Learning Multiclassifiers with Predictive Features that Vary with Data Distribution

Xuan-Hong Dang[†1], Omid Askarisichani[‡], Ambuj K. Singh[‡]
[†]*IBM T.J. Watson Research Center, NY 10598*
[‡]*Department of Computer Science, University of California Santa Barbara, CA 93106*
xuan-hong.dang@ibm.com, {omid55, ambuj}@cs.ucsb.edu

***Abstract*—In many real-world big data applications, the data distribution is not homogeneous over entire data, but instead varies across groups/clusters of data samples. Although a model's predictive performance remains vital, there is also a need to learn succinct sets of features that evolve and capture smooth variations in data distribution. These small sets of features not only lead to high prediction accuracy, but also discover the important underlying processes. We investigate this challenging problem by developing a novel multi-task learning paradigm that trains multiple support vector machine (SVM) classifiers over a set of related data clusters, and directly imposes smoothness constraints on adjacent classifiers. We show that such patterns can be effectively learned in the dual form of the classical SVM, and further show that a parsimonious solution can be achieved in the primal form. Although a solution can be effectively optimized via gradient descent, the technical development is not straightforward, requiring a relaxation over the loss function of SVMs. We demonstrate the performance of our algorithm in two practical application domains: team performance and road traffic prediction. Empirical results show our model not only achieves competitive prediction accuracy, but its discovered patterns truly capture and give intuition about the variation in the data distribution across multiple data clusters.**

***Index Terms*—Big Data; Feature Extraction; Supervised Learning; Multi-task learning**

I. Introduction

With increasing advances in hardware and software technologies for data collection and management, practitioners in big data mining are now confronted with more challenges to discover actionable patterns from their collected data samples. In many practical domains, the collected data can be naturally grouped into clusters that reflect smooth changes in their heterogeneous data distribution. For example, teams along with their performance evaluation in solving tasks can be grouped into clusters based on the difficulty levels of the tasks as well as the experience and skill sets of team members [1]–[3]. Data snapshots captured from a busy road traffic network [4], [5] are often heterogeneous but can be naturally grouped into clusters based on time of day. National census data can be grouped into states or adjacent geographical regions. Although the ultimate goal of training a model that is highly predictive of the class labels (e.g., successful vs. unsuccessful teams, weekend vs. weekday traffic) remains unchanged, it is also desirable to discover smoothly evolving feature sets that capture the variation in the data distribution. Successfully uncovering such patterns not only leads to high prediction accuracy over the class labels of data samples within and across the data groups, but further provides an intuitive explanation and gives insights into mechanisms that govern the observed data. These patterns enhance our understanding of emerging data across numerous application domains spanning psychology, network science, sociology, to biology.

In this paper, we investigate the aforementioned problem of learning succinct sets of features that vary smoothly across clusters of data samples and that can be used as the predictors for the dependent class variable from a given large dataset. The dataset itself is partitioned into multiple related groups/clusters that naturally reflect the variation in the data distribution. This problem is related to the fields of multi-task learning (MTL) [6], transfer learning (TL) [7], and hypothesis transfer learning (HTL) [8]. Generally speaking, the learning objective in MTL is to improve the predictive performance of multiple related tasks simultaneously by exploiting the intrinsic relationships among tasks. Studies can be divided into two categories, those that explicitly maximize the similarity of parameter sharing across multi-tasks [9]–[11], and those that implicitly capture the common structures (e.g. low rank subspaces or attribute subsets) among tasks [12]–[14]. Transfer learning is closely related to MTL, yet focuses more on improving the predictive performance on a target source by transferring knowledge from multiple source tasks. Although our work in this paper can be viewed as falling into the umbrella of multi-task and transfer learning, it differs in three ways: (i) First, unlike both MTL and TL, we study the setting in which there is a clear and natural relationship among groups of data samples upon which we train our multiple classifiers; (ii) Second, we broaden the research scope to further address the more challenging problem of discovering *varying* predictive patterns that account for the variation in data distribution across data clusters, explaining their behaviors behind the observed data, rather than training a less explainable model that solely leads to a high prediction accuracy. (iii) Third, while most existing studies in MTL and TL widely adopt linear models of regression and logistic regression as the core algorithm to build the models on, we explore a more sophisticated technique of SVM in order to better deal with real-world data applications.

Toward our research objectives, we present a new learning

[1]Majority of this work was done at the UC Santa Barbara. Some minor work on experiments and writing was done while the first author was with the Institute of Information Technology, Hanoi, and IBM T.J.Watson Research.

paradigm that trains multiple SVM classifiers from a set of related data clusters, and directly imposes smoothness constraints on the classifiers in order to ensure the discovery of smoothly evolving patterns, represented as a succinct set of selected features. We show that such a challenging objective of learning can be effectively dealt with in the dual form of the conventional SVM technique, and further present that a parsimonious solution can be achieved in the primal form. In the latter case, the L1-norm Lasso condition is applied in order to ensure sparsity by keeping only the most informative and relevant features from the original feature set. Although a solution can be effectively optimized through gradient descent, the technical development is not straightforward, requiring a relaxation over the SVM optimization function. We demonstrate the performance of our proposed models on two important application domains. The first one is to analyze and predict team performance collected from a multi-player online battle area (MOBA) game [15], with the goal of investigating how teams' members interact with each other in order to win a game, and especially to understand how their strategy varies as the teams' members gain more experience and become more skillful. Analyzing this problem enhances our understanding of human collaboration in the virtual world, and has greatly gained attention in the interdisciplinary research of computer science, cognitive psychology and sociology [15], [16]. The second application is from the network science domain with road traffic data [4], [17], in which we want to discover evolving road segments (along with hours) that discriminate weekday from weekend. To summarize, in this work, we make the following contributions:

- We present a new problem in which a large-scale dataset can be naturally divided into multiple related groups of data samples and the task is to train multiple classifiers that not only lead to high prediction accuracy within each data cluster but also discover smoothly changing predictive features that capture the variation in data distribution across clusters.
- We propose a principled learning framework that trains multiple SVM classifiers, each with respect to a data cluster, while modeling the mutual relationship among the data clusters. This ensures that the final model truly captures smooth variation in data distribution across clusters.
- We present a parsimonious solution for the primal form SVM that allows the discovery of a small set of relevant features from each data cluster.
- We empirically evaluate the performance of our developed method against state-of-the-art algorithms. The empirical results show that our developed model is not only competitive in accuracy rates but further discovers the smooth changes of highly relevant features that are human-interpretable.

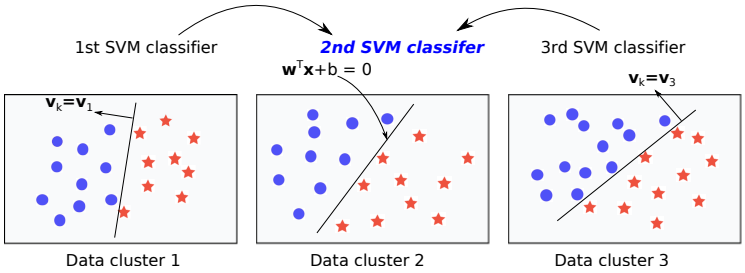

Fig. 1. A simple data set is naturally grouped into three adjacent data clusters. The second SVM classifier is under training whose learned decision boundary is impacted by those learned from the nearby data clusters 1 and 3. We assume there are smooth changes in data distribution across three data clusters. Hence, such patterns in cluster 2 should be captured by the margin vector $\mathbf{w}$ whose non-zero entries reflect the selected features. (Data shown here are in a 2-dimensional space, but they are often characterized in a very high dimensional space as demonstrated later in our experimental results).

II. Problem Setting

Let $\mathcal{DB} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a dataset that consists of n data samples, each $\mathbf{x}_i \in \mathbb{R}^d$. Without loss of generality, we assume that each $\mathbf{x}_i$ belongs to one of 2 classes $\{-1, +1\}$ [1]. We deal with a class of applications whose generated data samples $\mathcal{DB}$ can be naturally partitioned into clusters or groups. It is possible to identify relationships among these data clusters, and given a cluster, one can obtain all its neighboring clusters. For example, data cluster 2 shown in Fig. 1 has relationship with, or is related to both data clusters 1 and 3. Let us denote M as the number of such data clusters, and assume each cluster of data samples is a task upon which we will train a classifier.

Our objective is to learn a model that not only predicts accurately the class labels of data samples within each cluster, but also discovers succinct sets of predictive features that capture and reflect the variation in the data distribution across data clusters. For instance, three classifiers will be trained for a dataset shown in Fig. 1 along with three succinct sets of selected features that smoothly change across three data clusters. We assume the existence of a natural partitioning of the given dataset $\mathcal{DB}$ into data clusters. If such a data partition is not available, one can apply an unsupervised learning algorithm to group similar data samples into clusters prior to training our model.

III. Learning a Set of Smooth Classifiers

We adopt SVM for the classification task, and learn each classification boundary (encoded by the margin vector) for each cluster of data samples. Let us denote $\mathbf{w}$ for the margin vector of the current classifier under training, and $\mathbf{v}_k$ the margin vector learned from a $k-$th classifier that is related to the current classifier (as illustrated in Fig. 1). We learn $\mathbf{w}$ by optimizing the following function, which takes into account the smoothness among related classifiers:

$$\arg\min_{\mathbf{w},b,\xi_i} \frac{1}{4}\|\mathbf{w}\|_2^2 + \frac{1}{4K}\sum_{k=1}^{K}\|\mathbf{w}-\mathbf{v}_k\|_2^2 + C\sum_{i=1}^{n_c}\xi_i \qquad (1)$$
$$\text{such that} \quad y_i(b+\mathbf{w}^T\mathbf{x}_i) \geq 1-\xi_i, \xi_i \geq 0, \forall i$$

[1]For multiple classes, one can adopt the one-vs-rest strategy [18].

where ξ_i's are slack variables, $C > 0$ is the trade-off constant, n_c is the number of data samples belonging to the current cluster, and K is the number of neighboring clusters related to the current cluster. For instance, $K = 2$ for data cluster 2 illustrated in Fig. 1 due to its adjacency to two other clusters. $K = 1$ for both data cluster 1 and 3 since they only have one adjacent cluster [2]. This convex constrained minimization can be cast in the Lagrangian form:

$$L = \frac{1}{4}\|\mathbf{w}\|_2^2 + \frac{1}{4K}\sum_{k=1}^{K}\|\mathbf{w}-\mathbf{v}_k\|_2^2 + C\sum_{i=1}^{n_c}\xi_i - \sum_{i=1}^{n_c}\alpha_i(y_i(b+\mathbf{w}^T\mathbf{x}_i)-1+\xi_i) - \sum_i \beta_i\xi_i \quad (2)$$

where $\alpha_i, \beta_i \geq 0$ are collectively called the Lagrange multipliers or dual variables, and viewed as the cost associated with the constraint violation. The Lagrange dual associated with this function is formed by taking its partial derivative with respect to $\mathbf{w}, b, \xi_i$, and equating them to zero. It is straightforward to show that:

$$\frac{\partial L}{\partial \mathbf{w}} = \frac{1}{2}\mathbf{w} + \frac{1}{2K}\sum_{k=1}^{K}(\mathbf{w}-\mathbf{v}_k) - \sum_{i=1}^{n_c}\alpha_i y_i \mathbf{x}_i = 0 \quad \text{or}$$
$$\mathbf{w} = \frac{1}{2K}\sum_k \mathbf{v}_k + \sum_{i=1}^{n_c}\alpha_i y_i \mathbf{x}_i \quad (3)$$

Likewise, $\partial L/\partial b = \sum_{i=1}^{n_c}\alpha_i y_i = 0$, and:

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \quad \text{or} \quad \beta_i = C - \alpha_i$$

Since $C > 0$ and $\xi_i \geq 0$, it is easy to see that $0 \leq \alpha_i \leq C$. Plugging these results into Eq. (2) leads to:

$$L = \frac{1}{4}\|\mathbf{w}\|_2^2 + \frac{1}{4K}\sum_{k=1}^{K}\|\mathbf{w}-\mathbf{v}_k\|_2^2 + C\sum_{i=1}^{n_c}\xi_i - \sum_{i=1}^{n_c}\alpha_i y_i b - \mathbf{w}^T\sum_{i=1}^{n_c}\alpha_i y_i \mathbf{x}_i - \sum_{i=1}^{n_c}\alpha_i\xi_i + \sum_{i=1}^{n_c}\alpha_i - \sum_{i=1}^{n_c}\beta_i\xi_i$$

or simply:

$$L_D = \underbrace{\frac{1}{4}\|\mathbf{w}\|_2^2}_{T_1} + \underbrace{\frac{1}{4K}\sum_{k=1}^{K}\|\mathbf{w}-\mathbf{v}_k\|_2^2}_{T_2} \underbrace{- \mathbf{w}^T\sum_{i=1}^{n_c}\alpha_i y_i \mathbf{x}_i}_{T_3} + \underbrace{\sum_{i=1}^{n_c}\alpha_i}_{T_4} \quad (4)$$

where we use subscript D to denote the dual form. In order to represent the Lagrange dual function purely in terms of Lagrange multipliers, we replace $\mathbf{w}$ by the quantity in Eq. (3).

[2]Note that K generally can be larger than 2 for many practical datasets. For example, national census data collected at each state/county can form a data cluster, and one can be adjacent to multiple nearby states. Intuitively, a model learned for a state like New Jersey can be highly impacted by those learned at the nearby states such as New York, Pennsylvania, Maryland, rather than those studied at the far distance states such as Georgia and Florida.

To keep the derivation clear, let us denote four terms in Eq. (4) respectively from T_1 to T_4, and denote $\mathbf{z}_i = y_i\mathbf{x}_i$. We have:

$$T_1 = \frac{1}{4}\|\mathbf{w}\|_2^2 = \frac{1}{4}\left(\frac{1}{2K}\sum_k \mathbf{v}_k + \sum_i \alpha_i\mathbf{z}_i\right)^T\left(\frac{1}{2K}\sum_k \mathbf{v}_k + \sum_i \alpha_i\mathbf{z}_i\right)$$
$$= \frac{1}{4}\left(\frac{1}{4K^2}(\sum_k \mathbf{v}_k)^T\sum_k \mathbf{v}_k + \frac{1}{K}(\sum_k \mathbf{v}_k)^T\sum_i \alpha_i\mathbf{z}_i + \sum_{i,j}\alpha_i\alpha_j\mathbf{z}_i^T\mathbf{z}_j\right)$$
$$= \text{const} + \frac{1}{4K}(\sum_k \mathbf{v}_k)^T\sum_i \alpha_i\mathbf{z}_i + \frac{1}{4}\sum_{i,j}\alpha_i\alpha_j\mathbf{z}_i^T\mathbf{z}_j$$

where from the second to the third row, we consider the first term as a constant since all $\mathbf{v}_k$'s are known prior to optimizing $\mathbf{w}$.

Term T_2 in Eq. (4) is the summation over K terms and the derivation w.r.t. term j-th, denoted by T_{2j}, is as follows:

$$T_{2j} = \frac{1}{4K}\|\mathbf{w}-\mathbf{v}_j\|_2^2 = \frac{1}{4K}\left\|\frac{1}{2K}\sum_k \mathbf{v}_k - \mathbf{v}_j + \sum_i \alpha_i\mathbf{z}_i\right\|_2^2$$
$$= \text{const} + \frac{1}{4K^2}\left(\sum_k \mathbf{v}_k - 2K\mathbf{v}_j\right)^T\left(\sum_i \alpha_i\mathbf{z}_i\right) + \frac{1}{4K}\sum_{i,j}\alpha_i\alpha_j\mathbf{z}_i^T\mathbf{z}_j$$

Again, we have a constant for the terms solely calculated from $\mathbf{v}_k$'s. Term T_2, as the sum of all T_{2j}'s, is thus calculated by:

$$T_2 = \sum_{j=1}^{K} T_{2j} = \frac{1}{4K}\sum_j \|\mathbf{w}-\mathbf{v}_j\|_2^2$$
$$= \sum \text{const} - \frac{1}{4K}\left(\sum_k \mathbf{v}_k\right)^T\left(\sum_i \alpha_i\mathbf{z}_i\right) + \frac{1}{4}\sum_{i,j}\alpha_i\alpha_j\mathbf{z}_i^T\mathbf{z}_j$$

Term T_4 is already in a simple form and is equal to $\sum_i \alpha_i$ while T_3 is:

$$T_3 = -\sum_i \alpha_i\mathbf{z}_i^T\left(\frac{1}{2K}\sum_k \mathbf{v}_k + \sum_i \alpha_i\mathbf{z}_i\right)$$
$$= -\frac{1}{2K}(\sum_k \mathbf{v}_k)^T\sum_i \alpha_i\mathbf{z}_i - \sum_{i,j}\alpha_i\alpha_j\mathbf{z}_i^T\mathbf{z}_j$$

In the summation of all terms, we can omit the constants since they do not impact the goal of optimization:

$$T_1 + \ldots + T_4 = \frac{1}{4K}(\sum_k \mathbf{v}_k)^T\sum_i \alpha_i\mathbf{z}_i + \frac{1}{4}\sum_{i,j}\alpha_i\alpha_j\mathbf{z}_i^T\mathbf{z}_j$$
$$- \frac{1}{4K}(\sum_k \mathbf{v}_k)^T\sum_i \alpha_i\mathbf{z}_i + \frac{1}{4}\sum_{i,j}\alpha_i\alpha_j\mathbf{z}_i^T\mathbf{z}_j$$
$$+ \sum_i \alpha_i - \frac{1}{2K}(\sum_k \mathbf{v}_k)^T\sum_i \alpha_i\mathbf{z}_i - \sum_{i,j}\alpha_i\alpha_j\mathbf{z}_i^T\mathbf{z}_j$$
$$= \sum_i \alpha_i - \frac{1}{2K}(\sum_k \mathbf{v}_k)^T\sum_i \alpha_i\mathbf{z}_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j\mathbf{z}_i^T\mathbf{z}_j$$

The final form for the Lagrange dual objective is thus given by:

$$\arg\min_{\alpha} L_D(\alpha) = \sum_{i=1}^{n_c}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j\mathbf{z}_i^T\mathbf{z}_j - \frac{1}{2K}(\sum_{k=1}^{K}\mathbf{v}_k)^T\sum_{i=1}^{n_c}\alpha_i\mathbf{z}_i$$
$$\text{such that} \quad 0 \le \alpha_i \le C \text{ and } \sum_{i=1}^{n_c}\alpha_i y_i = 0$$

This is a quadratic programming problem with constraints and hence is straightforward to solve for α_i, and subsequently the weight vector $\mathbf{w}$ and the bias b.

IV. Solving the Problem in the Primal Form

Our solution achieved in the dual form of SVM involves smoothness while training multiple classifiers across data clusters. However, it does not offer us an elegant way to extract a succinct set of relevant features that capture the smooth change in clusters' data distribution. For this reason, we further develop a solution in the primal form of SVM. Specifically, we aim to minimize the following primal form of the objective function:

$$\arg\min_{\mathbf{w},b,\xi_i} \frac{1}{4}\|\mathbf{w}\|_2^2 + \frac{1}{4K}\sum_{k=1}^{K}\|\mathbf{w}-\mathbf{v}_k\|_2^2 + C\sum_{i=1}^{n_c}\xi_i \qquad (5)$$
$$\text{such that} \quad y_i(b+\mathbf{w}^T\mathbf{x}_i) \ge 1-\xi_i, \xi_i \ge 0, \forall i$$

The last two constraints can be combined into: $\xi_i \ge \max(0, 1-y_i(b+\mathbf{w}^T\mathbf{x}_i)) = [1-y_i(b+\mathbf{w}^T\mathbf{x}_i)]_+$, which is the Hinge loss function [18]. Given this setting, we rewrite the function as follows:

$$P(\mathbf{w},b) = \arg\min_{\mathbf{w},b}\sum_{i=1}^{n_c}[1-y_i(b+\mathbf{w}^T\mathbf{x}_i)]_+ + \frac{\lambda_2}{4}\|\mathbf{w}\|_2^2$$
$$+ \frac{\lambda_2}{4K}\sum_{k=1}^{K}\|\mathbf{w}-\mathbf{v}_k\|_2^2 + \lambda_1\|\mathbf{w}\|_1 \qquad (6)$$

where $\lambda_2 = 1/C$ and we have added L1-norm on $\mathbf{w}$ in order to sparsify its components. Such a constraint helps to remove irrelevant features that have little or no impact on predicting class labels of data samples.

In this form, the first term can be considered as the loss function while the remaining three terms are viewed as regularization. With regard to the Hinge loss function, its direct optimization is difficult since the derivative is discontinuous at $y_i(b+\mathbf{w}^T\mathbf{x}_i) = 1$. The sharp corner at value 1 of this function encourages sparsity, yet with respect to the *dual* variables. Thus, this property is less compelling since what we want to learn is a sparse model with respect to the *primal* variables $\mathbf{w}$, i.e. performing feature selection. We hence adopt a more practical approach by smoothing out the function and thus define the Smooth Hinge (shown in Fig. 2) as follows:

$$h(z) = \begin{cases} \frac{1}{2} - z & \text{if } z \le 0 \text{ (region } R_1\text{)} \\ \frac{1}{2}(1-z)^2 & \text{if } 0 < z < 1 \text{ (region } R_2\text{)} \\ 0 & \text{if } z \ge 1 \text{ (region } R_3\text{)} \end{cases} \qquad (7)$$

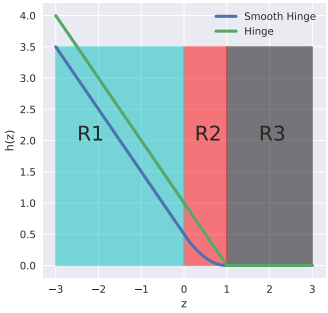

Fig. 2. Hinge loss function in Eq. (6) vs. Smooth Hinge in Eq. (7) within 3 regions.

where $z = y(b+\mathbf{w}^T\mathbf{x})$.

As such, Eq. (6) can be rewritten as:

$$P(\mathbf{w},b) = \arg\min_{\mathbf{w}}\sum_{i=1}^{n_c}h(z_i) + \frac{\lambda_2}{4}\|\mathbf{w}\|_2^2$$
$$+ \frac{\lambda_2}{4K}\sum_{k=1}^{K}\|\mathbf{w}-\mathbf{v}_k\|_2^2 + \lambda_1\|\mathbf{w}\|_1 \qquad (8)$$

Having the fact that $y_i^2 = 1$ since $y_i \in \{-1, +1\}$, the partial derivative of P w.r.t. b is given by:

$$\frac{\partial P}{\partial b} = -\sum_{z_i\in R_1} y_i - \sum_{z_i\in R_2} y_i(1-y_i(b+\mathbf{w}^T\mathbf{x}_i))$$
$$= -\sum_{z_i\in R_1} y_i + \sum_{z_i\in R_2}(b+\mathbf{w}^T\mathbf{x}_i - y_i)$$

Let us denote $|R_2|$ the number of samples falling into region R_2, then the update on b is given by:

$$b = |R_2|^{-1}\left(\sum_{z_i\in R_1} y_i - \sum_{z_i\in R_2}(\mathbf{w}^T\mathbf{x}_i - y_i)\right) \qquad (9)$$

The derivative of P with respect to the j-th component of $\mathbf{w}$ yields:

$$\frac{\partial P}{\partial w_j} = -\sum_{z_i\in R_1} y_i x_{ij} + \sum_{z_i\in R_2}(b+\mathbf{w}^T\mathbf{x}_i - y_i)x_{ij}$$
$$+ \frac{\lambda_2}{2}w_j + \frac{\lambda_2}{2K}\sum_k(w_j - v_{kj}) + \frac{\partial\lambda_1|\mathbf{w}|}{\partial w_j} \qquad (10)$$

Let us decompose $\mathbf{w}^T\mathbf{x}_i = \bar{\mathbf{w}}^T\bar{\mathbf{x}}_i + w_j x_{ij}$ where $\bar{\mathbf{w}}$ and $\bar{\mathbf{x}}_i$ respectively denote the vectors excluding their j-th component, we can re-arrange terms in Eq. (10) as follows:

$$\frac{\partial P}{\partial w_j} = \left(\sum_{z_i\in R_2} x_{ij}^2 + \lambda_2\right)w_j + \sum_{z_i\in R_2}(\bar{\mathbf{w}}^T\bar{\mathbf{x}}_i + b)x_{ij}$$
$$- \sum_{z_i\in R_{1,2}} y_i x_{ij} - \frac{\lambda_2}{2K}\sum_k v_{kj} + \frac{\partial\lambda_1|\mathbf{w}|}{\partial w_j} \qquad (11)$$

The derivative of the last term with respect to w_j, knowing not being smooth, is computed as:

$$\frac{\partial \lambda_1 |\mathbf{w}|}{\partial w_j} = \begin{cases} \{-\lambda_1\} & \text{if } w_j < 0 \\ \{+\lambda_1\} & \text{if } w_j > 0 \\ [-\lambda_1, +\lambda_1] & \text{if } w_j = 0 \end{cases} \quad (12)$$

For the sake of clarity, let us simplify:
$a = \sum_{z_i \in R_2} x_{ij}^2 + \lambda_2$,
$c = \sum_{z_i \in R_{1,2}} y_i x_{ij} + \frac{\lambda_2}{2K} \sum_k v_{kj} - \sum_{z_i \in R_2} (\bar{\mathbf{w}}^T \bar{\mathbf{x}}_i + b) x_{ij}$.
Obviously a is always positive and it is straightforward to show:

$$\frac{\partial P}{\partial w_j} = \begin{cases} \{a \times w_j - c - \lambda_1\} & \text{if } w_j < 0 \\ \{a \times w_j - c + \lambda_1\} & \text{if } w_j > 0 \\ [-c - \lambda_1, -c + \lambda_1] & \text{if } w_j = 0 \end{cases} \quad (13)$$

Setting this equation to zero and solving for w_j finally yields:

$$w_j = \begin{cases} (c + \lambda_1)/a & \text{if } c < -\lambda_1 \\ (c - \lambda_1)/a & \text{if } c > +\lambda_1 \\ 0 & \text{if } c \in [-\lambda_1; +\lambda_1] \end{cases} \quad (14)$$

a) Implementation in the primal form: Our solution in the primal form is optimized via the coordinate gradient descent. Each margin vector $\mathbf{w}$ is optimized w.r.t. its corresponding data cluster and the related margin vectors $\mathbf{v}_k$'s from nearby classifiers. At each step, we optimize one classifier (characterized by $\mathbf{w}$), while keeping other classifiers fixed ($\mathbf{v}_k$'s are fixed). The convergence of the classifier under training is defined w.r.t. ϵ, a threshold accounting for the average change in the magnitude of $\mathbf{w}$, or its optimization reaches the maximum number of iterations $iter$ defined at that step. Once this classifier converges (under this setting), we move to the next one to optimize, and once all classifiers are trained w.r.t. the current setting of ϵ and $iter$, we update these two thresholds and repeat the entire process at the next step. Training all component classifiers in this way ensures that our model will optimize all margin vectors concurrently, and especially the current states of related margin vectors will impact $\mathbf{w}$, the margin vector under optimization, immediately. Hence, our model does not set ϵ and $iter$ fixed all the time. Instead, it starts with ϵ relatively high while $iter$ small, say $\epsilon = 0.1$ and $iter = 10$. Then, it keeps updating $\epsilon = 0.1 \times \epsilon$ and $iter = 2 \times iter$ in the subsequent iterations (the final settings are $\epsilon = 1e^{-5}$ and $iter = 100$). This resembles the scenario where one can optimize all margin vectors in parallel, yet after a certain number of iterations, their current values are propagated through out the entire system. Our simple training strategy allows its algorithm to optimize each $\mathbf{w}$ locally within its own data clusters, but also to optimize all $\mathbf{w}$'s globally across data clusters gradually.

b) Algorithm complexity: The computational complexity of our dual form is similar to that of classical `SVM` since its function remains quadratic as presented in Section III. An analysis of the computational complexity of the dual form can be found in [19], [20]. Hence, we concentrate on the primal form. Our primal form requires the computation of the smooth Hinge loss in every iteration, which costs $O(n_c d) + O(n_c)$ (where n_c is the number of data samples in the current cluster on which the classifier is trained, and d is the number of original data dimensions) due to the projection of $\mathbf{x}_i$'s on $\mathbf{w}$, and the searching for the number of data samples falling into three regions of the Hinge loss (Eq. (7)). The bias computation needs at most $O(n_c d)$ for the inner product between $\mathbf{x}_i$'s in region R_2 and $\mathbf{w}$, which is also the computational cost for the update of each component of $\mathbf{w}$ (Eq. (11)). Since we need to compute for every component, the computation amounts to $O(n_c d^2)$. Let N be the average number of iterations to train a classifier till its convergence, and as we need to compute for all margin vectors of all M classifiers (whose total number of training samples is n), the overall complexity under gradient descent amounts to $O(N d^2 n)$.

V. Experiments

We denote our first algorithm `SSVMDual` and the second one `SSVMPrimal`. Their performance is compared against the following algorithms: (1) The classical `SVM` (`CSVM`), that imposes no smooth constraints among clusters of data samples; (2) Multi-task feature learning (`MTFL`) [21] dealing with multi-task datasets that may contain both static and dynamic features; (3) `Logistic-TGL` [22] that performs multi-task learning typically for disease progression models.

For each examined dataset described below, we split 70% for training data and 30% for an independent testing data. Such splitting is applied to every cluster of data samples. Within the training data, we further perform 5-fold cross validation to tune model's parameters. A final model is then trained on the entire training data using the optimal parameters and is used to make class prediction on the independent testing dataset. For `CSVM`, we tune the soft-margin coefficient C from the range $[1, 10^3]$. For our algorithms, λ_2 is tuned from the range $[0.01, 10]$. With `MTFL`, we choose optimal ρ_1 in the range $[0.001, 500]$ as suggested in [21], and for `Logistic-TGL`, we select θ_1, θ_2 and θ_3 in the range $[10^{-3}, 10^3]$ as recommended in [22][3].

We use classification accuracy as the evaluation metric for different methods. However, as our more important goal is to explore the features that not only lead to high prediction accuracy but can also capture and reflect the changes in the data distribution, we investigate the feature sets returned by each examined model. Our attempt is to interpret how they adapt to changes in the data distribution across multiple data clusters.

A. Team Performance on LoL Dataset

The first dataset we analyze is the League of Legends (`LoL`) that has gained much research attention recently [3], [15], [23] due to its rich information in studying human collaboration toward solving complex team-task problems in a virtual online world. The dataset involves data collected at various dimensions of team performance from 5620 games. The games are naturally clustered into five progressive levels

[3]For each parameter range, we choose 10 uniformly distributed values in its log-scale form.

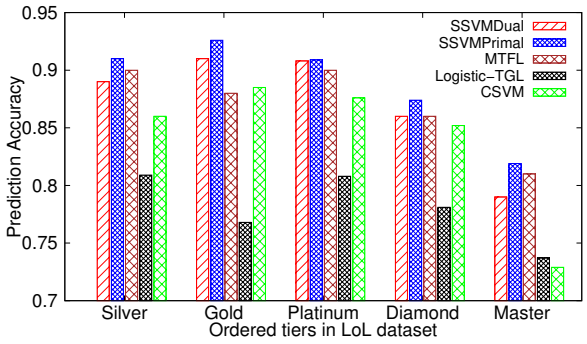

Fig. 3. Accuracy rates of all algorithms on five tiers of League of Legend.

ranged from Silver, Gold, Platinum, Diamond and Master, which reflect the increasing levels of task difficulties. We excluded data from the Bronze level since at this first level, there is no historical information about the game-players. Our task is to build a model that can predict which team will win a game and how its strategy changes as the difficulty of the task (game) advances. Generally, in order to win a game, a member of the team should choose a champion (the player-controlled character) whom (s)he is the most skilled at. Nevertheless, acting as a team, all members should negotiate and choose champions or the skill sets that can complement each other. Having do so can help to maximize their chance of winning the game or completing the complex task ahead. We therefore collect three sets of features: (i) 14 features of in-game statistics such as rate of acquiring gold per minute, damage taken per minute, number of killed creeps per minute, etc., collected within only the first 5 minutes of a game. These statistics are considered as the consequence of *actual collaboration* among team members; (ii) 48 features characterizing different aspects of champions (e.g., attack, defense, health, etc.) selected by the players; (iii) 178 features describing players' proficiency, which measure how skillful a player is with the selected champions by comparing his/her currently selected champions against the distribution of champions (s)he has played with in the past. These features are considered *individual skills/experiences*. In order to alleviate the data noise and to better understand the patterns associated with players' experience, we ensure that all members in a team are in the same rank (i.e., have similar experience) [16].

a) Prediction accuracy: In Fig. 3, we show the prediction accuracy of all algorithms on the `LoL` dataset. The accuracy is reported at the game-tier which forms 5 bar clusters annotated by Silver, Gold, Platinum, Diamond and Master. For `CSVM` and `SSVMDual`, we report their prediction accuracy using all original features. For `MTFL`, `Logistic-TGL`, and `SSVMPrimal`, we report their accuracy when a threshold of 25 features is imposed to each technique. From the plot, one can observe a general trend that predicting game outcomes at the low tiers (e.g., Silver and Gold) is more accurate than those at the high levels (e.g., Diamond and Master) in most examined algorithms. This result is intuitive since `LoL` is a year long league in which a player may need months in order to get skilled at a champion, and especially to learn how to collaborate with his/her teammates. At the low tiers, most players are novice and their performance can be anticipated based on the statistics collected at the beginning of the games. However, once players gain more experience, become elite and advance higher into the league, the game outcomes are harder to predict. As observed in Fig. 3, the prediction accuracy of most examined algorithms can be as high as 88% at the low levels of Silver and Gold clusters, but reduce considerably to around 80% at the last level of Master.

At the level of individual algorithm performance, we find that `SSVMDual` performs better than `CSVM` though both exploit all input features for their classification models. This clearly demonstrates the advantages of imposing smooth variation in training classifiers from multiple clusters of data samples. `SSVMDual` accuracy rates are also competitive to those of MTFL in most tiers. The performance of `Logistic-TGL` on this dataset is lower as compared to other techniques. This might be attributed to the fact that `Logistic-TGL` attempts to learn a single *fixed* set of features (their coefficients are also quite stable as shortly seen below) across all game tiers, despite the variation in the data distribution. Out of all algorithms, our `SSVMPrimal` algorithm produces the best prediction accuracy with more than 90% at the low tiers (Silver to Platinum), and 82% at the most challenging task, the Master tier. Looking ahead, these results can be justified by the set of smoothly changed features adapted to the variation in the data distribution across five tiers as presented below.

b) Interpreting patterns: In Fig. 4(a),(b) and (c), we plot 25 features respectively selected by algorithms `MTFL`, `Logistic-TGL` and `SSVMPrimal`. The length of each bar is the absolute coefficient of the corresponding feature. Purple, orange and green colors respectively encode the three feature categories (as described above): in-game statistics (reflecting the performance of the team), champions' features, and personal skills of players. It can be seen that, both `MTFL` and `Logistic-TGL` select a similar set of features across all game-tiers. However, unlike `Logistic-TGL` where the features' coefficients from each tier are quite stable, there are some significant changes in the features' coefficients learned by `MTFL`, which perhaps helps `MTFL` achieve better prediction accuracy. However, both techniques give less attention to features related to the team performance, regardless of how hard the games are. On the other hand, one can observe that the set of features discovered by `SSVMPrimal` not only vary in terms of the feature coefficients but also in the feature sets belonging to each of three categories mentioned above. Especially, such a variation in the selected feature sets is smooth and evolves along with the difficulty levels of the games. We found that the harder the prediction of a game is, the more in-game statistics features (in purple color) are selected. As observed in Fig. 4(c), there are only four features selected from this category at the Silver and Gold tiers, but that number is almost double in the subsequent higher levels of Diamond and Master. Note that the in-game statistics characterize the team's collaboration, and for more difficult games (more complex tasks), these features have gained more

Fig. 4. Features selected by (a) MTFL, (b) Logistic-TGL and (c) SSVMPrimal through all tiers in LoL dataset. Purple encodes in-game statistics features (team performance); Orange encodes champions' features; Green encodes individual player proficiency features (best visualized in color). The game-related features are described in (https://developer.riotgames.com/api-methodshttps://developer.riotgames.com/api-methods). In each row, five plots from left to right correspond to 5 tiers in LoL. Observation: the features selected by MTFL and Logistic-TGL are similar across tiers. For SSVMPrimal, team features (in purple) are less paid attention in the low tiers (less selected team features as well as small coefficients) but gain more and more attention in the high tiers, demonstrating the importance of collaboration among team's members in order to win a game (more explanation is given in text).

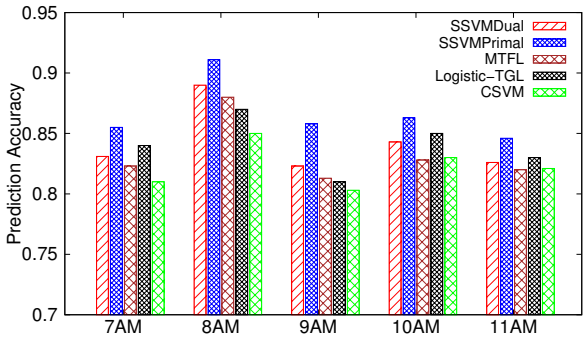

Fig. 5. Classification accuracy in predicting weekday vs. weekend snapshots of the LA traffic network. Five groups of traffic snapshots correspond to five hours between 7AM to 12PM. For simplicity, each group is denoted by its starting hour.

important roles, which are highly intuitive and interpretable. Moreover, Fig. 4(c) also shows that more related-champion features (in orange) but fewer personal skill-related features (in green) have been chosen as the game levels increase. This means experienced teams also prefer the correct selection and diversity among the selected champions in order to maximize its chance of winning a game. All these findings are quite consistent with those found in [15], [16], [23], which have exploited the domain expertise and game-players' surveys in order to find and validate these novel patterns. Our algorithm demonstrates that such important patterns can be discovered purely from a data-driven approach, which is clearly more preferable, in many applications, given the expensive cost and time of using domain expertise.

B. LA Traffic Networks

The second dataset we analyze is the `LAtraffic`—the highway traffic network data of Los Angeles, California (http://pems.dot.ca.gov) collected in the whole month of April 2011. It contains snapshots captured every 5 minutes from sensors placed at 100 road segments. Values collected at a sensor are the speeds averaged from vehicles traveling at that road segment within 5-minute resolution. In order to analyze and understand the evolution of traveling patterns between weekdays and weekends, we collect all snapshots between `7AM` and `12PM`, and consider snapshots within each hour as a group of similar data samples. This naturally forms 5 adjacent clusters of traffic network snapshots.

a) Prediction accuracy: We evaluate the performance of all algorithms in classifying the weekdays' snapshots from the weekends' ones. Fig. 5 reports the prediction accuracy from all algorithms. As observed, there is no big gap in performance among three methods of `Logistic-TGL`, `MTFL`, and `SSVMDual`. While `Logistic-TGL` regularizes its learning process through the temporal group Lasso, `MTFL` adopts the group hard constraints imposed on the parameter matrix, both techniques encourage the consistency of feature sets across multiple tasks. Our `SSVMDual` performs competitively against these two typical techniques, and better than that the conventional `SVM`; the smoothness constraint helps it narrow down the search space in order to converge to a better set of support vectors. Among all examined techniques,

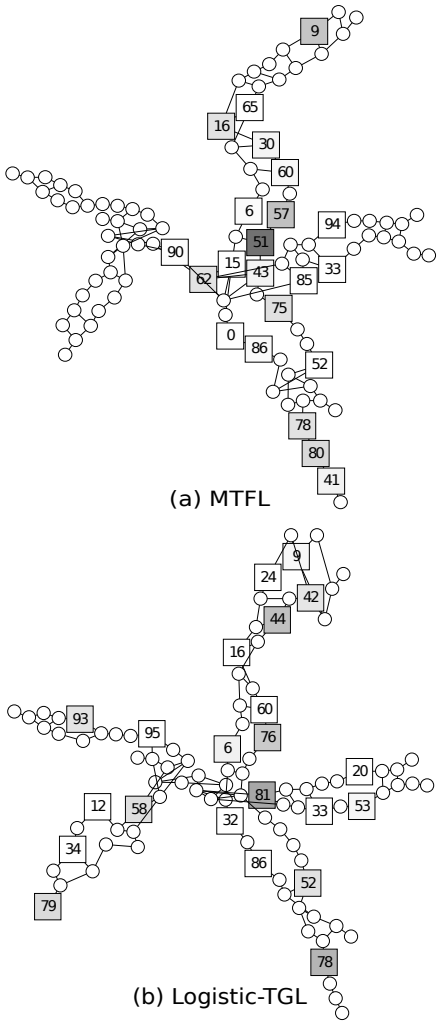

Fig. 6. Road segments selected by (a) MTFL and (b) Logistic-TGL in distinguishing weekday from weekend snapshots in LAtraffic. Only patterns at 9AM are plotted as both techniques select the same patterns across 5 hours. Explanation is given in text.

the `SSVMPrimal` offers the most convincing solution with an impressive improvement in prediction accuracy on all data clusters. It is not only radically different from the `Logistic-TGL` and `MTFL` in the learning objective function but also in the soft constraint imposed on the similarity among training classifiers learned from adjacent data clusters. Consequently, the feature sets (as visualized below) learnt from each data group adapt well to the variation in the data distribution.

b) Interpreting patterns: Our `SSVMPrimal` algorithm and the `Logistic-TGL` are discriminative models, as is the `MTFL` though it relies on a regression model with a discrete dependent variable. The road segments selected by three techniques hence can be interpreted as the subspace in which snapshots of weekdays can be well discriminated from those of weekends. In Fig. 6(a) and (b), we plot the road segments selected by `MTFL` and `Logistic-TGL` at `9AM`. We omit other hours since both techniques output the same set of selected nodes. In Fig. 7, we plot the road segments selected

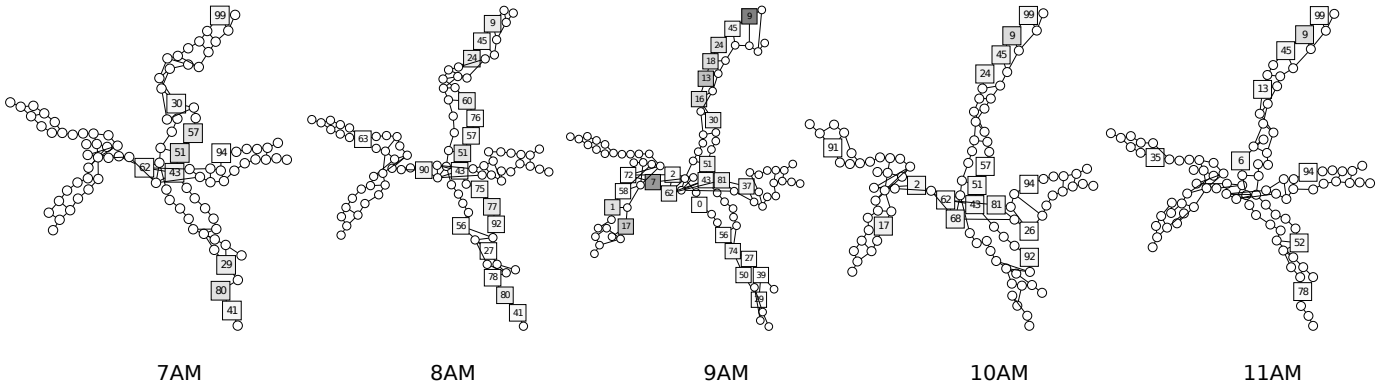

Fig. 7. Road segments selected by SSVMPrimal across 5 morning hours in distinguishing weekday snapshots from weekend ones in the Los Angeles highway traffic network data. Observation: selected road segments (square nodes with index) reveal traffic congestion areas, which form in the early hours, grow in the subsequent ones, shrink and vanish in the later hours. More detailed explanation is given in the text.

by SSVMPrimal across all 5 hours since unlike the other two competitors, there exists variation among its selected feature nodes. Each node corresponds to a road segment, and an edge connects two nodes if the two corresponding road segments are adjacent to each other. Square nodes denote the selected road segments while the node darkness reflects the magnitude of its absolute coefficient.

In Fig. 6(a) and (b) MTFL and Logistic-TGL share almost half of the selected road segments, namely segments (16, 34, 42, 44, 86, 95). Nonetheless, many of their selected road segments are not adjacent to each other. Though these patterns can help the algorithms discriminate weekend snapshots from weekdays, they do not carry much interpretive power from the viewpoint of traffic patterns. In contrast, it is more intuitive and interesting to observe the road segments discovered by our developed technique SSVMPrimal as visualized in Fig. 7. Across the the five hours from 7AM to 11AM, we can observe the formation of traffic congestion which helps to distinguish weekday snapshots from the weekend ones. Specifically, at 7AM, the formation of two large subnetworks (62, 43, 51, 57), and (29, 80, 41) can be easily spotted.

In the subsequent time slots from 8AM to 9AM, there is involvement of other new nodes in these existing subnetworks. For example, two novel road segments (60, 76) get involved in the first subnetwork (62, 43, 51, 57) at 8AM, in addition to the formation of subnetworks (78, 80, 41), and (9, 45, 24). At 9AM, we further observe nodes (18, 13, 16, 30) being attached to this latter subnetwork, demonstrating the formation and growth of traffic congestion. At 10AM, one can see the start of shrinkage of these congestion areas, and at 11AM, there are only 9 road segments selected by the SSVMPrimal. The traffic congestion at (45, 9, 99) seems to be severe as it lasts for hours. These traffic patterns obviously demonstrate the novel findings discovered by SSVMPrimal as compared to those selected by MTFL and Logistic-TGL. By relaxing the same feature set selected across data clusters, our algorithm allows the discovery of evolving patterns that effectively reflect and capture the smooth variation in the underlying distribution of the observed data.

VI. Related work

Our studies in this work relate to multi-task learning (MTL). Studies in multi-task learning aim at improving the generalization performance of one task by leveraging the useful information contained in other related tasks. Popular approaches in MTL are those enforcing similarity among model parameters of every task [9], [24]–[26], those capturing the common structures such as low rank subspaces or attribute subsets among all tasks [14], [27], [28], and those combining both these approaches [21], [22], [29].

In [9], [11], [24], there is a common assumption that all tasks are similar to each other and thus the learning objective is to train the model's parameters of one task as close as possible to the average model's parameters of all other tasks. This assumption is relaxed in the approaches developed in [26], [30] where the task similarity is defined for pairs of tasks. Pairwise regularization is imposed to train the MTL model so that model parameters of two related tasks are more similar than those of unrelated ones. However, none of these techniques perform feature selection for an individual task, which is the key difference to our work. Their derived models thus lack the important goal of interpretation and explanation, hence providing limited information to the user. In the studies performed in [14], [27], the authors attempt to learn a sparse model across multiple tasks by imposing the group Lasso condition (similar to the work done in [21], [29] as shortly reviewed below). These approaches are mostly different in the way they solve the objective function. While [14] adopts Nesterov's solution, [27] explores a blockwise path-following scheme to approximately trace the regularization path in its solution.

Our work is also related to [22] and its extension in [29]. These studies address the problem from logistic regression and further impose group Lasso in order to sparsify the model. However, the fundamental difference between these studies and ours is of the problem setting. In [22], [29], the authors address the variation in the *dependent* variable

(i.e., as class labels) while fixing the observable variables (i.e., the predictors). This setting is suitable for disease-related applications [31] where measurement over class labels can be performed periodically, yet the data collected at the baseline (used as the predictors) remain unchanged. Thus, there is no variation in the data distribution except over the class labels. In contrast, we address the setting in which the distribution of both predictors and class labels vary, and attempt to interpret the changes in the data distribution via the changes in the predictors. The study in [21] regularizes the linear regression with `L1`-norm in order to obtain a parsimonious solution. Nonetheless, the feature set selected by [21] (as also in [22], [29]) is stable across all tasks, as demonstrated in our experiments, implying the data distribution is consistent and homogeneous across tasks. Our work, in contrast, addresses the practical situations in which the data distribution is heterogeneous and keeps changing across clusters of data samples. Hence, it aims at discovering smoothly varying feature sets that well capture the smooth variation in the data distribution across multiple data clusters.

VII. Conclusions

In this paper, we investigated the novel problem of training multiple classifiers and discovering succinct sets of features that vary smoothly across clusters of data samples. We presented a novel learning framework by training multiple classifiers and imposing smoothness constraints among them. We developed two algorithms formulated in the dual and primal forms of `SVM`, and showed that our solution in the dual form is straightforward, while the solution in the primal form requires a relaxation of the hinge loss function. This relaxation is technically involved; however, the final solution can be effectively optimized and achieved through the approach of gradient descent. We demonstrated the performance of our novel learning framework on two important real-world applications of online team-task performance, and of the highway road traffic network. The empirical results demonstrate that our framework not only outperforms existing methods in term of prediction accuracy, but more importantly it can effectively discover succinct sets of smoothly varying features that truly capture and reflect the variation in the underlying data distribution. Such patterns considerably enhance our understanding regarding the mechanism behind the observed data.

Acknowledgement: This material is supported by the U.S. Army Research Laboratory and the U.S. Army Research Office under grant number W911NF-15-1-0577. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government.

References

[1] T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," in *SIGKDD*, 2009.

[2] A. Gajewar and A. Das Sarma, "Multi-skill collaborative teams based on densest subgraphs," in *SDM*, 2012, pp. 165–176.

[3] N. Pobiedina, J. Neidhardt, M. d. C. Calatrava Moreno, and H. Werthner, "Ranking factors of team success," in *ACM-WWW*, 2013.

[4] X.-H. Dang, A. Silva, A. Singh, A. Swami, and P. Basu, "Outlier detection from network data with subnetwork interpretation," in *IEEE ICDM*, 2016.

[5] A. Silva, X. H. Dang, P. Basu, A. Singh, and A. Swami, "Graph wavelets via sparse cuts," in *ACM SIGKDD*, 2016, pp. 1175–1184.

[6] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, 1997.

[7] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE TKDE*, vol. 22, no. 10, 2010.

[8] S. S. Du, J. Koushik, A. Singh, and B. Póczos, "Hypothesis transfer learning via transformation functions," in *NIPS*, 2017, pp. 574–584.

[9] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *J. Mach. Learn. Res.*, vol. 6, Dec. 2005.

[10] K. Yu, V. Tresp, and A. Schwaighofer, "Learning Gaussian processes from multiple tasks," in *ICML*. ACM, 2005.

[11] S. Parameswaran and K. Q. Weinberger, "Large margin multi-task metric learning," in *NIPS*, 2010.

[12] Z. Kang, K. Grauman, and F. Sha, "Learning with whom to share in multi-task feature learning." in *ICML*, 2011, pp. 521–528.

[13] S. Kim and E. P. Xing, "Tree-guided group lasso for multi-task regression with structured sparsity," in *ICML*, 2010.

[14] J. Liu, S. Ji, and J. Ye, "Multi-task feature learning via efficient l-2, 1-norm minimization," in *25th conference on uncertainty in artificial intelligence*, 2009.

[15] Y. J. Kim *et al.*, "What makes a strong team?: Using collective intelligence to predict team performance in league of legends." in *ACM-CSCW*, 2017.

[16] J. Kim, B. C. Keegan, S. Park, and A. Oh, "The proficiency-congruency dilemma: Virtual team design and performance in multiplayer online games," in *ACM-CHI*, 2016.

[17] M. Mongiovi *et al.*, "Netspot: Spotting significant anomalous regions on dynamic networks," in *SDM*, 2013.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag., 2006.

[19] L. Bottou and C.-j. Lin, "Support vector machine solvers," in *Large scale kernel machines*, 2007, pp. 301–320.

[20] Q. V. Le, A. J. Smola, and S. Vishwanathan, "Bundle methods for machine learning," in *Advances in neural information processing systems*, 2008, pp. 1377–1384.

[21] L. Zhao, Q. Sun, J. Ye, F. Chen, C.-T. Lu, and N. Ramakrishnan, "Multi-task learning for spatio-temporal event forecasting," in *SIGKDD*, 2015.

[22] J. Zhou, L. Yuan, J. Liu, and J. Ye, "A multi-task learning formulation for predicting disease progression," in *ACM-SIGKDD*, 2011.

[23] A. Leavitt, B. C. Keegan, and J. Clark, "Ping to win?: Non-verbal communication and team performance in competitive online multiplayer games," in *ACM-CHI*, 2016.

[24] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *SIGKDD*, 2004.

[25] Y. Zhang and D.-Y. Yeung, "Transfer metric learning by learning task relationships," in *SIGKDD*. ACM, 2010.

[26] T. Evgeniou, C. A. Micchelli, and M. Pontil, "Learning multiple tasks with kernel methods," *Journal of Machine Learning Research*, vol. 6, pp. 615–637, 2005.

[27] G. Obozinski, B. Taskar, and M. I. Jordan, "Joint covariate selection and joint subspace selection for multiple classification problems," *Statistics and Computing*, vol. 20, no. 2, 2010.

[28] S. Lee, J. Zhu, and E. P. Xing, "Adaptive multi-task lasso: with application to eqtl detection," in *NIPS*, 2010, pp. 1306–1314.

[29] J. Zhou, J. Liu, V. A. Narayan, J. Ye, A. D. N. Initiative *et al.*, "Modeling disease progression via multi-task learning," *NeuroImage*, vol. 78, pp. 233–248, 2013.

[30] T. Kato, H. Kashima, M. Sugiyama, and K. Asai, "Conic programming for multitask learning," *IEEE TKDE*, vol. 22, no. 7, pp. 957–968, 2010.

[31] X.-H. Dang, H. You, A. K. Singh, and S. Grafton, "Subnetwork mining with spatial and temporal smoothness," in *SIAM-SDM*, 2017.