LEAST-SQUARES PROBABILISTIC CLASSIFIER: A COMPUTATIONALLY EFFICIENT ALTERNATIVE TO KERNEL LOGISTIC REGRESSION

Masashi Sugiyama[1], *Hirotaka Hachiya*[1], *Makoto Yamada*[1], *Jaak Simm*[2], *and Hyunha Nam*[1]

[1]Tokyo Institute of Technology, Japan
[2]Tallinn University of Technology, Estonia

ABSTRACT

The *least-squares probabilistic classifier* (LSPC) is a computationally efficient alternative to kernel logistic regression (KLR). A key idea for the speedup is that, unlike KLR that uses maximum likelihood estimation for a log-linear model, LSPC uses least-squares estimation for a linear model. This allows us to obtain a global solution analytically in a class-wise manner. In exchange for the speedup, however, this linear least-squares formulation does not necessarily produce a non-negative estimate. Nevertheless, consistency of LSPC is guaranteed in the large sample limit, and rounding up a negative estimate to zero in finite sample cases was demonstrated not to degrade the classification performance in experiments. Thus, LSPC is a practically useful probabilistic classifier. In this paper, we give an overview of LSPC and its extentions to covariate shift, multi-task, and multi-label scenarios. A MATLAB implementation of LSPC is available from '`http://sugiyama-www.cs.titech.ac.jp/~sugi/software/LSPC/`'.

***Index Terms*—** Least-squares probabilistic classifier, kernel logistic regression, covariate shift, multi-task learning, multi-label classification

1. INTRODUCTION

Kernel logistic regression (KLR) is a popular probabilistic classification method for estimating class-posterior probabilities. KLR models the class-posterior probability by a log-linear combination of kernel functions and learns its parameters by penalized maximum likelihood via, e.g., (quasi-) Newton methods [1, 2]. However, training of KLR models is often time-consuming and is not scalable to large datasets.

To cope with this problem, we introduced an alternative method called the *least-squares probabilistic classifier* (LSPC) [3]. LSPC models the class-posterior probability by a linear combination of kernel functions and learns its parameters by regularized least-squares fitting of the true class-posterior probability. Thanks to this simple formulation, the solution of LSPC can be computed analytically in a class-wise manner just by solving a regularized system of linear equations.

So far, LSPC has been successfully applied to image classification [4, 5], audio tagging [4, 6], accelerometer-based human activity recognition [7], and face-based age prediction [8].

In this paper, we review LSPC and its extentions to covariate shift [7], multi-task [5], and multi-label [6] scenarios.

2. PROBABILISTIC CLASSIFICATION BY LSPC

In this section, we review the *least-squares probabilistic classifier* (LSPC) [3].

2.1. Formulation

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

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

2.2. Least-Squares Fitting of Class-Posterior Probabilities

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

$$\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_B(\boldsymbol{x}))^\top \in \mathbb{R}^B \tag{1}$$

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:

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

where $p(\boldsymbol{x})$ denotes the marginal density of feature vector $\boldsymbol{x}$. Expanding the squared term, we can express J_y as

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

where $p(y|\boldsymbol{x}) = p(\boldsymbol{x}|y)p(y)/p(\boldsymbol{x})$ is used 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:

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

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

$$\boldsymbol{\Phi} = (\boldsymbol{\phi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_N))^\top \in \mathbb{R}^{N \times B} \tag{3}$$

is the design matrix, and $\boldsymbol{\pi}_y$ is the N-dimensional class-indicator vector defined as

$$\pi_{y,n} = \begin{cases} 1 & (y_n = y), \\ 0 & (y_n \neq y). \end{cases} \tag{4}$$

Taking the derivative of $\widehat{J}_y$ with respect to $\boldsymbol{\theta}_y$ and setting it to zero, we can obtain the minimizer $\widehat{\boldsymbol{\theta}}_y$ 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. Note that this is essentially the same formulation as *ridge regression* (i.e., ℓ_2-regularized least-squares) [9] with target $\boldsymbol{\pi}_y$.

As the number of training samples 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 [3]. For a finite sample size, we obtain the final solution by rounding up a negative output to zero and normalization as follows [4]:

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

This method is called LSPC.

Thanks to the analytic solution, LSPC was demonstrated to be computationally much more efficient than kernel logistic regression, whereas the classification accuracy is kept comparable [3, 4].

A possible variation is to use the ℓ_1-regularizer,

$$\sum_{b=1}^{B} |\theta_{y,b}|,$$

instead of the ℓ_2-regularizer $\|\boldsymbol{\theta}_y\|^2 = \sum_{b=1}^{B} \theta_{y,b}^2$. Then this is essentially the same formulation as the *least absolute shrinkage and selection operator* (Lasso) [10] with target $\boldsymbol{\pi}_y$. Because Lasso tends to produce a sparse solution, its solution can be computed efficiently [11, 12, 13, 14, 15, 16, 17]. Furthermore, solutions for all regularization parameters can be computed efficiently by *parametric programming* [18].

3. COMPARISON WITH KERNEL LOGISTIC REGRESSION

In this section, we methodologically and experimentally compare LSPC with *kernel logistic regression* (KLR).

3.1. Methodological Comparison

KLR models the class-posterior probability $p(y|\boldsymbol{x})$ by a log-linear combination of kernel functions:

$$q'(y|\boldsymbol{x}; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_Y) := \frac{1}{Z(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_Y)} \exp\left(\sum_{b=1}^{B} \theta_{y,b} \phi_b(\boldsymbol{x})\right),$$

where $Z(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_Y)$ is the normalization factor defined by

$$Z(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_Y) := \sum_{y=1}^{Y} \exp\left(\sum_{b=1}^{B} \theta_{y,b} \phi_b(\boldsymbol{x})\right).$$

Then the parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_Y$ are learned so that the penalized log-likelihood is maximized:

$$\max_{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_Y} \left[\sum_{n=1}^{N} \log q'(y_n|\boldsymbol{x}_n; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_Y) - \sum_{y=1}^{Y} \eta_y \|\boldsymbol{\theta}_y\|^2\right],$$

where $\eta_y > 0$ is the penalty parameter for class y.

In the maximum likelihood formulation, the likelihood diverges to infinity if the normalization factor $Z(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_Y)$ is omitted. Thus, the normalization factor needs to be included. However, because $Z(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_Y)$ depends on parameters of all classes, training of KLR needs to be carried out simultaneously for all classes. This involves NY parameters, which can be cumbersome in multi-class problems.

On the other hand, the consistency of LSPC (i.e., convergence of the LSPC solution to the true class-posterior probability in the large sample limit) is theoretically guaranteed without normalization. This is a notable advantage of the least-squares formulation because this allows us to solve the LSPC optimization problem separately in a class-wise manner. Then the optimization problem for each class involves only N parameters. Thus, LSPC is computationally highly efficient in multi-class scenarios.

The penalized log-likelihood in KLR does not possess useful structure to speed up optimization. Thus, we need to simply use a generic non-linear optimization technique such as (quasi-) Newton methods, which is practically convenient but not necessarily computationally efficient. When the KLR optimization problem is solved by the Newton method, the optimization procedure is reduced to *iteratively-reweighted least-squares* [1], which requires to solve a system of linear equations (of size NY) in each iteration. On the other hand, LSPC requires to solve a system of linear equations (of size N) only once for each class y, which provides significant speedup in practice.

However, a potential weakness of LSPC is that, solutions can be negative particularly in small sample cases. In practice, such negative outputs are merely rounded-up to zero, which was demonstrated not to degrade the classification performance severely [3, 4].

3.2. Experimental Comparison

Next, we experimentally compare the performance of LSPC and KLR on real-world image classification and audio tagging.

3.2.1. Setup

For both LSPC and KLR, we use the Gaussian kernel model, i.e., we set $B = N$ and

$$\phi_b(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_b\|^2}{2\sigma^2}\right),$$

where σ is the Gaussian width. We determine the kernel width σ and regularization parameter λ based on 2-fold cross-validation.

For computing LSPC solutions, we use our own MATLAB implementation[1]. For KLR, we use the MATLAB implementation included in the 'minFunc' package [19].

We compare the classification performance and computation time of LSPC and KLR. When evaluating the classification performance, it is important to take into account both the false positive rate and the true positive rate. Here we adopt the *area under the ROC curve* (AUC) as our error metric [20]. For computation time, we evaluate the CPU computation time required for training each classifier after the Gaussian width and the regularization parameter are chosen by cross-validation.

3.2.2. PASCAL VOC 2010 Datasets

We use the *PASCAL Visual Object Classes (VOC) 2010* dataset [21] for image classification experiments, which consists of 20 binary classification tasks of identifying the existence of a person, aeroplane, etc. in each image. The total number of images in the dataset is 11319, and we use 1000 randomly chosen images for training and the rest for testing.

Feature extraction from images is carried out as follows. We first extract visual features from each image by the *Speed Up Robust Features* (SURF) algorithm [22]. We then run the *k-means* clustering algorithm in the SURF space and obtained 500 cluster centers as *visual words*. Finally, we construct a 500-dimensional *bag-of-feature* vector by counting the number of visual words in each image, which is used as feature vector $\boldsymbol{x}$ in LSPC and KLR.

Table 1 shows the mean AUC values (with standard deviations in parentheses) over 50 trials. Average computation time is also included at the bottom of the table. The results show that LSPC is slightly more accurate than KLR with much less computational cost.

3.2.3. Freesound Datasets

For audio-tagging experiments, we use the data collected by the *Freesound* project [23], which consists of various audio files annotated with word tags such as 'people', 'noisy', and 'restaurant'. The goal is to predict the existence of each tag for a new audio file.

We extract audio files from among all files in the dataset containing any of the 50 most used tags and between 3–60 seconds in length. We then use 180 randomly selected uncompressed audio files with a sampling rate greater than 44.1kHz as our training set, and 1500 randomly selected audio files that are stored in a compressed format for testing. We use the *hidden Markov kernel* [24], instead of the simple Gaussian kernel due to the sequential nature of audio files.

We compute the AUC value over all test samples for each tag, and this is averaged over all tags. Table 2 summarizes the accuracy and computation time of LSPC and KLR over 50

[1]The software is publicly available from '`http://sugiyama-www.cs.titech.ac.jp/~sugi/software/LSPC/`'.

Table 1. Mean AUC values (with standard deviations in parentheses) over 50 trials for the PASCAL VOC dataset. The best method in terms of the mean AUC and comparable methods according to the *t-test* at the significance level 5% are specified by bold face. Average computation time is also included at the bottom.

	LSPC	KLR
Aeroplane	**82.6**(1.0)	**83.0**(1.3)
Bicycle	**77.7**(1.7)	76.6(3.4)
Bird	68.7(2.0)	**70.8**(2.2)
Boat	**74.4**(2.0)	72.8(2.6)
Bottle	**65.4**(1.8)	62.1(4.3)
Bus	**85.4**(1.4)	**85.6**(1.4)
Car	**73.0**(0.8)	72.1(1.2)
Cat	**73.6**(1.4)	**74.1**(1.7)
Chair	**71.0**(1.0)	**70.5**(1.0)
Cow	**71.7**(3.2)	69.3(3.6)
Diningtable	**75.0**(1.6)	71.4(2.7)
Dog	**69.6**(1.0)	**69.4**(1.8)
Horse	**64.4**(2.5)	61.2(3.2)
Motorbike	**77.0**(1.7)	75.9(3.3)
Person	**67.6**(0.9)	67.0(0.8)
Pottedplant	**66.2**(2.6)	61.9(3.2)
Sheep	**77.8**(1.6)	74.0(3.8)
Sofa	**67.4**(2.7)	65.4(4.6)
Train	**79.2**(1.3)	**78.4**(3.0)
Tvmonitor	**76.7**(2.2)	**76.6**(2.3)
Average AUC	73.2	71.9
Average training time [sec]	0.7	24.6

Table 2. Mean AUC values (with standard deviations in parentheses) over all audio files for the Freesound dataset. The *t-test* at the significance level 5% says that there is no significant difference in AUC obtained by LSPC and KLR. Computation time is also included at the bottom.

	LSPC	KLR
AUC	70.1 (9.6)	66.7 (10.3)
Training time [sec]	0.005	0.612

runs, showing that LSPC provides comparable classification performance to KLR with significantly less computation time.

4. EXTENSIONS OF LSPC

In this section, we review extentions of LSPC to covariate shift [7], multi-task [5], and multi-label [6] scenarios.

4.1. Probabilistic Classification under Covariate Shift

The *covariate shift* [25, 26] is a situation where training and test input samples follow different distributions, but the class-posterior probability remains unchanged[2]. Here, we review an extention of LSPC to the covariate shift scenario called the *importance-weighted LSPC* (IW-LSPC) [7].

4.1.1. Formulation

Let us consider a *semi-supervised learning* setup [27], where unlabeled samples $\{\boldsymbol{x}'_n\}_{n=1}^{N'}$ are given in addition to labeled samples $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$. Suppose unlabeled samples $\{\boldsymbol{x}'_n\}_{n=1}^{N'}$ are drawn independently from a distribution with density $p'(\boldsymbol{x})$, which can be generally different from $p(\boldsymbol{x})$ that $\{\boldsymbol{x}_n\}_{n=1}^{N}$ follow. Under this covariate shift setup, naively estimating a class-posterior probability by LSPC causes a bias due to the distribution difference. Below, we explain how this bias can be reduced.

4.1.2. Importance-Weighted Least-Squares Fitting of Class-Posterior Probabilities

The key technique for covariate shift adaptation is *importance sampling* [28]—the expectation of a loss function $\mathrm{loss}(\boldsymbol{x})$ over test input density $p'(\boldsymbol{x})$ can be consistently estimated by samples $\{\boldsymbol{x}_n\}_{n=1}^{N}$ drawn from training input density $p(\boldsymbol{x})$ as

$$\int \mathrm{loss}(\boldsymbol{x})p'(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \int \mathrm{loss}(\boldsymbol{x})w(\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \approx \frac{1}{N}\sum_{n=1}^{N}\mathrm{loss}(\boldsymbol{x}_n)w(\boldsymbol{x}_n),$$

where $w(\boldsymbol{x})$ is called the *importance weight* defined by

$$w(\boldsymbol{x}) := \frac{p'(\boldsymbol{x})}{p(\boldsymbol{x})}.$$

Because various methods to estimate the importance weight from samples $\{\boldsymbol{x}_n\}_{n=1}^{N}$ and $\{\boldsymbol{x}'_n\}_{n=1}^{N'}$ are available [29, 30, 31, 32, 33, 34, 35], we treat the importance weight $w(\boldsymbol{x})$ as known below.

Under the covariate shift, the least-squares criterion is defined as

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

where 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

[2]The term 'covariate' refers to feature vector $\boldsymbol{x}$. Thus, the 'covariate shift' indicates the fact that the distribution of covariates 'shifts'.

factor $1/N$, and including an ℓ_2-regularizer, we have the following training criterion:

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

where $\rho' > 0$ is the regularization parameter, $\boldsymbol{\pi}_y$ is defined by Eq.(4), $\boldsymbol{\Phi}$ is defined by Eq.(3), and

$$\boldsymbol{W} := \mathrm{diag}\left(w(\boldsymbol{x}_1), \ldots, w(\boldsymbol{x}_N)\right).$$

Taking the derivative of $\widehat{J}'_y$ with respect to $\boldsymbol{\theta}_y$ and setting it to zero, we can obtain the minimizer $\widehat{\boldsymbol{\theta}}_y$ analytically as

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

Thanks to importance sampling, this importance-weighted LSPC solution is less biased than the plain LSPC solution. However, it tends to have a larger variance [26]. To optimally control the bias-variance trade-off, it is effective to slightly flatten the importance weights [25, 36]. The level of flattening may be controlled by a model selection method [25, 37, 38].

As experimentally demonstrated in [7], IW-LSPC can successfully mitigate the influence of covariate shift in a computationally efficient manner.

4.2. Multi-Task Probabilistic Classification

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* [39]. A computationally efficient multi-task learning method can be developed by combining multiple LSPCs. Here, we first review multi-task LSPC (MT-LSPC) [5] in a slightly generalized way, and then we derive another formulation of MT-LSPC.

4.2.1. Formulation

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 $t \in \{1, \ldots, T\}$.

4.2.2. MT-LSPC

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 $\boldsymbol{\phi}(\boldsymbol{x})$ is the B-dimensional basis function vector defined by Eq.(1) 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 follows the line of [40], i.e., 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

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

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.

Let

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

where $\boldsymbol{0}_B$ denotes the B-dimensional vector with all zeros. Then the MT-LSPC training criterion can be compactly expressed as

$$\begin{aligned}\widehat{J}_y^{\mathrm{MT}}(\boldsymbol{\beta}_y) &= \frac{1}{2}\boldsymbol{\beta}_y^\top \boldsymbol{\Xi}^\top \boldsymbol{\Xi}\boldsymbol{\beta}_y - \boldsymbol{\beta}_y^\top \boldsymbol{\Xi}^\top \boldsymbol{\pi}_y \\ &\quad + \frac{1}{2}\boldsymbol{\beta}_y^\top (\boldsymbol{\Omega} \otimes \boldsymbol{I}_B)\boldsymbol{\beta}_y, \qquad (5)\end{aligned}$$

where $\boldsymbol{\pi}_y$ is defined by Eq.(4) and $\otimes$ denotes the *Kronecker product*, i.e., for $\boldsymbol{E} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{F} \in \mathbb{R}^{p\times q}$, $\boldsymbol{E}\otimes\boldsymbol{F}$ is defined

as

$$\boldsymbol{E} \otimes \boldsymbol{F} = \begin{pmatrix} E_{1,1}\boldsymbol{F} & \cdots & E_{1,n}\boldsymbol{F} \\ \vdots & \ddots & \vdots \\ E_{m,1}\boldsymbol{F} & \cdots & E_{m,n}\boldsymbol{F} \end{pmatrix} \in \mathbb{R}^{mp \times nq}.$$

Note that Eq.(5) is essentially the same form as the original single-task LSPC training criterion (2). Thus, taking the derivative of this training criterion with respect to $\boldsymbol{\beta}_y$ and setting it to zero, we have 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.

Specifically, $q(y|\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_{y,t})$ can be expressed as follows (see Eq.(147) of [41]):

$$\begin{aligned} q(y|\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_{y,t}) &= \widehat{\boldsymbol{\theta}}_{y,t}^\top \boldsymbol{\phi}(\boldsymbol{x}) = \widehat{\boldsymbol{\beta}}_y^\top \boldsymbol{\xi}_t(\boldsymbol{x}) \\ &= \boldsymbol{\pi}_y^\top \boldsymbol{A}^{-1} \boldsymbol{b}_t, \end{aligned} \tag{6}$$

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

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

$\delta_{t,t'}$ is the Kronecker delta defined as

$$\delta_{t,t'} = \begin{cases} 1 & (t = t'), \\ 0 & (t \neq t'). \end{cases} \tag{7}$$

The computational complexity for computing the solution based on Eq.(6) is reduced to $\mathcal{O}(N^3)$, which is independent of T.

As experimentally demonstrated in [5], MT-LSPC is computationally much more efficient than an alternative multi-task approach based on kernel logistic regression [42], with comparable accuracy.

4.2.3. Pairwise Formulation of MT-LSPC

The above MT-LSPC formulation imposes solutions to be close to each other via the common part $\boldsymbol{\beta}_{0,y}$. Another way to impose such a multi-task penalty is to consider pairwise similarities between tasks [43].

More specifically, 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.

$$\begin{aligned} \widehat{J}_y^{\mathrm{MT}'}(\boldsymbol{\theta}_y) := \frac{1}{2}\sum_{n=1}^{N} q(y|\boldsymbol{x}_n; \boldsymbol{\theta}_{y,t_n})^2 - \sum_{n:y_n=y} q(y|\boldsymbol{x}_n; \boldsymbol{\theta}_{y,t_n}) \\ + \frac{1}{2}\sum_{t=1}^{T} \lambda_t \|\boldsymbol{\theta}_{y,t}\|^2 + \frac{1}{4}\sum_{t,t'=1}^{T} \gamma_{t,t'} \|\boldsymbol{\theta}_{y,t} - \boldsymbol{\theta}_{y,t'}\|^2, \end{aligned} \tag{8}$$

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

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

where $\boldsymbol{\pi}_y$ is defined by Eq.(4) and $\boldsymbol{C}$ is the $T \times T$ matrix defined as

$$C_{t,t'} := \delta_{t,t'} \left(\lambda_t + \sum_{t''=1}^{T} \gamma_{t,t''}\right) - \gamma_{t,t'}. \tag{9}$$

$\delta_{t,t'}$ is the Kronecker delta defined by Eq.(7).

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 Eq.(6), $q(y|\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_{y,t})$ can be efficiently computed based on the following expression:

$$\begin{aligned} q(y|\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_{y,t}) &= \widehat{\boldsymbol{\theta}}_{y,t}^\top \boldsymbol{\phi}(\boldsymbol{x}) = \widehat{\boldsymbol{\theta}}_y^\top \boldsymbol{\psi}_t(\boldsymbol{x}) \\ &= \boldsymbol{\pi}_y^\top \boldsymbol{A}'^{-1} \boldsymbol{b}'_t, \end{aligned} \tag{10}$$

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 based on Eq.(10) 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.

When the task similarity matrix $\boldsymbol{\Gamma}$ is unknown, we may jointly learn $\boldsymbol{\Gamma}$ and $\boldsymbol{\theta}_y$ as follows: Starting from the uniform task similarities $\gamma_{t,t'} = \gamma > 0$ for all $t, t' = 1, \ldots, T$, learn $\boldsymbol{\theta}_y$ based on the current $\boldsymbol{\Gamma}$, learn $\gamma_{t,t'}$ based on the distance between current $\boldsymbol{\theta}_{y,t}$ and $\boldsymbol{\theta}_{y,t'}$, and iterate this until convergence.

4.3. Multi-Label Probabilistic Classification

Multi-label classification allows a sample to belong to multiple classes simultaneously [44], which is often the case in real-world applications such as audio tagging and image annotation. In such a multi-label scenario, taking into account correlation between multiple labels can boost the classification accuracy. However, this makes classifier training more challenging because handling multiple labels tends to induce a high-dimensional optimization problem. Here, we review a computationally efficient multi-task classifier based on LSPC called *multi-label LSPC* (ML-LSPC) [6].

4.3.1. Formulation

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 $t \in \{1, \ldots, T\}$.

4.3.2. ML-LSPC

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 (8), we immediately have the training criterion for ML-LSPC:

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

However, a notable difference between the multi-task and multi-label formulations is that the number of training samples is N in the multi-task formulation (see Section 4.2.1), 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 expensive. Below, we explain how this computational bottleneck can be overcome.

4.3.3. Training ML-LSPC via Sylvester Equation

Let $\boldsymbol{\Theta}_y$ be the matrix of parameters $\boldsymbol{\theta}_{y,1}, \ldots, \boldsymbol{\theta}_{y,T}$:

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

Let $\boldsymbol{\pi}_{y,t}$ be the N-dimensional class-indicator vector for the t-th label:

$$\pi_{y,t,n} = \begin{cases} 1 & (y_{n,t} = y), \\ 0 & (y_{n,t} \neq y), \end{cases} \tag{11}$$

and let $\boldsymbol{\Pi}_y$ be the matrix of class-indicator vectors $\boldsymbol{\pi}_{y,1}, \ldots, \boldsymbol{\pi}_{y,T}$:

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

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

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

where $\boldsymbol{\Phi}$ is defined by Eq.(3) and $\boldsymbol{C}$ is defined by Eq.(9). Taking the derivative of the above equation with respect to $\boldsymbol{\Theta}_y$ and setting it to zero, we obtain

$$\boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{\Theta}_y + \boldsymbol{\Theta}_y \boldsymbol{C} = \boldsymbol{\Phi}^\top \boldsymbol{\Pi}_y. \tag{12}$$

This is called the *continuous Sylvester equation* with respect to $\boldsymbol{\Theta}_y$, which often arises in control theory [45].

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 $\widehat{\boldsymbol{\Theta}}_y$ of Eq.(12) is given analytically as

$$\widehat{\boldsymbol{\Theta}}_y = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_B) \boldsymbol{Q} (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_T)^\top, \tag{13}$$

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.(12) via Eq.(13) 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.

4.3.4. Training ML-LSPC via Conjugate Gradient

For large-scale data, Eq.(12) may be solved more efficiently by numerical optimization. Let $\boldsymbol{\theta}_y$ be the vector of parameters $\boldsymbol{\theta}_{y,1}, \ldots, \boldsymbol{\theta}_{y,T}$:

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

Then Eq.(12) can be expressed as

$$\boldsymbol{H}\boldsymbol{\theta}_y = \boldsymbol{h}_y, \tag{14}$$

where

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

$\boldsymbol{\Phi}$ is defined by Eq.(3), $\boldsymbol{C}$ is defined by Eq.(9), and $\boldsymbol{\pi}_{y,t}$ is defined by Eq.(11).

If a kernel model is used (i.e., $B = N$), naively solving Eq.(14) 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

$$\boldsymbol{H}\boldsymbol{\theta}_y = \begin{pmatrix} \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{\theta}_{y,1} + \sum_{t=1}^T C_{1,t} \boldsymbol{\theta}_{y,t} \\ \vdots \\ \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{\theta}_{y,T} + \sum_{t=1}^T C_{T,t} \boldsymbol{\theta}_{y,t} \end{pmatrix}.$$

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.

As experimentally demonstrated in [6], ML-LSPC with the above implementation contributes highly to reducing the computation time.

5. CONCLUSIONS

In this paper, we reviewed a computationally efficient alternative to kernel logistic regression (KLR) called the least-squares probabilistic classifier (LSPC), and its extentions to covariate shift, multi-task, and multi-label scenarios. The computational efficiency of LSPC is particularly useful in multi-task and multi-label scenarios because a large number of samples need to be processed.

Recently, *learning from crowds* has gathered a great deal of attention [46, 47, 48, 49], where a large number of training samples are provided by multiple (possibly unreliable) labelers. In a such scenario, a multi-task formulation was shown to be useful [50]. We expect that high computational efficiency of LSPC will play a key role in the crowdsourcing era.

6. REFERENCES

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, New York, NY, USA, 2001.

[2] T. P. Minka, "A comparison of numerical optimizers for logistic regression," Tech. Rep., Microsoft Research, 2007.

[3] 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.

[4] 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.

[5] 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.

[6] H. Nam, H. Hachiya, and M. Sugiyama, "Computationally efficient multi-label classification by least-squares probabilistic classifier," in *Proceedings of 2012 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP2012)*, Kyoto, Japan, Mar. 25–30 2012.

[7] H. Hachiya, M. Sugiyama, and N. Ueda, "Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition," *Neurocomputing*, vol. 80, pp. 93–101, 2012.

[8] K. Ueki, M. Sugiyama, Y. Ihara, and M. Fujita, "Multi-race age estimation based on the combination of multiple classifiers," in *Proceedings of the First Asian Conference on Pattern Recognition (ACPR2011)*, Beijing, China, Nov. 28–30 2011, pp. 633–637.

[9] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 3, pp. 55–67, 1970.

[10] R. Tibshirani, "Regression shrinkage and subset selection with the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, no. 1, pp. 267–288, 1996.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[12] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics*, vol. LVII, no. 11, pp. 1413–1457, 2004.

[13] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Modeling and Simulation*, vol. 4, no. 4, pp. 1168–1200, 2005.

[14] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinvesky, "An interior-point method for large-scale ℓ_1-regularized least squares," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, 2007.

[15] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, "Bregman iterative algorithms for L1-minimization with applications to compressed sensing," *SIAM Journal of Imaging Sciences*, vol. 1, no. 1, pp. 143–168, 2008.

[16] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, 2009.

[17] R. Tomioka, T. Suzuki, and M. Sugiyama, "Super-linear convergence of dual augmented Lagrangian algorithm for sparsity regularized estimation," *Journal of Machine Learning Research*, vol. 12, pp. 1537–1586, May 2011.

[18] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[19] M. Schmidt, *minFunc*, 2005, `http://people.cs.ubc.ca/\~{}schmidtm/Software/minFunc.html`.

[20] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[21] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results," 2010, `http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html`,

[22] H.t Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[23] The Freesound Project, "Freesound," 2011, `http://www.freesound.org`.

[24] G. Wichern, J. Xue, H. Thornburg, B. Mechtley, and A. Spanias, "Segmentation, indexing and retrieval of environmental and natural sounds," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 3, pp. 688–707, 2010.

[25] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of Statistical Planning and Inference*, vol. 90, no. 2, pp. 227–244, 2000.

[26] M. Sugiyama and M. Kawanabe, *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*, MIT Press, Cambridge, MA, USA, 2012.

[27] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*, MIT Press, Cambridge, MA, USA, 2006.

[28] G. S. Fishman, *Monte Carlo: Concepts, Algorithms, and Applications*, Springer-Verlag, Berlin, Germany, 1996.

[29] J. Qin, "Inferences for case-control and semiparametric two-sample density ratio models," *Biometrika*, vol. 85, no. 3, pp. 619–630, 1998.

[30] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, "Covariate shift by kernel mean matching," in *Dataset Shift in Machine Learning*, J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, Eds., chapter 8, pp. 131–160. MIT Press, Cambridge, MA, USA, 2009.

[31] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe, "Direct importance estimation for covariate shift adaptation," *Annals of the Institute of Statistical Mathematics*, vol. 60, no. 4, pp. 699–746, 2008.

[32] X. Nguyen, M. J. Wainwright, and M. I. Jordan, "Estimating divergence functionals and the likelihood ratio by convex risk minimization," *IEEE Transactions on Information Theory*, vol. 56, no. 11, pp. 5847–5861, 2010.

[33] T. Kanamori, S. Hido, and M. Sugiyama, "A least-squares approach to direct importance estimation," *Journal of Machine Learning Research*, vol. 10, pp. 1391–1445, Jul. 2009.

[34] M. Sugiyama, T. Suzuki, and T. Kanamori, "Density ratio matching under the Bregman divergence: A unified framework of density ratio estimation," *Annals of the Institute of Statistical Mathematics*, 2012, to appear.

[35] M. Sugiyama, T. Suzuki, and T. Kanamori, *Density Ratio Estimation in Machine Learning*, Cambridge University Press, Cambridge, UK, 2012.

[36] M. Yamada, T. Suzuki, T. Kanamori, H. Hachiya, and M. Sugiyama, "Relative density-ratio estimation for robust distribution comparison," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, Eds., 2011, pp. 594–602.

[37] M. Sugiyama and K.-R. Müller, "Input-dependent estimation of generalization error under covariate shift," *Statistics & Decisions*, vol. 23, no. 4, pp. 249–279, 2005.

[38] M. Sugiyama, M. Krauledat, and K.-R. Müller, "Covariate shift adaptation by importance weighted cross validation," *Journal of Machine Learning Research*, vol. 8, pp. 985–1005, May 2007.

[39] R. Caruana, L. Pratt, and S. Thrun, "Multitask learning," *Machine Learning*, vol. 28, pp. 41–75, 1997.

[40] T. Evgeniou and M. Pontil, "Regularized multi-task learning," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2004)*. 2004, pp. 109–117, ACM.

[41] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," Tech. Rep., Technical University of Denmark, 2008.

[42] À. Lapedriza, D. Masip, and J. Vitrià, "A hierarchical approach for multi-task logistic regression," in *Proceedings of the 3rd Iberian Conference on Pattern Recognition and Image Analysis, Part II*, J. Mart, J. M. Bened, A. M. Mendonga, and J. Serrat, Eds., Berlin, Germany, 2007, vol. 4478 of *Lecture Notes in Computer Science*, pp. 258–265, Springer-Verlag.

[43] T. Kato, H. Kashima, M. Sugiyama, and K. Asai, "Conic programming for multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 7, pp. 957–968, 2010.

[44] 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.

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

[46] M. Steyvers, M. Lee, B. Miller, and P. Hemmer, "The wisdom of crowds in the recollection of order information," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, Eds., pp. 1785–1793. 2009.

[47] J. Whitehill, P. Ruvolo, T.-F. Wu, J. Bergsma, and J. Movellan, "Whose vote should count more: Optimal integration of labels from labelers of unknown expertise," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, Eds., pp. 2035–2043. 2009.

[48] P. Welinder, S. Branson, S. Belongie, and P. Perona, "The multidimensional wisdom of crowds," in *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. K. I. Williams, R. Zemel, J. Shawe-Taylor, and A. Culotta, Eds., pp. 2424–2432. 2010.

[49] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, "Learning from crowds," *Journal of Machine Learning Research*, vol. 11, pp. 1297–1322, 2010.

[50] H. Kajino and H. Kashima, "Convex formulations of multi-task learning from crowds," *Journal of Japanese Society of Artificial Intelligence*, vol. 27, no. 3, 2012, to appear.