Computationally Efficient Multi-task Learning with Least-Squares Probabilistic Classifiers

Jaak Simm
Centre for Biology of Integrated Systems,
Tallinn University of Technology, Estonia.

Masashi Sugiyama
Department of Computer Science,
Tokyo Institute of Technology, Japan.
sugi@cs.titech.ac.jp http://sugiyama-www.cs.titech.ac.jp/~sugi

Tsuyoshi Kato
Department of Computer Science,
Gunma University, Japan

Abstract

Probabilistic classification and multi-task learning are two important branches of machine learning research. Probabilistic classification is useful when the 'confidence' of decision is necessary. On the other hand, the idea of multi-task learning is beneficial if multiple related learning tasks exist. So far, kernelized logistic regression has been a vital probabilistic classifier for the use in multi-task learning scenarios. However, its training tends to be computationally expensive, which prevented its use in large-scale problems. To overcome this limitation, we propose to employ a recently-proposed probabilistic classifier called the least-squares probabilistic classifier in multi-task learning scenarios. Through image classification experiments, we show that our method achieves comparable classification performance to the existing method, with much less training time.

1 Introduction

Probabilistic classification (PC) and *multi-task learning* (MTL) are two important research topics in the area of machine learning.

In real-world classification scenarios, one often wants to know the 'confidence' of classification results. This is because if the confidence is turned out to be low, one may give up automatically classifying the pattern and instead manually classify it. A standard PC approach tries to learn the class-posterior probability (i.e., the probability of a test pattern belonging to each class), which can be directly translated into the confidence of classification. *Logistic regression* (LR) [6] would be a representative PC method.

On the other hand, MTL deals with the case where multiple related learning tasks exist. The rationale behind MTL is that, rather than solving multiple learning tasks separately, solving them simultaneously by sharing some common information behind the tasks may improve the classification accuracy [2]. A popular approach to MTL is to impose the solutions of related tasks to be similar to each other. This allows related tasks to implicitly share training samples effectively [4].

In this paper, we focus on the MTL problem in PC scenarios. So far, LR classifiers have been applied to MTL and shown to perform well in experiments [7]. However, when a kernelized version of LR (KLR) is used for non-linear classification, its training is computationally highly expensive for kernel functions producing *dense* kernel matrices (e.g., the Gaussian kernel). Although sophisticated non-linear optimization techniques such as Newton's method and quasi-Newton methods can be employed for training KLR classifiers [6, 8], applying KLR to large-scale data is still challenging. This computational inefficiency of KLR becomes more critical in MTL scenarios since a large number of training data gathered from many tasks need to be handled at the same time.

The goal of this paper is to propose a computationally-efficient alternative to the KLR-based MTL method. More specifically, we propose to use an alternative non-linear PC method called *least-squares probabilistic classifiers* (LSPCs) [11], instead of the KLR classifiers, in the MTL scenarios. An advantage of LSPC is that its solution can be computed *analytically* just by solving a regularized system of linear equations. Thus, it is computationally very efficient and stable. We combine the MTL idea proposed in the paper [4] with LSPCs, and develop a computationally-efficient MTL method for PC.

However, naively combining the MTL idea with LSPC still requires a high computational cost—indeed, the computational complexity grows *cubically* with respect to the number of tasks. To ease this problem, we reformulate the optimization problem in the *dual* domain, and show that the solution can be computed exactly with the computational complexity *independent* of the number of tasks. This is the same computational complexity as that of the single-task LSPC method, and therefore the proposed MTL method is computationally highly efficient when a large number of tasks exist.

Through image classification experiments, we demonstrate that the proposed LSPC-based MTL method achieves comparable classification performance to the existing KLR-based MTL method, with the computational cost smaller in two orders of magnitude.

After reviewing LSPC for ordinary single-task classification scenarios in Section 2, we extend it to multi-task classification scenarios in Section 3. Experimental results are reported in Section 4, and we conclude in Section 5 by summarizing our contributions.

2 Single-Task Classification

In this section, we review the original LSPC method for ordinary single-task classification scenarios.

Let us consider a single-task binary classification problem. Suppose we are given N

training points

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

where $\mathbf{x}_n \in \mathbb{R}^d$ are the inputs and $y_n \in \{-1, +1\}$ are the class labels. The goal is to estimate the class-posterior probability $p(y|\mathbf{x})$.

LSPC[1], proposed in the paper[11], models the class-posterior probability $p(y|\mathbf{x})$ by using a linear model

$$\boldsymbol{\alpha}_y^\top \boldsymbol{\phi}(\mathbf{x}),$$

where $\boldsymbol{\alpha}_y \in \mathbb{R}^N$ is the N-dimensional parameter vector for class y, $^\top$ denotes the transpose, and $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^N$ is the N-dimensional feature vector. For example, in the paper [11], the *Gaussian kernel* was used as the feature vector $\boldsymbol{\phi}(\mathbf{x})$:

$$\boldsymbol{\phi}(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_N))^\top,$$

where

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2). \tag{1}$$

For each class $y \in \{-1, +1\}$, LSPC finds the parameter $\boldsymbol{\alpha}_y$ that minimizes the squared error between the true class probability $p(y|\mathbf{x})$ and its model $\boldsymbol{\alpha}_y^\top \boldsymbol{\phi}(\mathbf{x})$ by solving the following optimization problem:

$$\widehat{\boldsymbol{\alpha}}_y = \underset{\boldsymbol{\alpha}_y}{\operatorname{argmin}} \ \frac{1}{2N} \sum_{n=1}^{N} \boldsymbol{\alpha}_y^\top \boldsymbol{\phi}(\mathbf{x}_n) \boldsymbol{\phi}(\mathbf{x}_n)^\top \boldsymbol{\alpha}_y - \frac{1}{N} \sum_{n: y_n = y} \boldsymbol{\alpha}_y^\top \boldsymbol{\phi}(\mathbf{x}_n) + \frac{\lambda}{2} \|\boldsymbol{\alpha}_y\|^2, \tag{2}$$

where λ (≥ 0) is the regularizer parameter. Let

$$\begin{aligned} \boldsymbol{\Phi} &= [\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_N)]^\top, \\ \mathbf{z}_y &= [\delta_{y,y_1}, \ldots, \delta_{y,y_N}]^\top, \end{aligned}$$

where $\delta_{y,y'}$ denotes *Kronecker's delta*:

$$\delta_{y,y'} = \begin{cases} 1 & y = y', \\ 0 & y \neq y'. \end{cases}$$

Then the problem (2) can be compactly rewritten as[2]

$$\widehat{\boldsymbol{\alpha}}_y = \underset{\boldsymbol{\alpha}_y}{\operatorname{argmin}} \ \frac{1}{2N} \boldsymbol{\alpha}_y^\top \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{\alpha}_y - \frac{1}{N} \boldsymbol{\alpha}_y^\top \boldsymbol{\Phi}^\top \mathbf{z}_y + \frac{\lambda}{2} \|\boldsymbol{\alpha}_y\|^2. \tag{3}$$

[1] Note that the LSPC method we are reviewing here is the 'LSPC(full)' method described in the paper [11], where 'full' means that all kernels are used for learning. On the other hand, a more practical version of LSPC where irrelevant kernels are removed was also proposed in the original paper. Here we chose 'LSPC(full)' since this is more suitable in the MTL formulation.

[2] Note that $\boldsymbol{\Phi}$ is a symmetric matrix in the current setup, i.e., $\boldsymbol{\Phi}^\top = \boldsymbol{\Phi}$. However, when the number of kernel functions is reduced, e.g., by random subset selection, $\boldsymbol{\Phi}$ could be a rectangular matrix. For this reason, we decided to explicitly use its transpose $\boldsymbol{\Phi}^\top$ throughout the paper.

λ and σ will be chosen based on cross-validation.

The solution to (3) is given analytically by

$$\widehat{\boldsymbol{\alpha}}_y = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda N \mathbf{I}_N)^{-1} \boldsymbol{\Phi}^\top \mathbf{z}_y,$$

where $\mathbf{I}_N$ denotes the N-dimensional identity matrix. Finally, the class-posterior probability is estimated as follows [13].

$$\widehat{p}(y|\mathbf{x}) = \frac{\max(0, \widehat{\boldsymbol{\alpha}}_y^\top \boldsymbol{\phi}(\mathbf{x}))}{\max(0, \widehat{\boldsymbol{\alpha}}_{-1}^\top \boldsymbol{\phi}(\mathbf{x})) + \max(0, \widehat{\boldsymbol{\alpha}}_{+1}^\top \boldsymbol{\phi}(\mathbf{x}))}.$$

The computational complexity of LSPC is $O(N^3)$.

On the other hand, KLR involves non-linear optimization, and the solution is usually computed using iterative algorithms. Its typical implementation based on Newton's method iteratively solves a weighted least-squares problem [6], which requires $O(N^3)$ computational costs in each iteration. Thus, the computational complexity of LSPC training corresponds to a single iteration of KLR training.

3 Multi-task Classification

When multiple related learning tasks exist, solving them simultaneously by sharing some common information behind the tasks could be more beneficial than solving them separately. Here, we extend the LSPC method to the multi-task scenarios. We first describe our basic idea in Section 3.1, and then we introduce a trick to improve the computational efficiency in Section 3.2.

3.1 Basic Formulation

Suppose there are T binary classification tasks, and each task has a different class-posterior probability $p(y|\mathbf{x}, t)$, where $t \in \{1, \ldots, T\}$ is the task index. The training samples are now accompanied with the task index, i.e.,

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

where $t_n \in \{1, \ldots, T\}$.

The key idea of multi-task learning is to impose solutions of different tasks to be similar to each other [4], by which training samples can be implicitly shared across different tasks. Here, we apply this idea to LSPC, which we refer to as LSPC-MT. More specifically, let us model the class-posterior probability $p(y|\mathbf{x}, t)$ by the following linear model:

$$(\boldsymbol{\beta}_{y,0} + \boldsymbol{\beta}_{y,t})^\top \boldsymbol{\phi}(\mathbf{x}),$$

where $\boldsymbol{\beta}_{y,0}$ is the common part of the solutions for all tasks, and $\boldsymbol{\beta}_{y,t}$ is the individual part of the solution for task t. Then we can express the LSPC training criterion (2) for the multi-task model as

$$\begin{aligned}\widehat{\boldsymbol{\beta}}_y = \underset{\boldsymbol{\beta}_y}{\operatorname{argmin}}\ & \frac{1}{2N}\sum_{n=1}^{N}(\boldsymbol{\beta}_{y,0}+\boldsymbol{\beta}_{y,t_n})^\top \boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^\top(\boldsymbol{\beta}_{y,0}+\boldsymbol{\beta}_{y,t_n}) \\ & -\frac{1}{N}\sum_{n:y_n=y}(\boldsymbol{\beta}_{y,0}+\boldsymbol{\beta}_{y,t_n})^\top \boldsymbol{\phi}(\mathbf{x}_n) \\ & +\frac{\lambda}{2}\|\boldsymbol{\beta}_{y,0}\|^2+\frac{\gamma}{2T}\sum_{t=1}^{T}\|\boldsymbol{\beta}_{y,t}\|^2,\end{aligned} \tag{4}$$

where λ (≥ 0) is the regularization parameter for the shared parameter $\boldsymbol{\beta}_{y,0}$. γ (≥ 0) is the multi-task parameter which controls the strength of the multi-task penalty, i.e., the solutions $\{\boldsymbol{\beta}_{y,0}+\boldsymbol{\beta}_{y,t}\}_{t=1}^{T}$ are imposed to be close to each other. If λ is large enough, the shared component $\boldsymbol{\beta}_{y,0}$ vanishes, and thus we merely have T single-task LSPC models (with a common 'regularization' parameter γ). On the other hand, if γ is large enough, the individual components $\{\boldsymbol{\beta}_{y,t}\}_{t=1}^{T}$ vanish, and thus we have a single LSPC model trained using samples from all tasks. Otherwise, the solution for each task is generally forced to be close to each other.

Naively obtaining the solutions of (4) requires to solve a system of $N(T+1)$ linear equations. This requires $O(N^3T^3)$ computational complexity, which may be intractable when T is large.

3.2 Improving Computational Efficiency

To improve the computational complexity of LSPC-MT, we reformulate the parameters of LSPC-MT as

$$\begin{aligned}\boldsymbol{\omega}_y &= \left(\sqrt{\frac{T\lambda}{\gamma}}\boldsymbol{\beta}_{y,0}^\top, \boldsymbol{\beta}_{y,1}^\top, \ldots, \boldsymbol{\beta}_{y,T}^\top\right)^\top, \\ \boldsymbol{\psi}(\mathbf{x},t) &= \left(\sqrt{\frac{\gamma}{T\lambda}}\boldsymbol{\phi}(\mathbf{x})^\top, \underbrace{\mathbf{0}_N^\top,\ldots,\mathbf{0}_N^\top}_{t-1}, \boldsymbol{\phi}(\mathbf{x})^\top, \underbrace{\mathbf{0}_N^\top,\ldots,\mathbf{0}_N^\top}_{T-t}\right)^\top,\end{aligned}$$

where $\mathbf{0}_N$ denotes the N-dimensional zero vector. This reformulation idea follows a similar line to the paper [4], which focused on MTL for support vector machines. By using the facts that

$$\begin{aligned}\|\boldsymbol{\omega}_y\|^2 &= \frac{T\lambda}{\gamma}\|\boldsymbol{\beta}_{y,0}\|^2+\sum_{t=1}^{T}\|\boldsymbol{\beta}_{y,t}\|^2, \\ \boldsymbol{\omega}_y^\top\boldsymbol{\psi}(\mathbf{x},t) &= (\boldsymbol{\beta}_{y,0}+\boldsymbol{\beta}_{y,t})^\top\boldsymbol{\phi}(\mathbf{x}),\end{aligned}$$

we can express (4) as

$$\begin{aligned}\widehat{\boldsymbol{\omega}}_y = \underset{\boldsymbol{\omega}_y}{\operatorname{argmin}}\ & \frac{1}{2N}\sum_{n=1}^{N}\boldsymbol{\omega}_y^\top \boldsymbol{\psi}(\mathbf{x}_n, t_n)\boldsymbol{\psi}(\mathbf{x}_n, t_n)^\top \boldsymbol{\omega}_y \\ & - \frac{1}{N}\sum_{n:y_n=y}\boldsymbol{\omega}_y^\top \boldsymbol{\psi}(\mathbf{x}_n, t_n) + \frac{\gamma}{2T}\|\boldsymbol{\omega}_y\|^2.\end{aligned}$$

Similarly to the original LSPC, by denoting

$$\boldsymbol{\Psi} = [\boldsymbol{\psi}(\mathbf{x}_1, t_1), \ldots, \boldsymbol{\psi}(\mathbf{x}_N, t_N)]^\top,$$

we have

$$\begin{aligned}\widehat{\boldsymbol{\omega}}_y &= \underset{\boldsymbol{\omega}_y}{\operatorname{argmin}}\ \frac{1}{2N}\boldsymbol{\omega}_y^\top \boldsymbol{\Psi}^\top \boldsymbol{\Psi}\boldsymbol{\omega}_y - \frac{1}{N}\boldsymbol{\omega}_y^\top \boldsymbol{\Psi}^\top \mathbf{z}_y + \frac{\gamma}{2T}\|\boldsymbol{\omega}_y\|^2 \\ &= \left(\boldsymbol{\Psi}^\top\boldsymbol{\Psi} + \frac{\gamma N}{T}\mathbf{I}_{N(T+1)}\right)^{-1}\boldsymbol{\Psi}^\top \mathbf{z}_y. \end{aligned} \tag{5}$$

However, calculating (5) still requires $O(N^3T^3)$ time. In order to reduce the computational cost, let us consider a *dual* expression of (5).

Lemma 1 (5) *can be equivalently expressed as*

$$\widehat{\boldsymbol{\omega}}_y = \boldsymbol{\Psi}^\top\left(\boldsymbol{\Psi}\boldsymbol{\Psi}^\top + \frac{\gamma N}{T}\mathbf{I}_N\right)^{-1}\mathbf{z}_y.$$

Proof: According to (147) in the paper [9], the following matrix inversion formula holds for some matrix $\mathbf{B}$ and invertible matrices $\mathbf{R}$ and $\mathbf{P}$:

$$(\mathbf{B}^\top\mathbf{R}^{-1}\mathbf{B} + \mathbf{P}^{-1})^{-1}\mathbf{B}^\top\mathbf{R}^{-1} = \mathbf{P}\mathbf{B}^\top(\mathbf{B}\mathbf{P}\mathbf{B}^\top + \mathbf{R})^{-1}.$$

Let us put

$$\mathbf{B} = \boldsymbol{\Psi}, \quad \mathbf{R} = \mathbf{I}_N, \quad \text{and} \quad \mathbf{P} = \frac{T}{\gamma N}\mathbf{I}_{N(T+1)}.$$

Then we have

$$\left(\boldsymbol{\Psi}^\top\boldsymbol{\Psi} + \frac{\gamma N}{T}\mathbf{I}_{N(T+1)}\right)^{-1}\boldsymbol{\Psi}^\top = \boldsymbol{\Psi}^\top\left(\boldsymbol{\Psi}\boldsymbol{\Psi}^\top + \frac{\gamma N}{T}\mathbf{I}_N\right)^{-1},$$

which concludes the proof. ■

This dual representation allows us to write the (un-normalized) estimator of $p(y|\mathbf{x}, t)$ as

$$\begin{aligned}\widehat{\boldsymbol{\omega}}_y^\top \boldsymbol{\psi}(\mathbf{x}, t) &= \mathbf{z}_y^\top\left(\boldsymbol{\Psi}\boldsymbol{\Psi}^\top + \frac{\gamma N}{T}\mathbf{I}_N\right)^{-1}\boldsymbol{\Psi}\boldsymbol{\psi}(\mathbf{x}, t) \\ &= \widehat{\boldsymbol{\mu}}_y^\top \boldsymbol{\xi}(\mathbf{x}, t),\end{aligned} \tag{6}$$

where

$$\boldsymbol{\xi}(\mathbf{x},t) = [\boldsymbol{\psi}(\mathbf{x}_1,t_1)^\top \boldsymbol{\psi}(\mathbf{x},t), \ldots, \boldsymbol{\psi}(\mathbf{x}_N,t_N)^\top \boldsymbol{\psi}(\mathbf{x},t)]^\top,$$
$$\widehat{\boldsymbol{\mu}}_y = \left(\boldsymbol{\Psi}\boldsymbol{\Psi}^\top + \frac{\gamma N}{T}\mathbf{I}_N\right)^{-1} \mathbf{z}_y.$$

Then, since

$$\boldsymbol{\psi}(\mathbf{x},t)^\top \boldsymbol{\psi}(\mathbf{x}',t') = \left(\frac{\gamma}{T\lambda} + \delta_{t,t'}\right) \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}'),$$
$$[\boldsymbol{\Psi}\boldsymbol{\Psi}^\top]_{n,n'} = \boldsymbol{\psi}(\mathbf{x}_n,t_n)^\top \boldsymbol{\psi}(\mathbf{x}_{n'},t_{n'}),$$

(6) can be computed with $O(N^3)$ computational costs.

Finally, the class-posterior probability is estimated as

$$\widehat{p}(y|\mathbf{x}) = \frac{\max(0, \widehat{\boldsymbol{\mu}}_y^\top \boldsymbol{\xi}(\mathbf{x},t))}{\max(0, \widehat{\boldsymbol{\mu}}_{-1}^\top \boldsymbol{\xi}(\mathbf{x},t)) + \max(0, \widehat{\boldsymbol{\mu}}_{+1}^\top \boldsymbol{\xi}(\mathbf{x},t))}. \tag{7}$$

The computational complexity required for this formulation of LSPC-MT is $O(N^3)$, which is the same as the single-task LSPC. This implies that the computational complexity of LSPC-MT is *independent* of the number of tasks, and thus it is computationally highly efficient when T is large.

4 Experiments

In this section, we report the results of experimental performance evaluation on two real-world image classification problems.

4.1 UMIST Face Recognition

In the first set of experiments, we used the *UMIST face recognition* dataset [5].

The UMIST dataset contains images of 20 different people, 575 images in total. Images were appropriately cropped into 112×92 (= 10304) pixels. Each pixel takes 8-bit intensity values from 0 to 255.

The database contains 4 female subjects among the 20 subjects. In our experiments, we chose a male subject from the 16 male subjects for each of the 4 female subjects, and constructed 4 binary classification tasks between male (class +1) and female (class −1). We expect that MTL captures some common structure behind different male-female classifiers. Examples of face images are depicted in Figure 1.

As inputs, the raw pixel values of the gray-scale images were directly used, i.e., $\mathbf{x} \in \mathbb{R}^{10304}$. Training images were randomly chosen from the images of the target male and female subjects, and the rest of the images were used as test samples. In each task, the numbers of male and female samples were set to be equal both for training and testing.

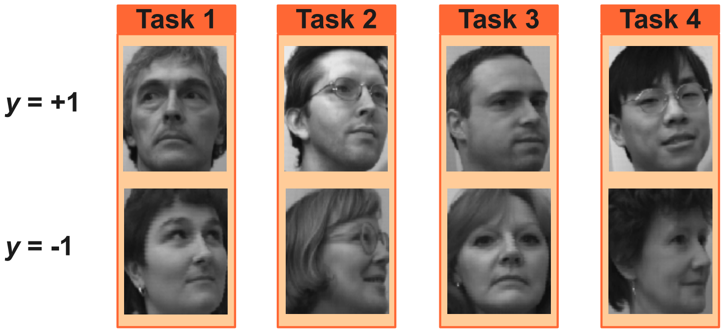


Figure 1: Examples of face images taken from the UMIST face datasets. We constructed four binary classification tasks between male (class $+1$) and female (class -1), each contains face images from a single male subject (the upper row) and a single female subject (the lower row).

We compared the correct classification rate (i.e., classification accuracy) and computation time of the proposed LSPC-MT method with those of the KLR multi-task method (KLR-MT) [7] as a function of the number of training samples. As baselines, we also included in our comparison the single-task counterparts: LSPC-STI, KLR-STI, LSPC-STC, and KLR-STC. 'STI' denotes 'single task, independent', meaning that each task is treated independently and a classifier is trained for each task only using samples of that task (this corresponds to setting λ in (4) large enough). On the other hand, 'STC' denotes 'single task, combined', meaning that all tasks are combined together and a single common classifier is trained using samples from all tasks (this corresponds to setting γ in (4) large enough).

In all the six methods, LSPC-MT, KLR-MT, LSPC-STI, KLR-STI, LSPC-STC, and KLR-STC, 5-fold cross-validation (CV) with respect to the classification accuracy was used to choose the regularization parameter

$$\lambda \in \{0.01, 0.03, 0.1, 0.3, 1.0, 3.0\},$$

and the Gaussian kernel bandwidth

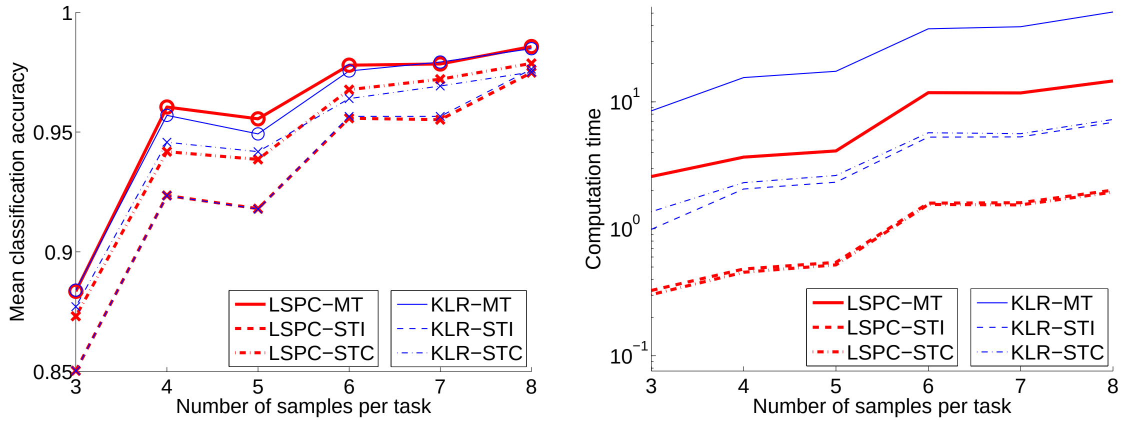
$$\sigma \in \{\tfrac{1}{2}m, \tfrac{2}{3}m, \tfrac{5}{6}m, m, \tfrac{4}{3}m, \tfrac{5}{3}m\},$$

where m is the median distance between all pairs of training samples. Additionally, for LSPC-MT and KLR-MT, we selected the multi-task parameter

$$\gamma \in \{0, 0.01, 0.03, 0.1, 0.3, 1.0, 3.0\}$$

by CV.

We implemented all the methods using MATLAB®. KLR solutions were numerically computed by the *limited-memory Broyden-Fletcher-Goldfarb-Shanno* (L-BFGS) method using the '*minFunc*' package [10]. We repeated the experiments 200 times with different random seeds, and evaluated the mean classification accuracy and computation time.



(a) Accuracy

(b) Computation time

Figure 2: Experimental results for the UMIST dataset. (a) Mean classification accuracy over 200 runs. '∘' indicates the best performing method or a tie with the best performance (by t-test with 1% level of significance). '×' indicates that the method is significantly weaker than the best one. (b) The computation time (in seconds).

The classification accuracy are summarized in Figure 2(a), showing that both MTL methods significantly outperform the single-task learning counterparts. On the other hand, the accuracy of LSPC-MT and KLR-MT is comparable to each other. Figure 2(b) summarizes the computation time, showing that LSPC and LSPC-MT were 2–3 times faster than KLR and KLR-MT, respectively. The minimum and maximum values for the classification accuracy and computation time are reported in Tables 1 and 2, respectively.

4.2 Landmine Image Classification

In the second set of experiments, we used the *landmine image classification* dataset [12].

The Landmine dataset consists of 29 binary classification tasks about various landmine fields. Each input sample $\mathbf{x}$ is a 9-dimensional feature vector corresponding to a region of landmine fields, and the binary class y corresponds to whether there is a landmine or not in that region. The feature vectors are extracted from radar images, concatenating four moment-based features, three correlation-based features, one energy ratio feature, and one spatial variance feature (see the paper [12] for details). The goal is to estimate whether a test landmine field contains landmines or not based on the region features. In the 29 landmine classification tasks, the first 15 tasks are highly foliated and the last 14 tasks are regions that are bare earth or desert. Here we use the first 17 tasks for our experiments: all 15 highly foliated regions and the first 2 tasks from bare earth regions. In the latter 2 datasets, we completely reversed the class labels and evaluated the robustness of MTL methods against noisy tasks.

We again compared the performance of LSPC-MT, KLR-MT, LSPC-STI, KLR-STI, LSPC-STC, and KLR-STC. The experimental setup was the same as the previous UMIST

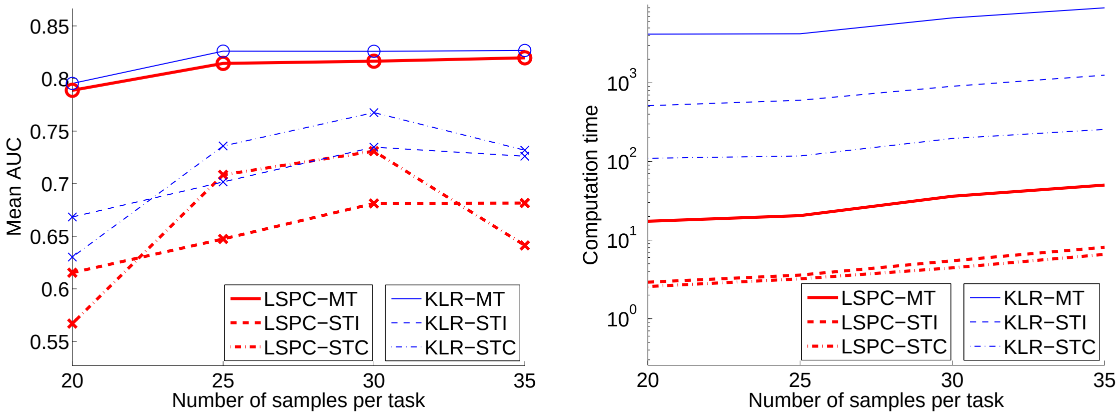
Table 1: Experimental results for the UMIST dataset. Minimum and maximum classification accuracy over 200 runs. The first column shows the number of samples per task.

	LSPC-MT	KLR-MT	LSPC-STI	KLR-STI	LSPC-STC	KLR-STC
3	0.763	0.812	0.830	0.867	0.754	0.796
	1.000	1.000	1.000	1.000	1.000	1.000
4	0.915	0.921	0.896	0.901	0.837	0.908
	1.000	1.000	1.000	1.000	1.000	1.000
5	0.893	0.925	0.903	0.915	0.909	0.930
	1.000	1.000	1.000	1.000	1.000	1.000
6	0.918	0.907	0.909	0.911	0.912	0.912
	1.000	1.000	1.000	1.000	1.000	1.000
7	0.899	0.915	0.897	0.905	0.897	0.903
	1.000	1.000	1.000	1.000	1.000	1.000
8	0.967	0.962	0.938	0.942	0.979	0.979
	1.000	1.000	1.000	1.000	1.000	1.000

Table 2: Experimental results for the UMIST dataset. Minimum and maximum computation time (in seconds) over 200 runs. The first column shows the number of samples per task.

	LSPC-MT	KLR-MT	LSPC-STI	KLR-STI	LSPC-STC	KLR-STC
3	2.52e+00	8.04e+00	3.22e-01	8.88e-01	2.97e-01	1.26e+00
	2.67e+00	9.11e+00	3.69e-01	1.20e+00	3.30e-01	1.53e+00
4	3.09e+00	1.26e+01	4.11e-01	1.68e+00	3.88e-01	1.86e+00
	1.45e+01	5.29e+01	1.87e+00	7.86e+00	1.93e+00	8.64e+00
5	3.09e+00	1.25e+01	4.09e-01	1.63e+00	3.88e-01	1.91e+00
	1.18e+01	5.11e+01	1.67e+00	6.88e+00	1.60e+00	7.81e+00
6	7.53e+00	2.65e+01	1.02e+00	3.61e+00	1.01e+00	3.87e+00
	3.76e+01	1.14e+02	5.42e+00	1.68e+01	5.17e+00	1.92e+01
7	7.58e+00	2.69e+01	1.02e+00	3.58e+00	9.90e-01	3.86e+00
	3.91e+01	1.51e+02	6.57e+00	2.20e+01	6.65e+00	2.39e+01
8	8.47e+00	3.41e+01	1.17e+00	4.63e+00	1.13e+00	4.82e+00
	4.84e+01	1.48e+02	7.89e+00	2.63e+01	6.02e+00	2.20e+01

experiments, except that instead of the correct classification rate, we adopted the *Area Under the receiver operating characteristic Curve* (AUC) [1] as the performance measure. The reason for this choice is as follows. In the landmine datasets, only about 6% of samples are from the landmine class and the rest are from the non-landmine class. For such *imbalanced classification* problems [3], merely using the classification accuracy is not appropriate since just predicting all test samples to be non-landmine achieves 94% accuracy, which is obviously non-sense. In imbalanced classification scenarios, it is important to take into account the *coverage* of true landmine fields, in addition to the classification accuracy. Since there is a trade-off between the coverage and classification accuracy, we



(a) AUC score

(b) Computation time

Figure 3: Experimental results for the Landmine dataset. (a) Mean AUC score over 10 runs. '∘' indicates the best performing method or a tie with the best performance (by t-test with 1% level of significance). '×' indicates that the method is significantly weaker than the best one. (b) The computation time (in seconds).

decided to adopt the AUC as our error metric here, which reflects all possible trade-offs[3]. In our experiments, we first calculated the AUC score on the test samples for each task separately, and then took the mean of the AUC values over all tasks.

To be consistent with the above performance measure, we performed CV also with respect to the AUC score. Since the landmine datasets are highly imbalanced, the validation data in the CV procedure can contain no landmine sample, which causes inappropriate choice of tuning parameters. To avoid this problem, we combined all estimated class-posterior probabilities from different tasks and calculated a single AUC score in the CV procedure, instead of merely taking the mean of the AUC scores over all tasks.

The number of landmine samples contained in each task is 445–690. We randomly selected a subset of the samples for training and used the rest for evaluating the AUC score. We repeated the experiments 10 times with different random seeds, and evaluated the mean AUC score and computation time.

Figure 3(a) summarizes the AUC scores, showing that the AUC scores of LSPC-MT and KLR-MT are comparable to each other, and the MTL methods are significantly better than the single-task counterparts. Figure 3(b) summarizes the computation time, showing that LSPC-MT is faster than KLR-MT in two orders of magnitude. The minimum and maximum values for AUC and computation time are shown in Tables 3 and 4.

[3] Note that we did *not* round up classifiers' negative outputs to zero (see Eq.(7)) since negative values can also be utilized for computing the AUC scores.

Table 3: Experimental results for the Landmine dataset. Minimum and maximum AUC values over 10 runs. The first column shows the number of samples per task.

	LSPC-MT	KLR-MT	LSPC-STI	KLR-STI	LSPC-STC	KLR-STC
20	0.714	0.642	0.543	0.642	0.442	0.532
	0.830	0.832	0.645	0.706	0.759	0.773
25	0.762	0.800	0.587	0.658	0.602	0.598
	0.839	0.842	0.694	0.750	0.788	0.800
30	0.790	0.811	0.646	0.687	0.661	0.713
	0.837	0.850	0.756	0.789	0.783	0.805
35	0.795	0.806	0.627	0.684	0.428	0.663
	0.838	0.849	0.734	0.771	0.742	0.797

Table 4: Experimental results for the Landmine dataset. Minimum and maximum computation time (in seconds) over 10 runs. The first column shows the number of samples per task.

	LSPC-MT	KLR-MT	LSPC-STI	KLR-STI	LSPC-STC	KLR-STC
20	1.20e+01	2.84e+03	1.86e+00	3.72e+02	1.59e+00	7.99e+01
	3.67e+01	8.22e+03	6.41e+00	1.11e+03	6.08e+00	2.60e+02
25	1.78e+01	3.90e+03	3.15e+00	5.28e+02	2.49e+00	9.36e+01
	2.69e+01	4.91e+03	4.28e+00	6.99e+02	4.18e+00	1.49e+02
30	2.36e+01	4.77e+03	3.50e+00	6.57e+02	3.29e+00	1.37e+02
	7.86e+01	1.26e+04	9.29e+00	1.80e+03	7.67e+00	3.61e+02
35	3.72e+01	5.96e+03	5.03e+00	8.48e+02	4.45e+00	1.76e+02
	9.89e+01	1.82e+04	1.44e+01	2.49e+03	1.09e+01	5.91e+02

5 Conclusions

We extended a recently-proposed probabilistic classification method called the *least-squares probabilistic classifier* (LSPC) to multi-task setting. Although a naive multi-task extension of LSPC given in Section 3.1 may still be computationally more advantageous than the multi-task method based on kernel logistic regression (KLR) [7] with dense kernel matrices, it significantly increased the computational complexity compared with the single-task LSPC method. In order to improve the computational efficiency, we introduced a dual formulation for the LSPC multi-task method in Section 3.2, allowing us to keep the computational complexity the same as the single task case.

Through experiments, we confirmed that the proposed LSPC-based multi-task method has comparable performance to the KLR-based multi-task method in terms of the classification accuracy, while the proposed method is computationally much more efficient.

Acknowledgments

JS was supported by MEXT, and MS was supported by AOARD, SCAT, and the JST PRESTO program.

References

[1] Bradley, A. P.: 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).

[2] Caruana, R., Pratt, L. and Thrun, S.: Multitask Learning, *Machine Learning*, Vol. 28, p. 41 (1997).

[3] Chawla, N. V., Japkowicz, N. and Kotcz, A.: Editorial: Special Issue on Learning from Imbalanced Data Sets, *ACM SIGKDD Explorations Newsletter*, Vol. 6, No. 1, pp. 1–6 (2004).

[4] Evgeniou, T. and Pontil, M.: Regularized multi–task learning, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, ACM, pp. 109–117 (2004).

[5] Graham, D. B. and Allinson, N. M.: Characterizing Virtual Eigensignatures for General Purpose Face Recognition, *Computer and Systems Sciences*, NATO ASI Series F, Vol. 163, Springer, Berlin, Germany, pp. 446–456 (1998).

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

[7] Lapedriza, A., Masip, D. and Vitrià, J.: A Hierarchical Approach for Multi-task Logistic Regression, *Proceedings of the 3rd Iberian Conference on Pattern Recognition and Image Analysis, Part II*, Berlin, Germany, Springer-Verlag, pp. 258–265 (2007).

[8] Minka, T. P.: A Comparison of Numerical Optimizers for Logistic Regression, Technical report, Microsoft Research (2007). `http://research.microsoft.com/~minka/papers/logreg/minka-logreg.pdf`.

[9] Petersen, K. B. and Pedersen, M. S.: The Matrix Cookbook, Technical report, Technical University of Denmark (2008). `http://matrixcookbook.com/`.

[10] Schmidt, M.: minFunc (2005). `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`.

[11] Sugiyama, M.: 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).

[12] Xue, Y., Liao, X., Carin, L. and Krishnapuram, B.: Multi-Task Learning for Classification with Dirichlet Process Priors, *Journal of Machine Learning Research*, Vol. 8, pp. 35–63 (2007).

[13] Yamada, M., Sugiyama, M., Wichern, G. and Simm, J.: Improving the Accuracy of Least-Squares Probabilistic Classifiers, Technical Report IBISML2010-32, IEICE Technical Report (2010).