extensive grid search on a smaller dataset to study the influence of parameters. This enabled us to narrow down the range of parameters to test with the larger datasets. We then performed a grid search on a single random training and test partition with the parameters approximately in the range $\left[10^k,\right.$ where $k \in \{-1,-2,-3,-4,-5\}\left.\right]$. With the best parameters selected for each method, we repeated the experiments with 30 random training and test partitions.

Further, to test whether there was any benefit of adding the family (SCOP) and homologous superfamily (CATH) levels, we removed the tasks belonging to these levels and tested them using the same setup. We will call this partition L3-only, which stands for level 3 in the hierarchies as opposed to L3+L4 which includes both level 3 and level 4 in the hierarchies.

### 4.4 Metrics

Accuracy of classification is not a good measure of performance when the class distribution is highly unbalanced. In these cases Area Under Curve (AUC) of Receiver Operating Characteristics curve (ROC) [50] gives a better indication of classification performance. As stated above we repeated the training and testing with 30 randomly derived partitions. For each task we collected the AUC scores from each run and calculated their mean and standard error. The AUC scores averaged across all the tasks, separated by the two hierarchies, are reported in Table 3. The standard error value for each score is reported inside parenthesis, next to the AUC score.

## 5 Results and Discussion

### 5.1 Performance of MHMTL

We performed a comprehensive set of experiments evaluating the performance of different MTL approaches under various settings. Table 3 shows the average AUC scores for the different MTL models evaluated across the SCOP and CATH nodes, and for the L3+L4 and L3-only datasets. Detailed results with AUC scores for each task have been made available on the supplementary website [1]. The following observations can be made from the results.

Sparse MTL based on $l_{2,1}$-norm regularization does not improve the results in SHMTL and MHMTL compared to STL. The sparse formulation used here, attempts to extract sparse features across all the tasks. It assumes a uniform relationship between all the tasks. However, it is unlikely that any subset for the features derived from the amino-acid sequences are more important for the classification. During the parameter optimization with $l_{2,1}$ regularization we found that better results were produced when we lowered the value of the parameter which controls sparsity. Therefore, the best performing models are not sparse. However, these experiments act as a control set to verify that no bias is being introduced by the selection of tasks within the datasets.

Fig. 2 compares the AUC scores of SHMTL and STL methods. As can be seen from Fig. 2 and Table 3, SHMTL with the Trace and Graph based regularization performs better than the STL. We notice that improvement for CATH nodes is significantly greater ($\approx 2-3\%$) than that for SCOP nodes ($\leq 1\%$) which can also be noticed in Fig. 2. We also see that the Trace regularization performs slightly better in comparison to the Graph regularization for SHMTL with the L3+L4 dataset. This difference is also evident from Fig. 4(a) which compares the improvement in AUC scores in SHMTL over STL for Graph and Trace regularization. We also notice that for some CATH

1. http://www.cs.gmu.edu/~mlbio/supplements/mhmtl/

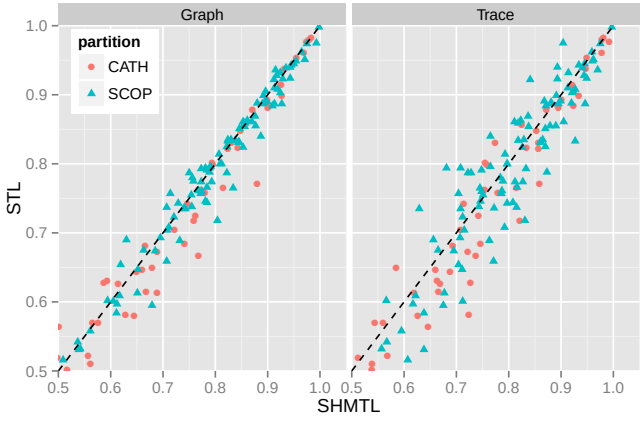



Figure 2: AUC SHMTL vs STL for L3 tasks

| | | L3 + L4 | | | | | | L3-only | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | STL | | SHMTL | | MHMTL | | STL | | SHMTL | | MHMTL | |
| SCOP | Sparse | 0.791 | (0.009) | 0.792 | (0.009) | 0.791 | (0.009) | 0.792 | (0.009) | 0.792 | (0.009) | 0.789 | (0.010) |
| | Graph | | | 0.797 | (0.009) | 0.844 | (0.012) | | | 0.797 | (0.008) | 0.844 | (0.012) |
| | Trace | | | 0.801 | (0.011) | 0.853 | (0.015) | | | 0.792 | (0.013) | 0.840 | (0.014) |
| CATH | Sparse | 0.725 | (0.010) | 0.723 | (0.011) | 0.725 | (0.010) | 0.725 | (0.010) | 0.722 | (0.011) | 0.723 | (0.011) |
| | Graph | | | 0.741 | (0.012) | 0.775 | (0.012) | | | 0.743 | (0.012) | 0.777 | (0.012) |
| | Trace | | | 0.745 | (0.012) | 0.765 | (0.011) | | | 0.746 | (0.012) | 0.766 | (0.012) |

Table 3: Average AUC scores

nodes the improvement is significantly greater for Trace than for Graph.

We see an improvement of approximately $5\%$ in the AUC scores for SCOP using both the Trace and Graph regularizations for the MHMTL in comparison to SHMTL. For the CATH dataset we see an improvement of $2-3\%$ (refer to L3+L4 dataset results in Table 3). The percentage improvement observed for MHMTL in comparison to STL is $5-6\%$ and $4-5\%$ for the SCOP and CATH data, respectively. Fig. 3 compares the AUC scores for MHMTL and SHMTL for both Trace and Graph formulations. This suggests that there is an advantage in combined MTL over both the hierarchies.

Fig. 4(b) compares the AUC improvement from MHMTL over SHMTL for the Graph and Trace formulations. There is a linear correlation in the improvements in AUC scores from Graph and Trace formulations in MHMTL when compared to SHMTL, *i.e.,* the same L3 nodes are improving and by roughly the same amount using two independent methods. Therefore, it is very unlikely that the improvement is random. Interestingly, as can be seen in Fig. 4(b) CATH nodes benefit more from Graph than from Trace. Whereas in the AUC improvement of SHMTL over STL case shown in Fig. 4(a), Trace formulation helps CATH nodes more than Graph formulation.

### 5.2 L3-only versus L3+L4

When we remove the L4 nodes from training the results are not significantly affected. Fig. 5 compares the improvement in AUC scores across the L3-only and L3+L4 datasets (difference of MHMTL over SHMTL given by "MHMTL - SHMTL") for the Graph and Trace formulations. The performance of Graph is very similar in both the cases, but Trace performed slightly better, especially for SCOP nodes with L4 nodes included in training. Removing L4 nodes significantly reduces the training times (reported in Table 4), therefore the cost to benefit ratio for including L4 nodes is very high. Runtime values reported in Table 4 were obtained using Intel Xeon 2.40GHz dual core processors running 64 bit Linux operating system.

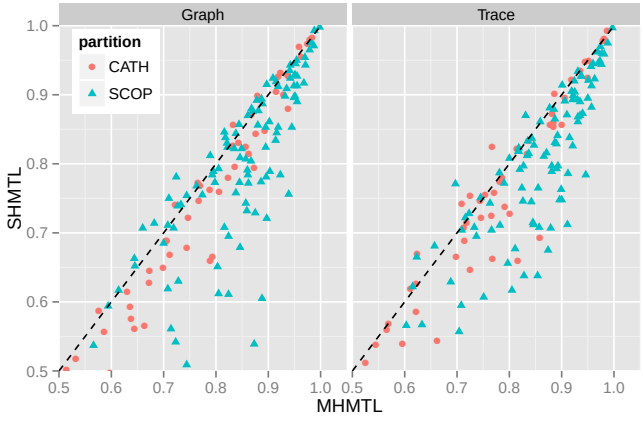



Figure 3: AUC MHMTL vs SHMTL for L3 tasks

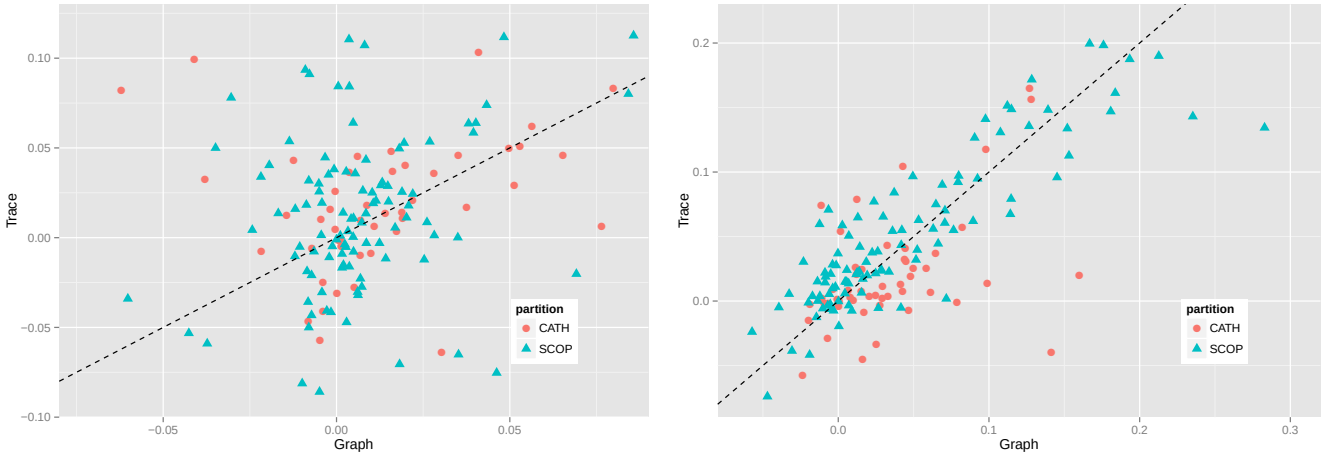
### 5.3 Graph regularized MTL using Inner node mappings

In the previous set of experiments using Graph MHMTL, we created the cross links between the same level of the hierarchy. However, the nodes at different levels between SCOP and CATH are not necessarily equivalent to each other . In their comparison of SCOP and CATH, Casaba et al. [28] consider the following levels to be mappable to each other - SCOP superfamily/family map to CATH homologous superfamily and SCOP fold map to CATH topology and SCOP class to CATH class. Their evaluation reveals that this mapping of levels is not strict. To evaluate the performance of Graph MHMTL with respect to Graph SHMTL when the cross links between SCOP and CATH are defined by node mappings based on shared protein domains we created another dataset with tasks corresponding to fold, superfamily and family nodes from SCOP and topology and homologous superfamily nodes from CATH.

To map the nodes, we computed an F1-score metric between pair of nodes similar to the mapping described by Casaba et al. [28]. This mapping is not symmetric, hence, we consider two maps, one from SCOP$\rightarrow$ CATH and similarly from CATH $\rightarrow$ SCOP. To map the nodes we use the set of similar domains. Let $S_1$ represent the set of domains belonging a SCOP node $N_1$ and $S_2$ be the set of domains belonging to a CATH node $N_2$. To compute the F1-score of mapping from $N_1 \rightarrow N_2$, we first determine the set of SCOP domains similar to the CATH domains in $S_2$, and denote this set by $\bar{S}_2$.

The F1-score of mapping is calculated as

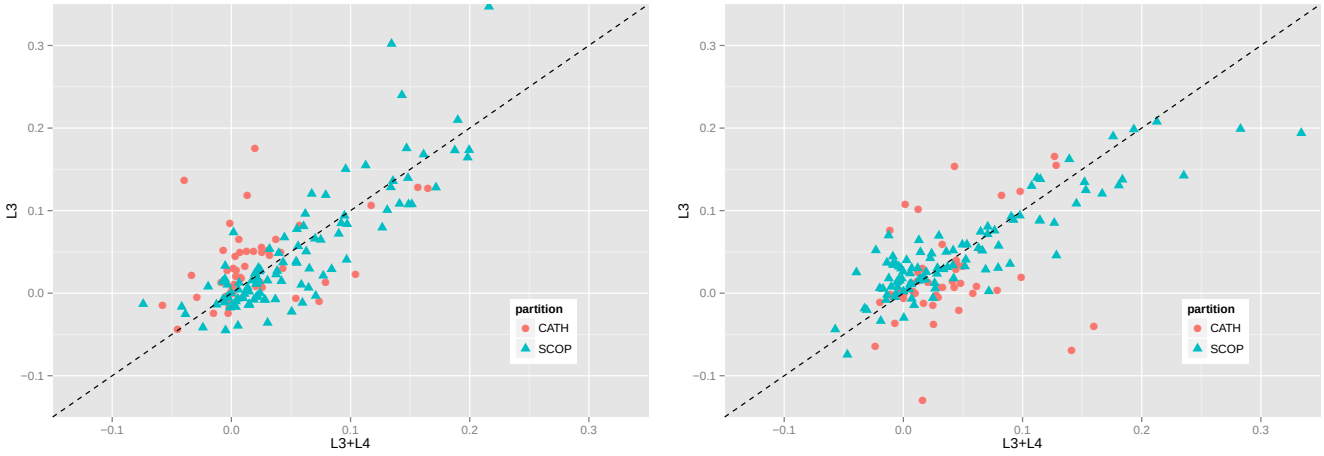
$$F1 - score = \frac{2 * sensitivity * specificity}{sensitivity + specificity}$$




(a) SHMTL - STL

(b) MHMTL - SHMTL

Figure 4: AUC improvement for Graph vs Trace




(a) Graph

(b) Trace

Figure 5: AUC improvement (MHMTL - SHMTL) for different set of nodes in training L3 vs L3+L4

Where $sensitivity = \left|S_1 \cap \bar{S}_2\right| / \left|S_1\right|$ and $specificity = \left|S_1 \cap \bar{S}_2\right| / \left|S_1 \cup \bar{S}_2\right|$. The mapping of nodes from CATH to SCOP is similarly calculated. Finally, we created an edge between the nodes where the F1-score of a pair of nodes in the mappings in both the directions is $\geq 0.5$. We extracted a total of 154 such cross links.

The experiments with this dataset did not show any statistically significant improvement compared to the previous dataset where the edges were defined between a pair of nodes if they shared even a single domain. The detailed results are available on the supplementary website.

### 5.4 Graph-based Link Analysis

In the context of Graph based MTL, we also studied the effect of sibling relationships on AUC improvement of SHMTL over STL (SHMTL - STL) and the effect of number of cross links on AUC improvement of MHMTL over STL (MHMTL - STL). Fig. 6 and Fig. 7 depict the improvements with respect to the number of links. In Fig. 6-Graph, which shows the improvements for Graph formulation, we see a uniform improvement in all the nodes with non-zero cross links. CATH seems to benefit more from the presence of cross edge and Graph formulation which can also be seen from Fig. 4(b). This suggests that Graph formulation is able to leverage the relatedness between tasks across the hierarchies. The Trace formulation does not take into account these relationship, Fig. 6-Trace does not seem to suggest this pattern.

Fig. 7 shows the improvement in AUC for SHMTL over STL (SHMTL - STL) with respect to the number of siblings. Both Fig. 6 and Fig. 7 suggest a positive inductive bias being introduced by related tasks and Graph based formulation is doing slightly better when the relationships between tasks are known.

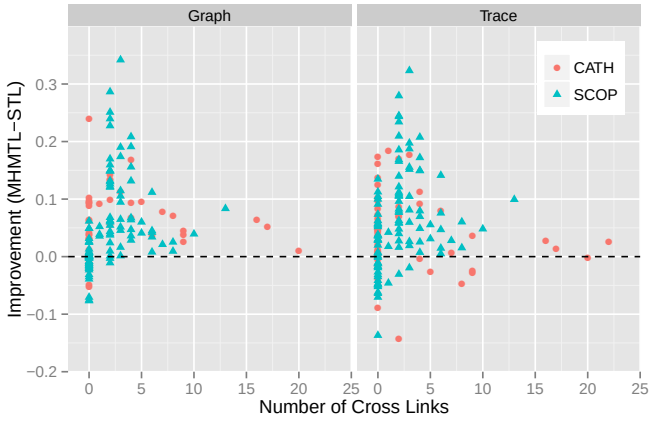



Figure 6: AUC improvement (MHMTL - STL) with number of Cross Links

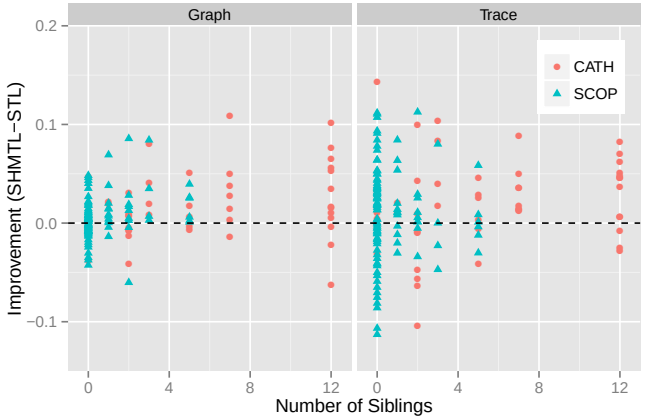



Figure 7: AUC improvement (SHMTL - STL) with number of siblings

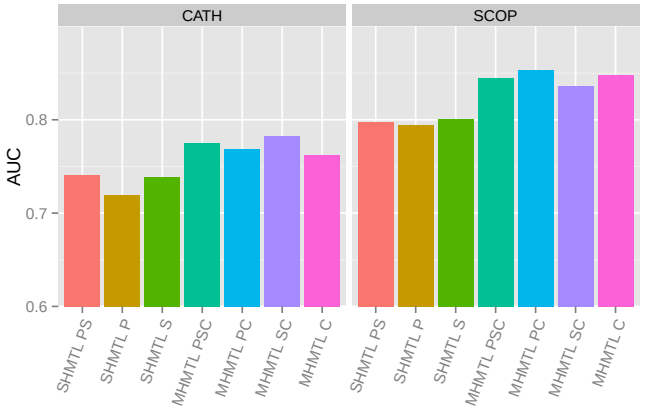



Figure 8: Average AUC for SCOP and CATH with different links in training set for Graph MTL

| | | STL | SHMTL | MHMTL |
|---|---|---|---|---|
| L3 | Sparse | 873 | 214 | 227 |
| | Graph | - | 371 | 895 |
| | Trace | - | 1,233 | 1,721 |
| L3+L4 | Sparse | 876 | 536 | 656 |
| | Graph | | 1,159 | 3,463 |
| | Trace | | 5,498 | 7,049 |

Table 4: Execution Times

To study the effect of different relationships within the Graph-based formulation for SHMTL and MHMTL during the training phase, we repeated the experiments with various combinations of link types. The bar chart of average AUC scores are show in Fig. 8. P, S, and C denote the parent, sibling, and cross links respectively. For example *MHMTL PC* indicates that the MHMTL experiment was conducted only with parent links and sibling links were removed. We observed that for both CATH and SCOP, the sibling links helped more than the parent links. However, for MHMTL the effects on SCOP and CATH due to the removal of sibling or parent links was different. SCOP performance improved with removal of sibling links and decreased with removal of parent links while the effect on CATH was the opposite.

### 5.5 Run Time Analysis

Table 4 reports the runtimes in seconds for the STL, SHMTL and MHMTL with different regularization penalties and for the L3-only and L3+L4 dataset. We can observe that the Graph-MHMTL takes longer in comparison to Graph-SHMTL. This is because there are more relationships (edges) that are added for the MHMTL method in comparison to the SHMTL method. With the addition of more tasks in L3+L4 dataset in comparison to the L3-only dataset, we notice a 3-5 fold increase in run time. Our analysis of AUC performance suggests that use of L4 nodes may not be necessary given the increase in run time.

## 6 Conclusions

In this work, we have shown the benefit of combining two hierarchical classification schemes to achieve better classification performance using the protein structure classification problem. Due to the nature of protein structure classification, the Sparse feature learning MTL formulation could not achieve any performance improvement over the baseline STL method. However, both the Graph and Trace MTL methods were able to learn better classifiers in both single hierarchy settings as well as multiple hierarchy settings. In our analysis we found that introducing the Family (SCOP) and Homologous Superfamily (CATH) tasks did not noticeably improve the classification, but incurred some increase in runtime. For Graph based MTL formulation, which explicitly utilizes task relationships, we also examined the effect of various relationships on the classification performance.

## Acknowledgments

This work was supported by NSF IIS-0905117.

## REFERENCES


[1] A.G. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536–540, 1995.

[2] C.A. Orengo, AD Michie, S. Jones, D.T. Jones, MB Swindells, and J.M. Thornton. CATH-a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

[3] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1-2):95–114, 2000.

[4] Hiroto Saigo, Jean-Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–9, 2004.

[5] C. Leslie, E. Eskin, and W.S. Noble. The spectrum kernel: A string kernel for svm protein classification. In *Proceedings of the pacific symposium on biocomputing*, volume 7, pages 566–575. Hawaii, USA., 2002.

[6] H. Rangwala and G. Karypis. Profile-based direct kernels for remote homology detection and fold recognition. *Bioinformatics*, 21(23):4239–4247, 2005.

[7] J. Weston, C. Leslie, E. Ie, D. Zhou, A. Elisseeff, and W.S. Noble. Semi-supervised protein classification using cluster kernels. *Bioinformatics*, 21(15):3241, 2005.

[8] A. Charuvaka and H. Rangwala. Multi-task learning for classifying proteins using dual hierarchies. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE, 2012.

[9] J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12(1):149–198, 2000.

[10] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[11] Sebastian Thrun. Is learning the $n$-th thing any easier than learning the first? In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 640–646. The MIT Press, 1996.

[12] S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. *Learning Theory and Kernel Machines*, pages 567–580, 2003.

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

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

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

[16] S. Thrun and J. O'Sullivan. Clustering learning tasks and the selective cross-task transfer of knowledge. *Learning to learn*, pages 181–209, 1998.

[17] Edwin Bonilla, Kian Ming Chai, and Chris Williams. Multi-task gaussian process prediction. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 153–160. MIT Press, Cambridge, MA, 2008.

[18] Yu Zhang and Dit-Yan Yeung. A convex formulation for learning task relationships in multi-task learning. In *Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 733–442, Corvallis, Oregon, 2010. AUAI Press.

[19] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Conference on Knowledge Discovery in Data: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117, 2004.

[20] T. Jebara. Multitask sparsity via maximum entropy discrimination. *The Journal of Machine Learning Research*, 12:75–110, 2011.

[21] G. Obozinski and B. Taskar. Multi-task feature selection. In *The workshop of structural Knowledge Transfer for Machine Learning in the 23rd International Conference on Machine Learning (ICML*, 2006.

[22] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[23] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, 19:41, 2007.

[24] T. Jebara. Multi-task feature and kernel selection for svms. In *Proceedings of the twenty-first international conference on Machine learning*, page 55. ACM, 2004.

[25] T. Kato, H. Kashima, M. Sugiyama, and K. Asai. Multi-task learning via conic programming. *Advances in Neural Information Processing Systems*, 20:737–744, 2008.

[26] Laurent Jacob, Francis Bach, and Jean-Philippe Vert. Clustered multi-task learning: A convex formulation. *CoRR*, abs/0809.2085, 2008.

[27] Jiayu Zhou, Jianhui Chen, and Jieping Ye. Clustered multi-task learning via alternating structure optimization. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 702–710, 2011.

[28] G. Csaba, F. Birzele, and R. Zimmer. Systematic comparison of SCOP and CATH: a new gold standard for protein structure analysis. *BMC Structural Biology*, 9(1):23, 2009.

[29] R. Day, D.A.C. Beck, R.S. Armen, and V. Daggett. A consensus view of fold space: Combining SCOP, CATH, and the Dali Domain Dictionary. *Protein Science*, 12(10):2150–2160, 2003.

[30] C. Hadley and D.T. Jones. A systematic comparison of protein structure classifications: Scop, cath and fssp. *Structure*, 7(9):1099–1112, 1999.

[31] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[32] SF Altschul, TL Madden, AA Schaffer, J. Zhang, Z. Zhang, W. Miller, and DJ Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389, 1997.

[33] C. Leslie and R. Kuang. Fast kernels for inexact string matching. *Learning Theory and Kernel Machines*, pages 114–128, 2003.

[34] R. Kuang, IE EUGENE, K. Wang, K. Wang, M. Siddiqi, Y. Freund, and C. Leslie. Profile-based string kernels for remote homology detection and motif extraction. *Journal of bioinformatics and computational biology*, 3(03):527–550, 2005.

[35] Pavel P. Kuksa, Pai-Hsi Huang, and Vladimir Pavlovic. Scalable algorithms for string kernels with inexact matching. In *Advances in Neural Information Processing Systems 20*, pages 881–888, 2008.

[36] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(2):1453, 2006.

[37] H. Rangwala and G. Karypis. Building multiclass classifiers for remote homology detection and fold recognition. *BMC bioinformatics*, 7(1):455, 2006.

[38] I. Melvin, E. Ie, J. Weston, W.S. Noble, and C. Leslie. Multi-class protein classification using adaptive codes. *Journal of Machine Learning Research*, 8(1557-1581):6, 2007.

[39] L. Rosasco, E.D. Vito, A. Caponnetto, M. Piana, and A. Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–1076, 2004.

[40] D.L. Donoho. For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. *Communications on pure and applied mathematics*, 59(6):797–829, 2006.

[41] T. Kato, H. Kashima, M. Sugiyama, and K. Asai. Conic programming for multitask learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(7):957–968, 2010.

[42] S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge Univ Pr, 2004.

[43] M. Fazel, H. Hindi, and S.P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference, 2001. Proceedings of the 2001*, volume 6, pages 4734–4739. IEEE, 2001.

[44] Jacob Abernethy, Francis Bach, Theodoros Evgeniou, and Jean-Philippe Vert. Low-rank matrix factorization with attributes. *CoRR*, abs/cs/0611124, 2006.

[45] S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 457–464. ACM, 2009.

[46] T.K. Pong, P. Tseng, S. Ji, and J. Ye. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization*, 2009.

[47] J.Zhou, J.Chen, and J. Ye. *MALSAR: Multi-tAsk Learning via StructurAl Regularization*. Arizona State University, 2011.

[48] Y. NESTEROV. Gradient methods for minimizing composite objective function. Technical report, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2007.
[49] C.S. Leslie, E. Eskin, A. Cohen, J. Weston, and W.S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.
[50] M.H. Zweig and G. Campbell. Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine. *Clinical chemistry*, 39(4):561–577, 1993.

**Anveshi Charuvaka** is a Ph.D. student in the Department of Computer Science, George Mason University since 2009. He received his Bacherlor of Technology degree from Acharya Nagarjuna University from India in 2006. His research interests include Multi-task learning, Structured output prediction and their applications in Bio-informatics.

**Huzefa Rangwala** is an Assistant Professor in the Department of Computer Science and in the Department of Bioengineering at George Mason University, VA, USA. He received a B.E. degree in Computer Engineering from Mumbai University (V.J.T.I), India in 2003, and a Ph.D. degree in Computer Science from the University of Minnesota, Minneapolis, USA in 2008. His research interests include data mining, bioinformatics and high performance computing. He has published extensively in premier data mining and bioinformatics conferences and journals. His research is sponsored by NSF, USDA, DARPA and NIH.