IEICE Transactions on Information and Systems,
vol.E96-D, no.8, pp.1871–1874, 2013.

Computationally Efficient Multi-label Classification by Least-Squares Probabilistic Classifiers

Hyunha Nam
Hirotaka Hachiya
Masashi Sugiyama

Tokyo Institute of Technology, Japan.
{hyunha@sg., sugi@}cs.titech.ac.jp
http://sugiyama-www.cs.titech.ac.jp/˜sugi

Abstract

Multi-label classification allows a sample to belong to multiple classes simultaneously, which is often the case in real-world applications such as text categorization and image annotation. In multi-label scenarios, taking into account correlations among multiple labels can boost the classification accuracy. However, this makes classifier training more challenging because handling multiple labels induces a high-dimensional optimization problem. In this paper, we propose a scalable multi-label method based on the least-squares probabilistic classifier. Through experiments, we show the usefulness of our proposed method.

Keywords

Multi-Label Classification, Least-Squares Probabilistic Classifier

1 Introduction

In some recent applications of pattern recognition, a sample can belong to more than one category at the same time. For example, in text mining, a news article about *Transformer* can be categorized into the "car", "robot", and "movie" categories. The classification problem where a single sample can belong to multiple classes is called *multi-label classification*, and it has attracted a great deal of attention recently [5].

However, multi-label classification is computationally expensive, and overcoming the computational bottleneck is a common challenge. In this paper, we thus propose a novel multi-label method. Our approach is to extend the computationally efficient multi-task method [3] based on the *least-squares probabilistic classifier* (LSPC) [4, 6] to multi-label scenarios, and to achieve a method to compute its solution efficiently.

2 Probabilistic Classification by LSPC

In this section, we review the *least-squares probabilistic classifier* (LSPC) for single-label classification [4, 6].

Suppose that we are given a set of training samples

$$\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$$

drawn independently from a joint probability distribution with density $p(\boldsymbol{x}, y)$, where $\boldsymbol{x}_n \in \mathbb{R}^D$ is a feature vector, D is the dimensionality of feature vector $\boldsymbol{x}$, $y_n \in \{1, \ldots, Y\}$ is a class label, and Y is the number of classes. The objective of probabilistic classification is to learn the class-posterior probability $p(y|\boldsymbol{x})$ from the training samples. Based on the class-posterior probability, classification of a new sample $\boldsymbol{x}$ can be carried out by

$$\widehat{y} := \underset{y \in \{1,\ldots,Y\}}{\operatorname{argmax}}\ p(y|\boldsymbol{x}),$$

with confidence $p(y = \widehat{y}|\boldsymbol{x})$.

For each $y \in \{1, \ldots, Y\}$, we model $p(y|\boldsymbol{x})$ by

$$q(y|\boldsymbol{x}; \boldsymbol{\theta}_y) := \sum_{b=1}^{B} \theta_{y,b} \phi_b(\boldsymbol{x}) = \boldsymbol{\theta}_y^\top \boldsymbol{\phi}(\boldsymbol{x}),$$

where B denotes the number of parameters,

$$\boldsymbol{\theta}_y = (\theta_{y,1}, \ldots, \theta_{y,B})^\top \in \mathbb{R}^B$$

is the parameter vector, and

$$\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_B(\boldsymbol{x}))^\top \in \mathbb{R}^B$$

is the basis function vector. In practice, we may use a kernel model, i.e., we set $B = N$ and $\phi_b(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{x}_b)$, where $K(\boldsymbol{x}, \boldsymbol{x}')$ is a kernel function.

We fit the above model to the true class-posterior probability $p(y|\boldsymbol{x})$ under the following squared loss:

$$\begin{aligned} J_y(\boldsymbol{\theta}_y) &:= \frac{1}{2} \int \left(q(y|\boldsymbol{x}; \boldsymbol{\theta}_y) - p(y|\boldsymbol{x})\right)^2 p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\ &= \frac{1}{2} \int q(y|\boldsymbol{x}; \boldsymbol{\theta}_y)^2 p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} - \int q(y|\boldsymbol{x}; \boldsymbol{\theta}_y) p(\boldsymbol{x}|y) p(y) \mathrm{d}\boldsymbol{x} + C, \end{aligned}$$

where $p(\boldsymbol{x})$ denotes the marginal density of feature vector $\boldsymbol{x}$ and C is a constant independent of $\boldsymbol{\theta}_y$. Approximating the expectations over $\boldsymbol{x}$ by sample averages and the class-prior probability $p(y)$ by sample ratios, ignoring constant C and factor $1/N$, and including an ℓ_2-regularizer, we have the following training criterion:

$$\begin{aligned} \widehat{J}_y(\boldsymbol{\theta}_y) &:= \frac{1}{2} \sum_{n=1}^{N} q(y|\boldsymbol{x}_n; \boldsymbol{\theta}_y)^2 - \sum_{n:y_n=y} q(y|\boldsymbol{x}_n; \boldsymbol{\theta}_y) + \frac{\rho}{2} \|\boldsymbol{\theta}_y\|^2 \\ &= \frac{1}{2} \boldsymbol{\theta}_y^\top \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{\theta}_y - \boldsymbol{\theta}_y^\top \boldsymbol{\Phi}^\top \boldsymbol{\pi}_y + \frac{\rho}{2} \|\boldsymbol{\theta}_y\|^2, \end{aligned}$$

where $\rho > 0$ is the regularization parameter,

$$\boldsymbol{\Phi} = (\boldsymbol{\phi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_N))^\top \in \mathbb{R}^{N \times B}$$

is the design matrix, and $\boldsymbol{\pi}_y$ is the N-dimensional class-indicator vector, i.e., $\pi_{y,n} = 1$ if $y_n = y$ and $\pi_{y,n} = 0$ otherwise:

$$\pi_{y,n} = \begin{cases} 1 & (y_n = y), \\ 0 & (y_n \neq y). \end{cases}$$

We can obtain the minimizer $\widehat{\boldsymbol{\theta}}_y$ of $\widehat{J}_y$ analytically as

$$\widehat{\boldsymbol{\theta}}_y = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \rho \boldsymbol{I}_B\right)^{-1} \boldsymbol{\Phi}^\top \boldsymbol{\pi}_y,$$

where $\boldsymbol{I}_B$ denotes the B-dimensional identity matrix.

As the number of training samples, N, increases, the solution $q(y|\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_y)$ was shown to converge to the true class-posterior probability $p(y|\boldsymbol{x})$ with the optimal convergence rate [4]. For a finite sample size, we obtain the final solution by rounding up a negative output to zero and normalization as follows [6]:

$$\widehat{p}(y|\boldsymbol{x}) = \frac{\max(0, q(y|\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_y))}{\sum_{y'=1}^{Y} \max(0, q(y'|\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_{y'}))}.$$

3 Multi-Task LSPC

When multiple related learning tasks exist, solving them simultaneously by sharing some common information behind the tasks is expected to be more promising than solving them separately. This is the idea of *multi-task learning*. A computationally efficient multi-task learning method can be developed by combining multiple LSPCs. Here, we review multi-task LSPC (MT-LSPC) [3] in a slightly generalized way.

Suppose that we are given a set of training samples

$$\{(\boldsymbol{x}_n, y_n, t_n)\}_{n=1}^N,$$

where $t_n \in \{1, \ldots, T\}$ denotes the task index. We assume that $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ are drawn independently from a joint probability distribution with density $p_{t_n}(\boldsymbol{x}, y)$. The objective of multi-task probabilistic classification is to learn the class-posterior probabilities $p_t(y|\boldsymbol{x})$ for $t \in \{1, \ldots, T\}$.

Let us model $p_t(y|\boldsymbol{x})$ for each $t \in \{1, \ldots, T\}$ and $y \in \{1, \ldots, Y\}$ as

$$q(y|\boldsymbol{x}; \boldsymbol{\theta}_{y,t}) := \sum_{b=1}^{B} \theta_{y,b,t} \phi_b(\boldsymbol{x}) = \boldsymbol{\theta}_{y,t}^\top \boldsymbol{\phi}(\boldsymbol{x}),$$

where

$$\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_B(\boldsymbol{x}))^\top \in \mathbb{R}^B$$

and

$$\boldsymbol{\theta}_{y,t} := (\theta_{y,1,t}, \ldots, \theta_{y,B,t})^\top \in \mathbb{R}^B.$$

The basic idea of MT-LSPC is that solutions of all tasks are imposed to be close to each other in terms of the ℓ_2-norm. More specifically, let us decompose $\boldsymbol{\theta}_{y,t}$ as

$$\boldsymbol{\theta}_{y,t} = \boldsymbol{\beta}_{y,0} + \boldsymbol{\beta}_{y,t},$$

where $\boldsymbol{\beta}_{y,0}$ is the common part of solutions for all tasks and $\boldsymbol{\beta}_{y,t}$ is the individual part of solutions for task t. Then, for

$$\boldsymbol{\beta}_y := \left(\boldsymbol{\beta}_{y,0}^\top, \boldsymbol{\beta}_{y,1}^\top, \ldots, \boldsymbol{\beta}_{y,T}^\top\right)^\top \in \mathbb{R}^{B(T+1)},$$

the training criterion of MT-LSPC is given by

$$\widehat{J}_y^{\mathrm{MT}}(\boldsymbol{\beta}_y) := \frac{1}{2}\sum_{n=1}^{N} q(y|\boldsymbol{x}_n; \boldsymbol{\beta}_{y,0} + \boldsymbol{\beta}_{y,t_n})^2 - \sum_{n:y_n=y} q(y|\boldsymbol{x}_n; \boldsymbol{\beta}_{y,0} + \boldsymbol{\beta}_{y,t_n}) + \frac{\omega_0}{2}\|\boldsymbol{\beta}_{y,0}\|^2 + \frac{1}{2}\sum_{t=1}^{T}\omega_t\|\boldsymbol{\beta}_{y,t}\|^2,$$

where $\omega_0 > 0$ is the regularization parameter for the task-independent part and $\omega_t > 0$ $(t = 1, \ldots, T)$ is the regularization parameter for the task-dependent parts.

For $\mathbf{0}_B$ denoting the B-dimensional zero vector, let

$$\begin{aligned}
\boldsymbol{\xi}_t(\boldsymbol{x}) &:= \left(\boldsymbol{\phi}(\boldsymbol{x})^\top, \mathbf{0}_{B(t-1)}^\top, \boldsymbol{\phi}(\boldsymbol{x})^\top, \mathbf{0}_{B(T-t)}^\top\right)^\top \in \mathbb{R}^{B(T+1)},\\
\boldsymbol{\Xi} &:= \left(\boldsymbol{\xi}_{t_1}(\boldsymbol{x}_1), \ldots, \boldsymbol{\xi}_{t_N}(\boldsymbol{x}_N)\right)^\top \in \mathbb{R}^{N \times B(T+1)},\\
\boldsymbol{\Omega} &:= \mathrm{diag}\left(\omega_0, \omega_1, \ldots, \omega_T\right) \in \mathbb{R}^{(T+1)\times(T+1)}.
\end{aligned}$$

Then the MT-LSPC training criterion can be compactly expressed as

$$\widehat{J}_y^{\mathrm{MT}}(\boldsymbol{\beta}_y) = \frac{1}{2}\boldsymbol{\beta}_y^\top \boldsymbol{\Xi}^\top \boldsymbol{\Xi} \boldsymbol{\beta}_y - \boldsymbol{\beta}_y^\top \boldsymbol{\Xi}^\top \boldsymbol{\pi}_y + \frac{1}{2}\boldsymbol{\beta}_y^\top (\boldsymbol{\Omega} \otimes \boldsymbol{I}_B)\boldsymbol{\beta}_y,$$

where $\otimes$ denotes the *Kronecker product*. Because the above $\widehat{J}_y^{\mathrm{MT}}(\boldsymbol{\beta}_y)$ is essentially the same form as the original single-task LSPC training criterion, we can similarly obtain the minimizer $\widehat{\boldsymbol{\beta}}_y$ analytically as

$$\widehat{\boldsymbol{\beta}}_y = \left(\boldsymbol{\Xi}^\top \boldsymbol{\Xi} + \boldsymbol{\Omega} \otimes \boldsymbol{I}_B\right)^{-1} \boldsymbol{\Xi}^\top \boldsymbol{\pi}_y.$$

Suppose that we use a kernel model (i.e., $B = N$). Then, the size of the matrix to be inverted in the above equation is $N(T+1) \times N(T+1)$. Thus, the computational complexity for naively computing the solution $\widehat{\boldsymbol{\beta}}_y$ is $\mathcal{O}(N^3T^3)$, which can be expensive. However, because the rank of $\boldsymbol{\Xi}^\top\boldsymbol{\Xi}$ is at most N, the solution can be computed more efficiently. More specifically, $q(y|\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_{y,t})$ can be expressed as follows:

$$q(y|\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_{y,t}) = \widehat{\boldsymbol{\theta}}_{y,t}^\top \boldsymbol{\phi}(\boldsymbol{x}) = \widehat{\boldsymbol{\beta}}_y^\top \boldsymbol{\xi}_t(\boldsymbol{x}) = \boldsymbol{\pi}_y^\top \boldsymbol{A}^{-1} \boldsymbol{b}_t,$$

where $\boldsymbol{A}$ is the $N \times N$ matrix and $\boldsymbol{b}_t$ is the N-dimensional vector defined as

$$\begin{aligned} A_{n,n'} &:= [\boldsymbol{\Xi}(\boldsymbol{\Omega}^{-1} \otimes \boldsymbol{I}_B)\boldsymbol{\Xi}^\top + \boldsymbol{I}_N]_{n,n'} \\ &= \left(\frac{1}{\omega_0} + \frac{\delta_{t_n,t_{n'}}}{\omega_{t_n}}\right) \boldsymbol{\phi}(\boldsymbol{x}_n)^\top \boldsymbol{\phi}(\boldsymbol{x}_{n'}) + \delta_{n,n'}, \\ b_{t,n} &:= [\boldsymbol{\Xi}(\boldsymbol{\Omega}^{-1} \otimes \boldsymbol{I}_B)\boldsymbol{\xi}_t(\boldsymbol{x})]_n = \left(\frac{1}{\omega_0} + \frac{\delta_{t,t_n}}{\omega_t}\right) \boldsymbol{\phi}(\boldsymbol{x}_n)^\top \boldsymbol{\phi}(\boldsymbol{x}). \end{aligned}$$

Here $\delta_{t,t'}$ denotes the *Kronecker delta*. The computational complexity for computing the solution in this way is reduced to $\mathcal{O}(N^3)$, which is independent of T.

4 Reformulation of MT-LSPC

In this paper, we develop a multi-label method based on MT-LSPC. However, the original MT-LSPC imposes all solutions to be close to each other via the common part, which is not necessarily appropriate in the multi-label scenario. Here, we derive an extention of MT-LSPC that imposes a multi-task penalty via pairwise similarities between tasks. This pairwise version will be used for developing a multi-label method later.

For $\boldsymbol{\theta}_y := (\boldsymbol{\theta}_{y,1}^\top, \ldots, \boldsymbol{\theta}_{y,T}^\top)^\top \in \mathbb{R}^{BT}$, let us consider the following training criterion:

$$\begin{aligned} \widehat{J}_y^{\mathrm{MT}'}(\boldsymbol{\theta}_y) := \frac{1}{2}\sum_{n=1}^{N} q(y|\boldsymbol{x}_n;\boldsymbol{\theta}_{y,t_n})^2 - \sum_{n:y_n=y} q(y|\boldsymbol{x}_n;\boldsymbol{\theta}_{y,t_n}) \\ + \frac{1}{2}\sum_{t=1}^{T} \lambda_t \|\boldsymbol{\theta}_{y,t}\|^2 + \frac{1}{4}\sum_{t,t'=1}^{T} \gamma_{t,t'} \|\boldsymbol{\theta}_{y,t} - \boldsymbol{\theta}_{y,t'}\|^2, \end{aligned}$$

where $\lambda_t > 0$ is the regularization parameter for task t and $\gamma_{t,t'} > 0$ is the similarity between tasks t and t' (large $\gamma_{t,t'}$ corresponds to similar tasks). Let

$$\begin{aligned} \boldsymbol{\psi}_t(\boldsymbol{x}) &:= \left(\boldsymbol{0}_{B(t-1)}^\top, \boldsymbol{\phi}(\boldsymbol{x})^\top, \boldsymbol{0}_{B(T-t)}^\top\right)^\top \in \mathbb{R}^{BT}, \\ \boldsymbol{\Psi} &:= (\boldsymbol{\psi}_{t_1}(\boldsymbol{x}_1), \ldots, \boldsymbol{\psi}_{t_N}(\boldsymbol{x}_N))^\top \in \mathbb{R}^{N \times BT}. \end{aligned}$$

Then $\widehat{J}_y^{\mathrm{MT}'}$ can be compactly expressed as

$$\widehat{J}_y^{\mathrm{MT}'}(\boldsymbol{\theta}_y) = \frac{1}{2}\boldsymbol{\theta}_y^\top \boldsymbol{\Psi}^\top \boldsymbol{\Psi} \boldsymbol{\theta}_y - \boldsymbol{\theta}_y^\top \boldsymbol{\Psi}^\top \boldsymbol{\pi}_y + \frac{1}{2}\boldsymbol{\theta}_y^\top (\boldsymbol{C} \otimes \boldsymbol{I}_B)\boldsymbol{\theta}_y,$$

where $\boldsymbol{C}$ is the $T \times T$ matrix defined as

$$C_{t,t'} := \delta_{t,t'}\left(\lambda_t + \sum_{t''=1}^{T} \gamma_{t,t''}\right) - \gamma_{t,t'}.$$

Taking the derivative of $\widetilde{J}_y^{\mathrm{MT}}$ with respect to $\boldsymbol{\theta}_y$ and setting it to zero, we have the minimizer $\widehat{\boldsymbol{\theta}}_y$ analytically as

$$\widehat{\boldsymbol{\theta}}_y = \left(\boldsymbol{\Psi}^\top \boldsymbol{\Psi} + \boldsymbol{C} \otimes \boldsymbol{I}_B\right)^{-1} \boldsymbol{\Psi}^\top \boldsymbol{\pi}_y.$$

Using the same trick as MT-LSPC, $q(y|\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_{y,t})$ can be efficiently computed based on the following expression:

$$q(y|\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_{y,t}) = \widehat{\boldsymbol{\theta}}_{y,t}^\top \boldsymbol{\phi}(\boldsymbol{x}) = \widehat{\boldsymbol{\theta}}_y^\top \boldsymbol{\psi}_t(\boldsymbol{x}) = \boldsymbol{\pi}_y^\top \boldsymbol{A}'^{-1} \boldsymbol{b}'_t,$$

where $\boldsymbol{A}'$ is the $N \times N$ matrix and $\boldsymbol{b}'_t$ is the N-dimensional vector defined as

$$\begin{aligned} A'_{n,n'} &:= [\boldsymbol{\Psi}(\boldsymbol{C}^{-1} \otimes \boldsymbol{I}_B)\boldsymbol{\Psi}^\top + \boldsymbol{I}_N]_{n,n'} \\ &= [\boldsymbol{C}^{-1}]_{t_n,t_{n'}} \boldsymbol{\phi}(\boldsymbol{x}_n)^\top \boldsymbol{\phi}(\boldsymbol{x}_{n'}) + \delta_{n,n'}, \\ b'_{t,n} &:= [\boldsymbol{\Psi}(\boldsymbol{C}^{-1} \otimes \boldsymbol{I}_B)\boldsymbol{\psi}_t(\boldsymbol{x})]_n = [\boldsymbol{C}^{-1}]_{t,t_n} \boldsymbol{\phi}(\boldsymbol{x}_n)^\top \boldsymbol{\phi}(\boldsymbol{x}). \end{aligned}$$

The computational complexity for computing the solution in this way is reduced to $\mathcal{O}(N^3 + T^3)$. Note that the factor T^3 comes from the computation of $\boldsymbol{C}^{-1}$; if the task similarity matrix $\boldsymbol{\Gamma}$ (with $\Gamma_{t,t'} = \gamma_{t,t'}$) enjoys nice structure such as being low-rank or sparse, it may be computed more efficiently.

5 Multi-Label LSPC

In this section, we propose a computationally efficient multi-task classifier based on the pairwise MT-LSPC called *multi-label LSPC* (ML-LSPC).

Suppose that we are given a set of training samples

$$\{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N,$$

where $\boldsymbol{y}_n = (y_{n,1}, \ldots, y_{n,T})^\top \in \{1, \ldots, Y\}^T$ is the class-label vector for the n-th sample and T is the number of labels. Input vector $\boldsymbol{x}$ is assumed to be drawn independently from $p(\boldsymbol{x})$, and the t-th element y_t of $\boldsymbol{y} = (y_1, \ldots, y_t)^\top$ is assumed to be drawn from $p_t(y|\boldsymbol{x})$. The objective of multi-label probabilistic classification is to learn the class-posterior probabilities $p_t(y|\boldsymbol{x})$ for $t \in \{1, \ldots, T\}$.

Requiring that similar labels should have similar classification solutions, we can employ a multi-task learning method to solve the multi-label learning problem. Indeed, from the MT-LSPC training criterion, we immediately have the training criterion for ML-LSPC:

$$\begin{aligned} \widehat{J}_y^{\mathrm{ML}}(\boldsymbol{\theta}_y) := \sum_{t=1}^T \left(\frac{1}{2} \sum_{n=1}^N q(y|\boldsymbol{x}_n; \boldsymbol{\theta}_{y,t})^2 - \sum_{n:y_{n,t}=y} q(y|\boldsymbol{x}_n; \boldsymbol{\theta}_{y,t}) + \frac{1}{2}\lambda_t \|\boldsymbol{\theta}_{y,t}\|^2 \right) \\ + \frac{1}{4} \sum_{t,t'=1}^T \gamma_{t,t'} \|\boldsymbol{\theta}_{y,t} - \boldsymbol{\theta}_{y,t'}\|^2. \end{aligned}$$

However, a notable difference between multi-task and multi-label formulations is that the number of training samples is N in the multi-task formulation, whereas that in the multi-label formulation is essentially NT. Thus, if we naively apply MT-LSPC to the multi-label problem, the computational complexity is $\mathcal{O}(N^3T^3)$ for a kernel model (i.e., $B = N$), which is prohibitively expensive. Below, we explain how to mitigate this problem.

Let

$$\boldsymbol{\Theta}_y := (\boldsymbol{\theta}_{y,1}, \ldots, \boldsymbol{\theta}_{y,T}) \in \mathbb{R}^{B \times T},$$

and let $\boldsymbol{\pi}_{y,t}$ be the N-dimensional class-indicator vector for the t-th label, i.e., $\pi_{y,t,n} = 1$ if $y_{n,t} = y$, and $\pi_{y,t,n} = 0$ otherwise:

$$\pi_{y,t,n} = \begin{cases} 1 & (y_{n,t} = y), \\ 0 & (y_{n,t} \neq y). \end{cases}$$

Let

$$\boldsymbol{\Pi}_y := (\boldsymbol{\pi}_{y,1}, \ldots, \boldsymbol{\pi}_{y,T}) \in \mathbb{R}^{N \times T}.$$

Then $\widehat{J}_y^{\mathrm{ML}}$ can be compactly expressed as

$$\widehat{J}_y^{\mathrm{ML}}(\boldsymbol{\theta}_y) = \frac{1}{2}\mathrm{tr}(\boldsymbol{\Theta}_y^\top \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{\Theta}_y) - \mathrm{tr}(\boldsymbol{\Theta}_y^\top \boldsymbol{\Phi}^\top \boldsymbol{\Pi}_y) + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Theta}_y \boldsymbol{C} \boldsymbol{\Theta}_y^\top).$$

Taking the derivative of the above equation with respect to $\boldsymbol{\Theta}_y$ and setting it to zero, we obtain

$$\boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{\Theta}_y + \boldsymbol{\Theta}_y \boldsymbol{C} = \boldsymbol{\Phi}^\top \boldsymbol{\Pi}_y. \tag{1}$$

This is called the *continuous Sylvester equation* with respect to $\boldsymbol{\Theta}_y$, which often arises in control theory [2].

Various algorithms for solving the Sylvester equation have been developed. One of the simplest methods is based on the eigenvalue decompositions of $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ and $\boldsymbol{C}$ as follows: Let $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_B$ be eigenvectors of $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ associated with eigenvalues $f_1, \ldots, f_B$, and let $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_T$ be eigenvectors of $\boldsymbol{C}$ associated with eigenvalues $g_1, \ldots, g_T$. Then the solution $\widehat{\boldsymbol{\Theta}}_y$ to Eq.(1) is given analytically as

$$\widehat{\boldsymbol{\Theta}}_y = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_B) \boldsymbol{Q} (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_T)^\top,$$

where $\boldsymbol{Q}$ is the $B \times T$ matrix defined as

$$Q_{b,t} := \frac{\boldsymbol{f}_b^\top \boldsymbol{\Phi}^\top \boldsymbol{\Pi}_y \boldsymbol{g}_t}{f_b + g_t}.$$

If a kernel model is used (i.e., $B = N$), the computational complexity for solving Eq.(1) in this way is $\mathcal{O}(N^3 + N^2T + NT^2 + T^3)$. Note that the terms N^3 and T^3 come from the eigenvalue decompositions of $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ and $\boldsymbol{C}$, which can be performed more efficiently if they enjoy nice structure such as being low-rank or sparse.

For large-scale data, Eq.(1) may be solved more efficiently by numerical optimization. Let

$$\boldsymbol{\theta}_y := (\boldsymbol{\theta}_{y,1}^\top, \ldots, \boldsymbol{\theta}_{y,T}^\top)^\top \in \mathbb{R}^{BT}.$$

Then Eq.(1) can be expressed as

$$\boldsymbol{H}\boldsymbol{\theta}_y = \boldsymbol{h}_y,$$

where

$$\boldsymbol{H} := \boldsymbol{I}_T \otimes (\boldsymbol{\Phi}^\top \boldsymbol{\Phi}) + \boldsymbol{C} \otimes \boldsymbol{I}_B \in \mathbb{R}^{BT \times BT},$$
$$\boldsymbol{h}_y := ((\boldsymbol{\Phi}^\top \boldsymbol{\pi}_{y,1})^\top, \ldots, (\boldsymbol{\Phi}^\top \boldsymbol{\pi}_{y,T})^\top)^\top \in \mathbb{R}^{BT}.$$

If a kernel model is used (i.e., $B = N$), naively solving $\boldsymbol{H}\boldsymbol{\theta}_y = \boldsymbol{h}_y$ takes $\mathcal{O}(N^3T^3)$ time. Here, we take into account the Kronecker structure of $\boldsymbol{H}$, and solve the equation numerically by the *conjugate gradient* method. More specifically, we can compute the matrix-vector product $\boldsymbol{H}\boldsymbol{\theta}_y$ as

$$[\boldsymbol{H}\boldsymbol{\theta}_y]_t = \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{\theta}_{y,t} + \sum_{t'=1}^{T} C_{t,t'} \boldsymbol{\theta}_{y,t'}.$$

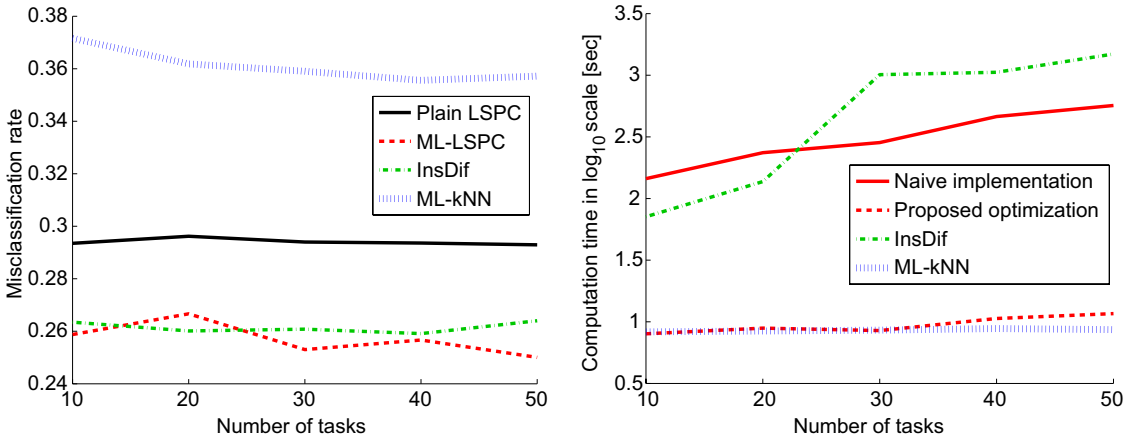
Although the computational complexity for naively computing $\boldsymbol{H}\boldsymbol{\theta}_y$ is $\mathcal{O}(N^3 + N^2T^2)$ including the computation of $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$, that for computing $\boldsymbol{H}\boldsymbol{\theta}_y$ based on the above expression is reduced to $\mathcal{O}(N^2T + NT^2)$. Note that the term N^2T comes from the computation $\boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{\theta}_{y,t}$ and the term NT^2 comes from the computation $\sum_{t'=1}^{T} C_{t,t'} \boldsymbol{\theta}_{y,t'}$. If $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ is approximated by a low-rank matrix and the task similarity matrix $\boldsymbol{\Gamma}$ enjoys nice structure such as being approximately low-rank or sparse, $\boldsymbol{H}\boldsymbol{\theta}_y$ may be approximately computed even more efficiently.

6 Experiments

In this section, we experimentally evaluate the performance of the proposed ML-LSPC.

6.1 Toy Dataset

Let the feature dimension be $D = 300$, and we consider T binary classification tasks. Training samples of the t-th task is created as follows: $\boldsymbol{x}_n = (x_{1,n}, \ldots, x_{D,n})^\top$ is independently drawn from the standard normal distribution and $y_{t,n}$ is determined by linear decision boundary $\cos(2\pi t/T)x_{1,n} + \sin(2\pi t/T)x_{2,n}$ (i.e., the decision boundaries are rotated in the subspace spanned by the first two dimensions). We set the number of training samples to $N = 2000$. The label similarity $W_{t,t'}$ is set to $\max(0, \rho_{t,t'})$, where $\rho_{t,t'}$ is the Pearson correlation coefficient between $\{y_{t,n}\}_{n=1}^N$ and $\{y_{t',n}\}_{n=1}^N$. We use the Gaussian kernel model in LSPCs.



(a) Misclassification rate (b) Computation time

Figure 1: Toy dataset.

As functions of the number of tasks, we compare the classification performance of the plain LSPC (i.e., each task is solved separately), the proposed ML-LSPC, the k-*nearest neighbor classifier for multi-label learning* (ML-kNN) [7] which treats a multi-label problem as a set of single-label problems, and the *instance differentiation* method (InsDif) [8] which utilizes a multi-instance formulation in multi-label problems.

All tuning parameters were optimized based on 5-fold cross-validation in terms of the misclassification rate. Fig. 1(a) plots the average misclassification rate over 50 runs, showing that ML-LSPC and InsDif perform well. Plain LSPC and ML-kNN performed poorly because they did not explicitly take label correlations into account. Fig. 1(b) plots the computation time of ML-LSPC with naive implementation (we used the left-division function '`mldivide`' in MATLAB®), the proposed optimization method (we used the conjugate gradient function '`pcg`' in MATLAB®), InsDif, and ML-kNN. This shows that the proposed optimization method is computationally much more efficient than the naive implementation of ML-LSPC. InsDif is slow because it includes clustering of a bag of samples.

6.2 Enron Email Dataset

Finally, we test the performance of the proposed method on the *Enron Email Dataset*, which consists of 1702 real-world email messages [1]. Each email message is represented as a 1001-dimensional feature vector, accompanied with 53 labels. We randomly chose $N = 1000$ samples for training, and used the remaining 702 samples for performance evaluation. Because the presence and absence of labels were highly imbalanced in this dataset, we decided to evaluate the test performance in terms of the *F-measure*. The average F-scores (and computation time) for plain LSPC, ML-LSPC, ML-kNN, and InsDif over 150 runs were 0.556 (3.3 sec.), 0.561 (5.5), 0.372 (9.6), and 0.526 (470.7), where ML-LSPC was significantly better than others according to the t-test at the significance level 5%.

7 Conclusions

Multi-label classification is useful in various real-world problems such as audio tagging, image annotation, video search, and text mining. However, because the essential number of training samples for T-dimensional label vectors of size N is NT, naive implementation of multi-label classification is computationally expensive when N and T are large. To overcome this computational bottleneck, we developed a multi-label method based on LSPC [4, 6]. Our key idea was to utilize the block structure of the system of linear equations to improve the computational efficiency. Through experiments, we showed that the proposed method, ML-LSPC, is promising.

Acknowledgments

HN was supported by the GCOE program, the Hasegawa scholarship, and JST PRESTO, HH was supported by the FIRST program, and MS was supported by MEXT KAKENHI 23300069 and AOARD.

References

[1] The CALO Project, "Enron email dataset," 2009. http://www.cs. cmu.edu/~enron/.

[2] V. Sima, Algorithms for Linear-Quadratic Optimization, Marcel Dekker, New York, NY, USA, 1996.

[3] J. Simm, M. Sugiyama, and T. Kato, "Computationally efficient multi-task learning with least-squares probabilistic classifiers," IPSJ Transactions on Computer Vision and Applications, vol.3, pp.1–8, 2011.

[4] M. Sugiyama, "Superfast-trainable multi-class probabilistic classifier by least-squares posterior fitting," IEICE Transactions on Information and Systems, vol.E93-D, no.10, pp.2690–2701, 2010.

[5] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," International Journal of Data Warehousing and Mining, vol.3, no.3, pp.1–13, 2007.

[6] M. Yamada, M. Sugiyama, G. Wichern, and J. Simm, "Improving the accuracy of least-squares probabilistic classifiers," IEICE Transactions on Information and Systems, vol.E94-D, no.6, pp.1337–1340, 2011.

[7] M.L. Zhang and Z.H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," Pattern Recognition, vol.40, no.7, pp.2038–2048, 2007.

[8] M.L. Zhang and Z.H. Zhou, "Multi-label learning by instance differentiation," Proceedings of the AAAI Conference on Artificial Intelligence, pp.669–674, 2007.