

Conic Programming for Multi-Task Learning

Tsuyoshi Kato

Center for Informational Biology, Ochanomizu University
2-1-1, Otsuka, Bunkyo, Tokyo 112-8610, Japan
kato-tsuyoshi@aist.go.jp

Hisashi Kashima

IBM Research, Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato-shi, Kanagawa 242-8502, Japan
hkashima@jp.ibm.com

Masashi Sugiyama

Department of Computer Science, Tokyo Institute of Technology
2-12-1 O-okayama, Meguro-ku, Tokyo 152-8552, Japan
sugi@cs.titech.ac.jp

Kiyoshi Asai

Department of Computational Biology, The University of Tokyo
and
Computational Biology Research Center,
National Institute of Advanced Industrial Science and Technology
asai@k.u-tokyo.ac.jp

Abstract

When we have several related tasks, solving them simultaneously has been shown to be more effective than solving them individually. This approach is called *multi-task learning* (MTL). In this paper, we propose a novel MTL algorithm. Our method controls the relatedness among the tasks *locally*, so all pairs of related tasks are guaranteed to have similar solutions. We apply the above idea to support vector machines and show that the optimization problem can be cast as a *second-order cone program*, which is convex and can be solved efficiently. The usefulness of our approach is demonstrated in ordinal regression, link prediction and collaborative filtering, each of which can be formulated as a structured multi-task problem.

Keywords

Multi-Task Learning, Second-Order Cone Programming, Ordinal Regression, Link Prediction, Collaborative Filtering

1 Introduction

We often encounter several *related* classification tasks. Since the related tasks tend to share common factors, solving them together is expected to be more advantageous than solving them independently. This approach is called *multi-task learning* (MTL, a.k.a. *inductive transfer* or *learning to learn*) and has been theoretically and experimentally proven to be useful [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].

Typically, the “relatedness” among tasks is implemented by requiring that the solutions of related tasks be similar. However, the MTL methods developed so far have a limitation—the related tasks are often required to be close in the sense that the *sum* of the distances between solutions over all related tasks is upper-bounded [8]. Such a constraint is often referred to as a *global* constraint [11]. This implies that all the solutions of related tasks are not necessarily close, but that some can be quite different.

In this paper, we propose a new MTL method that overcomes the above limitation. We solve the problem by directly upper-bounding *each* distance between the solutions of every task pair (which we call *local* constraints). Furthermore, structured task relation in the form of a *task-relation network* could be naturally handled in our approach.

We apply this idea in the framework of support vector machines (SVM) and show that linear SVMs can be trained via a *second-order cone program* (SOCP) [12] in the primal. An SOCP is a convex problem and the global solution can be computed efficiently. We further show that the kernelized version of the proposed method can be formulated as a *matrix-fractional program* (MFP) [12] in the dual, which can also be cast as an SOCP; thus the optimization problem of the kernelized variant is still convex and the global solution can be computed efficiently. Through experiments, we show that the proposed MTL method compares favorably with existing MTL methods.

The MTL idea is also useful in

Multi-class classification: Predicting class labels out of several classes [13],

Ordinal regression: Predicting ordinal class labels such as users’ preferences (“like”/“neutral”/“dislike”) [14, 8],

Link prediction: Given a partially linked network, predicting the existence of links between unchecked nodes [15, 16, 17, 18],

Collaborative filtering: Predicting a user’s taste using relevant contents for the purpose of making recommendations [19].

We experimentally show that the proposed method tends to outperform existing ordinal regression, link prediction, and collaborative filtering methods.

This paper is organized as follows. In Section 2, we propose a new multi-task learning method for linear SVMs. In Section 3, we extend the proposed method to kernelized SVMs. In Section 4, we discuss the relation between the proposed and existing methods in standard multi-task, ordinary regression, link prediction, and collaborative filtering scenarios. In Section 5, we show the usefulness of the proposed method through experiments. Finally in Section 6, we make concluding remarks.

Notation We denote vectors by bold-faced lower-case letters and matrices by bold-faced upper-case letters. Elements of vectors and matrices are not bold-faced. The transposition of a matrix \mathbf{A} is denoted by \mathbf{A}^\top , and the inverse of \mathbf{A} is by \mathbf{A}^{-1} . The $n \times n$ identity matrix is denoted by \mathbf{I}_n . We use \mathbf{E}_{ij} to denote a matrix in which (i, j) element is one and all the others are zero. The n -dimensional vector all of whose elements are one is denoted by $\mathbf{1}_n$. We use \mathbb{R} and \mathbb{N} to denote the set of real and natural numbers, \mathbb{R}^n and \mathbb{N}^n to denote the set of n -dimensional real and natural vectors, and $\mathbb{R}^{m \times n}$ to denote the set of $m \times n$ real matrices. The set of real nonnegative numbers is denoted by \mathbb{R}_+ . For $\forall n \in \mathbb{N}$, we use \mathbb{N}_n to denote the set of natural numbers less than or equal to n . We use \mathbb{S}^n to denote the set of symmetric $n \times n$ matrices, \mathbb{S}_+^n to denote the set of symmetric positive semi-definite $n \times n$ matrices, and \mathbb{S}_{++}^n to denote the set of symmetric strictly positive definite $n \times n$ matrices. The symbols \leq and \geq are used to denote not only the standard inequalities between scalars, but also the componentwise inequalities between vectors.

Problem Setting Let us consider M binary classification tasks that all share the common input-output space $\mathcal{X} \times \{\pm 1\}$ [20]. For the time being, we assume $\mathcal{X} \subset \mathbb{R}^d$ for simplicity; later in Section 3, we extend \mathbb{R}^d to reproducing kernel Hilbert spaces. Assume we have M tasks to learn, and i -th task has n_i data samples $(\mathbf{x}_{t,i}, y_{t,i}) \in \mathcal{X} \times \{\pm 1\}$ ($t = 1, \dots, n_i$). Let ℓ the total number of samples, i.e. $\ell \equiv \sum_{i=1}^M n_i$.

The goal is to learn the score function of each classification task:

$$f_i(\mathbf{x}; \mathbf{w}_i, b_i) = \mathbf{w}_i^\top \mathbf{x} + b_i, \quad \text{for } i = 1, \dots, M,$$

where $\mathbf{w}_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$ are the model parameters of the i -th task.

2 Local MTL with a Task Network: Linear Version

In this section, we propose a new MTL method.

2.1 Basic Idea

When the relation among tasks is not available, we may just solve M penalized fitting problems individually:

$$\frac{1}{2} \|\mathbf{w}_i\|^2 + C_\alpha \sum_{t=1}^{n_i} \text{Hinge}(f_i(\mathbf{x}_{t,i}; \mathbf{w}_i, b_i), y_{t,i}), \quad (1)$$

for $i = 1, \dots, M$,

where a positive scalar $C_\alpha \in \mathbb{R}_+$ is a regularization constant and $\text{Hinge}(\cdot, \cdot)$ is the *hinge* loss function:

$$\text{Hinge}(f, y) \equiv \max(1 - fy, 0).$$

Eq.(1) is known as the support vector machine (SVM) [21]. The first term is the inverse of the margin between two classes. Thus SVM tries to find the hyper-plane which separates two classes with the maximum margin.

This individual approach tends to perform poorly if the number of training samples in each task is limited—the performance is expected to be improved if more training samples are available.

To cope with this problem, here we take another approach that is based on the expectation that the *solutions* of related tasks are close to each other. More specifically, we impose the following constraint on the optimization problem (1):

$$\forall i, \forall j : \frac{1}{2} \|\mathbf{w}_i - \mathbf{w}_j\|^2 \leq \rho. \quad (2)$$

Namely, we upper-bound the difference between the solutions of tasks by a non-negative scalar $\rho \in \mathbb{R}_+$. We refer to this constraint as a *local constraint* following Tsuda and Noble [11]. Note that we do not impose a constraint on the bias parameter b_i since the bias could be significantly different even among related tasks. In the rest of this paper, we focus on using the single upper bound ρ . But it is straightforward to generalize this by replacing $\rho_{i,j}$. The constraint (2) allows us to *implicitly* increase the number of training samples through the solutions of related tasks.

Following convention [8], we blend Eqs.(1) and (2) as

$$\frac{1}{2M} \sum_{i=1}^M \|\mathbf{w}_i\|^2 + C_\alpha \sum_{i=1}^M \sum_{t=1}^{n_i} \text{Hinge}(f_i(\mathbf{x}_{t,i}; \boldsymbol{\theta}), y_{t,i}) + C_\rho \rho,$$

where $C_\rho \in \mathbb{R}_+$ is a non-negative trade-off parameter. Then our optimization problem is summarized as follows:

Problem 1

$$\begin{aligned} \min \quad & \frac{1}{2M} \sum_{i=1}^M \|\mathbf{w}_i\|^2 + C_\alpha \|\boldsymbol{\xi}\|_1 + C_\rho \rho, \\ \text{wrt} \quad & \mathbf{w} \in \mathbb{R}^{Md}, \mathbf{b} \in \mathbb{R}^M, \boldsymbol{\xi}_\alpha \in \mathbb{R}_+^\ell, \rho \in \mathbb{R}_+, \\ \text{subj. to} \quad & \forall i, \forall j \in \mathbb{N}_M : \frac{1}{2} \|\mathbf{w}_i - \mathbf{w}_j\|^2 \leq \rho, \\ & \forall i \in \mathbb{N}_M, \forall t \in \mathbb{N}_{n_i} : y_{t,i} (\mathbf{w}_i^\top \mathbf{x}_{t,i} + b_i) \geq 1 - \xi_{t,i}^\alpha, \\ \text{where} \quad & \mathbf{w} \equiv [\mathbf{w}_1^\top, \dots, \mathbf{w}_M^\top]^\top, \\ & \boldsymbol{\xi}_\alpha \equiv [\xi_{1,1}^\alpha, \dots, \xi_{n_1,1}^\alpha, \xi_{1,2}^\alpha, \dots, \xi_{n_M,M}^\alpha]^\top. \end{aligned}$$

Generally, fewer constraints lead to more efficient computation for a convex problem. A bottleneck of Problem 1 is constraints for all the pairs of \mathbf{w}_i . Hence the problem requires enormous computational time. We associate the constraints with a fully connected network of tasks. We will show that the generalization performance does not degenerate even

if some edges are removed. Let us denote the set of the remaining edges by

$$\mathcal{E} \equiv \{i_k, j_k\}_{k=1}^K.$$

The optimization problem is then expressed as follows:

Problem 2

$$\begin{aligned} \min \quad & \frac{1}{2M} \sum_{i=1}^M \|\mathbf{w}_i\|^2 + C_\alpha \|\boldsymbol{\xi}\|_1 + C_\rho \rho, \\ \text{wrt} \quad & \mathbf{w} \in \mathbb{R}^{Md}, \mathbf{b} \in \mathbb{R}^M, \boldsymbol{\xi}_\alpha \in \mathbb{R}_+^\ell, \rho \in \mathbb{R}_+, \\ \text{subj. to} \quad & \forall k \in \mathbb{N}_K : \frac{1}{2} \|\mathbf{w}_{i_k} - \mathbf{w}_{j_k}\|^2 \leq \rho, \\ & \forall i \in \mathbb{N}_M, \forall t \in \mathbb{N}_{n_i} : y_{t,i} (\mathbf{w}_i^\top \mathbf{x}_{t,i} + b_i) \geq 1 - \xi_{t,i}^\alpha, \\ \text{where} \quad & \mathbf{w} \equiv [\mathbf{w}_1^\top, \dots, \mathbf{w}_M^\top]^\top, \\ & \boldsymbol{\xi}_\alpha \equiv [\xi_{1,1}^\alpha, \dots, \xi_{n_1,1}^\alpha, \xi_{1,2}^\alpha, \dots, \xi_{n_M,M}^\alpha]^\top. \end{aligned}$$

Hereinafter, the network with \mathcal{E} is referred to as the *task network*. We assume that the task network is given a priori. Learning task networks is a challenging open research issue and this is beyond the scope of the current paper.

2.2 Primal MTL Learning by SOCP

The SOCP is a class of convex programs for minimizing a linear function over an intersection of second-order cones [12].¹

Problem 3

$$\begin{aligned} \min \quad & \mathbf{f}^\top \mathbf{z} \quad \text{wrt} \quad \mathbf{z} \in \mathbb{R}^n, \\ \text{subj. to} \quad & \forall i \in \mathbb{N} : \|\mathbf{A}_i \mathbf{z} + \mathbf{b}_i\| \leq \mathbf{c}_i^\top \mathbf{z} + d_i, \\ \text{where} \quad & \mathbf{f} \in \mathbb{R}^n, \\ & \forall i \in \mathbb{N} : \mathbf{A}_i \in \mathbb{R}^{(n_i-1) \times n}, \mathbf{b}_i \in \mathbb{R}^{n_i-1}, \\ & \mathbf{c}_i \in \mathbb{R}^n, d_i \in \mathbb{R}. \end{aligned}$$

Linear programs, quadratic programs, and quadratically-constrained quadratic programs are special cases of SOCPs. SOCPs, a sub-class of semidefinite programs (SDP) [12], can be solved more efficiently than most other SDPs. Interior-point algorithms are successful optimization algorithms for both SDPs and SOCPs. Standard SDP solvers (e.g. [22]) consume $O(n^2 \sum_i n_i^2)$ time² for solving Problem 3, but the SOCP-specialized solvers that directly solve Problem 3 take only $O(n^2 \sum_i n_i)$ computation time [23]. Thus, SOCPs can indeed be solved more efficiently than SDPs.

We show that Problem 2 can be cast as an SOCP.

¹More generally, an SOCP can include linear equality constraints, but they can be eliminated, for example, by some projection method.

²This corresponds to the computational complexity per iteration.

Theorem 1 *Problem 2 can be reduced to the following SOCP:*

Problem 4

$$\begin{aligned}
 \min \quad & \frac{1}{M}t_w + C_\alpha \mathbf{1}_\ell^\top \boldsymbol{\xi} + C_\rho \rho, \\
 \text{wrt} \quad & t_w \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^{Md}, \mathbf{b} \in \mathbb{R}^M, \boldsymbol{\xi}_\alpha \in \mathbb{R}_+^\ell, \rho \in \mathbb{R}_+, \\
 \text{subj. to} \quad & \left\| \begin{bmatrix} 2\mathbf{w} \\ 2t_w - 1 \end{bmatrix} \right\| \leq 2t_w + 1, \\
 & \forall k \in \mathbb{N}_K : \left\| \begin{bmatrix} 2(\mathbf{w}_{i_k} - \mathbf{w}_{j_k}) \\ 2\rho - 1 \end{bmatrix} \right\| \leq 2\rho + 1, \\
 & \forall i \in \mathbb{N}_M \forall t \in \mathbb{N}_{n_i} : y_{t,i} (\mathbf{w}_i^\top \mathbf{x}_{t,i} + b_i) \geq 1 - \xi_{t,i}^\alpha, \\
 \text{where} \quad & \mathbf{w} \equiv [\mathbf{w}_1^\top, \dots, \mathbf{w}_M^\top]^\top, \\
 & \boldsymbol{\xi}_\alpha \equiv [\xi_{1,1}^\alpha, \dots, \xi_{n_1,1}^\alpha, \xi_{1,2}^\alpha, \dots, \xi_{n_M,M}^\alpha]^\top.
 \end{aligned}$$

This theorem can be proved directly from the following lemma:

Lemma 1 (Hyperbolic constraints) [12] *For $\forall \mathbf{w} \in \mathbb{R}^d, \forall x, \forall y \in \mathbb{R}$,*

$$\|\mathbf{w}\|^2 \leq xy, \quad x \geq 0, \quad y \geq 0 \quad \iff \quad \left\| \begin{bmatrix} 2\mathbf{w} \\ x - y \end{bmatrix} \right\| \leq x + y.$$

Problem 4 can be solved with $O((Md + \ell)^2(Kd + \ell))$ computation time.

3 Local MTL with a Task Network: Kernelization

In the previous section, we showed that a linear version of the proposed MTL method can be cast as an SOCP. In this section, we show how the kernel trick can be employed for obtaining a non-linear variant.

3.1 Dual Formulation

Let \mathbf{K}_{fea} be a positive semidefinite matrix with the (s, t) -th element being the inner-product of feature vectors \mathbf{x}_s and \mathbf{x}_t :

$$K_{s,t}^{\text{fea}} \equiv \langle \mathbf{x}_s, \mathbf{x}_t \rangle.$$

This is a kernel matrix of feature vectors. We also introduce a kernel among tasks. Using a new K -dimensional non-negative parameter vector $\boldsymbol{\lambda} \in \mathbb{R}_+^K$, we define the kernel matrix of tasks by

$$\mathbf{K}_{\text{net}}(\boldsymbol{\lambda}) \equiv \left(\frac{1}{M} \mathbf{I}_M + \mathcal{U} \boldsymbol{\lambda} \right)^{-1},$$

where

$$\begin{aligned}\mathcal{U}\boldsymbol{\lambda} &\equiv \sum_{k=1}^K \lambda_k \mathbf{U}_k, \\ \mathbf{U}_k &\equiv \mathbf{E}^{i_k i_k} + \mathbf{E}^{j_k j_k} - \mathbf{E}^{i_k j_k} - \mathbf{E}^{j_k i_k}.\end{aligned}$$

$\mathbf{E}^{(i,j)} \in \mathbb{R}^{M \times M}$ is the sparse matrix whose (i, j) -th element is one and all the others are zero. Note that this is the graph Laplacian kernel [24], where the k -th edge is weighted according to λ_k . Let $\mathbf{Z} \in \mathbb{N}^{M \times \ell}$ be the indicator of a task and a sample such that

$$\mathbf{Z}^\top \equiv \begin{bmatrix} \mathbf{1}_{n_1} & \mathbf{0}_{n_1} & \cdots & \mathbf{0}_{n_1} \\ \mathbf{0}_{n_2} & \mathbf{1}_{n_2} & \cdots & \mathbf{0}_{n_2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{n_M} & \mathbf{0}_{n_M} & \cdots & \mathbf{1}_{n_M} \end{bmatrix}.$$

Then the information about the tasks are expressed by the $\ell \times \ell$ kernel matrix

$$\mathbf{Z}^\top \mathbf{K}_{\text{net}}(\boldsymbol{\lambda}) \mathbf{Z}.$$

We integrate the two kernel matrices \mathbf{K}_{fea} and $\mathbf{Z}^\top \mathbf{K}_{\text{net}}(\boldsymbol{\lambda}) \mathbf{Z}$ by

$$\mathbf{K}_{\text{int}}(\boldsymbol{\lambda}) \equiv \mathbf{K}_{\text{fea}} \circ (\mathbf{Z}^\top \mathbf{K}_{\text{net}}(\boldsymbol{\lambda}) \mathbf{Z}),$$

where \circ denotes the *Hadamard product* (a.k.a the *element-wise product*). This parameterized matrix $\mathbf{K}_{\text{int}}(\boldsymbol{\lambda})$ is guaranteed to be positive semidefinite [25].

Based on the above notations, we have the following theorem.

Theorem 2 *The dual formulation of Problem 2 can be expressed using the parameterized integrated kernel matrix $\mathbf{K}_{\text{int}}(\boldsymbol{\lambda})$ as follows:*

Problem 5

$$\begin{aligned}\min & \quad \frac{1}{2} \boldsymbol{\alpha}^\top \text{diag}(\mathbf{y}) \mathbf{K}_{\text{int}}(\boldsymbol{\lambda}) \text{diag}(\mathbf{y}) \boldsymbol{\alpha} - \|\boldsymbol{\alpha}\|_1, \\ \text{wrt} & \quad \boldsymbol{\alpha} \in \mathbb{R}_+^\ell, \boldsymbol{\lambda} \in \mathbb{R}_+^K, \\ \text{subj. to} & \quad \boldsymbol{\alpha} \leq C_\alpha \mathbf{1}_\ell, \quad \mathbf{Z} \text{diag}(\mathbf{y}) \boldsymbol{\alpha} = \mathbf{0}_M, \quad \|\boldsymbol{\lambda}\|_1 \leq C_\rho.\end{aligned}$$

For $\forall i, \forall t \in \mathbb{N}_{n_i}$, we use $\alpha_{t,i}$ to denote the elements of the dual variable $\boldsymbol{\alpha} \in \mathbb{R}_+^\ell$ such that

$$\boldsymbol{\alpha} = [\alpha_{1,1}, \dots, \alpha_{n_1,1}, \alpha_{1,2}, \dots, \alpha_{n_{M-1},M-1}, \alpha_{1,M}, \dots, \alpha_{1,M}].$$

The proof of the above theorem is summarized in Appendix A. We note that the solutions $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ tend to be sparse due to the ℓ_1 norm.

Changing the definition of \mathbf{K}_{fea} from the linear kernel to an arbitrary kernel, we can extend the proposed linear MTL method to non-linear domains. Furthermore, we can

also deal with non-vectorial structured data by employing a suitable kernel such as the string kernel or the Fisher kernel [26, 27, 28, 29, 30, 31, 32, 33].

In the test stage, a new sample \mathbf{x} in the j -th task is classified by

$$f_j(\mathbf{x}) = \sum_{i=1}^M \sum_{t=1}^{n_i} \alpha_{t,i} y_{t,i} k_{\text{fea}}(\mathbf{x}_{t,i}, \mathbf{x}) k_{\text{net}}(i, j) + b_j,$$

where $k_{\text{fea}}(\cdot, \cdot)$ and $k_{\text{net}}(\cdot, \cdot)$ are the kernel functions over features and tasks, respectively.

3.2 Dual MTL Learning by SOCP

Here, we show that the above dual problem can also be reduced to an SOCP. To this end, we first introduce a *matrix-fractional program* (MFP) [23]:

Problem 6

$$\begin{aligned} \min \quad & (\mathbf{F}\mathbf{z} + \mathbf{g})^\top \mathbf{P}(\mathbf{z})^{-1} (\mathbf{F}\mathbf{z} + \mathbf{g}), \\ \text{wrt} \quad & \mathbf{z} \in \mathbb{R}_+^p, \\ \text{subj. to} \quad & \mathbf{P}(\mathbf{z}) \equiv \mathbf{P}_0 + \sum_{i=1}^p z_i \mathbf{P}_i \in \mathbb{S}_{++}^n, \\ \text{where} \quad & \mathbf{P}_i \in \mathbb{S}_+^n, \quad \mathbf{F} \in \mathbb{R}^{n \times p}, \mathbf{g} \in \mathbb{R}^n. \end{aligned}$$

Let us re-define d as the rank of the feature kernel matrix \mathbf{K}_{fea} . We introduce a matrix $\mathbf{V}_{\text{fea}} \in \mathbb{R}^{\ell \times d}$ which decomposes the feature kernel matrix as

$$\mathbf{K}_{\text{fea}} = \mathbf{V}_{\text{fea}} \mathbf{V}_{\text{fea}}^\top.$$

By defining the ℓ -dimensional vectors $\mathbf{f}_h \in \mathbb{R}^\ell$ corresponding to the h -th feature as

$$\mathbf{V}_{\text{fea}} \equiv [\mathbf{f}_1, \dots, \mathbf{f}_d] \in \mathbb{R}^{\ell \times d},$$

and the matrices

$$\mathbf{F}_h \equiv \mathbf{Z} \text{diag}(\mathbf{f}_h \circ \mathbf{y}), \quad \text{for } h = 1, \dots, d,$$

we have the following lemma.

Lemma 2 *The objective function of Problem 5 can be rewritten as*

$$\frac{1}{2} \sum_{h=1}^d \boldsymbol{\alpha}^\top \mathbf{F}_h^\top \left(\frac{1}{M} \mathbf{I}_M + \mathcal{U}\boldsymbol{\lambda} \right)^{-1} \mathbf{F}_h \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{1}_\ell.$$

A proof of the above lemma is given in Appendix B. Lemma 2 implies that Problem 5 can be transformed into a combination of a linear program and d MFPs.

Let us further introduce the vector $\mathbf{v}_k \in \mathbb{R}^M$ for each edge:

$$\mathbf{v}_k = \mathbf{e}_{i_k} - \mathbf{e}_{j_k},$$

where \mathbf{e}_{i_k} is a unit vector with the i_k -th element being one. Let \mathbf{V}_{lap} be a matrix defined by

$$\mathbf{V}_{\text{lap}} = [\mathbf{v}_1, \dots, \mathbf{v}_K] \in \mathbb{R}^{M \times K}.$$

Then we can re-express the graph Lagrangian matrix of tasks as

$$\mathcal{U}\boldsymbol{\lambda} = \mathbf{V}_{\text{lap}} \text{diag}(\boldsymbol{\lambda}) \mathbf{V}_{\text{lap}}^\top.$$

Given the fact that an MFP can be reduced to an SOCP [23], we have the following theorem.

Theorem 3 *The dual problem (Problem 5) can be reduced to the following SOCP:*

Problem 7

$$\begin{aligned} \min \quad & -\mathbf{1}_\ell^\top \boldsymbol{\alpha} + \frac{1}{2} \sum_{h=1}^d s_{0,h} + s_{1,h} + \dots + s_{K,h}, \\ \text{wrt} \quad & \forall h \in \mathbb{N}_d, \forall k \in \mathbb{N}_K : s_{0,h} \in \mathbb{R}, s_{k,h} \in \mathbb{R}, \\ & \forall h \in \mathbb{N}_d : \mathbf{u}_{0,h} \in \mathbb{R}^M, \mathbf{u}_h = [u_{1,h}, \dots, u_{K,h}]^\top \in \mathbb{R}^K, \\ & \boldsymbol{\lambda} \in \mathbb{R}_+^K, \boldsymbol{\alpha} \in \mathbb{R}_+^\ell, \\ \text{subj. to} \quad & \boldsymbol{\alpha} \leq C_\alpha \mathbf{1}_\ell, \quad \mathbf{Z} \text{diag}(\mathbf{y}) \boldsymbol{\alpha} = \mathbf{0}_M, \\ & \mathbf{1}_K^\top \boldsymbol{\lambda} \leq C_\rho, \\ & \forall h \in \mathbb{N}_d : M^{-1/2} \mathbf{u}_{0,h} + \mathbf{V}_{\text{lap}} \mathbf{u}_h = \mathbf{F}_h \boldsymbol{\alpha}, \\ & \forall h \in \mathbb{N}_d : \left\| \begin{bmatrix} 2\mathbf{u}_{0,h} \\ s_{0,h} - 1 \end{bmatrix} \right\| \leq s_{0,h} + 1, \\ & \forall h \in \mathbb{N}_d, \forall k \in \mathbb{N}_K : \left\| \begin{bmatrix} 2u_{k,h} \\ s_{k,h} - \lambda_k \end{bmatrix} \right\| \leq s_{k,h} + \lambda_k, \end{aligned}$$

The proof is given in Appendix C. Problem 7 can be solved with $O((Kd + \ell)^2((M + K)d + \ell))$ computation time.

4 Discussions

In this section, we discuss the properties of the proposed MTL method and the relation to existing methods.

4.1 MTL with Common Bias

A possible variant of the proposed MTL method would be to share the bias parameter with all tasks (i.e., $b_1 = b_2 = \dots = b_M$). The idea is expected to be useful when the number of samples in each task is very small since overfitting can be avoided. We can also apply the common bias idea in the kernelized version just by replacing the constraint

$$\mathbf{Z} \text{diag}(\mathbf{y})\boldsymbol{\alpha} = \mathbf{0}_M$$

in Problem 5 by

$$\mathbf{y}^\top \boldsymbol{\alpha} = 0.$$

4.2 Relation to Standard SVMs

By construction, the proposed MTL method includes the standard SVM learning algorithm as a special case. Indeed, when the number of tasks is one, Problem 5 is reduced to the standard SVM optimization problem. Thus, the proposed method may be regarded as a natural extension of SVMs.

When the task network is completely unconnected, we end up in training SVMs individually:

$$\frac{1}{2} \|\mathbf{w}_i\|^2 + C_\alpha \sum_{t=1}^{n_i} \text{Hinge}(f_i(\mathbf{x}_{t,i}; \mathbf{w}_i, b_i), y_{t,i}),$$

for $i = 1, \dots, M$.

We refer to this as individually learned SVM (**IL-SVM**) for comparison purposes. IL-SVM can also be recovered by setting $C_\rho = 0$ in the proposed method.

4.3 Global/Local Constraints

In the papers [5, 6] an MTL method has been proposed for SVMs that upper-bounds the *sum* of the distances between solutions over all pairs of related tasks:

$$\frac{1}{2} \sum_{i,j=1}^M \|\mathbf{w}_i - \mathbf{w}_j\|^2 \leq \rho.$$

Upper-bounding the sum is often referred to as a *global* constraint [11], and requiring the solutions of all tasks to be close can be regarded as using a *fully-connected* task network. For this reason, we refer to the above approach as **MTL-SVM (global/full)**. The global constraint can allow some of the distances to be large since only the sum is globally upper-bounded. In practice, this causes a significant performance degradation, which will be experimentally shown in Section 5. In contrast, we proposed upper-bounding *each* distance between the solutions of task pairs:

$$\frac{1}{2} \|\mathbf{w}_i - \mathbf{w}_j\|^2 \leq \rho, \quad \text{for } \forall i, \forall j \in \mathbb{N}_M.$$

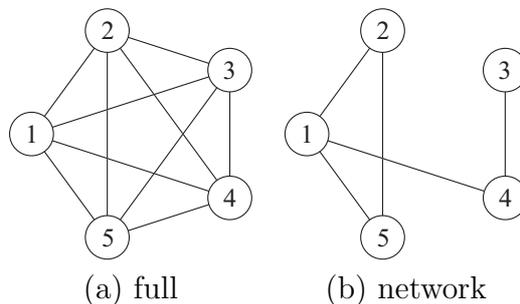


Figure 1: A main feature of our MTL-SVM is to upper-bound the difference between model parameters of each task pair. The upper-bounds in MTL-SVM (local/full) are applied to all task pairs as in (a), whereas those in MTL-SVM (local/network) are to a part of task pairs as in (b). MTL-SVM (global/full) and MTL-SVM (global/network) upper-bound the sum of differences for all task pairs and a part of task pairs, respectively.

Thus, all the task pairs are guaranteed to have similar solutions. Upper-bounding each component is often referred to as a *local* constraint [11], and we refer to our approach as **MTL-SVM (local/full)** below.

Micchelli and Pontil [8] proposed another formulation to represent particular pairs being strongly related:

$$\frac{1}{2} \sum_{k=1}^K \|\mathbf{w}_{i_k} - \mathbf{w}_{j_k}\|^2 \leq \rho,$$

where the tasks i_k and j_k ($k = 1, \dots, K$) are related to each other, but the rest are not. Since task network information is utilized, we refer to this approach as **MTL-SVM (global/network)**. Our local-constraint method could also utilize a task network as

$$\frac{1}{2} \|\mathbf{w}_{i_k} - \mathbf{w}_{j_k}\|^2 \leq \rho, \quad \text{for } \forall k \in \mathbb{N}_K.$$

We refer to this approach as **MTL-SVM (local/network)** below. Our algorithm is not included in the formulation of Micchelli and Pontil [8] and therefore is novel. The four methods are summarized in Figure 1.

4.4 Ordinal Regression

MTL approaches are also useful in ordinal regression problems. The goal of ordinal regression is to predict ordinal class labels such as users' preferences ("like"/"neutral"/"dislike") or students' grades (from "A" to "F"). The ordinal regression problems can be formulated as a set of binary classification problems. For example, in the case of students' grades

from A to F, we have the following five binary classification problems:

$$\begin{aligned} & A \text{ vs. } B, C, D, E, F, \\ & A, B \text{ vs. } C, D, E, F, \\ & A, B, C \text{ vs. } D, E, F, \\ & A, B, C, D \text{ vs. } E, F, \\ & A, B, C, D, E \text{ vs. } F. \end{aligned}$$

A naive approach to ordinal regression is IL-SVM, but the approach does not utilize the relatedness among tasks. Alternatively, Shashua and Levin [14] have proposed an ordinal regression method called the *support vector ordinal regression* (SVOR), where the weight vectors are shared by all SVMs (i.e., $\mathbf{w}_1 = \mathbf{w}_2 = \dots = \mathbf{w}_M$) and only the bias parameter is learned individually.

If we allow \mathbf{w}_i ($i = 1, \dots, M$) to be slightly different from each other, the ordinal regression problem becomes an MTL problem. Since the task network is not fully connected in the ordinal regression setup (i.e., the task network only has a weight between consecutive tasks), MTL-SVM (local/network) would be a suitable choice:

$$\|\mathbf{w}_i - \mathbf{w}_{i+1}\|^2 \leq \rho, \quad i = 1, \dots, M - 1,$$

This method includes the above two ordinal regression approaches as special cases. $C_\rho = 0$ (i.e., ignoring the task network) yields IL-SVM and $C_\rho = \infty$ (i.e., the weight vectors of all SVMs agree) is reduced to SVOR. Thus, in the context of ordinal regression, the proposed method smoothly bridges two extremes.

In the next section, we experimentally show that the proposed MTL-SVM (local/full) and MTL-SVM (local/network) method tends to outperform existing approaches.

4.5 Link Prediction

We can also use the MTL methods in link prediction problems. Suppose we are given a graph \mathcal{G} with n nodes and undirected links. We denote the set of vertices by

$$\mathcal{V} = \{1, \dots, n\},$$

and all the pairs of nodes by

$$\mathcal{P} \equiv \{(i, j) \mid 1 \leq i < j \leq n\}.$$

Note that the number of elements in \mathcal{P} is $n(n-1)/2$. The set of links \mathcal{E} is a subset of \mathcal{P} , and the node pairs in the complementary set $\bar{\mathcal{E}} \equiv \mathcal{P} \setminus \mathcal{E}$ are not linked. We express the link information as

$$y_i^j = \begin{cases} +1 & \text{if } (i, j) \in \mathcal{E}, \\ -1 & \text{if } (i, j) \notin \mathcal{E}. \end{cases}$$

Each node is assumed to be given a d -dimensional feature vector \mathbf{x}_i . Let \mathcal{P}_{tra} be a subset of \mathcal{P} such that we know whether each pair of nodes in \mathcal{P}_{tra} is linked or not (i.e., pairs in \mathcal{P}_{tra} are “labeled”). The goal is to predict the presence or the absence of the links of the “unlabeled” pairs of nodes in $\mathcal{P}_{\text{tst}} \equiv \mathcal{P} \setminus \mathcal{P}_{\text{tra}}$.

A standard approach to link prediction is to employ a global model that accounts for the entire network [15, 16, 17]. However, a global model trained with all of the training information can be heavily affected by irrelevant (or noisy) information. To cope with this problem, Bleakley et al. [18] proposed using local models (i.e., one local model for one node). In this method, each local model is trained with only local information (i.e., the training samples related to the target node) and therefore the effect of noise may be alleviated. However, since the amount of training information available in a local region is limited, local models trained with a limited amount of information are unreliable. Indeed, the above local-model approach corresponds to IL-SVM in the MTL context and is therefore not effective.

In contrast, our proposed MTL-SVM can be effectively used in link prediction by regarding local models of the linked neighbors in the network as related tasks. This approach is motivated by the following consideration. Many of the complex networks in nature share interesting properties [34]. One of these properties is that a node A is likely to be linked with node B when they share a common neighbor (e.g., A and B are likely to be friends if they share a common friend). We exploit this property in the context of MTL learning by requiring the model parameter of node A to be similar to the model parameter of node B. This requirement can be effectively fulfilled by our MTL-SVM (local/network) approach.

4.6 Collaborative Filtering

Collaborative filtering is a machine-learning problem to fill in missing elements of an incomplete table. Let us denote an $m \times n$ incomplete binary table by $\mathbf{Y} \in \mathbb{R}^{m \times n}$. Each element Y_{ji} takes ± 1 or 0, where 0 represents a missing value. Thus, the goal of the collaborative filtering task addressed here is to predict the binary labels of the missing elements with $Y_{ji} = 0$.

Let us denote the matrix \mathbf{Y} as

$$\mathbf{Y} = [\mathbf{x}_1, \dots, \mathbf{x}_n] = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_m \end{bmatrix}, \quad (3)$$

where the i -th column and the j -th row are denoted by $\mathbf{x}_i \in \mathbb{R}^{m \times 1}$ and $\mathbf{y}_j \in \mathbb{R}^{1 \times n}$, respectively. A supervised learning formulation of this collaborative filtering problem is to regard each column \mathbf{x}_i as an m -dimensional feature vector and train a classifier. More specifically, let us regard each row \mathbf{y}_j in the table as a task. For the j -th task, a binary classifier is trained using features \mathbf{x}_i and labels Y_{ji} for all $i \in \mathbb{N}$ such that $Y_{ji} = \pm 1$ (i.e., non-missing entries). Then the missing values are predicted using the trained classifier.

We can directly extend this formulation to a multi-task problem by finding related rows (tasks); then an MTL classifier could be employed in collaborative filtering. A practical idea of building a task network is that a task j' is judged to be related to the task j if the row j' has many non-missing elements with the same labels as the task j . We will show through experiments that the proposed MTL method is useful in this scenario.

5 Experiments

In this section, we experimentally investigate the usefulness of the proposed method.

5.1 Protein Super-Family Classification

We test the performance of the proposed method using real-world protein super-family classification problems. Proteins are macromolecules in a cell. There are many kinds of proteins which function differently and cooperatively for survival of the cells. The function of a protein is determined by the tertiary structure, which is three dimensional coordinates of the atoms included in the macromolecules. Hence, the tertiary structures are a powerful clue to investigate the function of the proteins. However, the tertiary structure of a protein is not always available because measuring three dimensional coordinates of the atoms needs extremely expensive experiments. On the other hand, protein amino acid sequences are abundantly available. We here infer the class of tertiary structures from the amino acid sequences.

We counted 2-mers for extraction of feature vectors from the amino acid sequences. There are 20 kinds of amino acids. Hence, the number of features is $20^2 = 400$. We predict the super-families of a protein classified in the *SCOP database* [35] (not SOCP). That is a multi-class classification problems. In this experiment, we used seven super-families in ‘Flavodoxin’ and ‘OB-Fold.’ If we take the one-vs-rest approach, the number of tasks is equal to the number of super-families.

We compare the proposed MTL-SVM (local/network) with IL-SVM, MTL-SVM (global/full), and MTL-SVM (global/network). In the network methods, we generate a random tree of tasks and use it as a task network. The values of the parameters, C_α and C_ρ , were determined by 3-fold cross-validation over the training set. We used RBF kernels (a.k.a. Gaussian kernels), where the parameter σ_{rbf}^2 was set to the median of the squared distances among all the training samples. We randomly picked five samples for each class of the training set. We compute the classification accuracies of the remaining test sequences. We repeat this procedure 50 times and take the average of the areas under the ROC (Receiver Operating Characteristic) curves. We use the one-sample t -test at the significance level 1% [36] to detect statistically significant differences.

The results are shown in Table 1. Each row shows the performance for each super-family. The last row is the average. MTL-SVM (local/network) generally outperforms all the other methods. Table 2 shows the computational time. The algorithms with local constraints are not fast. Especially, MTL-SVM (local/full) takes the longer computational

time than MTL-SVM (local/network). That is because MTL-SVM (local/full) has more dual variables. MTL-SVM (local/network) is faster, yet the prediction performance is not significantly different from MTL-SVM (local/full) for most of cases.

5.2 Ordinal Regression

We created six ordinal regression data sets as described in Table 3, where all the data sets were originally regression of real output values and the output values were divided into five quantiles. Therefore, the overall task could be divided into four isolated classification tasks, each of which estimates a quantile. We compare MTL-SVM (local/full) and MTL-SVM (local/network) with IL-SVM, *SVOR* [14] (see Section 4), MTL-SVM (global/full), and MTL-SVM (global/network). The values of the parameters, C_α and C_ρ , were determined by 3-fold cross-validation over the training set. We used RBF kernels (a.k.a. Gaussian kernels), where the parameter σ_{rbf}^2 was set to the median of the squared distances among all the training examples. We randomly picked ten samples for training. The remaining samples were used for evaluating the classification accuracies. Statistically significant differences were evaluated by using the one-sample t -test at the significance level 1% [36].

The averaged performance over 100 runs is described in Table 3, showing that the proposed MTL-SVM (local/full) is promising also in ordinal regression scenarios. The performance of MTL-SVM (local/network) is not significantly different from MTL-SVM (local/full), although it is computationally more expensive.

5.3 Link Prediction

We use two kinds of protein networks in yeast. One is the metabolic gene network. Genes encode various proteins, and some of them are enzymes. Enzymes accelerate many biochemical reactions occurring within cells. A series of reactions form metabolic pathways. Through the metabolic pathways, the first molecules are modified into the last products. Some products are used as a metabolic product. Some are stored in a cell. Some became substrates of another pathway. In the metabolic gene network, the nodes are enzymes. An edge is established if two enzymes catalyze successive reactions in any metabolic pathway. This network, which contains 769 proteins and 3702 undirected edges, is identical to the one used in Yamanishi et al [17].

We also tested our algorithms on a protein-protein interaction network. The network we used in this experiment has been provided by von Mering et al [37]. This interaction network is produced from the results of various biological experiments. Each edge is rated according to the confidence level: high, middle or low. We only used high confidence edges because they are supported by multiple experiments. By removing isolated proteins with no edge, we obtained a network of 984 proteins and 2438 edges. The edge in the interaction network indicates the physical interaction of two proteins.

We compared our algorithm with the approach of Bleakley et al. [18]. We determined C_α and C_ρ by 3-fold cross-validation over the training set. We took 100 nodes with the

highest degree-of-nodes, and learned the local model for each node to predict the links between that node and the other nodes. The average areas under the ROC (Receiver Operating Characteristic) curves for prediction of the enzyme network and protein interactions are shown in Tables 4 and 5. All the datasets except *ady* were also used by Lanckriet et al. [38]. The *ady* dataset is the set of feature vectors extracted by the approach described in Section 4.6. MTL-SVM (local/network) generally outperforms all the other methods. We confirmed this by using the one-sample *t*-test at the significance level 1% [36]. We can not show the performance of MTL-SVM (local/full) because MTL-SVM (local/full) is expected to require several months for learning this task.

5.4 Collaborative Filtering

We used the MovieLens dataset³, which includes a table consisting of ratings of movies by users. Our goal is to predict whether or not a target user rated (bought and watched) a target movie; thus we are predicting purchase information [39].

We constructed five tables, A, B, C, D, and E, from the entire dataset. Table A includes four categories, Action, Adventure, Animation, and Children’s. Table B includes Comedy and Crime. Table C includes Documentary, Drama, Fantasy, Film-Noir, Horror, and Musical. Table D includes Mystery, Romance, and Sci-Fi. Table E includes Thriller, War, and Western. We then selected users who rated more than 50 movies for each table. We created a binary table with the rows and the columns representing users and movies, respectively. The elements of the table are binary values whether the movie is rated by the corresponding user (+1) or the movie is not rated (−1). We randomly picked up 30% of the elements as the known elements, and the binary values of the remaining 70% were treated as missing and were predicted. We regard the task j' to be related to the task j if the users j and j' share the positive label for more than two movies. We determined C_α and C_ρ in our algorithm by 3-fold cross-validation over the training set.

We compared our algorithm with the latent semantic indexing (LSI) [19] and maximum margin matrix factorization (MMMF) [40] which are popular methods for collaborative filtering. We determined the hyper-parameter of LSI and MMMF by 3-fold cross-validation. The average areas under the ROC curves for collaborative filtering are shown in Table 6. This shows that MTL-SVM (local/network) achieved the best performance, and statistically significant differences were detected using the one-sample *t*-test at the significance level 1% [36]. MTL-SVM (local/full) obtained good results, but the algorithm took much of computational time. Indeed, we could not finish the experiments for Dataset A and Dataset D.

³<http://www.grouplens.org/>

Table 1: Protein super-family classification. The task networks are randomly generated trees. The bold-faced figures in the table represent the best accuracy. The underlined figures in the table indicate that the performance is not significantly different from the best accuracy.

Class	IL-SVM	MTL-SVM (global/full)	MTL-SVM (global/network)	MTL-SVM (local/full)	MTL-SVM (local/network)
1	<u>0.893</u> (0.033)	<u>0.896</u> (0.032)	<u>0.885</u> (0.053)	0.887 (0.040)	0.884 (0.043)
2	0.803 (0.066)	<u>0.827</u> (0.040)	0.820 (0.043)	0.833 (0.035)	<u>0.831</u> (0.038)
3	0.916 (0.013)	0.919 (0.016)	0.918 (0.018)	0.926 (0.012)	0.925 (0.012)
4	0.702 (0.103)	<u>0.762</u> (0.046)	<u>0.769</u> (0.041)	0.776 (0.055)	<u>0.771</u> (0.067)
5	0.938 (0.030)	0.948 (0.024)	<u>0.947</u> (0.027)	0.942 (0.023)	0.942 (0.026)
6	0.755 (0.085)	0.763 (0.057)	<u>0.782</u> (0.041)	0.786 (0.048)	<u>0.785</u> (0.061)
7	0.591 (0.061)	0.612 (0.067)	<u>0.616</u> (0.073)	0.629 (0.055)	<u>0.627</u> (0.060)
ave	0.800 (0.024)	0.818 (0.019)	0.819 (0.021)	0.825 (0.017)	<u>0.823</u> (0.020)

Table 2: Computation time.

Class	IL-SVM	MTL-SVM (global/full)	MTL-SVM (global/network)	MTL-SVM (local/full)	MTL-SVM (local/network)
Time (sec)	0.069 (0.006)	0.269 (0.007)	0.288 (0.016)	2.339 (0.089)	0.884 (0.057)

Table 3: The accuracy of each method in ordinal regression tasks. The bold-faced figures in the table represent the best accuracy. The underlined figures in the table indicate that the performance is not significantly different from the best accuracy.

Dataset	LSI	IL-SVM	MTL-SVM (global/full)	MTL-SVM (global/network)	MTL-SVM (local/full)	MTL-SVM (local/network)
abalone	0.965 (0.016)	0.965 (0.016)	0.965 (0.016)	0.965 (0.016)	0.972 (0.012)	0.966 (0.016)
bodyfat	0.957 (0.016)	0.958 (0.013)	0.958 (0.016)	0.958 (0.015)	0.962 (0.013)	0.962 (0.013)
cadata	0.974 (0.010)	0.974 (0.010)	0.974 (0.010)	0.974 (0.010)	0.975 (0.009)	0.975 (0.010)
housing	0.969 (0.012)	0.970 (0.012)	0.969 (0.012)	0.969 (0.012)	0.973 (0.011)	0.972 (0.011)
mg	0.969 (0.010)	0.969 (0.010)	0.969 (0.010)	0.969 (0.010)	0.969 (0.009)	0.970 (0.010)
mpg	0.964 (0.012)	0.966 (0.012)	0.963 (0.011)	0.963 (0.011)	0.969 (0.011)	0.969 (0.011)

Table 4: The AUCs for enzyme network prediction. The bold-faced figures in the table represent the best AUCs. The underlined figures in the table indicate that the performance is not significantly different from the best AUCs.

Dataset	IL-SVM	MTL-SVM (global/full)	MTL-SVM (global/network)	MTL-SVM (local/full)	MTL-SVM (local/network)
ady	0.733 (0.113)	0.744 (0.122)	0.746 (0.120)	n/a	0.752 (0.118)
blast	0.786 (0.102)	0.792 (0.104)	0.800 (0.097)	n/a	0.806 (0.098)
diff	0.620 (0.130)	0.630 (0.124)	0.639 (0.122)	n/a	0.654 (0.109)
expr	0.630 (0.104)	0.635 (0.109)	0.636 (0.107)	n/a	0.645 (0.100)
fft	0.652 (0.111)	0.663 (0.112)	0.668 (0.109)	n/a	0.675 (0.103)
lin_int	0.588 (0.111)	0.609 (0.129)	0.611 (0.127)	n/a	0.614 (0.119)
pfam_hmm	0.740 (0.123)	0.748 (0.123)	0.749 (0.125)	n/a	0.760 (0.124)
sw	0.732 (0.126)	0.731 (0.135)	0.725 (0.146)	n/a	0.753 (0.122)

Table 5: The AUCs for prediction of protein interactions. The bold-faced figures in the table represent the best AUCs. The underlined figures in the table indicate that the performance is not significantly different from the best AUCs.

Dataset	IL-SVM	MTL-SVM (global/full)	MTL-SVM (global/network)	MTL-SVM (local/full)	MTL-SVM (local/network)
ady	0.748 (0.139)	0.748 (0.151)	0.750 (0.151)	n/a	<u>0.761</u> (0.144)
blast	0.621 (0.095)	0.657 (0.096)	0.655 (0.095)	n/a	0.656 (0.095)
expr	0.820 (0.117)	0.832 (0.124)	<u>0.831</u> (0.123)	n/a	0.832 (0.123)
fft	0.591 (0.109)	<u>0.604</u> (0.109)	<u>0.603</u> (0.110)	n/a	0.612 (0.104)
pfam_hmm	0.628 (0.137)	<u>0.647</u> (0.136)	0.650 (0.128)	n/a	0.657 (0.127)
sw	0.693 (0.114)	0.678 (0.173)	0.672 (0.178)	n/a	0.717 (0.132)

Table 6: The AUCs for collaborative filtering. The bold-faced figures in the table represent the best performance. The underlined figures in the table indicate that the performance is not significantly different from the best AUCs.

Dataset	LSI	MMMMF	IL-SVM	MTL-SVM (global/full)	MTL-SVM (global/network)	MTL-SVM (local/full)	MTL-SVM (local/network)
A	0.764 (0.097)	0.760 (0.038)	0.796 (0.043)	0.822 (0.038)	0.813 (0.044)	n/a	0.827 (0.035)
B	0.770 (0.076)	<u>0.808</u> (0.046)	0.785 (0.037)	0.814 (0.029)	0.814 (0.029)	0.816 (0.029)	0.817 (0.028)
C	<u>0.801</u> (0.088)	0.809 (0.028)	0.803 (0.043)	0.835 (0.040)	0.835 (0.040)	<u>0.832</u> (0.042)	<u>0.832</u> (0.042)
D	0.776 (0.081)	0.778 (0.047)	0.786 (0.043)	0.816 (0.041)	0.806 (0.045)	n/a	0.817 (0.039)
E	0.770 (0.086)	0.777 (0.050)	0.767 (0.048)	0.798 (0.042)	0.793 (0.044)	0.802 (0.041)	0.803 (0.041)

6 Conclusions

We proposed a new multi-task learning method that overcomes the limitation of existing approaches by making use of local constraints. We demonstrated through simulations that the proposed method is useful in various multi-task learning scenarios.

There are several methods which learn the tasks and estimate the task network simultaneously using non-convex Bayesian approach [9, 41, 42, 43]. Future work includes development of a convex algorithm for it. In the papers [5, 6], global-constraint MTL has been addressed in the context of multiple kernel learning. A possible extension of our work along this line would be to formulate local-constraint MTL as multiple kernel learning and investigate their relation.

The standard SVMs have a variety of extensions and have been combined with various techniques, for example, one-class classification [44, 45], regression [46, 21, 47], and the ν -trick [48, 49, 50]. We expect that such extensions and techniques can also be applied similarly to the proposed method. Other possible future works include the elucidation of the entire regularization path [51, 52] and the application to learning from multiple networks [53, 11, 54]. Developing algorithms for learning probabilistic models with a task network is also a promising direction to be explored.

A Proof of Theorem 2

Let

$$\tilde{\mathbf{X}} \equiv (\mathbf{Z} \otimes \mathbf{1}_d) \circ (\mathbf{1}_M \otimes \mathbf{X}),$$

where we have defined

$$\mathbf{X} \equiv [\mathbf{x}_{1,1}, \dots, \mathbf{x}_{n_1,1}, \mathbf{x}_{1,2}, \dots, \mathbf{x}_{n_{M-1},M-1}, \mathbf{x}_{1,M}, \dots, \mathbf{x}_{n_M,M}]$$

and the operators \otimes and \circ are the Kronecker and Hadamard product, respectively. Define

$$\mathbf{D}_{\mathbf{x}} \equiv \text{diag}\{\mathbf{x}\} \in \mathbb{R}^{n \times n}, \quad \text{for } \forall \mathbf{x} \in \mathbb{R}^n.$$

The constraints associated with Hinge loss can be rewritten as

$$\mathbf{D}_{\alpha} \left(\tilde{\mathbf{X}}^{\top} \mathbf{w} + \mathbf{Z}^{\top} \mathbf{b} \right) \geq \mathbf{1}_{\ell} - \xi_{\alpha}.$$

The squared Euclidean distance can be expressed as

$$\begin{aligned} \|\mathbf{w}_{i_k} - \mathbf{w}_{j_k}\|^2 &= \mathbf{w}_{i_k}^{\top} \mathbf{w}_{i_k} + \mathbf{w}_{j_k}^{\top} \mathbf{w}_{j_k} - \mathbf{w}_{i_k}^{\top} \mathbf{w}_{j_k} - \mathbf{w}_{j_k}^{\top} \mathbf{w}_{i_k} \\ &= \mathbf{w}^{\top} (\mathbf{E}^{i_k, i_k} \otimes \mathbf{I}_d) \mathbf{w} + \mathbf{w}^{\top} (\mathbf{E}^{j_k, j_k} \otimes \mathbf{I}_d) \mathbf{w} \\ &\quad - \mathbf{w}^{\top} (\mathbf{E}^{i_k, j_k} \otimes \mathbf{I}_d) \mathbf{w} - \mathbf{w}^{\top} (\mathbf{E}^{j_k, i_k} \otimes \mathbf{I}_d) \mathbf{w} \\ &= \mathbf{w}^{\top} (\mathbf{U}_k \otimes \mathbf{I}_d) \mathbf{w}. \end{aligned}$$

If we denote the primal and dual variables by

$$\begin{aligned}\boldsymbol{\theta}_P &\equiv [\mathbf{w}^\top, \mathbf{b}^\top, \boldsymbol{\xi}_\alpha^\top, \rho]^\top, \\ \boldsymbol{\theta}_D &\equiv [\boldsymbol{\alpha}^\top, \boldsymbol{\lambda}^\top, \mu, \boldsymbol{\eta}_\alpha^\top]^\top \geq \mathbf{0},\end{aligned}$$

respectively, the Lagrangian function [12] is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_P, \boldsymbol{\theta}_D) &= \frac{1}{2M} \|\mathbf{w}\|^2 + C_\alpha \|\boldsymbol{\xi}_\alpha\|_1 + C_\rho \rho + \boldsymbol{\alpha}^\top \left(\mathbf{1}_\ell - \boldsymbol{\xi}_\alpha - \mathbf{D}_\alpha \left(\tilde{\mathbf{X}}^\top \mathbf{w} + \mathbf{Z}^\top \mathbf{b} \right) \right) \\ &\quad + \sum_{k=1}^K \lambda_k \left(\frac{1}{2} \mathbf{w}^\top (\mathbf{U}_k \otimes \mathbf{I}_d) \mathbf{w} - \rho \right) - \rho \mu - \boldsymbol{\xi}_\alpha^\top \boldsymbol{\eta}_\alpha \\ &= \frac{1}{2} \mathbf{w}^\top \left(\left(\frac{1}{M} \mathbf{I}_M + \mathcal{U} \boldsymbol{\lambda} \right) \otimes \mathbf{I}_d \right) \mathbf{w} - \boldsymbol{\alpha}^\top \mathbf{D}_y \left(\tilde{\mathbf{X}}^\top \mathbf{w} + \mathbf{Z}^\top \mathbf{b} \right) + \|\boldsymbol{\alpha}\|_1 \\ &\quad + \boldsymbol{\xi}_\alpha^\top (C_\alpha \mathbf{1}_K - \boldsymbol{\alpha} - \boldsymbol{\eta}_\alpha) + \rho (C_\rho - \mu - \mathbf{1}_K^\top \boldsymbol{\lambda}). \\ &= \frac{1}{2} \mathbf{w}^\top (\mathbf{K}_{\text{net}}(\boldsymbol{\lambda})^{-1} \otimes \mathbf{I}_d) \mathbf{w} - \boldsymbol{\alpha}^\top \mathbf{D}_y \left(\tilde{\mathbf{X}}^\top \mathbf{w} + \mathbf{Z}^\top \mathbf{b} \right) + \|\boldsymbol{\alpha}\|_1 \\ &\quad + \boldsymbol{\xi}_\alpha^\top (C_\alpha \mathbf{1}_K - \boldsymbol{\alpha} - \boldsymbol{\eta}_\alpha) + \rho (C_\rho - \mu - \mathbf{1}_K^\top \boldsymbol{\lambda}).\end{aligned}$$

The derivatives of $\boldsymbol{\theta}_P$ are written as

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{w}} &= (\mathbf{K}_{\text{net}}(\boldsymbol{\lambda})^{-1} \otimes \mathbf{I}_d) \mathbf{w} - \tilde{\mathbf{X}} \mathbf{D}_y \boldsymbol{\alpha}, \\ \frac{\partial \mathcal{L}}{\partial \mathbf{b}} &= \mathbf{Z} \mathbf{D}_y \boldsymbol{\alpha}, \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}_\alpha} &= C_\alpha \mathbf{1}_K - \boldsymbol{\lambda}_\alpha - \boldsymbol{\eta}_\alpha, \\ \frac{\partial \mathcal{L}}{\partial \rho} &= C_\rho - \mu - \mathbf{1}_K^\top \boldsymbol{\lambda}.\end{aligned}$$

Setting the derivatives to zero yields

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_P, \boldsymbol{\theta}_D) &= -\frac{1}{2} \boldsymbol{\alpha}^\top \mathbf{D}_y \tilde{\mathbf{X}}^\top (\mathbf{K}_{\text{net}}(\boldsymbol{\lambda}) \otimes \mathbf{I}_d) \tilde{\mathbf{X}} \mathbf{D}_y \boldsymbol{\alpha} + \boldsymbol{\alpha}^\top \mathbf{1}_\ell \\ &= -\frac{1}{2} \boldsymbol{\alpha}^\top \mathbf{D}_y (\mathbf{K}_{\text{fea}} \circ (\mathbf{Z}^\top \mathbf{K}_{\text{net}}(\boldsymbol{\lambda}) \mathbf{Z})) \mathbf{D}_y \boldsymbol{\alpha} + \boldsymbol{\alpha}^\top \mathbf{1}_\ell.\end{aligned}$$

Negating the Lagrangian, the dual form in Problem 5 is derived.

B Proof of Lemma 2

The objective function of Problem 5 can be rearranged as

$$\begin{aligned}
& \frac{1}{2} \boldsymbol{\alpha}^\top \text{diag}(\mathbf{y}) \mathbf{K}_{\text{int}}(\boldsymbol{\lambda}) \text{diag}(\mathbf{y}) \boldsymbol{\alpha} - \|\boldsymbol{\alpha}\|_1 \\
&= \frac{1}{2} \boldsymbol{\alpha}^\top \text{diag}(\mathbf{y}) (\mathbf{K}_{\text{fea}} \circ (\mathbf{Z}^\top \mathbf{K}_{\text{net}}(\boldsymbol{\lambda}) \mathbf{Z})) \text{diag}(\mathbf{y}) \boldsymbol{\alpha} - \|\boldsymbol{\alpha}\|_1 \\
&= \frac{1}{2} \boldsymbol{\alpha}^\top \text{diag}(\mathbf{y}) \left(\sum_{i=1}^d (\mathbf{f}_i \mathbf{f}_i^\top) \circ (\mathbf{Z}^\top \mathbf{K}_{\text{net}}(\boldsymbol{\lambda}) \mathbf{Z}) \right) \text{diag}(\mathbf{y}) \boldsymbol{\alpha} - \|\boldsymbol{\alpha}\|_1 \\
&= \frac{1}{2} \sum_{i=1}^d \boldsymbol{\alpha}^\top \text{diag}(\mathbf{y}) ((\mathbf{f}_i \mathbf{f}_i^\top) \circ (\mathbf{Z}^\top \mathbf{K}_{\text{net}}(\boldsymbol{\lambda}) \mathbf{Z})) \text{diag}(\mathbf{y}) \boldsymbol{\alpha} - \|\boldsymbol{\alpha}\|_1 \\
&= \frac{1}{2} \sum_{i=1}^d \boldsymbol{\alpha}^\top \text{diag}(\mathbf{y} \circ \mathbf{f}_i) \mathbf{Z}^\top \mathbf{K}_{\text{net}}(\boldsymbol{\lambda}) \mathbf{Z} \text{diag}(\mathbf{y} \circ \mathbf{f}_i) \boldsymbol{\alpha} - \|\boldsymbol{\alpha}\|_1 \\
&= \frac{1}{2} \sum_{h=1}^d \boldsymbol{\alpha}^\top \mathbf{F}_h^\top \mathbf{K}_{\text{net}}(\boldsymbol{\lambda}) \mathbf{F}_h \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{1}_\ell \\
&= \frac{1}{2} \sum_{h=1}^d \boldsymbol{\alpha}^\top \mathbf{F}_h^\top \left(\frac{1}{M} \mathbf{I}_M + \mathcal{U} \boldsymbol{\lambda} \right)^{-1} \mathbf{F}_h \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{1}_\ell,
\end{aligned}$$

which concludes the proof.

C Proof of Theorem 3

The graph Laplacian is expressed as

$$\begin{aligned}
\mathcal{U} \boldsymbol{\lambda} &= \sum_{k=1}^K \lambda_k (\mathbf{E}^{i_k i_k} + \mathbf{E}^{j_k j_k} - \mathbf{E}^{i_k j_k} - \mathbf{E}^{j_k i_k}) \\
&= \sum_{k=1}^K \lambda_k \mathbf{v}_k \mathbf{v}_k^\top \\
&= \mathbf{V}_{\text{lap}} \text{diag}(\boldsymbol{\lambda}) \mathbf{V}_{\text{lap}}^\top.
\end{aligned}$$

If we let

$$\begin{aligned}
\mathbf{P}_0 &\equiv \frac{1}{M} \mathbf{I}_M, \\
\mathbf{P}_k &\equiv \mathbf{v}_k \mathbf{v}_k^\top, \quad \forall k, \\
\mathbf{g} &\equiv \mathbf{0}_M,
\end{aligned}$$

the objective function of the dual problem is expressed as

$$\frac{1}{2} \sum_{h=1}^d (\mathbf{F}_h \boldsymbol{\alpha} + \mathbf{g})^\top \left(\mathbf{P}_0 + \sum_{k=1}^K \lambda_k \mathbf{P}_k \right)^{-1} (\mathbf{F}_h \boldsymbol{\alpha} + \mathbf{g}) - \boldsymbol{\alpha}^\top \mathbf{1}_\ell,$$

which implies that Problem 5 can be transformed into the combination of a linear program and d MFPs. Following Lobo et al [23], we show that the equivalence between Problem 7 and Problem 5 as follows. We rewrite Problem 7 as

$$\begin{aligned} \min \quad & -\mathbf{1}_\ell^\top \boldsymbol{\alpha} + \frac{1}{2} \sum_{h=1}^d s_{0,h} + s_{1,h} + \cdots + s_{K,h}, \\ \text{wrt} \quad & s_{0,h} \in \mathbb{R}, s_{k,h} \in \mathbb{R}, \mathbf{u}_{0,h} \in \mathbb{R}^M, \\ & \mathbf{u}_h = [u_{1,h}, \dots, u_{K,h}]^\top \in \mathbb{R}^K, \quad \forall k, \forall h, \\ & \boldsymbol{\lambda} \in \mathbb{R}_+^K, \boldsymbol{\alpha} \in \mathbb{R}_+^\ell, \\ \text{subj. to} \quad & \boldsymbol{\alpha} \leq C_\alpha \mathbf{1}_\ell, \quad \mathbf{Z} \text{diag}(\mathbf{y}) \boldsymbol{\alpha} = \mathbf{0}_M, \quad \mathbf{1}_K^\top \boldsymbol{\lambda} \leq C_\rho, \\ & \frac{1}{\sqrt{M}} \mathbf{I}_M \mathbf{u}_{0,h} + \sum_{k=1}^K u_{k,h} \mathbf{v}_k = \mathbf{F}_h \boldsymbol{\alpha} + \mathbf{g}, \\ & \|\mathbf{u}_{0,h}\|^2 \leq s_{0,h}^2, \quad \forall h, \\ & \|u_{k,h}\|^2 \leq s_{k,h} \lambda_k, \quad \forall k, \forall h. \end{aligned}$$

Eliminating $s_{0,h}, s_{k,h} \forall k, \forall h$, the problem can be reduced to

$$\begin{aligned} \min \quad & -\mathbf{1}_\ell^\top \boldsymbol{\alpha} + \frac{1}{2} \sum_h \left(\|\mathbf{u}_{0,h}\|^2 + \sum_k \frac{1}{\lambda_k} \|\mathbf{u}_{k,h}\|^2 \right), \\ \text{wrt} \quad & \mathbf{u}_{0,h} \in \mathbb{R}^M, \mathbf{u}_h = [u_{1,h}, \dots, u_{K,h}]^\top \in \mathbb{R}^K, \quad \forall k, \forall h, \\ & \boldsymbol{\lambda} \in \mathbb{R}_+^K, \boldsymbol{\alpha} \in \mathbb{R}_+^\ell, \\ \text{subj. to} \quad & \boldsymbol{\alpha} \leq C_\alpha \mathbf{1}_\ell, \quad \mathbf{Z} \text{diag}(\mathbf{y}) \boldsymbol{\alpha} = \mathbf{0}_M, \quad \mathbf{1}_K^\top \boldsymbol{\lambda} \leq C_\rho, \\ & \frac{1}{\sqrt{M}} \mathbf{u}_{0,h} + \sum_{k=1}^K u_{k,h} \mathbf{v}_k = \mathbf{F}_h \boldsymbol{\alpha} + \mathbf{g}, \end{aligned}$$

where $0/0$ is here treated as 0. Introducing d Lagrangian multipliers $\boldsymbol{\zeta}_h \in \mathbb{R}^d$ for the linear equality constraints of $\mathbf{u}_{0,h}$ and $\mathbf{u}_{k,h}$, the Lagrangian function is expressed as

$$\begin{aligned} & -\mathbf{1}_\ell^\top \boldsymbol{\alpha} + \frac{1}{2} \sum_h \left(\|\mathbf{u}_{0,h}\|^2 + \sum_k \frac{1}{\lambda_k} \|\mathbf{u}_{k,h}\|^2 \right) \\ & + \sum_h \boldsymbol{\zeta}_h^\top \left(\frac{1}{\sqrt{M}} \mathbf{u}_{0,h} + \sum_{k=1}^K u_{k,h} \mathbf{v}_k - \mathbf{F}_h \boldsymbol{\alpha} - \mathbf{g} \right). \end{aligned}$$

Setting the derivatives to zero yields

$$\begin{aligned}\forall h \in \mathbb{N}_d : \mathbf{u}_{0,h} &= -\frac{1}{\sqrt{M}} \boldsymbol{\zeta}_h, \\ \forall h \in \mathbb{N}_d, \forall k \in \mathbb{N}_K : \mathbf{u}_{k,h} &= -\lambda_k \mathbf{v}_k^\top \boldsymbol{\zeta}_h,\end{aligned}$$

Substituting them back into the objective function, the problem is obtained as

$$\begin{aligned}\min \quad & -\mathbf{1}_\ell^\top \boldsymbol{\alpha} + \frac{1}{2} \sum_h \boldsymbol{\zeta}_h^\top \left(\mathbf{P}_0 + \sum_k \lambda_k \mathbf{P}_k \right) \boldsymbol{\zeta}_h, \\ \text{wrt} \quad & \boldsymbol{\lambda} \in \mathbb{R}_+^K, \boldsymbol{\alpha} \in \mathbb{R}_+^\ell, \\ \text{subj. to} \quad & \boldsymbol{\alpha} \leq C_\alpha \mathbf{1}_\ell, \quad \mathbf{Z} \text{diag}(\mathbf{y}) \boldsymbol{\alpha} = \mathbf{0}_M, \quad \mathbf{1}_K^\top \boldsymbol{\lambda} \leq C_\rho, \\ & \forall h \in \mathbb{N}_d : \left(\mathbf{P}_0 + \sum_k \lambda_k \mathbf{P}_k \right) \boldsymbol{\zeta}_h = \mathbf{F}_h \boldsymbol{\alpha} + \mathbf{g},\end{aligned}$$

Then the theorem is established by substituting the last equality constraint into the objective function.

Acknowledgment

This work was supported by a Grant-in-Aid for Young Scientists (B), number 18700287, from the Ministry of Education, Culture, Sports, Science and Technology, Japan and BIRD of Japan Science and Technology Agency (JST).

References

- [1] R. Caruana, “Multitask learning,” *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.
- [2] S. Thrun and L. Pratt, *Learning to Learn*. Springer, 1997.
- [3] J. Baxter, “A model of inductive bias learning,” *Journal of Artificial Intelligence Research*, vol. 12, pp. 149–198, 2000.
- [4] B. Bakker and T. Heskes, “Task clustering and gating for Bayesian multitask learning,” *Journal of Machine Learning Research*, vol. 4, pp. 83–99, 2003.
- [5] T. Evgeniou and M. Pontil, “Regularized multitask learning,” in *Proceedings of the 17th SIGKDD Conference on Knowledge Discovery and Data Mining*, 2004, pp. 109–117.
- [6] 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.

- [7] N. D. Lawrence and J. C. Platt, “Learning to learn with the informative vector machine,” in *Proceedings of the Twenty First International Conference on Machine Learning*, 2004, pp. 512–519.
- [8] C. A. Micchelli and M. Pontil, “Kernels for multi-task learning,” in *Advances in Neural Information Processing Systems 17*. Cambridge, MA: MIT Press, 2005, pp. 921–928.
- [9] K. Yu, V. Tresp, and A. Schwaighofer, “Learning Gaussian processes from multiple tasks,” in *Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 1012–1019.
- [10] E. V. Bonilla, F. V. Agakov, and C. K. I. Williams, “Kernel multi-task learning using task-specific features,” in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, 2007, pp. 43–50.
- [11] K. Tsuda and W. S. Noble, “Learning kernels from biological networks by maximizing entropy,” *Bioinformatics*, vol. 20, no. Suppl. 1, pp. i326–i333, 2004.
- [12] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
- [13] Y. Amit, M. Fink, N. Srebro, and S. Ullman, “Uncovering shared structures in multi-class classification,” in *Proceedings of the 24th International Conference on Machine Learning*, 2007, pp. 17–24.
- [14] A. Shashua and A. Levin, “Ranking with large margin principle: two approaches,” in *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press, 2003, pp. 937–944.
- [15] T. Kato, K. Tsuda, and K. Asai, “Selective integration of multiple biological data for supervised network inference,” *Bioinformatics*, vol. 21, pp. 2488–2495, 2005.
- [16] J.-P. Vert and Y. Yamanishi, “Supervised graph inference,” in *Advances in Neural Information Processing Systems 17*. Cambridge, MA: MIT Press, 2005.
- [17] Y. Yamanishi, J. P. Vert, and M. Kanehisa, “Supervised enzyme network inference from the integration of genomic data and chemical information,” *Bioinformatics*, vol. 21 Suppl. 1, pp. i468–i477, Jun 2005.
- [18] K. Bleakley, G. Biau, and J.-P. Vert, “Supervised reconstruction of biological networks with local models,” *Bioinformatics*, vol. 23, no. 13, pp. i57–i65, 2007.
- [19] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis.” *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

- [20] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram, “Multi-task learning for classification with Dirichlet process priors,” *Journal of Machine Learning Research*, vol. 8, pp. 35–63, 2007.
- [21] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.
- [22] B. Borchers, “CSDP, a C library for semidefinite programming,” *Optimization Methods and Software*, vol. 11, no. 1, pp. 613–623, 1999.
- [23] M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebet, “Applications of second-order cone programming,” *Linear Algebra and its Applications*, vol. 284, pp. 193–228, 1998.
- [24] X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty, “Nonparametric transforms of graph kernels for semi-supervised learning,” in *Advances in Neural Information Processing Systems 17*. Cambridge, MA: MIT Press, 2004, pp. 1641–1648.
- [25] D. Haussler, “Convolution kernels on discrete structures,” UC Santa Cruz, Tech. Rep. UCSC-CRL-99-10, July 1999.
- [26] T. Jaakkola and D. Haussler, “Exploiting generative models in discriminative classifiers,” in *Advances in Neural Information Processing Systems 11*, M. S. Kearns, S. A. Solla, and D. A. Cohn, Eds. Cambridge, MA.: MIT Press, 1999, pp. 487–493.
- [27] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, “Text classification using string kernels,” *Journal of Machine Learning Research*, vol. 2, pp. 419–444, 2002.
- [28] R. I. Kondor and J. Lafferty, “Diffusion kernels on graphs and other discrete input spaces,” in *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002, pp. 315–322.
- [29] 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*, 2002, pp. 566–575.
- [30] H. Kashima and T. Koyanagi, “Kernels for semi-structured data,” in *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002, pp. 291–298.
- [31] T. Gärtner, “A survey of kernels for structured data,” *SIGKDD Explorations*, vol. 5, no. 1, pp. S268–S275, 2003.
- [32] T. Gärtner, P. Flach, and S. Wrobel, “On graph kernels: Hardness results and efficient alternatives,” in *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory*, 2003, pp. 129–143.
- [33] H. Kashima, K. Tsuda, and A. Inokuchi, “Marginalized kernels between labeled graphs,” in *Proceedings of the Twentieth International Conference on Machine Learning*, 2003, pp. 321–328.

- [34] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, “Network motifs: Simple building blocks of complex networks,” *Science*, vol. 298, pp. 824–827, Jan. 2002.
- [35] A. Andreeva, D. Howorth, S. E. Brenner, T. J. P. Hubbard, C. Chothia, and A. G. Murzin, “SCOP database in 2004: refinements integrate structure and sequence family data,” *Nuclear Acid Research*, vol. 32, pp. D226–D229, 2004.
- [36] E. L. Lehmann and J. P. Romano, *Testing Statistical Hypotheses*. Springer, 2005.
- [37] C. von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork, “Comparative assessment of large-scale data sets of protein-protein interactions,” *Nature*, vol. 417, pp. 399–403, 2002.
- [38] G. R. G. Lanckriet, T. D. Bie, N. Cristianini, M. I. Jordan, and W. S. Noble, “A statistical framework for genomic data fusion,” *Bioinformatics*, vol. 20, pp. 2626–2635, 2004.
- [39] M. Kurucz, A. A. Benczúr, T. Kiss, I. Nagy, A. Szabó, and B. Torma, “KDD cup 2007 task1 winner report,” in *ACM SIGKDD Explorations Newsletter*, vol. 9, no. 2, 2008, pp. 53–56.
- [40] N. Srebro, J. D. M. Rennie, and T. S. Jaakkola, “Maximum-margin matrix factorization,” in *Advances in Neural Information Processing Systems 17*, L. Saul, Y. Weiss, and L. Bottou, Eds. MIT Press, 2005, pp. 1329–1336.
- [41] E. Bonilla, K. M. Chai, and C. Williams, “Multi-task Gaussian process prediction,” in *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA: MIT Press, 2008, pp. 153–160.
- [42] X. Liao, Y. Xue, and L. Carin, “Logistic regression with an auxiliary data source,” in *Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 505–512.
- [43] Q. Liu, X. Liao, and L. Carin, “Semi-supervised multitask learning,” in *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA: MIT Press, 2008, pp. 937–944.
- [44] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the support of a high-dimensional distribution,” *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
- [45] D. M. J. Tax and R. P. W. Duin, “Support vector data description,” *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.
- [46] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, “Support vector regression machines,” in *Advances in Neural Information Processing Systems 9*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. MIT Press, 1997, pp. 155–161.

- [47] C.-C. Chang and C.-J. Lin, “Training ν -support vector regression: Theory and algorithms,” *Neural Computation*, vol. 14, no. 8, pp. 1959–1977, 2002.
- [48] B. Schölkopf, A. Smola, R. Williamson, and P. Bartlett, “New support vector algorithms,” *Neural Computation*, vol. 12, no. 5, pp. 1207–1245, 2000.
- [49] F. Perez-Cruz, D. J. L. H. J. Weston, and B. Schölkopf, “Extension of the ν -SVM range for classification,” in *Advances in Learning Theory: Methods, Models and Applications 190*, J. A. K. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, Eds. Amsterdam: IOS Press, 2003, pp. 179–196.
- [50] P. H. Chen, C. J. Lin, and B. Schölkopf, “A tutorial on ν -support vector machines,” *Applied Stochastic Models in Business and Industry*, vol. 21, no. 2, pp. 111–136, 2005.
- [51] B. Efron, T. Hastie, R. Tibshirani, and I. Johnstone, “Least angle regression,” *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.
- [52] T. Hastie, S. Rosset, R. Tibshirani, and J. ZHU, “The entire regularization path for the support vector machine,” *Journal of Machine Learning Research*, vol. 5, pp. 1391–1415, 2004.
- [53] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, “Learning the kernel matrix with semidefinite programming,” *Journal of Machine Learning Research*, vol. 5, pp. 27–72, Jan 2004.
- [54] T. Kato, H. Kashima, and M. Sugiyama, “Integration of multiple networks for robust label propagation,” in *Proceedings of 2008 SIAM International Conference on Data Mining (SDM2008)*, Atlanta, Georgia, USA, 2008, pp. 716–726.