Revisit of Logistic Regression: Efficient Optimization and Kernel Extensions

Takumi Kobayashi
National Institute of Advanced Industrial Science and Technology
Umezono 1-1-1, Tsukuba, 305-8568, Japan
Email: takumi.kobayashi@aist.go.jp

Kenji Watanabe
Wakayama University
Sakaedani 930, Wakayama, 640-8510, Japan
Email: k-watanabe@vrl.sys.wakayama-u.ac.jp

Nobuyuki Otsu
National Institute of Advanced Industrial Science and Technology
Umezono 1-1-1, Tsukuba, 305-8568, Japan
Email: otsu.n@aist.go.jp

***Abstract*—Logistic regression (LR) is widely applied as a powerful classification method in various fields, and a variety of optimization methods have been developed. To cope with large-scale problems, an efficient optimization method for LR is required in terms of computational cost and memory usage. In this paper, we propose an efficient optimization method using non-linear conjugate gradient (CG) descent. In each CG iteration, the proposed method employs the optimized step size without exhaustive line search, which significantly reduces the number of iterations, making the whole optimization process fast. In addition, on the basis of such CG-based optimization scheme, a novel optimization method for kernel logistic regression (KLR) is proposed. Unlike the ordinary KLR methods, the proposed method optimizes the kernel-based classifier, which is naturally formulated as the linear combination of sample kernel functions, directly in the reproducing kernel Hilbert space (RKHS), not the linear coefficients. Subsequently, we also propose the multiple-kernel logistic regression (MKLR) along with the optimization of KLR. The MKLR effectively combines the multiple types of kernels with optimizing the weights for the kernels in the framework of the logistic regression. These proposed methods are all based on CG-based optimization and matrix-matrix computation which is easily parallelized such as by using multi-thread programming. In the experimental results on multi-class classifications using various datasets, the proposed methods exhibit favorable performances in terms of classification accuracies and computation times.**

I. Introduction

A classification problem is an intensive research topic in the pattern recognition field. Especially, classifying the feature vectors extracted from input data plays an important role; e.g., for image (object) recognition [1] and detection [2], motion recognition [3], natural language processing [4]. Nowadays, we can collect a large amount of data such as via internet, and thus large-scale problems have being frequently addressed in those fields to improve classification performances.

In the last decade, the classification problems have been often addressed in the large margin framework [5] as represented by support vector machine (SVM) [6]. While those methods are basically formulated for linear classification, they are also extended to kernel-based methods by employing kernel functions and produce promising performances. However, they are mainly intended for binary (two) class problems and it is generally difficult to extend the method toward the multi-class problems without heuristics such as a one-versus-rest approach. Several methods, however, are proposed to cope with the multi-class problems, e.g., in [7]. Another drawback is that the optimization in those methods has difficulty in parallelization. The SVM-based methods are formulated in quadratic programming (QP). Some successful optimization methods to solve the QP, such as sequential minimal optimization (SMO) [8], are based on a sequential optimization approach which can not be easily parallelized. Parallel computing currently developed such as by using GPGPU would be a key tool to effectively treat large-scale data.

On the other hand, logistic regression has also been successfully applied in various classification tasks. Apart from the margin-based criterion for the classifiers, the logistic regression is formulated in the probabilistic framework. Therefore, it is advantageous in that 1) the classifier outputs (class) posterior probabilities and 2) the method is naturally generalized to the multi-class classifiers by employing a multi-nominal logistic function which takes into account the correlations among classes. While the optimization problem, i.e., objective cost function, for logistic regression is well-defined, there is still room to argue about its optimization method in terms of computational cost and memory usage, especially to cope with large-scale problems. A popular method, *iterative reweighted least squares*, is based on the Newton-Raphson method [9] requiring significant computation cost due to the Hessian.

In this paper, we propose an efficient optimization method for the logistic regression. The proposed method is based on non-linear conjugate gradient (CG) descent [10] which is directly applied to minimize the objective cost. The non-linear CG is widely applied to unconstrained optimization problems, though requiring an exhaustive line search to determine a step size in each iteration. In the proposed method, we employ the optimum step size without the line search, which makes the whole optimization process more efficient by significantly reducing the number of iterations. In addition, we propose a novel optimization method for kernel logistic regression (KLR) on the basis of the CG-based optimization scheme. Unlike the ordinary KLR methods, the proposed method optimizes the kernel-based classifier, which is naturally formulated as the linear combination of sample kernel functions as in SVM, di-

TABLE I. NOTATIONS

N	Number of samples
C	Number of classes
$\boldsymbol{x}_i$	Feature vector of the i-th sample ($\in \mathbb{R}^L$)
$\boldsymbol{y}_i$	Class indicator vector of the i-th sample ($\in \{0,1\}^C$) in which only the assigned class component is 1 and the others 0
$\boldsymbol{X}$	Matrix containing feature vectors $\boldsymbol{x}_i$ in its columns ($\in \mathbb{R}^{L\times N}$)
$\boldsymbol{Y}$	Matrix containing class vectors $\boldsymbol{y}_i$ in its columns ($\in \{0,1\}^{C\times N}$)
$\mathcal{H}$	Reproducing kernel Hilbert space (RKHS)
$\mathsf{k}(\cdot,\cdot)$	Kernel function in RKHS $\mathcal{H}$
$\boldsymbol{K}$	Kernel Gram matrix of $[\mathsf{k}(\boldsymbol{x}_i,\boldsymbol{x}_j)]_{i=1,..,N}^{j=1,..,N}$ ($\in \mathbb{R}^{N\times N}$)
$\lceil\cdot\rceil_{1:C-1}$	Operator extracting the $1\sim C-1$-th rows of a matrix/vector
$[\cdot_{ij}]_{i=1,..,H}^{j=1,..,W}$	Operator constructing the matrix of the size $\mathbb{R}^{H\times W}$ where the lower/upper index is for the row/column.
$\cdot^\top$	Transpose of a matrix/vector
$\langle\cdot,\cdot\rangle$	Frobenius inner product of matrices, i.e., $\langle\boldsymbol{A},\boldsymbol{B}\rangle = \mathrm{trace}(\boldsymbol{A}^\top\boldsymbol{B})$

rectly in the reproducing kernel Hilbert space (RKHS), not the linear coefficients of samples. Subsequently, multiple-kernel logistic regression (MKLR) is also proposed as multiple-kernel learning (MKL). The MKL combines the multiple types of kernels with optimizing the weights for the kernels, and it has been addressed mainly in the large margin framework [11]. The proposed MKLR is formulated as a convex form in the framework of logistic regression. In the proposed formulation, by resorting to the optimization method in the KLR, we optimize the kernel-based classifier in sum of multiple RKHSs and consequently the linear weights for the multiple kernels. In summary, the contributions of this paper are as follows;

- Non-linear CG in combination with the optimum step size for optimizing logistic regression.
- Novel method for kernel logistic regression to directly optimize the classifier in RKHS.
- Novel method of multiple-kernel logistic regression.

Note that all the proposed methods are based on the CG-based optimization and the computation cost is dominated by matrix-matrix computation which is easily parallelized.

The rest of this paper is organized as follows: the next section briefly reviews the related works of optimization for logistic regression. In Section III, we describe the details of the proposed method using non-linear CG. And then in Section IV and Section V we propose the novel optimization methods for kernel logistic regression and for multiple-kernel logistic regression. In Section VI, we mention the parallel computing in the proposed methods. The experimental results on various types of multi-class classification are shown in Section VII. Finally, Section VIII contains our concluding remarks.

This paper contains substantial improvements over the preliminary version [12] in that we develop the kernel-based methods including MKL and give new experimental results.

A. Notations

We use the notations shown in Table I. Basically, the big bold letter, e.g., $\boldsymbol{X}$, indicates a matrix, its small bold letter with the index, e.g., $\boldsymbol{x}_i$, denotes the i-th column vector, and the small letter with two indexes, e.g., x_{ic}, indicates the c-th component of the i-th column vector $\boldsymbol{x}_i$, corresponding to the c-th row and i-th column element of $\boldsymbol{X}$.

To cope with multi-class problems, we apply the following multi-nominal logistic function for the input $\boldsymbol{z}\in\mathbb{R}^{C-1}$:

$$\sigma_c(\boldsymbol{z}) = \begin{cases} \frac{\exp(z_c)}{1+\sum_{k=1}^{C-1}\exp(z_k)} (c<C) \\ \frac{1}{1+\sum_{k=1}^{C-1}\exp(z_k)} (c=C) \end{cases}, \quad \boldsymbol{\sigma}(\boldsymbol{z}) = \begin{bmatrix} \sigma_1(\boldsymbol{z}) \\ \vdots \\ \sigma_C(\boldsymbol{z}) \end{bmatrix} \in \mathbb{R}^C,$$

where $\sigma_c(\boldsymbol{z})$ outputs a posterior probability on the c-th class and $\boldsymbol{\sigma}(\boldsymbol{z})$ produces the probabilities over the whole C classes.

II. RELATED WORKS

The (multi-class) logistic regression is also mentioned in the context of the maximum entropy model [13] and the conditional random field [14]. We first describe the formulation of linear logistic regression. The linear logistic regression estimates the class posterior probabilities $\hat{\boldsymbol{y}}$ from the input feature vector $\boldsymbol{x}\in\mathbb{R}^L$ by using the above logistic function:

$$\hat{\boldsymbol{y}} = \boldsymbol{\sigma}(\boldsymbol{W}^\top\boldsymbol{x}) \in \mathbb{R}^C,$$

where $\boldsymbol{W}\in\mathbb{R}^{L\times C-1}$ is the classifier weight matrix. To optimize $\boldsymbol{W}$, the following objective cost is minimized:

$$J(\boldsymbol{W}) = -\sum_i^N\sum_c^C y_{ic}\log\sigma_c(\boldsymbol{W}^\top\boldsymbol{x}_i) \to \min_{\boldsymbol{W}}. \qquad (1)$$

There exists various methods for optimizing the logistic regression, as described below. Comparative studies on those optimization methods are shown in [13], [15].

A. Newton-Raphson method

For simplicity, we unfold the weight matrix $\boldsymbol{W}$ into the long vector $\boldsymbol{w} = [\boldsymbol{w}_1^\top,\cdots,\boldsymbol{w}_{C-1}^\top]^\top \in \mathbb{R}^{L(C-1)}$. The derivatives of the cost function (1) is given by

$$\nabla_{\boldsymbol{w}_c}J = \sum_i^N \boldsymbol{x}_i(\hat{y}_{ic}-y_{ic}) \in \mathbb{R}^L, \quad \nabla_{\boldsymbol{w}}J = \begin{bmatrix} \nabla_{\boldsymbol{w}_1}J \\ \vdots \\ \nabla_{\boldsymbol{w}_{C-1}}J \end{bmatrix} \in \mathbb{R}^{L(C-1)},$$

where $\hat{y}_{ic} = \sigma_c(\boldsymbol{W}^\top\boldsymbol{x}_i)$, and the Hessian of J is obtained as

$$\boldsymbol{H}_{c,k} = \nabla_{\boldsymbol{w}_c}\nabla_{\boldsymbol{w}_k}^\top J = \sum_i^N y_{ic}(\delta_{ck}-y_{ik})\boldsymbol{x}_i\boldsymbol{x}_i^\top \in \mathbb{R}^{L\times L},$$

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{H}_{1,1} & \cdots & \boldsymbol{H}_{1,C-1} \\ \vdots & \cdots & \vdots \\ \boldsymbol{H}_{C-1,1} & \cdots & \boldsymbol{H}_{C-1,C-1} \end{pmatrix} = [\boldsymbol{H}_{ck}]_{c=1,..,C-1}^{k=1,..,C-1} \in \mathbb{R}^{L(C-1)\times L(C-1)},$$

where δ_{ck} is the Kronecker delta. This Hessian matrix is positive definite, and thus the optimization problem in (1) is convex. For the optimization, the Newton-Raphson update is described by

$$\boldsymbol{w}^{new} = \boldsymbol{w}^{old} - \boldsymbol{H}^{-1}\nabla_{\boldsymbol{w}}J = \boldsymbol{H}^{-1}(\boldsymbol{H}\boldsymbol{w}^{old}-\nabla_{\boldsymbol{w}}J) = \boldsymbol{H}^{-1}\boldsymbol{z}, \qquad (2)$$

where $\boldsymbol{z} \triangleq \boldsymbol{H}\boldsymbol{w}^{old} - \nabla_{\boldsymbol{w}}J$. This update procedure, which can be regarded as reweighted least squares, is repeated until convergence. Such a method based on Newton-Raphson, called *iterative reweighted least squares* (IRLS) [16], is one of the commonly used optimization methods.

This updating of $\boldsymbol{w}$ in (2) requires the inverse matrix computation for the Hessian. In the case of large-dimensional feature vectors and large number of classes, it requires much computational cost to compute the inverse of the large Hessian matrix. To cope with such difficulty in large-scale data, various optimization methods have been proposed by making the update (2) efficient. Komarek and Moore [17] regarded (2) as the solution of the following linear equations, $\boldsymbol{H}\boldsymbol{w}^{new} = \boldsymbol{z}$, and they apply Cholesky decomposition to efficiently solve it. On the other hand, Komarek and Moore [18] applied linear conjugate-gradient (CG) descent to solve these linear equations [19]. The CG method is applicable even to large-dimensional Hessian $\boldsymbol{H}$. Recently, Lin et al. [20] employed

trust-region method [9] to increase the efficiency of the Newton-Raphson update using the linear CG. Note that the method in [20] deals with multi-class problems in a slightly different way from ordinary multi-class LR by considering one-against-rest approach.

B. Quasi Newton method

As described above, it is inefficient to explicitly compute the Hessian for multi-class large dimensional data. To remedy it, Malouf [13] and Daumé III [21] presented the optimization method using limited memory BFGS [22]. In the limited memory BFGS, the Hessian is approximately estimated in a computationally efficient manner and the weight $\boldsymbol{W}$ is updated by using the approximated Hessian $\tilde{\boldsymbol{H}}$.

C. Other optimization methods

Besides those Newton-based methods, the other optimization methods are also applied. For example, Pietra et al. [23] proposed the method of *improved iterative scaling*, and Minka [15] and Daumé III [21] presented the method using non-linear CG [10] with an exhaustive line search.

In this study, we focus on the non-linear CG based optimization due to its favorable performance reported in [15] and its simple formulation which facilitates the extensions to the kernel-based methods.

D. Kernel logistic regression

Kernel logistic regression [24], [25], [26] is an extension of the linear logistic regression by using kernel function. By considering the classifier function $\mathsf{f}_c(\cdot)$, $c \in \{1, \cdots, C-1\}$, the class posterior probabilities are estimated from $\boldsymbol{x}$ by

$$\hat{\boldsymbol{y}} = \boldsymbol{\sigma}\big([\mathsf{f}_c(\boldsymbol{x})]_{c=1,..,C-1}\big) \in \mathbb{R}^C.$$

As in the other kernel-based methods [27], $\mathsf{f}_c(\cdot)$ is represented by the linear combinations of sample kernel functions $\mathsf{k}(\boldsymbol{x}_i, \cdot)$ in the reproducing kernel Hilbert space (RKHS) $\mathcal{H}$:

$$\mathsf{f}_c(\cdot) = \sum_i^N w_{ci}\mathsf{k}(\boldsymbol{x}_i, \cdot) \Rightarrow \boldsymbol{\sigma}\big([\mathsf{f}_c(\boldsymbol{x})]_{c=1,..,C-1}\big) = \boldsymbol{\sigma}\left(\boldsymbol{W}^\top \boldsymbol{k}(\boldsymbol{x})\right),$$

where $\boldsymbol{W} = [w_{ci}]_{i=1,..,N}^{c=1,..,C-1} \in \mathbb{R}^{N\times C-1}$ indicates the (linear) coefficients of the samples for the classifier and $\boldsymbol{k}(\boldsymbol{x}) = [\mathsf{k}(\boldsymbol{x}_i, \boldsymbol{x})]_{i=1,..,N} \in \mathbb{R}^N$ is a kernel feature vector. Ordinary kernel logistic regression is formulated in the following optimization problem;

$$J(\boldsymbol{W}) = -\sum_i^N \sum_c^C y_{ic} \log\left\{\sigma_c\left(\boldsymbol{W}^\top \boldsymbol{k}(\boldsymbol{x}_i)\right)\right\} \rightarrow \min_{\boldsymbol{W}}.$$

This corresponds to the linear logistic regression in (1) except that the feature vectors are replaced by the kernel feature vectors $\boldsymbol{x}_i \mapsto \boldsymbol{k}(\boldsymbol{x}_i)$ and the classifier weights $\boldsymbol{W} \in \mathbb{R}^{N\times C-1}$ are formulated as the coefficients for the samples.

III. Efficient Optimization for Linear Logistic Regression

In this section, we propose the optimization method for linear logistic regression which efficiently minimizes the cost even for the large-scale data. The proposed method is based on non-linear CG method [10] directly applicable to the optimization as in [15], [21]. Our contribution is that the step size required in CG updates is optimized without an exhaustive line search employed in an ordinary non-linear CG method, in order to significantly reduce the number of iterations and speed-up the optimization. The non-linear CG can also save memory usage without relying on the Hessian matrix. The proposed method described in this section serves as a basis for kernel-based extensions in Section IV and V.

A. Non-linear CG optimization for linear logistic regression

We minimize the following objective cost with the regularization term, L_2-norm of the classifier weights $\boldsymbol{W} \in \mathbb{R}^{L\times C-1}$:

$$J(\boldsymbol{W}) = \frac{\lambda}{2}\|\boldsymbol{W}\|_F^2 - \sum_i^N \sum_c^C y_{ic} \log\left\{\sigma_c(\boldsymbol{W}^\top \boldsymbol{x}_i)\right\} \rightarrow \min_{\boldsymbol{W}}, \tag{3}$$

where $\|\boldsymbol{W}\|_F^2 = \langle \boldsymbol{W}, \boldsymbol{W}\rangle$ and λ is a regularization parameter. The gradient of J with respect to $\boldsymbol{W}$ is given by

$$\nabla_{\boldsymbol{W}} J = \lambda \boldsymbol{W} + \boldsymbol{X}\lceil \hat{\boldsymbol{Y}} - \boldsymbol{Y}\rceil_{1:C-1}^\top \in \mathbb{R}^{L\times C-1},$$

where $\hat{\boldsymbol{Y}} = [\hat{\boldsymbol{y}}_i = \boldsymbol{\sigma}(\boldsymbol{W}^\top \boldsymbol{x}_i)]^{i=1,..,N} \in \mathbb{R}^{C\times N}$.

The non-linear CG method utilizes the gradient $\nabla_{\boldsymbol{W}} J$ to construct the conjugate gradient, and the cost (3) is minimized iteratively. At the l-th iteration, letting $\boldsymbol{G}^{(l)} \triangleq \nabla_{\boldsymbol{W}} J(\boldsymbol{W}^{(l)})$, the conjugate gradient $\boldsymbol{D}^{(l)} \in \mathbb{R}^{L\times C-1}$ is provided by

$$\boldsymbol{D}^{(l)} = -\boldsymbol{G}^{(l)} + \beta \boldsymbol{D}^{(l-1)}, \quad \boldsymbol{D}^{(0)} = -\boldsymbol{G}^{(0)},$$

where β is a CG update parameter. There are various choices for β [10]; we employ the update parameter in [28]:

$$\beta = \max\left\{\frac{\langle \boldsymbol{G}^{(l)}, \boldsymbol{G}^{(l)} - \boldsymbol{G}^{(l-1)}\rangle}{\langle \boldsymbol{D}^{(l-1)}, \boldsymbol{G}^{(l)} - \boldsymbol{G}^{(l-1)}\rangle}, 0\right\} - \theta \frac{\langle \boldsymbol{G}^{(l)}, \boldsymbol{W}^{(l)} - \boldsymbol{W}^{(l-1)}\rangle}{\langle \boldsymbol{D}^{(l-1)}, \boldsymbol{G}^{(l)} - \boldsymbol{G}^{(l-1)}\rangle}, \tag{4}$$

where we set $\theta = 0.5$ in this study. Then, the classifier weight $\boldsymbol{W}$ is updated by using the conjugate gradient:

$$\boldsymbol{W}^{(l+1)} = \boldsymbol{W}^{(l)} + \alpha \boldsymbol{D}^{(l)}, \tag{5}$$

where α is a step size, the determination of which is described in the next section. These non-linear CG iterations are repeated until convergence.

B. Optimum step size α

The step size α in (5) is critical for efficiency in the optimization, and it is usually determined by an exhaustive line search satisfying Wolfe condition in an ordinary non-linear CG [10]. We optimize the step size α so as to minimize the cost function:

$$\alpha = \arg\min_{\alpha} J(\boldsymbol{W} + \alpha \boldsymbol{D}), \tag{6}$$

$$J(\boldsymbol{W} + \alpha \boldsymbol{D}) = \frac{\lambda}{2}\|\boldsymbol{W} + \alpha \boldsymbol{D}\|_F^2 - \sum_i^N \sum_c^C y_{ic} \log\left\{\sigma_c\left(\boldsymbol{W}^\top \boldsymbol{x}_i + \alpha \boldsymbol{D}^\top \boldsymbol{x}_i\right)\right\}.$$

Here, we introduce auxiliary variables, $\boldsymbol{P} = \boldsymbol{W}^\top \boldsymbol{X} \in \mathbb{R}^{C-1\times N}$, $\boldsymbol{Q} = \boldsymbol{D}^\top \boldsymbol{X} \in \mathbb{R}^{C-1\times N}$ and $\hat{\boldsymbol{Y}} = \left[\hat{\boldsymbol{y}}_i = \boldsymbol{\sigma}((\boldsymbol{W} + \alpha \boldsymbol{D})^\top \boldsymbol{x}_i) = \boldsymbol{\sigma}(\boldsymbol{p}_i + \alpha \boldsymbol{q}_i)\right]^{i=1,..,N}$, and thereby the gradient and Hessian of J with respect to α are written by

$$\frac{dJ}{d\alpha} = \lambda\left\{\alpha\langle \boldsymbol{D}, \boldsymbol{D}\rangle + \langle \boldsymbol{W}, \boldsymbol{D}\rangle\right\} + \sum_i^N \boldsymbol{q}_i^\top \lceil \hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\rceil_{1:C-1} \triangleq g(\alpha),$$

$$\frac{d^2J}{d\alpha^2} = \lambda\langle \boldsymbol{D}, \boldsymbol{D}\rangle + \sum_i^N \sum_c^{C-1} \hat{y}_{ic} q_{ic}\left(q_{ic} - \sum_k^{C-1} \hat{y}_{ik} q_{ik}\right) \triangleq h(\alpha).$$

Algorithm 1 : Logistic Regression by non-linear CG

Input: $\boldsymbol{X} = [\boldsymbol{x}_i]^{i=1,..,N} \in \mathbb{R}^{L\times N},\ \boldsymbol{Y} = [\boldsymbol{y}_i]^{i=1,..,N} \in \{0,1\}^{C\times N}$
1: **Initialize** $\boldsymbol{W}^{(0)} = \boldsymbol{0} \in \mathbb{R}^{L\times C-1},\ \hat{\boldsymbol{Y}} = \left[\frac{1}{C}\right] \in \mathbb{R}^{C\times N}$
$\boldsymbol{G}^{(0)} = \boldsymbol{X}\lceil\hat{\boldsymbol{Y}} - \boldsymbol{Y}\rceil^{\top}_{1:C-1} \in \mathbb{R}^{L\times C-1},$
$\boldsymbol{D}^{(0)} = -\boldsymbol{G}^{(0)} \in \mathbb{R}^{L\times C-1}$
$\boldsymbol{P} = {\boldsymbol{W}^{(0)}}^{\top}\boldsymbol{X} = \boldsymbol{0} \in \mathbb{R}^{C-1\times N},\ l = 1$
2: **repeat**
3: $\boldsymbol{Q} = {\boldsymbol{D}^{(l-1)}}^{\top}\boldsymbol{X} \in \mathbb{R}^{C-1\times N}$
4: $\alpha = \arg\min_{\alpha} J(\boldsymbol{W}^{(l-1)} + \alpha\boldsymbol{D}^{(l-1)})$: see Section III-B
5: $\boldsymbol{W}^{(l)} = \boldsymbol{W}^{(l-1)} + \alpha\boldsymbol{D}^{(l-1)},\ \boldsymbol{P} \leftarrow \boldsymbol{P} + \alpha\boldsymbol{Q}$
6: $\hat{\boldsymbol{Y}} = [\hat{\boldsymbol{y}}_i = \boldsymbol{\sigma}(\boldsymbol{p}_i)]^{i=1,..,N}$
$J^{(l)} = J(\boldsymbol{W}^{(l)}) = \frac{\lambda}{2}\|\boldsymbol{W}^{(l)}\|_F^2 - \sum_i^N\sum_c^C y_{ic}\log\hat{y}_{ic}$
7: $\boldsymbol{G}^{(l)} = \boldsymbol{X}\lceil\hat{\boldsymbol{Y}} - \boldsymbol{Y}\rceil^{\top}_{1:C-1}$
8: $\beta = \max\left\{\frac{\langle\boldsymbol{G}^{(l)},\boldsymbol{G}^{(l)}-\boldsymbol{G}^{(l-1)}\rangle}{\langle\boldsymbol{D}^{(l-1)},\boldsymbol{G}^{(l)}-\boldsymbol{G}^{(l-1)}\rangle}, 0\right\} - \theta\frac{\langle\boldsymbol{G}^{(l)},\boldsymbol{W}^{(l)}-\boldsymbol{W}^{(l-1)}\rangle}{\langle\boldsymbol{D}^{(l-1)},\boldsymbol{G}^{(l)}-\boldsymbol{G}^{(l-1)}\rangle}$
9: $\boldsymbol{D}^{(l)} = -\boldsymbol{G}^{(l)} + \beta\boldsymbol{D}^{(l-1)},\ l \leftarrow l+1$
10: **until** convergence
Output: $\boldsymbol{W} = \boldsymbol{W}^{(l)}$

Since this Hessian is non-negative, the optimization problem in (6) is convex. Based on these quantities, we apply Newton-Raphson method to (6),

$$\alpha^{new} = \alpha^{old} - \frac{g(\alpha^{old})}{h(\alpha^{old})}. \tag{7}$$

This is a one-dimensional optimization and it terminates in only a few iterations in most cases. By employing so optimized step size α, the number of CG iterations is significantly reduced compared to the ordinary non-linear CG method using a line search [15], [21].

The overall algorithm is shown in Algorithm 1. In this algorithm, the number of matrix multiplication which requires large computation time is reduced via updating the quantities $\boldsymbol{P}, \boldsymbol{Q}$; as a result, the matrix multiplication is required only two times (line 3 and 7 in Algorithm 1) per iteration.

IV. Novel Optimization for Kernel Logistic Regression

As reviewed in Section II-D, the kernel logistic regression has been formulated in the optimization problem with respect to the coefficients $\boldsymbol{W} \in \mathbb{R}^{N\times C-1}$ over the samples by simply substituting the kernel features $\boldsymbol{k}(\boldsymbol{x}_i)$ for the feature vectors $\boldsymbol{x}_i$. It optimizes the classifier in the subspace spanned by the kernel functions of samples, which tends to cause numerically unfavorable issues such as plateau. We will discuss these issues in the experiments. In contrast to the ordinary method, we propose a novel method for kernel logistic regression that directly optimizes the classifier f_c in RKHS $\mathcal{H}$, not the coefficients of the samples, by employing the scheme of the non-linear CG-based optimization described in Section III.

By introducing regularization on the classifier f_c, $c \in \{1,\cdots,C-1\}$, the kernel logistic regression is optimized by

$$J(\{\mathsf{f}_c\}_{c=1,..,C-1}) = \frac{\lambda}{2}\sum_c^{C-1}\|\mathsf{f}_c\|_{\mathcal{H}}^2 - \sum_i^N\sum_c^C y_{ic}\log\Big\{\sigma_c\big([\mathsf{f}_c(\boldsymbol{x}_i)]_{c=1,..,C-1}\big)\Big\} \rightarrow \min_{\{\mathsf{f}_c\}},$$

and the gradient of J with respect to f_c is given by

$$\mathsf{g}_c(\cdot) = \lambda\mathsf{f}_c(\cdot) + \sum_i^N(\hat{y}_{ic} - y_{ic})\mathsf{k}(\boldsymbol{x}_i,\cdot), \tag{8}$$

where $\hat{y}_{ic} = \sigma_c\big([\mathsf{f}_c(\boldsymbol{x}_i)]_{c=1,..,C-1}\big)$ and we use $\mathsf{f}_c(\boldsymbol{x}) = \langle\mathsf{f}_c(\cdot),\mathsf{k}(\boldsymbol{x},\cdot)\rangle_{\mathcal{H}}$. The conjugate gradient is obtained as

$$\mathsf{d}_c^{(l)}(\cdot) = -\mathsf{g}_c^{(l)}(\cdot) + \beta\mathsf{d}_c^{(l-1)}(\cdot) = -\lambda\mathsf{f}_c^{(l)}(\cdot) - \sum_i^N(\hat{y}_{ic}^{(l)} - y_{ic})\mathsf{k}(\boldsymbol{x}_i,\cdot) + \beta\mathsf{d}_c^{(l-1)}(\cdot), \tag{9}$$

$$\mathsf{d}_c^{(0)}(\cdot) = -\mathsf{g}_c^{(0)}(\cdot) = -\lambda\mathsf{f}_c^{(0)}(\cdot) - \sum_i^N(\hat{y}_{ic}^{(0)} - y_{ic})\mathsf{k}(\boldsymbol{x}_i,\cdot),$$

and the classifier f_c is updated by

$$\mathsf{f}_c^{(l)}(\cdot) = \mathsf{f}_c^{(l-1)}(\cdot) + \alpha\mathsf{d}_c^{(l-1)}(\cdot). \tag{10}$$

Based on these update formula, if the initial classifier $\mathsf{f}_c^{(0)}(\cdot)$ is a linear combination of the sample kernel functions $\mathsf{k}(\boldsymbol{x}_i,\cdot)$, it is recursively ensured that all of the functions $\mathsf{f}_c^{(l)}(\cdot)$, $\mathsf{g}_c^{(l)}(\cdot)$ and $\mathsf{d}_c^{(l)}(\cdot)$ can also be represented by such linear combinations as well. In addition, at the optimum, the classifier function eventually takes the following form,

$$\lambda\mathsf{f}_c(\cdot) + \sum_i^N(\hat{y}_{ic} - y_{ic})\mathsf{k}(\boldsymbol{x}_i,\cdot) = 0, \therefore \mathsf{f}_c(\cdot) = \frac{1}{\lambda}\sum_i^N(y_{ic} - \hat{y}_{ic})\mathsf{k}(\boldsymbol{x}_i,\cdot).$$

Thus, the above-mentioned linear combination is actually essential to represent $\mathsf{f}_c^{(l)}$. In this study, by initializing the classifier $\mathsf{f}_c^{(0)} = 0$, such representations are realized; we denote $\mathsf{f}_c(\cdot) = \sum_i^N w_{ci}\mathsf{k}(\boldsymbol{x}_i,\cdot)$, $\mathsf{g}_c(\cdot) = \sum_i^N g_{ci}\mathsf{k}(\boldsymbol{x}_i,\cdot)$ and $\mathsf{d}_c(\cdot) = \sum_i^N d_{ci}\mathsf{k}(\boldsymbol{x}_i,\cdot)$. Consequently, the updates (8), (9) and (10) are applied only to those coefficients:

$$\boldsymbol{G}^{(l)} = \lambda\boldsymbol{W}^{(l)} + \lceil\hat{\boldsymbol{Y}} - \boldsymbol{Y}\rceil^{\top}_{1:C-1} \in \mathbb{R}^{N\times C-1}, \tag{11}$$

$$\boldsymbol{D}^{(l+1)} = -\boldsymbol{G}^{(l)} + \beta\boldsymbol{D}^{(l)},\ \boldsymbol{D}^{(0)} = -\boldsymbol{G}^{(0)} \in \mathbb{R}^{N\times C-1}, \tag{12}$$

$$\boldsymbol{W}^{(l+1)} = \boldsymbol{W}^{(l)} + \alpha\boldsymbol{D}^{(l)} \in \mathbb{R}^{N\times C-1}, \tag{13}$$

where $\hat{\boldsymbol{Y}} = \big[\hat{\boldsymbol{y}}_i = \boldsymbol{\sigma}({\boldsymbol{W}^{(l)}}^{\top}\boldsymbol{k}_i)\big]^{i=1,..,N} \in \mathbb{R}^{C\times N}$, α is a step size and the CG update parameter β is given in a manner similar to (4) by

$$\beta = \max\left\{\frac{\langle\boldsymbol{K}\boldsymbol{G}^{(l)},\boldsymbol{G}^{(l)} - \boldsymbol{G}^{(l-1)}\rangle}{\langle\boldsymbol{K}\boldsymbol{D}^{(l-1)},\boldsymbol{G}^{(l)} - \boldsymbol{G}^{(l-1)}\rangle}, 0\right\} - \theta\frac{\langle\boldsymbol{K}\boldsymbol{G}^{(l)},\boldsymbol{W}^{(l)} - \boldsymbol{W}^{(l-1)}\rangle}{\langle\boldsymbol{K}\boldsymbol{D}^{(l-1)},\boldsymbol{G}^{(l)} - \boldsymbol{G}^{(l-1)}\rangle}.$$

A. Optimum step size α

As in Section III-B, the step size α is determined so as to minimize the cost:

$$\alpha = \arg\min_{\alpha} J\big(\{\mathsf{f}_c + \alpha\mathsf{d}_c\}_{c=1,..,C-1}\big).$$

Let $\boldsymbol{P} = \big[\mathsf{f}_c(\boldsymbol{x}_i)\big]_{c=1,..,C-1}^{i=1,..,N} = \boldsymbol{W}^{\top}\boldsymbol{K} \in \mathbb{R}^{C-1\times N}$, $\boldsymbol{Q} = \big[\mathsf{d}_c(\boldsymbol{x}_i)\big]_{c=1,..,C-1}^{i=1,..,N} = \boldsymbol{D}^{\top}\boldsymbol{K} \in \mathbb{R}^{C-1\times N}$ and $\hat{\boldsymbol{Y}} = \big[\hat{\boldsymbol{y}}_i = \boldsymbol{\sigma}(\boldsymbol{p}_i + \alpha\boldsymbol{q}_i)\big]^{i=1,..,N} \in \mathbb{R}^{C\times N}$, and the gradient and Hessian of J with respect to α are written by

$$J\big(\{\mathsf{f}_c + \alpha\mathsf{d}_c\}_{c=1,..,C-1}\big) = \frac{\lambda}{2}(\alpha^2\langle\boldsymbol{Q}^{\top},\boldsymbol{D}\rangle + 2\alpha\langle\boldsymbol{Q}^{\top},\boldsymbol{W}\rangle + \langle\boldsymbol{P}^{\top},\boldsymbol{W}\rangle) - \sum_i^N\sum_c^C y_{ic}\log\hat{y}_{ic},$$

$$\frac{dJ}{d\alpha} = \lambda\big\{\alpha\langle\boldsymbol{Q}^{\top},\boldsymbol{W}\rangle + \langle\boldsymbol{Q}^{\top},\boldsymbol{D}\rangle\big\} + \langle\boldsymbol{Q},\lceil\hat{\boldsymbol{Y}} - \boldsymbol{Y}\rceil_{1:C-1}\rangle \triangleq g(\alpha),$$

$$\frac{d^2J}{d\alpha^2} = \lambda\langle\boldsymbol{Q}^{\top},\boldsymbol{D}\rangle + \sum_i^N\sum_c^{C-1}\hat{y}_{ic}q_{ic}\Big(q_{ic} - \sum_k^{C-1}\hat{y}_{ik}q_{ik}\Big) \triangleq h(\alpha).$$

The step size α is optimized by Newton-Raphson in (7).

The overall algorithm is shown in Algorithm 2. Although

Algorithm 2 : Kernel Logistic Regression by non-linear CG

Input: $\boldsymbol{K} \in \mathbb{R}^{N\times N}$, $\boldsymbol{Y} = [\boldsymbol{y}_i]^{i=1,..,N} \in \{0,1\}^{C\times N}$

1: **Initialize** $\boldsymbol{W}^{(0)} = \boldsymbol{0} \in \mathbb{R}^{N\times C-1}$, $\hat{\boldsymbol{Y}} = \left[\frac{1}{C}\right] \in \mathbb{R}^{C\times N}$
$\boldsymbol{G}^{(0)} = \lceil \hat{\boldsymbol{Y}} - \boldsymbol{Y} \rceil^{\top}_{1:C-1} \in \mathbb{R}^{N\times C-1}$,
$\boldsymbol{D}^{(0)} = -\boldsymbol{G}^{(0)} \in \mathbb{R}^{N\times C-1}$
$\boldsymbol{P} = {\boldsymbol{W}^{(0)}}^{\top} \boldsymbol{K} = \boldsymbol{0} \in \mathbb{R}^{C-1\times N}$,
$\boldsymbol{Q} = {\boldsymbol{D}^{(0)}}^{\top} \boldsymbol{K} \in \mathbb{R}^{C-1\times N}$, $l = 1$,
2: **repeat**
3: $\alpha = \arg\min_{\alpha} J(\{\mathsf{f}_c^{(l-1)} + \alpha \mathsf{d}_c^{(l-1)}\}_{c=1,..,C-1})$: see Section IV-A
4: $\boldsymbol{W}^{(l)} = \boldsymbol{W}^{(l-1)} + \alpha \boldsymbol{D}^{(l-1)}$, $\boldsymbol{P} \leftarrow \boldsymbol{P} + \alpha \boldsymbol{Q}$
5: $\hat{\boldsymbol{Y}} = \left[\hat{\boldsymbol{y}}_i = \boldsymbol{\sigma}(\boldsymbol{p}_i)\right]^{i=1,..,N}$,
$J^{(l)} = J\left(\{\mathsf{f}_c^{(l)}\}_{c=1,..,C-1}\right) = \frac{\lambda}{2}\langle \boldsymbol{P}, \boldsymbol{W}^{(l)}\rangle - \sum_i^N \sum_c^C y_{ic} \log \hat{y}_{ic}$
6: $\boldsymbol{G}^{(l)} = \lceil \hat{\boldsymbol{Y}}^{(l)} - \boldsymbol{Y} \rceil^{\top}_{1:C-1}$
7: $\boldsymbol{R} = {\boldsymbol{G}^{(l)}}^{\top} \boldsymbol{K}$
8: $\beta = \max\left\{ \frac{\langle \boldsymbol{R}^{\top}, \boldsymbol{G}^{(l)} - \boldsymbol{G}^{(l-1)}\rangle}{\langle \boldsymbol{Q}^{\top}, \boldsymbol{G}^{(l)} - \boldsymbol{G}^{(l-1)}\rangle}, 0 \right\} - \theta \frac{\langle \boldsymbol{R}^{\top}, \boldsymbol{W}^{(l)} - \boldsymbol{W}^{(l-1)}\rangle}{\langle \boldsymbol{Q}^{\top}, \boldsymbol{G}^{(l)} - \boldsymbol{G}^{(l-1)}\rangle}$
9: $\boldsymbol{D}^{(l)} = -\boldsymbol{G}^{(l)} + \beta \boldsymbol{D}^{(l-1)}$
10: $\boldsymbol{Q} \leftarrow -\boldsymbol{R} + \beta \boldsymbol{Q}$, $l \leftarrow l + 1$
11: **until** convergence

Output: $\mathsf{f}_c = \sum_i^N w_{ci}^{(l)} \mathsf{k}(\boldsymbol{x}_i, \cdot)$

only the coefficients are updated, the proposed method directly optimizes the classifier f_c itself by minimizing J with respect to f_c. In that point, the method differs from the ordinary optimization in kernel logistic regression.

V. Multiple Kernel Logistic Regression

In recent years, such a method that integrates different kernel functions with the optimized weights for a novel kernel has attracted keen attentions, which is called multiple kernel learning (MKL). By combining multiple types of kernels, the heterogeneous information, which is complementary to each other, can be effectively incorporated to improve the performance. The MKL has been mainly addressed in the framework of large margin classifiers [11]. In this section, we formulate MKL in the proposed scheme of kernel logistic regression described in Section IV.

For MKL, we first consider combined RKHS as in [29]. Suppose we have M types of kernel functions, $\mathsf{k}_1, \cdots, \mathsf{k}_M$, and corresponding RKHS's $\mathcal{H}_1, \cdots, \mathcal{H}_M$ each of which is endowed with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_m}$. We further introduce the slightly modified Hilbert space $\mathcal{H}'_m$ in which the following inner product with a scalar value $v_m \geq 0$ is embedded:

$$\mathcal{H}'_m = \left\{ \mathsf{f} | \mathsf{f} \in \mathcal{H}_m, \frac{\|\mathsf{f}\|_{\mathcal{H}_m}}{v_m} < \infty \right\}, \ \langle \mathsf{f}, \mathsf{g} \rangle_{\mathcal{H}'_m} = \frac{\langle \mathsf{f}, \mathsf{g} \rangle_{\mathcal{H}_m}}{v_m}.$$

This Hilbert space $\mathcal{H}'_m$ is a RKHS with the kernel $\mathsf{k}'_m(\boldsymbol{x}, \cdot) = v_m \mathsf{k}_m(\boldsymbol{x}, \cdot)$ since

$$\mathsf{f}(\boldsymbol{x}) = \frac{\langle \mathsf{f}(\cdot), v_m \mathsf{k}_m(\boldsymbol{x}, \cdot)\rangle_m}{v_m} = \langle \mathsf{f}(\cdot), v_m \mathsf{k}_m(\boldsymbol{x}, \cdot)\rangle_{\mathcal{H}'_m}.$$

Finally, we define the RKHS $\bar{\mathcal{H}}$ as direct sum of $\mathcal{H}'_m$: $\bar{\mathcal{H}} = \bigoplus_m^M \mathcal{H}'_m$, in which the associated kernel function is given by

$$\bar{\mathsf{k}}(\boldsymbol{x}, \cdot) = \sum_m^M \mathsf{k}'_m(\boldsymbol{x}, \cdot) = \sum_m^M v_m \mathsf{k}_m(\boldsymbol{x}, \cdot).$$

Based on the $\bar{\mathcal{H}}$, we estimate the class posterior probabilities as

$$\hat{\boldsymbol{y}} = \boldsymbol{\sigma}\left([\bar{\mathsf{f}}_c(\boldsymbol{x})]_{c=1,..,C-1}\right),$$

where $\bar{\mathsf{f}}_c \in \bar{\mathcal{H}}$ is the classifier function in the combined RKHS. We formulate the multiple-kernel logistic regression (MKLR) in

$$J\left(\{\bar{\mathsf{f}}_c \in \bar{\mathcal{H}}\}_{c=1,..,C-1}, \boldsymbol{v}\right) \tag{14}$$
$$= \frac{\lambda}{2} \sum_c^{C-1} \|\bar{\mathsf{f}}_c\|^2_{\bar{\mathcal{H}}} - \sum_i^N \sum_c^C y_{ic} \log\left[\sigma_c\left([\bar{\mathsf{f}}_c(\boldsymbol{x}_i)]_{c=1,..,C-1}\right)\right] \rightarrow \min_{\{\bar{\mathsf{f}}_c\}, \boldsymbol{v}}$$
$$\Leftrightarrow J\left(\{\mathsf{f}_{mc} \in \mathcal{H}_m\}_{c=1,..,C-1}^{m=1,..,M}, \boldsymbol{v}\right) \tag{15}$$
$$= \frac{\lambda}{2} \sum_m^M \frac{1}{v_m} \sum_c^{C-1} \|\mathsf{f}_{mc}\|^2_{\mathcal{H}_m} - \sum_i^N \sum_c^C y_{ic} \log\left[\sigma_c\left(\left[\sum_m^M \mathsf{f}_{mc}(\boldsymbol{x}_i)\right]_{c=1,..,C-1}\right)\right] \rightarrow \min_{\{\mathsf{f}_{mc}\}, \boldsymbol{v}}$$
$$s.t., \sum_m^M v_m = 1, \ v_m \geq 0, \ \forall m$$

where $\bar{\mathsf{f}}_c(\boldsymbol{x}) = \sum_m^M \mathsf{f}_{mc}(\boldsymbol{x})$ and f_{mc} belongs to each RKHS $\mathcal{H}_m$. The derivative of the cost J in (15) with respect to f_{mc} is given by

$$\frac{\partial J}{\partial \mathsf{f}_{mc}} = \frac{\lambda}{v_m} \mathsf{f}_{mc} + \sum_i^N (\hat{y}_{ic} - y_{ic}) \mathsf{k}_m(\boldsymbol{x}_i, \cdot),$$

where $\hat{y}_i = \sigma\left(\left[\sum_m^M \mathsf{f}_{mc}(\boldsymbol{x}_i)\right]_{c=1,..,C-1}\right)$ and we use $\mathsf{f}_{mc}(\boldsymbol{x}) = \langle \mathsf{f}_{mc}(\cdot), \mathsf{k}_m(\boldsymbol{x}, \cdot)\rangle_{\mathcal{H}_m}$. At the optimum $\frac{\partial J}{\partial \mathsf{f}_{mc}} = \mathsf{0}$, the classifier eventually takes the following form;

$$\bar{\mathsf{f}}_c = \sum_m^M \mathsf{f}_{mc} = \frac{1}{\lambda} \sum_i^N (y_{ic} - \hat{y}_{ic}) \sum_m^M v_m \mathsf{k}_m(\boldsymbol{x}_i, \cdot).$$

Multiple kernels are linearly combined with the weight v_m. Thus, the above-defined MKLR enables us to effectively combine multiple kernels, which can be regarded as multiple kernel learning (MKL).

The regularization term in the costs (14) and (15) is an upper bound of the mixed norm as follows.

$$\frac{\lambda}{2} \sum_c^{C-1} \|\bar{\mathsf{f}}_c\|^2_{\bar{\mathcal{H}}} = \frac{\lambda}{2} \sum_m^M \frac{1}{v_m} \sum_c^C \|\mathsf{f}_{mc}\|^2_{\mathcal{H}_m} \geq \frac{\lambda}{2} \left(\sum_m^M \sqrt{\sum_c^C \|\mathsf{f}_{mc}\|^2_{\mathcal{H}_m}} \right)^2$$

where the equality holds for $v_m = \frac{\sqrt{\sum_c^C \|\mathsf{f}_{mc}\|^2_{\mathcal{H}_m}}}{\sum_m^M \sqrt{\sum_c^C \|\mathsf{f}_{mc}\|^2_{\mathcal{H}_m}}}$. The right-hand-side is similar to group LASSO, and such regularization induces sparseness on the multiple kernels [30]; namely, we can obtain the sparse kernel weights in MKLR. It is noteworthy that the proposed MKLR in (14) and (15) is a convex optimization problem since the regularization term as well as the second term are convex (ref. Appendix in [29]).

We alternately minimize the objective cost (14) with respect to two variables $\{\bar{\mathsf{f}}_c\}_{c=1,..,C-1}$ and $\boldsymbol{v} = [v_m]_{m=1,..,M}$.

A. Optimization for $\bar{\mathsf{f}}$

The gradients of the cost J in (14) with respect to $\bar{\mathsf{f}}_c$ in the RKHS $\bar{\mathcal{H}}$ is given by

$$\frac{\partial J}{\partial \bar{\mathsf{f}}_c} = \lambda \bar{\mathsf{f}}_c(\cdot) + \sum_i^N \{\hat{y}_{ic} - y_{ic}\} \bar{\mathsf{k}}(\boldsymbol{x}_i, \cdot),$$

where we use $\bar{\mathsf{f}}_c(\boldsymbol{x}) = \langle \bar{\mathsf{f}}_c(\cdot), \bar{\mathsf{k}}(\boldsymbol{x}, \cdot)\rangle_{\bar{\mathcal{H}}}$. This is the same form as in the kernel logistic regression in (8) by replacing kernel function $\mathsf{k}(\boldsymbol{x}, \cdot) \mapsto \bar{\mathsf{k}}(\boldsymbol{x}, \cdot)$ and $\boldsymbol{K} \mapsto \bar{\boldsymbol{K}} = \sum_m^M v_m \boldsymbol{K}^{[m]}$ where $\boldsymbol{K}^{[m]} \in \mathbb{R}^{N\times N}$ is the Gram matrix of the m-th type of kernel k_m. Therefore, the optimization procedure described

in Section IV is also applicable to this optimization. The classifiers $\bar{\mathsf{f}}_c$ and the conjugate gradients d_c are represented by linear combinations of kernel functions $\bar{\mathsf{k}}(\boldsymbol{x}_i, \cdot)$;

$$\bar{\mathsf{f}}_c^{(l)} = \sum_i^N w_{ci}^{(l)} \bar{\mathsf{k}}(\boldsymbol{x}_i, \cdot) = \sum_i^N w_{ci}^{(l)} \sum_m^M v_m \mathsf{k}_m(\boldsymbol{x}_i, \cdot), \quad (16)$$
$$\mathsf{d}_c^{(l)} = \sum_i^N d_{ci}^{(l)} \bar{\mathsf{k}}(\boldsymbol{x}_i, \cdot),$$

and the update for $\bar{\mathsf{f}}$ is performed by

$$\bar{\mathsf{f}}_c^{(l)}(\cdot) = \bar{\mathsf{f}}_c^{(l-1)}(\cdot) + \alpha \mathsf{d}_c^{(l-1)}(\cdot).$$

Note that only the coefficients $\boldsymbol{W}^{(l)} = [w_{ci}^{(l)}]_{i=1,..,N}^{c=1,..,C-1}$, $\boldsymbol{D}^{(l)} = [d_{ci}^{(l)}]_{i=1,..,N}^{c=1,..,C-1}$ are updated by (11)∼(13).

B. Optimization for $\boldsymbol{v}$

To update the kernel weights $\boldsymbol{v}$, the following cost using the updated $\bar{\mathsf{f}}_c$ in (16) is minimized with respect to $\boldsymbol{v}$:

$$J(\boldsymbol{v}) = \frac{\lambda}{2} \sum_m^M v_m \langle {\boldsymbol{P}^{[m]}}^\top, \boldsymbol{W}^{(l)} \rangle - \sum_i^N \sum_c^C y_{ic} \log \left\{ \sigma_c \left(\textstyle\sum_m^M v_m \boldsymbol{p}_i^{[m]} \right) \right\},$$

where $\boldsymbol{P}^{[m]} = {\boldsymbol{W}^{(l)}}^\top \boldsymbol{K}^{[m]} \in \mathbb{R}^{C-1 \times N}$. The derivative of this cost function with respect to $\boldsymbol{v}$ is

$$\frac{\partial J}{\partial v_m} = \frac{\lambda}{2} \langle {\boldsymbol{P}^{[m]}}^\top, \boldsymbol{W}^{(l)} \rangle + \langle \boldsymbol{P}^{[m]}, \lceil \hat{\boldsymbol{Y}} - \boldsymbol{Y} \rceil_{1:C-1} \rangle,$$
$$s.t., \sum_m^M v_m = 1,\ v_m \geq 0, \quad (17)$$

where $\hat{\boldsymbol{Y}} = \left[\hat{\boldsymbol{y}}_i = \boldsymbol{\sigma}(\sum_m^M v_m {\boldsymbol{W}^{(l)}}^\top \boldsymbol{k}_i^{[m]}) \right]^{i=1,..,N}$. We apply reduced gradient descent method [31] to minimize the cost while ensuring the constraints (17). The descent direction denoted by $\boldsymbol{e}$ is computed in a manner similar to [29] as follows.

$$\mu = \arg\max_m \left\{ \sum_c^{C-1} \|\mathsf{f}_{mc}\|_{\mathcal{H}_m}^2 = v_m^2 \langle {\boldsymbol{P}^{[m]}}^\top, \boldsymbol{W} \rangle \right\},$$

$$e_m = \begin{cases} 0 & (v_m = 0 \wedge \frac{\partial J}{\partial v_m} - \frac{\partial J}{\partial v_\mu} > 0) \\ -\frac{\partial J}{\partial v_m} + \frac{\partial J}{\partial v_\mu} & (v_m > 0 \wedge m \neq \mu) \\ \sum_{\nu \neq \mu, v_\nu > 0} \frac{\partial J}{\partial v_\nu} - \frac{\partial J}{\partial v_\mu} & (m = \mu) \end{cases}.$$

After computing the descent direction $\boldsymbol{e}$ first, we then check whether the maximal admissible step size (to set a certain component, say v_ν, to 0 in that direction) decreases the objective cost value. In that case, v_ν is updated by setting $v_\nu = 0$ and $\boldsymbol{e}$ is normalized to meet the equality constraint. By repeating this procedure until the objective cost stops decreasing, we obtain both the modified $\boldsymbol{v}'$ and the final descent direction $\boldsymbol{e}$. Then, the kernel weights are updated by $\boldsymbol{v}^{new} = \boldsymbol{v}' + \alpha \boldsymbol{e}$, where α is the step size.

The optimal step size is also computed in a manner similar to the method in linear logistic regression (Section III-B). Let $\boldsymbol{P}^{[m]} = {\boldsymbol{W}^{(l)}}^\top \boldsymbol{K}^{[m]} \in \mathbb{R}^{C-1 \times N}$, $\boldsymbol{P} = \sum_m^M v'_m {\boldsymbol{W}^{(l)}}^\top \boldsymbol{K}^{[m]} = \sum_m^M v'_m \boldsymbol{P}^{[m]}$, $\boldsymbol{Q} = \sum_m^M e_m \boldsymbol{P}^{[m]}$ and $\hat{\boldsymbol{Y}} = \left[\hat{\boldsymbol{y}}_i = \boldsymbol{\sigma}(\boldsymbol{p}_i + \alpha \boldsymbol{q}_i) \right]^{i=1,..,N}$, and the step size α is optimized by (7) using the following derivatives,

$$\frac{dJ}{d\alpha} = \frac{\lambda}{2} \langle \boldsymbol{P}^\top, \boldsymbol{W}^{(l)} \rangle + \langle \boldsymbol{Q}, \lceil \hat{\boldsymbol{Y}} - \boldsymbol{Y} \rceil_{1:C-1} \rangle \triangleq g(\alpha),$$
$$\frac{d^2 J}{d\alpha^2} = \sum_i^N \sum_c^{C-1} \hat{y}_{ic} q_{ic} \left(q_{ic} - \sum_k^{C-1} \hat{y}_{ik} q_{ik} \right) \triangleq h(\alpha).$$

Algorithm 3 : Multiple Kernel Logistic Regression by non-linear CG

Input: $\boldsymbol{K}^{[m]} \in \mathbb{R}^{N \times N}, m \in \{1, \cdots, M\}$,
$\boldsymbol{Y} = [\boldsymbol{y}_i]^{i=1,..,N} \in \{0,1\}^{C \times N}$

1: **Initialize** $\boldsymbol{v} = [\frac{1}{M}] \in \mathbb{R}^M$, $\bar{\boldsymbol{K}} = \sum_m^M v_m \boldsymbol{K}^{[m]}$,
$\boldsymbol{W}^{(0)} = \boldsymbol{0} \in \mathbb{R}^{N \times C-1}$, $\hat{\boldsymbol{Y}} = [\frac{1}{C}] \in \mathbb{R}^{C \times N}$,
$\boldsymbol{G}^{(0)} = \lceil \hat{\boldsymbol{Y}} - \boldsymbol{Y} \rceil_{1:C-1}^\top \in \mathbb{R}^{N \times C-1}$,
$\boldsymbol{D}^{(0)} = -\boldsymbol{G}^{(0)} \in \mathbb{R}^{N \times C-1}$,
$\boldsymbol{P} = {\boldsymbol{W}^{(0)}}^\top \bar{\boldsymbol{K}} = \boldsymbol{0} \in \mathbb{R}^{C-1 \times N}$,
$\boldsymbol{Q} = {\boldsymbol{D}^{(0)}}^\top \bar{\boldsymbol{K}} \in \mathbb{R}^{C-1 \times N}$, $l = 1$
2: **repeat**
3: $\alpha = \arg\min_\alpha J(\{\bar{\mathsf{f}}_c^{(l-1)} + \alpha \mathsf{d}_c^{(l-1)}\}_{c=1,..,C-1})$: see Section V-A
4: $\boldsymbol{W}^{(l)} = \boldsymbol{W}^{(l-1)} + \alpha \boldsymbol{D}^{(l-1)}$, $\boldsymbol{P} \leftarrow \boldsymbol{P} + \alpha \boldsymbol{Q}$
5: **if** l mod $\tau = 0$ **then**
6: /* Optimization for $\boldsymbol{v}$ */
7: $\boldsymbol{P}^{[m]} = {\boldsymbol{W}^{(l)}}^\top \boldsymbol{K}^{[m]}$, $\forall m$
8: Calculate reduced gradient $\boldsymbol{e}$ and $\boldsymbol{v}'$: see Section V-B
9: $\alpha = \arg\min_\alpha J(\{\mathsf{f}_{mc} \in \mathcal{H}_m\}_{c=1,..,C-1}^{m=1,..,M}, \boldsymbol{v}' + \alpha \boldsymbol{e})$
10: $\boldsymbol{v} = \boldsymbol{v}' + \alpha \boldsymbol{e}$
11: $\bar{\boldsymbol{K}} = \sum_m^M v_m \boldsymbol{K}^{[m]}$, $\boldsymbol{P} = \sum_m^M v_m \boldsymbol{P}^{[m]}$
12: $\hat{\boldsymbol{Y}} = \left[\hat{\boldsymbol{y}}_i = \boldsymbol{\sigma}(\boldsymbol{p}_i) \right]^{i=1,..,N}$,
$J^{(l)} = J(\{\bar{\mathsf{f}}_c^{(l)}\}_{c=1,..,C-1}, \boldsymbol{v}) = \frac{\lambda}{2} \langle \boldsymbol{P}, \boldsymbol{W}^{(l)} \rangle - \sum_i^N \sum_c^C y_{ic} \log \hat{y}_{ic}$
13: $\boldsymbol{G}^{(l)} = \lceil \hat{\boldsymbol{Y}} - \boldsymbol{Y} \rceil_{1:C-1}^\top$, $\boldsymbol{D}^{(l)} = -\boldsymbol{G}^{(l)}$
14: $\boldsymbol{Q} = {\boldsymbol{D}^{(l)}}^\top \bar{\boldsymbol{K}}$
15: **else**
16: /* Optimization for $\bar{\mathsf{f}}$ */
17: $\hat{\boldsymbol{Y}} = [\hat{\boldsymbol{y}}_i = \boldsymbol{\sigma}(\boldsymbol{p}_i)]^{i=1,..,N}$,
$J^{(l)} = J(\{\bar{\mathsf{f}}_c^{(l)}\}_{c=1,..,C-1}, \boldsymbol{v}) = \frac{\lambda}{2} \langle \boldsymbol{P}, \boldsymbol{W}^{(l)} \rangle - \sum_i^N \sum_c^C y_{ic} \log \hat{y}_{ic}$
18: $\boldsymbol{G}^{(l)} = \lceil \hat{\boldsymbol{Y}} - \boldsymbol{Y} \rceil_{1:C-1}^\top$
19: $\boldsymbol{R} = {\boldsymbol{G}^{(l)}}^\top \bar{\boldsymbol{K}}$
20: $\beta = \max \left\{ \frac{\langle \boldsymbol{R}^\top, \boldsymbol{G}^{(l)} - \boldsymbol{G}^{(l-1)} \rangle}{\langle \boldsymbol{Q}^\top, \boldsymbol{G}^{(l)} - \boldsymbol{G}^{(l-1)} \rangle}, 0 \right\} - \theta \frac{\langle \boldsymbol{R}^\top, \boldsymbol{W}^{(l)} - \boldsymbol{W}^{(l-1)} \rangle}{\langle \boldsymbol{Q}^\top, \boldsymbol{G}^{(l)} - \boldsymbol{G}^{(l-1)} \rangle}$
21: $\boldsymbol{D}^{(l)} = -\boldsymbol{G}^{(l)} + \beta \boldsymbol{D}^{(l-1)}$
22: $\boldsymbol{Q} \leftarrow -\boldsymbol{R} + \beta \boldsymbol{Q}$
23: **end if**
24: $l \leftarrow l + 1$
25: **until** convergence

Output: $\bar{\mathsf{f}}_c = \sum_i^N w_{ci}^{(l)} \sum_m^M v_m \mathsf{k}_m(\boldsymbol{x}_i, \cdot)$

The overall algorithm is shown in Algorithm 3. Since the dimensionality of $\bar{\mathsf{f}}$ is larger than that of $\boldsymbol{v}$, the optimization for $\boldsymbol{v}$ is performed every τ iterations; we set $\tau = 5$ in this study. It should be noted that the optimizations both of $\bar{\mathsf{f}}$ and $\boldsymbol{v}$ are ensured to monotonically decrease the objective cost J via iterations.

VI. Parallel Computing

Although the non-linear CG sequentially minimizes the objective cost in an iterative manner, each step of iteration can be easily parallelized. The computational cost per iteration is dominated by the (large) matrix multiplications: lines 3 and 7 in Algorithm 1, line 7 in Algorithm 2, and lines 7, 14 and 19 in Algorithm 3. Those multiplications are parallelized such as by multi-thread programming especially in GPGPU, which effectively scales up the whole optimization procedure.

VII. Experimental Results

We conducted various experiments on multi-class classification by using linear logistic regression (LR), kernel LR (KLR) and multiple-kernel LR (MKLR). The proposed methods were compared to the other related methods in terms of the classification accuracy and computation time.

TABLE II. DATASETS OF DENSE FEATURES. WE APPLY FIVE-FOLD CROSS VALIDATION ON THE DATASETS MARKED BY *, WHILE USING GIVEN TRAINING/TEST SPLITS ON THE OTHER DATASETS.

Dataset	#class	#feature	#training sample	#test sample
SENSIT-VEHICLE	3	100	78,823	19,705
SEMEION*	10	256	1,275	318
ISOLET	26	617	6,238	1,559
MNIST	10	784	60,000	10,000
P53*	2	5,408	13,274	3,318

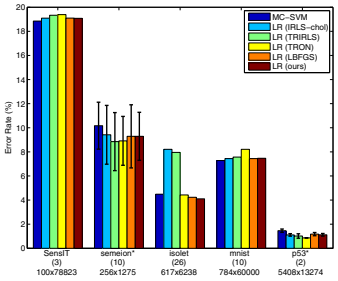
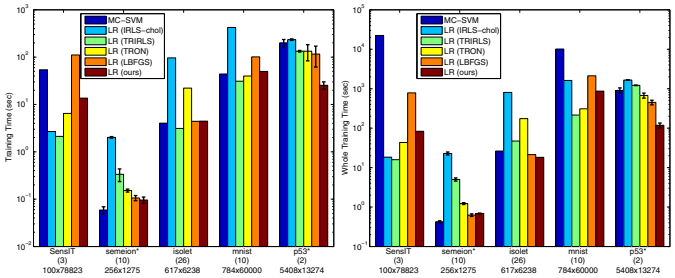


Fig. 1. Error rates on linear classification for dense features. The numbers of classes are indicated in parentheses and the sizes of $\boldsymbol{X}$ (#feature×#sample) are shown in the bottom.



(a) On final learning (b) On whole learning

Fig. 2. Computation times (*log-scale*) on linear classification for dense features. The computation time for learning final classifier is shown in (a), while that for whole training including 3-CV to determine λ is in (b).

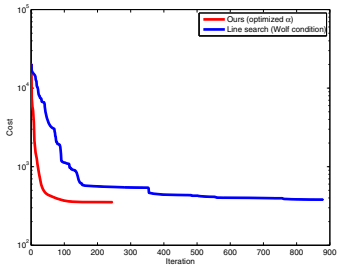


Fig. 3. Comparison to the method using an exhaustive line search. The plot shows the objective cost values through iterations on ISOLET.

A. Linear classification

As a preliminary experiment to the subsequent kernel-based methods, we applied linear classification methods.

For comparison, we applied multi-class support vector machine (MC-SVM) [7] and for LR, four types of optimization methods other than the proposed method in Section III:

- IRLS with Cholesky decomposition (IRLS-chol) [17]
- IRLS with CG (TRIRLS) [18]
- IRLS with trust region newton method (TRON) [20]
- limited memory BFGS method (LBFGS) [13] and [21].

All of these methods introduce regularization with respect to classifier norm in a similar form to (3), of which the regularization parameter is determined by three-fold cross validation on training samples ($\lambda \in \{1, 10^{-2}, 10^{-4}\}$). We implemented all the methods by using MATLAB with C-mex on Xeon 3GHz (12 threading) PC; we used LIBLINEAR [32] for MC-SVM and TRON, and the code[1] provided by Liu and Nocedal [22] for LBFGS.

We first used the datasets[2] of the dense feature vectors, the details of which are shown in Table II. For evaluation, we used the given training/test splits on some datasets and applied five-fold cross validation on the others. The classification performances (error rates) and the computation times for training the classifier are shown in Fig. 1 and Fig. 2, respectively. The computation times are measured in two ways; Fig. 2(a) shows the computation time only for learning the final classifier and Fig. 2(b) is for 'whole' training process including both the final learning and three-fold cross validations to determine the regularization parameter. The proposed method is favorably compared to the other methods in terms of error rates and computation time; the method of LR with IRLS-chol which is quite close to the ordinary IRLS requires more training time.

We then investigated the effectiveness of the optimized step size α (Section III-B) which is one of our contributions in this paper. Fig. 3 shows how the proposed optimization method works, compared to that using an exhaustive line search. By employing the optimized step size, the objective cost drastically decreases in the fist few steps and reaches convergence in a smaller number of iterations.

In the same experimental protocol, we also applied the methods to datasets which contain sparse feature vectors. The details of the datasets[3] are shown in Table III. Note that the method of LR with IRLS-chol can not deal with such a huge feature vectors since the Hessian matrix is quite large, making it difficult to solve linear equations by Cholesky decomposition in a realistic time. As shown in Fig. 4 and Fig. 5, the computation times of the methods are all comparable (around 10 seconds) with similar classification accuracies.

Though the performances of the proposed method are favorably compared to the others as a whole, they are different from those of IRLS-based methods (TRIRLS and TRON). The reason is as follows. The objective costs of those methods[4] are shown in Table IV. The proposed method produces lower objective costs than those by TRIRLS, and thus we can say that the IRLS-based method does not fully converge to global minimum. Although the objective cost function is convex, there would exist plateau [38] which stop the optimization in the IRLS-based methods before converging to the global minimum. Thus, from the viewpoint of optimization, the proposed method produces favorable results.

[1]The code is available at http://www.ece.northwestern.edu/~nocedal.

[2]SEMEION, ISOLET and P53 are downloded from UCI-repository http://archive.ics.uci.edu/ml/datasets.html, and SENSIT-VEHICLE [33] and MNIST [34] are from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

[3]REUTERS21578 (UCI KDD Archive) and TDT2 (Nist Topic Detection and Tracking corpus) are downloaded from http://www.zjucadcg.cn/dengcai/Data/TextData.html, and RCV1 [35], SECTOR [36] and NEWS20 [37] are from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

[4]We do not show the cost of TRON [20] whose formulation is slightly different as described in Section II-A.

TABLE III. DATASETS OF SPARSE FEATURES. WE APPLY FIVE-FOLD CROSS VALIDATION ON THE DATASETS MARKED BY *, WHILE USING GIVEN TRAINING/TEST SPLITS ON THE OTHER DATASETS.

Dataset	#class	#feature	#training sample	#non zeros	#test sample
REUTERS21578	51	18,933	5,926	283,531	2,334
TDT2*	77	36,771	8,140	1,056,166	2,035
RCV1	51	47,236	15,564	1,028,284	518,571
SECTOR	105	55,197	6,412	1,045,412	3,207
NEWS20	20	62,060	15,935	1,272,568	3,993

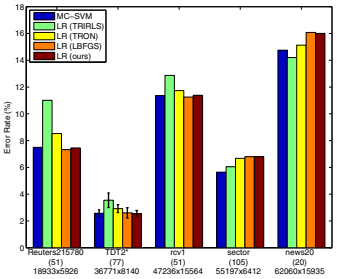
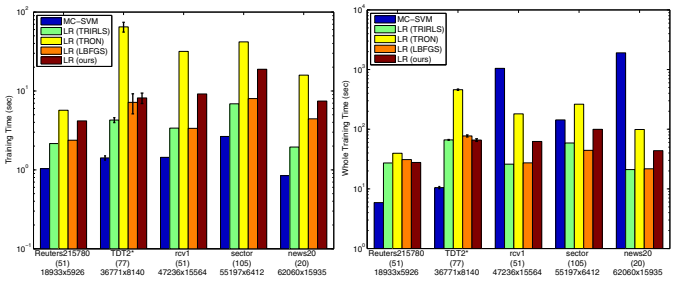


Fig. 4. Error rates on linear classification for sparse features.



(a) On final learning (b) On whole learning

Fig. 5. Computation times on linear classification for sparse features.

TABLE IV. OBJECTIVE COST VALUES OF LR METHODS WITH $\lambda = 10^{-2}$ ON SPARSE DATASETS.

Dataset	Ours	TRIRLS	LBFGS
REUTERS21578	**9.98**	389.26	10.32
TDT2	**12.13**	387.58	13.65
RCV1	**906.49**	15687.17	969.07
SECTOR	**1102.08**	29841.19	1167.35
NEWS20	**1949.60**	7139.07	2000.64

B. Kernel-based classification

Next, we conducted the experiments on kernel-based classifications. We applied the proposed kernel logistic regression (KLR) in Section IV and the kernelized methods of the above-mentioned linear classifiers;

- multi-class kernel support vector machine (MC-KSVM) [7]
- KLR using IRLS with CG (TRIRLS) [18]
- KLR using IRLS with trust region newton method (TRON) by [20]
- KLR using limited memory BFGS method (LBFGS) [13], [21].

Note that the KLR methods of TRIRLS, TRON and LBFGS are kernelized in the way described in Section II-D. Table V shows the details of the datasets[5] that we use, and in this experiment, we employed RBF kernel $\mathsf{k}(\boldsymbol{x}, \boldsymbol{\xi}) = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{\xi}\|^2}{2\sigma^2})$ where σ^2 is determined as the sample variance. The experimental protocol is the same as in Section VII-A.

As shown in Fig. 6, the classification performances of the proposed method are superior to the other KLR methods and are comparable to MC-KSVM, while the computation times of the proposed method are faster than that of MC-KSVM on most datasets (Fig. 7). As discussed in Section VI, we can employ GPGPU (NVIDIA Tesla C2050) to efficiently compute the matrix multiplications in our method on the datasets except for the huge dataset of SHUTTLE, and the computation time is significantly reduced as shown in Fig. 7.

While the proposed method optimizes the classifier in RKHS, the optimization in the other KLR methods is performed in the subspace spanned by the sample kernel functions (Section IV), possibly causing numerically unfavorable issues such as plateau [38], and the optimizations would terminate before fully converging to the global minimum. The objective costs shown in Table VI illustrates it; the proposed method provides lower costs than those of the other KLR methods. In addition, the obtained classifiers, i.e., coefficients $\boldsymbol{W}$ for samples, are shown in Fig. 8. The proposed method produces near sparse weights compared to those of the other methods and contribute to improve the performance similarly to MC-KSVM, even though any constraints to enhance sparseness are not imposed in the proposed method.

C. Multiple-kernel learning

Finally, we conducted the experiment on multiple-kernel learning. We applied the proposed multiple-kernel logistic regression (MKLR) described in Section V and simpleMKL [29] for comparison. For simpleMKL, we used the code[6] provided by the author with LIBSVM [41]. The details of the datasets[7] are shown in Table VII; for multi-class classification, in the dataset of PASCAL-VOC2007, we removed the samples to which multiple labels are assigned. In the datasets of PSORT-, NONPLANT and PASCAL-VOC2007, we used the precomputed kernel matrices provided in the authors' web sites. The dataset of PEN-DIGITS contains four types of feature vectors and correspondingly we constructed four types of RBF kernel in the same way as in Section VII-B.

The classification performances are shown in Fig. 9. As a reference, we also show the performances of KLR with the (single) averaged kernel matrix and the (single) best kernel matrix which produces the best performance among the multiple kernel matrices. The MKL methods produce superior performances compared to those of KLR with single kernel, and the proposed method is comparable to simpleMKL. The obtained kernel weights are also shown in Fig. 10. The weights by the proposed method are sparse similarly to those by simpleMKL, due to the formulation based on the combined RKHS $\bar{\mathcal{H}}$ in (14) and its efficient optimization using non-linear CG.

As shown in Fig. 11, the computation time of the proposed method is significantly ($10^2 \sim 10^4$ times) faster than

[5] USPS [39], LETTER (Statlog), PROTEIN [40] and SHUTTLE (Statlog) are downloaded from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/, and POKER is from UCI repository http://archive.ics.uci.edu/ml/datasets/.

[6] The code is available at http://asi.insa-rouen.fr/enseignants/~arakotom/code/mklindex.html

[7] PASCAL-VOC2007 [42] is downloaded from http://lear.inrialpes.fr/people/guillaumin/data.php, PEN-DIGITS [43] is from http://mkl.ucsd.edu/dataset/pendigits, and PSORT-, NONPLANT [44] are from http://www.fml.tuebingen.mpg.de/raetsch/suppl/protsubloc.

TABLE V. DATASETS FOR KERNEL-BASED CLASSIFICATION.

Dataset	#class	#feature	#training sample	#test sample
USPS	10	256	7,291	2,007
LETTER	26	16	15,000	5,000
PROTEIN	3	357	17,766	6,621
POKER	10	10	25,010	1,000,000
SHUTTLE	7	9	43,500	14,500

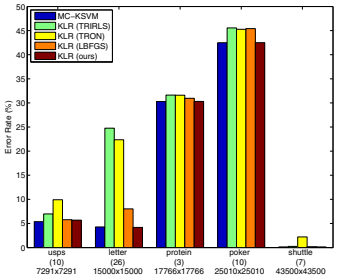
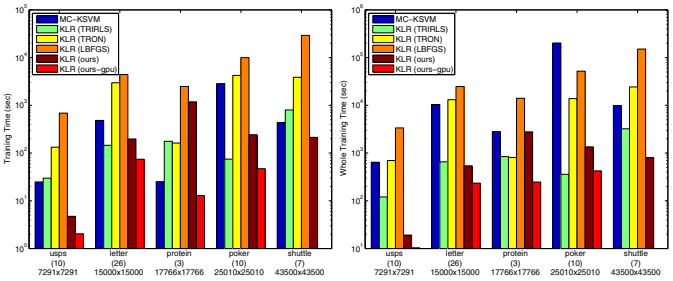


Fig. 6. Error rates on kernel-based classification.

that of simpleMKL. Thus, as is the case with kernel-based classification (Section VII-B), we can say that the proposed method produces comparable performances to simpleMKL with a significantly faster training time.

VIII. CONCLUDING REMARKS

In this paper, we have proposed an efficient optimization method using non-linear conjugate gradient (CG) descent for logistic regression. The proposed method efficiently minimizes the cost through CG iterations by using the optimized step size without an exhaustive line search. On the basis of the non-linear CG based optimization scheme, a novel optimization method for kernel logistic regression (KLR) is also proposed. Unlike the ordinary KLR methods, the proposed method naturally formulates the classifier as the linear combination of sample kernel functions and directly optimizes the kernel-based classifier in the reproducing kernel Hilbert space, not the linear coefficients for the samples. Thus, the optimization effectively performs while possibly avoiding the numerical issues such as plateau. We have further developed the KLR using single kernel to multiple-kernel LR (MKLR). The proposed MKLR, which is also optimized in the scheme of non-linear CG, produces the kernel-based classifier with optimized weights for multiple kernels. In the experiments on various multi-class classification tasks, the proposed methods produced favorable results in terms of classification performance and computation time, compared to the other methods.



(a) On final learning (b) On whole learning

Fig. 7. Computation times on kernel-based classification.

TABLE VI. OBJECTIVE COST VALUES OF KLR METHODS WITH $\lambda = 10^{-2}$ ON KERNEL DATASETS.

Dataset	Ours	TRIRLS	LBFGS
USPS	**446.37**	914.88	501.15
LETTER	**4746.13**	12476.41	5789.30
PROTEIN	**5866.16**	12576.97	10650.96
POKER	**22186.19**	30168.74	23345.94
SHUTTLE	**759.99**	1100.07	811.91

REFERENCES

[1] G. Wang, D. Hoiem, and D. Forsyth, "Building text features for object image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1367–1374.

[2] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert, "An empirical study of context in object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1271–1278.

[3] I. Laptev, M. Marszaek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[4] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale bayesian logistic regression for text categorization," *Technometrics*, vol. 49, no. 3, pp. 291–304, 2007.

[5] P. J. Bartlett, B. Schölkopf, D. Schuurmans, and A. J. Smola, Eds., *Advances in Large-Margin Classifiers*. MIT Press, 2000.

[6] V. Vapnik, *Statistical Learning Theory*. Wiley, 1998.

[7] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2001.

[8] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999, pp. 185–208.

[9] J. Nocedal and S. Wright, *Numerical Optimization*. Springer, 1999.

[10] W. W. Hager and H. Zhang, "A survey of nonlinear conjugate gradient methods," *Pacific Journal of Optimization*, vol. 2, pp. 35–58, 2006.

[11] G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[12] K. Watanabe, T. Kobayashi, and N. Otsu, "Efficient optimization of logistic regression by direct cg method," in *International Conference on Machine Learning and Applications*, 2011.

[13] R. Malouf, "A comparison of algorithms for maximum entropy parameter estimation," in *The Sixth Conference on Natural Language Learning*, 2002, pp. 49–55.

[14] C. Sutton and A. McCallum, *An introduction to conditional random fields for relational learning*, L. Getoor and B. Taskar, Eds. MIT Press, 2006.

[15] T. Minka, "A comparison of numerical optimizers for logistic regression," Carnegie Mellon University, Technical report, 2003.

[16] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2007.

[17] P. Komarek and A. Moore, "Fast robust logistic regression for large sparse datasets with binary outputs," in *The 9th International Workshop on Artificial Intelligence and Statistics*, 2003, pp. 3–6.

[18] ——, "Making logistic regression a core data mining tool," in *International Conference on Data Mining*, 2005, pp. 685–688.

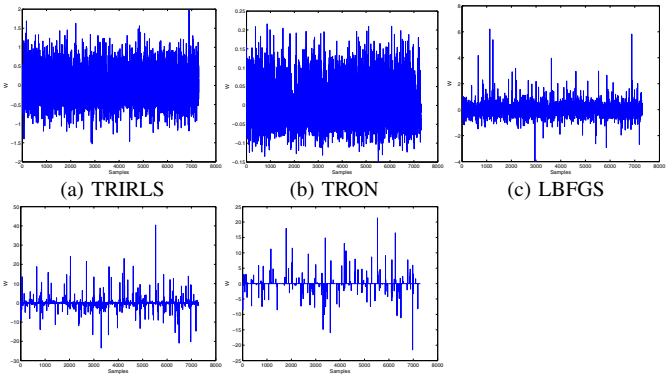
[19] M. R. Hestenes and E. L. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.

[20] C.-J. Lin, R. Weng, and S. Keerthi, "Trust region newton methods for large-scale logistic regression," in *International Conference on Machine Learning*, 2007, pp. 561–568.

[21] H. Daumé III, "Notes on CG and LM-BFGS optimization of logistic regression," Technical report, 2004. [Online]. Available: http://www.umiacs.umd.edu/~hal/docs/daume04cg-bfgs.pdf

[22] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical Programming*, vol. 45, pp. 503–528, 1989.

[23] S. D. Pietra, V. D. Pietra, and J. Lafferty, "Inducing features of



(a) TRIRLS (b) TRON (c) LBFGS

(d) ours (e) MC-KSVM

Fig. 8. Classifiers (coefficients $\boldsymbol{w}_1$ across samples) of class 1 on USPS.

TABLE VII. DATASETS FOR MULTIPLE-KERNEL LEARNING. WE APPLY FIVE-FOLD CROSS VALIDATION ON THE DATASETS MARKED BY *, WHILE USING GIVEN TRAINING/TEST SPLITS ON THE OTHER DATASETS.

Dataset	#class	#kernel	#training sample	#test sample
PSORT-*	5	69	1,155	289
NONPLANT*	3	69	2,186	546
PASCAL-VOC2007	20	15	2,954	3,192
PEN-DIGITS	10	4	7,494	3,498

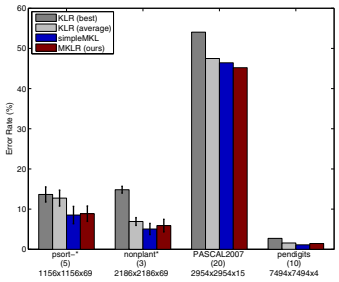
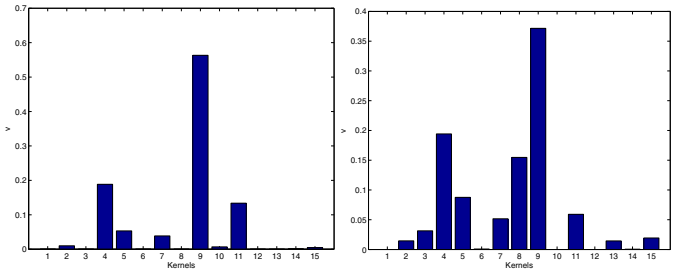
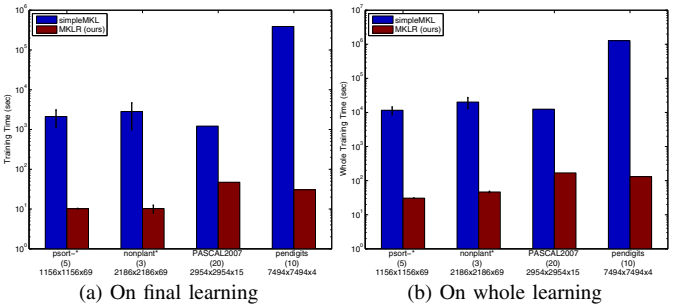


Fig. 9. Error rates and computation times on multiple-kernel learning.



(a) simpleMKL [29] (b) MKLR (ours)

Fig. 10. The obtained kernel weights $\boldsymbol{v}$ on PASCAL-VOC2007.



(a) On final learning (b) On whole learning

Fig. 11. Computation times on multiple-kernel learning.

random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 380–393, 1997.

[24] J. Zhu and T. Hastie, "Kernal logistic regression and the import vector machine," *Journal of Computational and Graphical Statistics*, vol. 14, no. 1, pp. 185–205, 2005.

[25] G. Wahba, C. Gu, Y. Wang, and R. Chappell, "Soft classification, a.k.a. risk estimation, via penalized log likelihood and smoothing spline analysis of variance," in *The Mathematics of Generalization*, D. Wolpert, Ed. Reading, MA, USA: Addison-Wesley, 1995, pp. 329–360.

[26] T. Hastie and R. Tibshirani, *Generalized Additive Models*. Chapman and Hall, 1990.

[27] B. Schölkopf and A. Smola, *Learning with Kernels*. MIT Press, 2001.

[28] Y. Dai and L. Liao, "New conjugacy conditions and related nonlinear conjugate gradient methods," *Applied Mathmatics and Optimization*, vol. 43, pp. 87–101, 2001.

[29] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, "Simplemkl," *Journal of Machine Learning Research*, vol. 9, pp. 2491–2521, 2008.

[30] F. Bach, "Consistency of the group lasso and multiple kernel learning," *Journal of Machine Learning Research*, vol. 9, pp. 1179–1225, 2008.

[31] J. Bonnans, J. Gilbert, C. Lemaréchal, and C. Sagastizábal, *Numerical Optimization: Theoritical and Practical Aspects*. Springer, 2006.

[32] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "Liblinear: A library for large linear classificatio," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008, Software available at http://www.csie.ntu.edu.tw/~cjlin/liblinear.

[33] M. Duarte and Y. H. Hu, "Vehicle classification in distributed sensor networks," *Journal of Parallel and Distributed Computing*, vol. 64, no. 7, pp. 826–838, 2004.

[34] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[35] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "Rcv1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5, pp. 361–397, 2004.

[36] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *AAAI'98 Workshop on Learning for Text categorization*, 1998.

[37] K. Lang, "Newsweeder: Learning to filter netnews," in *International Conference on Machine Learning*, 1995, pp. 331–339.

[38] K. Fukumizu and S. Amari, "Local minima and plateaus in hierarchical structures of multilayer perceptrons," *Neural Networks*, vol. 13, no. 3, pp. 317–327, 2000.

[39] J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[40] J.-Y. Wang, "Application of support vector machines in bioinformatics," Master's thesis, Department of Computer Science and Information Engineering, National Taiwan University, 2002.

[41] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[42] M. Guillaumin, J. Verbeek, and C. Schmid, "Multimodal semi-supervised learning for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 902–909.

[43] F. Alimoglu and E. Alpaydin, "Combining multiple representations and classifiers for pen-based handwritten digit recognition," in *International Conference on Document Analysis and Recognition*, 1997, pp. 637–640.

[44] A. Zien and C. S. Ong, "An automated combination of kernels for predicting protein subcellular localization," in *Proceedings of the 8th Workshop on Algorithms in Bioinformatics*, 2008, pp. 179–186.