JMLR: Workshop and Conference Proceedings 63:49–64, 2016 ACML 2016

Multiple Kernel Learning with Data Augmentation

Khanh Nguyen NKHANH@DEAKIN.EDU.AU
Trung Le TRUNG.L@DEAKIN.EDU.AU
Vu Nguyen V.NGUYEN@DEAKIN.EDU.AU
Tu Dinh Nguyen TU.NGUYEN@DEAKIN.EDU.AU
Dinh Phung DINH.PHUNG@DEAKIN.EDU.AU
Deakin University, Australia

Editors: Robert J. Durrant and Kee-Eung Kim

Abstract

The motivations of multiple kernel learning (MKL) approach are to increase kernel expressiveness capacity and to avoid the expensive grid search over a wide spectrum of kernels. A large amount of work has been proposed to improve the MKL in terms of the computational cost and the sparsity of the solution. However, these studies still either require an expensive grid search on the model parameters or scale unsatisfactorily with the numbers of kernels and training samples. In this paper, we address these issues by conjoining MKL, Stochastic Gradient Descent (SGD) framework, and data augmentation technique. The pathway of our proposed method is developed as follows. We first develop a maximum-a-posteriori (MAP) view for MKL under a probabilistic setting and described in a graphical model. This view allows us to develop data augmentation technique to make the inference for finding the optimal parameters feasible, as opposed to traditional approach of training MKL via convex optimization techniques. As a result, we can use the standard SGD framework to learn weight matrix and extend the model to support online learning. We validate our method on several benchmark datasets in both batch and online settings. The experimental results show that our proposed method can learn the parameters in a principled way to eliminate the expensive grid search while gaining a significant computational speedup comparing with the state-of-the-art baselines.

Keywords: Multiple Kernel Learning, Data Augmentation, Stochastic Gradient Descent

1. Introduction

Support Vector Machine (SVM) was first proposed in (Cortes and Vapnik, 1995). From then on, it has become a prevalent method for machine learning tasks (e.g., classification and regression) and been widely applied to a variety of domains. The primary issue of SVM is specifying the kernel function which best describes the similarity of any two data points. There are a wide spectrum of linear or nonlinear kernel functions to choose and each kernel function has its own parameters to tune. A common approach is to run a grid search over sets of parameters to obtain the optimal one. However, to obtain acceptable accuracy, the grid search requires us to scan through a huge number of kernel parameter sets, and hence it is computationally expensive.

. **Acknowledgement:** This work is partially supported by the Australian Research Council under the Discovery Project DP160109394

A notable approach to relax the grid search is to use multiple kernel learning (MKL) (Gönen and Alpaydın, 2011). In MKL approach, rather than using a single kernel, one prefers combining a wide spectrum of kernels into a linear weighted sum of kernels whose expressiveness capacity is increased. Many applications can take advantage from MKL especially when data are presented with different notions of similarity or from multiple channels. In these situations, it is more reasonable to combine multiple kernels, each of which corresponds to a data representation, than to specify a single kernel, which usually leads to a bias on the chosen kernel.

Different approaches have been proposed to combine multiple kernels. One of the remarkable formulations was first proposed in (Bach et al., 2004) for binary classification and then extended in (Zien and Ong, 2007) for multi-class case. The optimization problem of this formulation is written as follows

$$\min_{\mathbf{W}} \left(\frac{\lambda}{2} \Omega(\mathbf{W}) + \frac{1}{N} \sum_{n=1}^{N} l\left(\mathbf{W}; \mathbf{\Phi}(\mathbf{x}_n), y_n\right) \right)$$

where $\mathbf{W}$ is the weight matrix, and the inner product of two feature vectors $\mathbf{\Phi}(\mathbf{x}_i)$ and $\mathbf{\Phi}(\mathbf{x}_j)$: $\langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j) \rangle = k(\mathbf{x}_i, \mathbf{x}_j) = \sum_{f=1}^{F} d_f k_f(\mathbf{x}_i, \mathbf{x}_j)$ is a linear combination of F kernel functions $k_1, \ldots, k_F$ with $d_f \geq 0, \forall f$ and $\sum_{f=1}^{F} d_f = 1$. The regularization term $\Omega(\mathbf{W})$ is usually the group norm $\mathcal{L}_{2,p}$ $(p \geq 1)$ of $\mathbf{W}$ and the loss function $l(\mathbf{W}; \mathbf{\Phi}(\mathbf{x}_n), y_n)$ can be the Hinge loss function.

Besides predictive performance and expressiveness capacity, the sparsity of the weighted matrix $\mathbf{W}$ is also interesting in MKL. The reason is that a sparse solution not only reduces prediction time, but also gives an intuitive interpretation in some application domains where we need to find which similarity notion or representation type is more important than others (Kloft et al., 2009). To encourage the sparsity, the so-called Ultra-fast Multiple Kernel Learning (UFO-MKL) was proposed in (Orabona and Jie, 2011) which employed the elastic group norm $\Omega(\mathbf{W}) = {}^{\lambda}\!/_{2}\mathcal{L}_{2,p}(\mathbf{W}) + \alpha\mathcal{L}_{2,1}(\mathbf{W})$ to efficiently penalize redundant kernels. Then, the optimization problem of UFO-MKL can be efficiently solved with a good convergence rate using Stochastic Gradient Descent (SGD) framework (Shalev-Shwartz and Kakade, 2009).

Nevertheless, the existing MKL approaches fail to completely avoid the grid search. For example, besides the regularization parameter λ, the UFO-MKL introduces one more parameter α that also needs to be tuned. In batch mode, although one can parallelize the grid search, it takes an expensive computational resource. In addition, the ranges of parameters in the grid search are usually wide because they are unknown or uncertain. In online mode, since the dataset is growing continuously, these methods need to store many models at the same time, each for a parameter set, to obtain the optimal accuracy. To address the model selection problem, one notable approach is to leverage MKL with Bayesian inference. At its crux, Bayesian MKL methods view MKL under probabilistic perspective in a graphical model and aim to infer latent variables using Bayesian inference. One of the state-of-the-art methods in this line is BEMKL (Gonen, 2012), that employs a variational approximation to speed up the inference. However, it still requires inverting N by N and F by F matrices that costs $\mathcal{O}\left(N^3 + F^3 + N^2F^2\right)$, and hence heavily suffers the curse of dimensionality with the training size N and the number of kernel functions F. Moreover, BEMKL requires storing the full kernel matrices, thus it is also inefficient in memory usage.

In this paper, we present a new MKL framework utilizing the strengths of Bayesian inference with Data Augmentation and Stochastic Gradient Descent. Our pathway is as follows. First, we view MKL optimization of UFO-MKL under probabilistic perspective using a graphical model representation. We utilize data augmentation technique (Polson et al., 2011) which enables an efficient posterior inference by coupling the model $\mathbf{W}$ to appropriate auxiliary variables. Conveniently, we can avoid the group norm $\mathcal{L}_{2,1}$ that makes the optimization problem of UFO-MKL complicated. As a consequence, we can use a maximum-a-posteriori (MAP) estimate with the standard SGD to infer $\mathbf{W}$. Then, we use Gibbs sampling to infer the latent variables in the augmented graphical model. Comparing with UFO-MKL, our method shares the ability to scale with large-scale datasets and to run in online mode. Our model, however, completely eliminates the grid search since the hyperparameters can be inferred automatically. We validate our method on 8 benchmark datasets and compare with state-of-the-art baselines. The experimental results show that our method can automatically infer the appropriate parameters in a principled way under both batch and online contexts while simultaneously gaining a computational speedup comparing with the baselines.

2. Related work

In this section, we briefly review the literature mostly related to ours. In particular, we summarize MKL approach in Section 2.1 and data augmentation technique in Section 2.2.

2.1. Multiple Kernel Learning

In recent years, multiple kernel learning has attracted great attention due to its advantages over single kernel learning. The seminal works in MKL were proposed in (Cristianini et al., 2001; Crammer et al., 2002). Later, the influential work of (Lanckriet et al., 2004) proposed to formulate the MKL problem as a quadratically-constrained quadratic problem (QCQP) which can be efficiently solved using general-purpose optimization toolboxes (e.g., Mosek (Andersen and Andersen, 2000)). However, this work is only suitable for small-scale datasets. Bach et al. (2004) scaled up MKL by formulating the problem as a second-order cone programming problem but the proposed model is still limited to medium-scale datasets. To further scale up MKL, Sonnenburg et al. (2006) viewed the MKL problem as a semi-infinite linear programming (SILP). Subsequently, Rakotomamonjy et al. (2008) suggested applying sub-gradient descent (SD) to gain a MKL method faster than SILP. Then, Xu et al. (2009) improved the scalability by extending level method and achieved a method faster than SILP and SD. Regardless of the approaches used, most methods are based on an alternating optimization strategy containing two steps: i) updating the combination kernel function while the current learner-based solution is fixed and ii) finding the learner-based solution with the fixed combination kernel function. Although this strategy can utilize the existing efficient SVM solvers, it does not guarantee a bound on the maximum number of iterations needed (Orabona and Jie, 2011).

A sparse kernel combination is a preferable choice in MKL as a useful tool for feature selection or kernel selection (Kloft et al., 2009). A straightforward approach is to use $\mathcal{L}_{2,1}$ norm which allows the redundant kernels to be penalized and eliminated. Although the group norm $\mathcal{L}_{2,1}$ yields a sparse solution, the fact that this group norm is not smooth makes the optimization problem more complicated and slows down the convergence rate. To alleviate this problem, several works have replaced the group norm $\mathcal{L}_{2,1}$ by the group norm $\mathcal{L}_{2,p}$ ($p \geq 1$) (Kloft et al., 2009; Orabona et al., 2010; Sun et al., 2010). Particularly, Sun et al. (2010) developed an efficient method based on sequential minimal optimization (SMO). The proposed algorithm can solve the MKL problem directly instead of iteratively solving intermediate SVMs. However, the solution obtained from these methods is not truly sparse because the kernel weights do not become exactly zero although they can be extremely small. UFO-MKL was proposed by Orabona and Jie (2011) to address this issue by introducing an elastic group norm that allows the redundant kernels to be penalized and eliminated while being able to utilize the framework proposed in (Shalev-Shwartz and Kakade, 2009) to efficiently solve its optimization problem directly in the primal form.

It is noteworthy that the aforementioned methods can relax the grid search but cannot completely avoid it since there are still model parameters to be tuned. For example with UFO-MKL, the regularization parameter λ and the sparsity control parameter α need to be tuned using the grid search. A notable approach to address this issue is to use Bayesian inference. To enable Bayesian inference in MKL, it requires to view a MKL problem under the probabilistic perspective captured in a graphical model. Girolami and Rogers (2005) conducted a hierarchical probabilistic model for MKL and used variational method to estimate the posterior distribution. Subsequently, this model was extended by (Damoulas and Girolami, 2009) wherein multinomial logistic likelihood was replaced by multinomial probit likelihood and Gibbs sampling was used for inference. Girolami and Zhong (2007) assumed that each decision function of each representation was drawn from Gaussian Process. Based on this assumption, three methods was proposed to make inference including Gibbs sampling, variational approximation and expectation propagation. Zhang et al. (2011) proposed a fully Bayesian inference and used a Markov chain Monte Carlo for posterior predictions. A mixture of a point-mass distribution and Silverman's g-prior was employed to encourage the sparsity. Recently, Gonen (2012) proposed an efficient inference scheme using variational method. Although the work of (Gonen, 2012) is proven more efficient and faster than previous works, it still scales cubically with the training size and the number of kernels.

2.2. Data Augmentation Technique

The underlying premise of data augmentation technique is to integrate unobserved data or auxiliary variables in order to make computation tractable. This technique has increasingly drawn an attention in a variety of problems, including (Albert and Chib, 1993), (Meng and Van Dyk, 1999), (Holmes et al., 2006) and (Frühwirth-Schnatter et al., 2009). A recent notable work in SVM problem was proposed by (Polson et al., 2011). From that seminal contribution, much work has used this idea to apply other problems, such as (Perkins et al., 2015), (Chen et al., 2015) and (Nguyen et al., 2016a). In our work, we introduce the auxiliary variable λ_{mf} to engage with each weight $\mathbf{w}_{mf}$ for a computationally tractable joint distribution, and hence the weight matrix $\mathbf{W}$ can be sampled in a tractable form manner. By doing this, we can also avoid the group norm $\mathcal{L}_{2,1}$ optimization problem and thus make it easy to leverage the standard SGD framework to estimate $\mathbf{W}$.

3. Preliminary

In this section, we present some notions and mathematical definitions used throughout our paper. We use bold capital letters (e.g., $\mathbf{W}$, $\mathbf{X}$) to represent matrices. Vectors are displayed using bold lower letters (e.g., $\mathbf{y}$). For any positive integer number N, the set including the first N positive numbers is defined as $[N] \triangleq \{1, 2, \ldots, N\}$. Given a logical statement P, the indicator function $\mathbb{I}_{[P]}$ renders 1 if P is true and renders 0 if otherwise.

Given a m by n matrix $\mathbf{W} = [\mathbf{W}_1, \ldots, \mathbf{W}_n]$, the group norm $\mathcal{L}_{p,q}$ of the matrix $\mathbf{W}$ is defined as $\|\mathbf{W}\|_{p,q} \triangleq \left\| \left[\|\mathbf{W}_1\|_p, \ldots, \|\mathbf{W}_n\|_p \right] \right\|_q$. The Frobenius norm is a special case of the group norm when $p = q = 2$.

A random variable X is said to follow an inverse Gaussian distribution $\mathcal{IG}(\mu, \lambda)$ with mean $\mathbb{E}(X) = \mu$ and variance $D(X) = \mu^3/\lambda$ if its density distribution function is

$$p(x \mid \mu, \lambda) = \frac{1}{\sqrt{2\pi x^3}} \exp\left\{ -\frac{\lambda (x-\mu)^2}{2\mu^2 x} \right\}$$

A random variable Y is known to have generalized inverse Gaussian distribution $\mathcal{GIG}(\gamma, \psi, \chi)$ if it has the following probability density function

$$p(y \mid \gamma, \psi, \chi) = C(\lambda, \psi, \chi)\, y^{\lambda-1} \exp\left\{ -\frac{1}{2}\left(\frac{\chi}{y} + \psi y \right) \right\}$$

where $C(\lambda, \psi, \chi)$ is the normalization constant quantity. Their relationship is that if a random variable Y has generalized inverse Gaussian distribution $\mathcal{GIG}\left(\frac{1}{2}, \lambda, \chi\right)$ then $Y^{-1} = X \sim \mathcal{IG}(\lambda, \mu)$ where $\mu = (\lambda/\chi)^{1/2}$ (Polson et al., 2011).

4. Multiple Kernel Learning with Data Augmentation approach

In this section, we present our proposed MKL framework. We first start with the optimization problem. Then, we describe how to apply the data augmentation approach to infer the latent representation of the support vectors using Bayesian setting. Next, we present the graphical model and posterior inference. Finally, we extend our model for the online setting.

4.1. Optimization problem

Given training set $\mathfrak{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, where each instance $\mathbf{x}_n \in \mathbb{R}^D$ is a $D-$dimensional feature vector and $y_n \in \mathcal{Y} = \{1, \ldots, M\}$ is the corresponding label of $\mathbf{x}_n$. Following the work of (Crammer and Singer, 2002), with multi-class classification setting, we aim at learning M representative hyperplanes $\mathbf{W}_1, \ldots, \mathbf{W}_M$ in the feature space to give discriminative values for data in M classes[1]. Let us further define the weight matrix as $\mathbf{W} = [\mathbf{W}_1, \ldots, \mathbf{W}_M]^{\mathsf{T}}$. The decision function is as follows

$$f(\mathbf{x}) = \max_{m \in \{1,\ldots,M\}} \langle \mathbf{W}_m, \mathbf{\Phi}(\mathbf{x}) \rangle \tag{1}$$

where $\mathbf{\Phi}(\mathbf{x})$ is the representation of data instance $\mathbf{x}$ in the feature space. The feature map $\mathbf{\Phi}(.)$ from the input space to the feature space can be defined via a kernel function k: $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j) \rangle$ where $\mathcal{X}$ is the data domain.

In multiple kernel learning, the feature map $\mathbf{\Phi}(.)$ is constituted by F component feature maps (i.e., $\mathbf{\Phi}^f(.)$, $f = 1, \ldots, F$) and hence, each representative hyperplane $\mathbf{W}_m$ consistently consists of F components. In particular, we have the following representations

$$\mathbf{\Phi}(\mathbf{x}) = \left[\mathbf{\Phi}^1(\mathbf{x}), \ldots, \mathbf{\Phi}^F(\mathbf{x}) \right] \text{ and } \mathbf{W}_m = [\mathbf{w}_{m1}, \ldots, \mathbf{w}_{mF}]$$

It follows that the decision function in Equation (1) can be rewritten explicitly as

1. For consistency with remaining parts, we denote $\mathbf{W}_m$ instead of $\mathbf{w}_m$. As we will show, in multiple kernel learning, $\mathbf{W}_m$ is a matrix made of the concatenation of vectors.

$$f\left(\mathbf{x}\right)=\max_{m\in\{1,\dots,M\}}\left(\sum_{f=1}^{F}\left\langle \mathbf{w}_{mf},\mathbf{\Phi}^{f}\left(\mathbf{x}\right)\right\rangle \right)$$

To encourage the sparsity as in (Orabona and Jie, 2011), we employ an elastic group norm and achieve the following optimization problem

$$\min_{\mathbf{W}}\left(\frac{1}{N}\sum_{n=1}^{N}l\left(\mathbf{W};\,\mathbf{x}_{n},y_{n}\right)+\frac{\alpha'}{2}\left\Vert \mathbf{W}\right\Vert _{2,2}^{2}+\beta'\left\Vert \mathbf{W}\right\Vert _{2,1}\right) \tag{2}$$

where the group norms $\left\Vert \mathbf{W}\right\Vert _{2,2}^{2}=\sum_{m=1}^{M}\sum_{f=1}^{F}\left\Vert \mathbf{w}_{mf}\right\Vert _{2}^{2}$, $\left\Vert \mathbf{W}\right\Vert _{2,1}=\sum_{m=1}^{M}\sum_{f=1}^{F}\left\Vert \mathbf{w}_{mf}\right\Vert _{2}$, α' is a regularization parameter, and β' is a parameter for sparsity tuning.

In addition, the loss function $l\left(\mathbf{W};\,\mathbf{x}_{n},y_{n}\right)$ for the multi-class (Crammer and Singer, 2002) is defined as

$$l\left(\mathbf{W};\,\mathbf{x}_{n},y_{n}\right)=\max\left\{ 0,1+\max_{m\in\mathcal{Y}\backslash y_{n}}g\left(m,\mathbf{x}_{n}\right)-g\left(y_{n},\mathbf{x}_{n}\right)\right\} \tag{3}$$

where $g\left(m,\mathbf{x}_{n}\right)=\left\langle \mathbf{W}_{m},\mathbf{\Phi}\left(\mathbf{x}_{n}\right)\right\rangle$. The optimization problem in Equation (2) is rewritten as

$$\min_{\mathbf{W}}\left(J_{\alpha,\beta}\left(\mathbf{W}\right)=\sum_{n=1}^{N}l\left(\mathbf{W};\,\mathbf{x}_{n},y_{n}\right)+\sum_{m=1}^{M}\sum_{f=1}^{F}\left(\frac{\alpha}{2}\left\Vert \mathbf{w}_{mf}\right\Vert _{2}^{2}+\beta\left\Vert \mathbf{w}_{mf}\right\Vert _{2}\right)\right) \tag{4}$$

where $\alpha=\alpha'N$ and $\beta=\beta'N$.

We now view the MKL problem in Equation (4) under a probabilistic perspective. The solution of the optimization problem in Equation (4) is equal to the MAP estimate of the following pseudo posterior distribution

$$p\left(\mathbf{W}\mid\boldsymbol{X},\boldsymbol{y},\alpha,\beta\right)\propto\exp\left\{ -J_{\alpha,\beta}(\mathbf{W})\right\} \propto C\left(\alpha,\beta\right)p\left(\boldsymbol{y}\mid\boldsymbol{X},\mathbf{W}\right)p\left(\mathbf{W}\mid\alpha,\beta\right) \tag{5}$$

where we have defined

$$\begin{aligned}
p\left(\boldsymbol{y}\mid\boldsymbol{X},\mathbf{W}\right) & = \prod_{n=1}^{N}p\left(y_{n}\mid\mathbf{x}_{n},\mathbf{W}\right)\propto\prod_{n=1}^{N}\exp\left\{ -l\left(\mathbf{W};\,\mathbf{x}_{n},y_{n}\right)\right\} \\
 & = \prod_{n=1}^{N}\exp\left\{ -\max\left\{ 0,1+\max_{m\in\mathcal{Y}\backslash y_{n}}g\left(m,\mathbf{x}_{n}\right)-g\left(y_{n},\mathbf{x}_{n}\right)\right\} \right\} \qquad (6)\\
p\left(\mathbf{W}\mid\alpha,\beta\right) & \propto \prod_{m=1}^{M}\prod_{f=1}^{F}\exp\left\{ -\left(\frac{\alpha}{2}\left\Vert \mathbf{w}_{mf}\right\Vert ^{2}+\beta\left\Vert \mathbf{w}_{mf}\right\Vert \right)\right\}
\end{aligned}$$

and $C\left(\alpha,\beta\right)$ is the normalization term. We note that $p\left(y_{n}\mid\mathbf{x}_{n},\mathbf{W}\right)=\exp\left\{ -\max\left\{ 0,1+\max_{m\in\mathcal{Y}\backslash y_{n}}g\left(m,\mathbf{x}_{n}\right)-g\left(y_{n},\mathbf{x}_{n}\right)\right\} \right\}$ is a pseudo likelihood. However, we can use it as a proper likelihood because we can approximate $\exp\left\{ -\max\left\{ 0,z\right\} \right\} \approx\left(1+e^{-z}\right)^{-1}=S\left(-z\right)$ where S is the sigmoid function and $z=1+\max_{m\in\mathcal{Y}\backslash y_{n}}g\left(m,\mathbf{x}_{n}\right)-g\left(y_{n},\mathbf{x}_{n}\right)$.

4.2. Data Augmentation Approach

To view the above MAP under the standpoint of Bayesian inference, it requires to find the tractable form of the posterior distribution $p\left(\mathbf{W} \mid \boldsymbol{X}, \boldsymbol{y}, \alpha, \beta\right)$ shown in Equation (5). To this end, we utilize the data augmentation technique by coupling each component $\mathbf{w}_{mf}$ to a random variable λ_{mf} that induces a computationally tractable joint distribution $p\left(\mathbf{w}_{mf}, \lambda_{mf}\right)$. In particular, we depart from the following equation (Andrews and Mallows, 1974)

$$\int_0^{\infty} \frac{a}{\sqrt{2\pi\lambda_{mf}}} \exp\left\{-\frac{1}{2}\left(a^2\lambda_{mf} + b^2\lambda_{mf}^{-1}\right)\right\} d\lambda_{mf} = e^{-|ab|}$$

Substituting $a = 1$ and $b = \beta \left\|\mathbf{w}_{mf}\right\|$ to the above equation, we gain

$$\int_0^{\infty} \frac{1}{\sqrt{2\pi\lambda_{mf}}} \exp\left\{-\frac{1}{2}\left(\lambda_{mf} + \beta^2\lambda_{mf}^{-1}\left\|\mathbf{w}_{mf}\right\|^2\right)\right\} d\lambda_{mf} = e^{-\beta\left\|\mathbf{w}_{mf}\right\|} \tag{7}$$

Multiply Equation (7) with $e^{-\frac{\alpha}{2}\left\|\mathbf{w}_{mf}\right\|^2}$, we have

$$\int_0^{\infty} \frac{1}{\sqrt{2\pi\lambda_{mf}}} \exp\left\{-\frac{1}{2}\left(\lambda_{mf} + \left(\beta^2\lambda_{mf}^{-1} + \alpha\right)\left\|\mathbf{w}_{mf}\right\|^2\right)\right\} d\lambda_{mf} = e^{-\left(\frac{\alpha}{2}\left\|\mathbf{w}_{mf}\right\|^2 + \beta\left\|\mathbf{w}_{mf}\right\|\right)}$$

It follows that the conditional distribution of $\mathbf{w}_{mf}$ given α, β can be obtained by marginalizing out the random variable λ_{mf}. In particular, we have the following equation

$$p\left(\mathbf{w}_{mf} \mid \alpha, \beta\right) = \int_0^{\infty} p\left(\mathbf{w}_{mf}, \lambda_{mf} \mid \alpha, \beta\right) d\lambda_{mf}$$

where we have defined the joint distribution of $\mathbf{w}_{mf}$ and λ_{mf} as

$$p\left(\mathbf{w}_{mf}, \lambda_{mf} \mid \alpha, \beta\right) = \frac{1}{\sqrt{2\pi\lambda_{mf}}} \exp\left\{-\frac{1}{2}\left(\lambda_{mf} + \left(\beta^2\lambda_{mf}^{-1} + \alpha\right)\left\|\mathbf{w}_{mf}\right\|^2\right)\right\} \tag{8}$$

Therefore, we achieve the following formula

$$p\left(\mathbf{W}, \boldsymbol{\lambda} \mid \alpha, \beta\right) = \prod_{m=1}^{M}\prod_{f=1}^{F} \exp\left\{-\frac{1}{2}\left(\lambda_{mf} + \left(\beta^2\lambda_{mf}^{-1} + \alpha\right)\left\|\mathbf{w}_{mf}\right\|^2\right)\right\} \tag{9}$$

where M and F are the number of classes and the number of kernels, respectively. In what follows, we present the augmented graphical model and give the detail of the inference.

4.3. Graphical Model Representation for MKL via Data Augmentation

The augmented graphical model of our proposed method is illustrated in Figure 1. Our graphical model can be interpreted as follows:

- The regularization parameter α is drawn from the Gamma distribution with the shape parameter κ_0 and the scale parameter θ_0, i.e., $\alpha \sim \mathcal{G}\left(\kappa_0, \theta_0\right)$.
- The parameter β for sparsity tuning is drawn from Normal distribution with the mean parameter μ_0 and the variance parameter is σ_0^2, i.e., $\beta \sim \mathcal{N}\left(\mu_0, \sigma_0^2\right)$.

- Given α and β, each weight vector $\mathbf{w}_{mf}$ and auxiliary variable λ_{mf} $(m = 1, \ldots, M,\ f = 1, \ldots, F)$ is drawn from the distribution with probability density function described in Equation (8).
- For each data point $\mathbf{x}_n$ $(n = 1, \ldots, N)$, the corresponding output label y_n is drawn from pseudo likelihood described in Equation (6).

Based on this graphical model, we further show how to make inference and learn parameters in the following section. Note that we still need hyper-parameters (i.e., $\kappa_0, \theta_0, \mu_0, \sigma_0$) to fit the model. The underlying idea of building a hierarchy of parameters is to make model more robust with data. Comparing with regularization parameter α and sparsity tuning parameter β, these hyper-parameters are not sensitive to data. As in the BEMKL method, we fix the values of these hyper-parameters for all datasets.

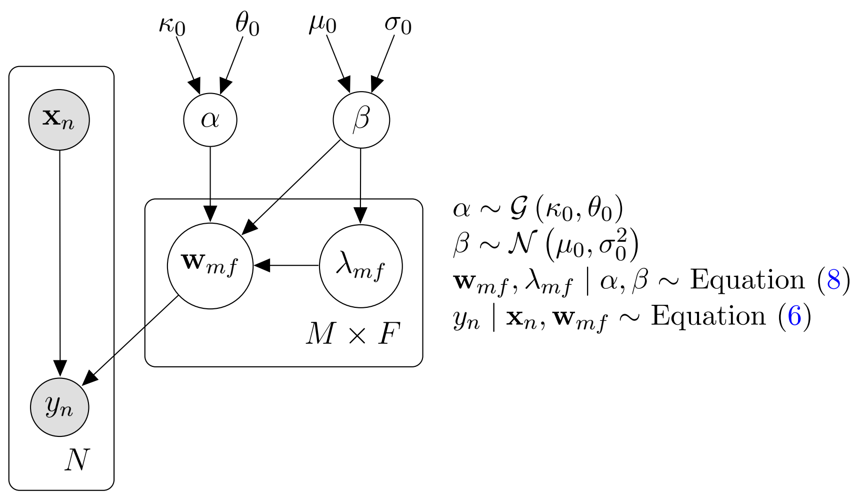
κ_0
θ_0
μ_0
σ_0
α
β
$\mathbf{x}_n$
$\mathbf{w}_{mf}$
λ_{mf}
$M \times F$
y_n
N
$\alpha \sim \mathcal{G}(\kappa_0, \theta_0)$
$\beta \sim \mathcal{N}(\mu_0, \sigma_0^2)$
$\mathbf{w}_{mf}, \lambda_{mf} \mid \alpha, \beta \sim$ Equation (8)
$y_n \mid \mathbf{x}_n, \mathbf{w}_{mf} \sim$ Equation (6)

Figure 1: Graphical model of MKL with Data Augmentation approach.

4.4. Model Inference and Parameter Learning

We use the Gibbs sampling for posterior inference. In what follows, we present the details of inference for each random variable.

4.4.1. Sampling W

We derive the posterior $p(\mathbf{W} \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\lambda}, \alpha, \beta) \propto p(\boldsymbol{y} \mid \boldsymbol{X}, \mathbf{W})\, p(\mathbf{W}, \boldsymbol{\lambda} \mid \alpha, \beta)$. To infer $\mathbf{W}$, we use MAP estimate as

$$\mathbf{W} = \underset{\mathbf{W}}{\operatorname{argmax}}\, p(\mathbf{W} \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\lambda}, \alpha, \beta) = \underset{\mathbf{W}}{\operatorname{argmax}} \left(\log p(\boldsymbol{y} \mid \boldsymbol{X}, \mathbf{W}) + \log p(\mathbf{W}, \boldsymbol{\lambda} \mid \alpha, \beta)\right)$$

Replacing the Equations (6) and (9) to the above equation, we achieve the following optimization problem

$$\min_{\mathbf{W}} \left(\frac{1}{N} \sum_{n=1}^{N} l(\mathbf{W}; \mathbf{x}_n, y_n) + \frac{1}{2} \sum_{m=1}^{M} \sum_{f=1}^{F} \left(\alpha' + \beta\beta'\lambda_{mf}^{-1}\right) \|\mathbf{w}_{mf}\|^2 \right)$$

Since the group norm $\mathcal{L}_{21}$ disappears, we can employ the standard SGD framework to efficiently solve the above optimization problem. The update rule for SGD is as follows

$$\mathbf{w}_{mf} = \left(1 - \frac{1}{t}\right)\mathbf{w}_{mf} - \frac{1}{\gamma_{mf} t}\left(\mathbb{I}_{\left[l\left(\mathbf{W};\mathbf{x}_{n_t},y_{n_t}\right)>0\,\wedge m=m_t\right]}\mathbf{\Phi}^f\left(\mathbf{x}_{n_t}\right) - \mathbb{I}_{\left[l\left(\mathbf{W};\mathbf{x}_{n_t},y_{n_t}\right)>0\,\wedge m=y_{n_t}\right]}\mathbf{\Phi}^f\left(\mathbf{x}_{n_t}\right)\right)$$

where $\gamma_{mf} = \alpha' + \beta\beta'\lambda_{mf}^{-1}$, and $m_t = \text{argmax}_{m\in\mathcal{Y}\setminus y_{n_t}} g\left(m, \mathbf{x}_{n_t}\right)$. Note that we handle $\mathbf{w}_{mf}$ indirectly through a set of weights and each weight is associated with a feature vector $\mathbf{\Phi}^f\left(\mathbf{x}_{n_t}\right)$. Then we apply kernel trick when calculate the loss function $l\left(\mathbf{W};\, \mathbf{x}_{n_t}, y_{n_t}\right)$.

4.4.2. Sampling $\boldsymbol{\lambda}$

From the Equation (9), we have

$$p\left(\lambda_{mf} \mid \mathbf{W}, \alpha, \beta\right) \propto \frac{1}{\sqrt{2\pi\lambda_{mf}}}\exp\left\{-\frac{1}{2}\left(\lambda_{mf} + \frac{\beta^2\left\|\mathbf{w}_{mf}\right\|^2}{\lambda_{mf}}\right)\right\} \sim \mathcal{GIG}\left(\frac{1}{2}, 1, \beta^2\left\|\mathbf{w}_{mf}\right\|^2\right)$$

For completeness, we recall the formula in Section 3 that a random variable Y has generalized inverse Gaussian distribution $\mathcal{GIG}\left(\frac{1}{2}, \lambda, \chi\right)$ then $Y^{-1} = X \sim \mathcal{IG}\left(\lambda, \mu\right)$ where $\mu = \left(\lambda/\chi\right)^{1/2}$. Therefore, we can sample λ_{mf}^{-1} from the inverse Gaussian distribution $\mathcal{IG}\left(1, 1/\beta\left\|\mathbf{w}_{mf}\right\|\right)$.

4.4.3. Sampling α and β

Finally, we need to sample parameter α and β. We derive the joint distribution as follows

$$p\left(\alpha, \beta \mid \mathbf{W}, \boldsymbol{\lambda}\right) \;\propto\; p\left(\mathbf{W}, \boldsymbol{\lambda} \mid \alpha, \beta\right)p\left(\alpha, \beta \mid .\,\right) \propto \exp\left\{-\frac{1}{2}\sum_{m=1}^{M}\sum_{f=1}^{F}\left(\beta^2\lambda_{mf}^{-1} + \alpha\right)\left\|\mathbf{w}_{mf}\right\|^2\right\}p\left(\alpha, \beta \mid .\,\right)$$

We note that $p\left(\alpha \mid .\,\right)$ is the Gamma distribution $\mathcal{G}\left(\kappa_0, \theta_0\right)$ and $p\left(\beta \mid .\,\right)$ is the Normal distribution $\mathcal{N}\left(\mu_0, \sigma_0^2\right)$. Using the conjugate prior property of Normal-Gamma distribution, we then have

$$\begin{aligned} p\left(\alpha, \beta \mid \mathbf{W}, \boldsymbol{\lambda}\right) &\propto \exp\left\{-\frac{1}{2}\sum_{m=1}^{M}\sum_{f=1}^{F}\left(\beta^2\lambda_{mf}^{-1} + \alpha\right)\left\|\mathbf{w}_{mf}\right\|^2\right\}\alpha^{\kappa_0 - 1}\exp\left\{-\frac{\alpha}{\theta_0}\right\}\exp\left\{-\frac{\left(\beta - \mu_0\right)^2}{2\sigma_0^2}\right\} \\ &\propto \alpha^{\kappa_0 - 1}\exp\left\{-\frac{\alpha}{2\theta_0/\left(2+\overline{w}\theta_0\right)}\right\}\exp\left\{-\frac{\left(\beta - \mu_0/\sqrt{1+\overline{\tau}\sigma_0^2}\right)^2}{2\sigma_0^2/\left(1+\overline{\tau}\sigma_0^2\right)}\right\} \end{aligned}$$

where $\bar{w} = \sum_{m=1}^{M}\sum_{f=1}^{F}\left\|\mathbf{w}_{mf}\right\|^2$ and $\bar{\tau} = \sum_{m=1}^{M}\sum_{f=1}^{F}\lambda_{mf}^{-1}\left\|\mathbf{w}_{mf}\right\|^2$.

Therefore, $p\left(\alpha \mid \mathbf{W}, \boldsymbol{\lambda}\right)$ is also Gamma distribution $\mathcal{G}\left(\kappa_l, \theta_l\right)$ with the shape parameter $\kappa_l = \kappa_0$ and the scale parameter $\theta_l = 2\theta_0/\left(2+\overline{w}\theta_0\right)$, and $p\left(\beta \mid \mathbf{W}, \boldsymbol{\lambda}\right)$ is also Normal distribution $\mathcal{N}\left(\mu_l, \sigma_l^2\right)$ with the mean $\mu_l = \mu_0/\sqrt{1+\overline{\tau}\sigma_0^2}$ and the variance $\sigma_l^2 = \sigma_0^2/\left(1+\overline{\tau}\sigma_0^2\right)$.

To summarize, we present the pseudo-code of our proposed method in Algorithm 1. We also note that the norm $\left\|\mathbf{w}_{mf}\right\|$ can be updated incrementally.

4.5. Multiple Kernel Learning under Online Setting

The pseudo-code in Algorithm 1 is designed for batch setting where the entire training set must be available at the training time. The reason is that to efficiently sample α, β and $\boldsymbol{\lambda}$ in the next step, the SGD needs to go through the entire training set one or some rounds to accurately update the matrix $\mathbf{W}$. However, under the online setting, data come continuously, sequentially and evolve rapidly. Consequently, the training sets at different

Algorithm 1 MKL with Data Augmentation approach for batch setting.

```
Input: $\mathfrak{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}, \kappa_0, \theta_0, \mu_0, \sigma_0, T$
Output: $\mathbf{W} = (\mathbf{w}_{mf})$
begin
    $l \leftarrow 1$ and $\mathbf{W} = \mathbf{0}$
    repeat
        Sampling $\alpha_l \sim \mathcal{G}(\kappa_{l-1}, \theta_{l-1})$ and $\beta_l \sim \mathcal{N}(\mu_{l-1}, \sigma_{l-1}^2)$
        Calculate $\alpha_l' = \alpha_l / N$ and $\beta_l' = \beta_l / N$
        for $t \leftarrow 1$ to $T$ do
            Sampling $n_t$ from $[N]$
            Calculate $\varphi \leftarrow \left(1 - \frac{1}{t}\right)$ ; $\gamma_{mf} \leftarrow \alpha_l' + \beta_l \beta_l' \lambda_{mf}^{-1}$ and $m_t \leftarrow \operatorname{argmax}_{m \in Y \setminus y_{n_t}} g(m, \mathbf{x}_{n_t})$
            $\mathbf{w}_{mf} \leftarrow \varphi \mathbf{w}_{mf} - \frac{1}{\gamma_{mft}} \left( \mathbb{I}_{[l(\mathbf{W}; \mathbf{x}_{n_t}, y_{n_t}) > 0 \wedge m = m_t]} \mathbf{\Phi}^f(\mathbf{x}_{n_t}) - \mathbb{I}_{[l(\mathbf{W}; \mathbf{x}_{n_t}, y_{n_t}) > 0 \wedge m = y_{n_t}]} \mathbf{\Phi}^f(\mathbf{x}_{n_t}) \right)$
        end
        Sampling $\lambda_{mf}^{-1} \sim \mathcal{IG}\left(1, \frac{1}{\beta_l \|\mathbf{w}_{mf}\|}\right)$
        Update $\bar{w} \leftarrow \sum_{m=1}^{M} \sum_{f=1}^{F} \|\mathbf{w}_{mf}\|^2$ and $\bar{\tau} \leftarrow \sum_{m=1}^{M} \sum_{f=1}^{F} \lambda_{mf}^{-1} \|\mathbf{w}_{mf}\|^2$
        Update $\kappa_l \leftarrow \kappa_{l-1}$ and $\theta_{l-1} \leftarrow \frac{2\theta_{l-1}}{2 + \bar{w}\theta_{l-1}}$
        Update $\mu_l \leftarrow \frac{\mu_{l-1}}{\sqrt{1 + \bar{\tau}\sigma_{l-1}^2}}$ and $\sigma_l \leftarrow \frac{\sigma_{l-1}}{\sqrt{1 + \bar{\tau}\sigma_{l-1}^2}}$
        $l \leftarrow l + 1$
    until $l = max\ loop$;
end
```

moments might be totally different and hence, if we allow sampling α, β and $\boldsymbol{\lambda}$ whenever the system receives data, the incremental information in the matrix $\mathbf{W}$ may not be sufficient to efficiently guide α, β and $\boldsymbol{\lambda}$. To address this issue, we propose to regularly update the matrix $\mathbf{W}$, but to periodically sample α, β and $\boldsymbol{\lambda}$. In fact, these variables are scheduled to periodically sample in the learning progress. The algorithm for our online version is presented in Algorithm 2.

5. Experiments

In this section, we present our evaluation on the proposed method and compare it against the other state-of-the-art methods. First, we describe our experiment setting in Section 5.1. Then, we report our experimental results on batch mode and online mode in Section 5.2 and 5.3, respectively.

5.1. Experiment Setting

We establish the experiments on 8 benchmark datasets from a variety of domains. All datasets are preprocessed to have zero mean and unit variance across features. Protein (Wang, 2002), pendigits (Alimoglu and Alpaydin, 1996), gisette (Guyon et al., 2004), german (A. Asuncion, 2007), mushrooms (Schlimmer, 1981), ijcnn1 (Prokhorov, 2001) and shuttle datasets are available at the LIBSVM Repository[2]. For flowers17 dataset (Nilsback and Zisserman, 2006), we obtained from Visual Geometry Group's website[3] where seven precomputed distance matrices on different types of representation of data are available. From these distance matrices, we generate 1400 kernel matrices by calculating

2. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
3. http://www.robots.ox.ac.uk/~vgg/data/flowers/17/

Algorithm 2 MKL with Data Augmentation approach for online setting

```
Input: $\kappa_0, \theta_0, \mu_0, \sigma_0, dt_{update}$
Output: $\mathbf{W} = (\mathbf{w}_{mf})$
begin
    $\mathbf{W} = \mathbf{0}$
    for $n \leftarrow 1$ to $\infty$ do
        Receive $(\mathbf{x}_n, y_n) \sim P_{X \times Y}$ where $P_{X \times Y}$ is joint distribution over $X \times Y$
        if $n \ mod \ dt_{update} = 0$ then
            Sampling $\alpha_l \sim \mathcal{G}(\kappa_{l-1}, \theta_{l-1})$ and $\beta_l \sim \mathcal{N}(\mu_{l-1}, \sigma_{l-1}^2)$
            Calculate $\alpha_l' = \alpha_l / n$ and $\beta_l' = \beta_l / n$
        end
        Calculate $\varphi \leftarrow \left(1 - \frac{1}{t}\right)$; $\gamma_{mf} \leftarrow \alpha_l' + \beta_l \beta_l' \lambda_{mf}^{-1}$ and $m_t \leftarrow \underset{m \in Y \setminus y_n}{\text{argmax}}\, g(m, \mathbf{x}_n)$
        $\mathbf{w}_{mf} \leftarrow \varphi \mathbf{w}_{mf} - \frac{1}{\gamma_{mf} t} \left( \mathbb{I}_{[l(\mathbf{W}; \mathbf{x}_n, y_n) > 0 \wedge m = m_t]} \mathbf{\Phi}^f(\mathbf{x}_n) - \mathbb{I}_{[l(\mathbf{W}; \mathbf{x}_n, y_n) > 0 \wedge m = y_n]} \mathbf{\Phi}^f(\mathbf{x}_n) \right)$
        if $n \ mod \ dt_{update} = 0$ then
            Sampling $\lambda_{mf}^{-1} \sim \mathcal{IG}\left(1, \frac{1}{\beta_l \|\mathbf{w}_{mf}\|}\right)$
            Update $\bar{w} \leftarrow \sum_{m=1}^{M} \sum_{f=1}^{F} \|\mathbf{w}_{mf}\|^2$ and $\bar{\tau} \leftarrow \sum_{m=1}^{M} \sum_{f=1}^{F} \lambda_{mf}^{-1} \|\mathbf{w}_{mf}\|^2$
            Update $\kappa_l \leftarrow \kappa_{l-1}$ and $\theta_{l-1} \leftarrow \frac{2\theta_{l-1}}{2 + \bar{w}\theta_{l-1}}$
            Update $\mu_l \leftarrow \frac{\mu_{l-1}}{\sqrt{1 + \bar{\tau}\sigma_{l-1}^2}}$ and $\sigma_l \leftarrow \frac{\sigma_{l-1}}{\sqrt{1 + \bar{\tau}\sigma_{l-1}^2}}$
        end
    end
end
```

$\exp\left(-\gamma \times d\left(\mathbf{x}_i, \mathbf{x}_j\right) / s\right)$ where s is the mean distance between training point pairs and γ is chosen in the grid of 200 instances from 2^{-5} to 2^5. For other datasets, we employ Gaussian kernels with F different widths in the range from 2^{-15} to 2^{15}, where the value of F for each dataset is reported in Table 1. We compare our proposed method MKL-DA with UFO-MKL (Orabona and Jie, 2011) and BEMKL (Gonen, 2012). The codes of baseline methods are achieved from the corresponding authors. All experiments are performed on the computer with the configuration of Xeon E5 2.6 GHz and 96 GB of RAM. We also repeat 5 times and record the corresponding mean value.

5.2. Batch mode comparison

We carry out a set of experiments to study the behavior of our proposed method compared with other baselines. To select trade-off parameters for UFO-MKL, we do cross validation with 5 folds in considered ranges of the regularization parameter λ in the grid $\left\{2^{-5}, 2^{-3}, \ldots, 2^5\right\}$ and the sparsity tuning parameter α in the grid $\{0.0001, \ldots, 0.02\}$ as recommended in (Orabona and Jie, 2011). With BEMKL, we fix the hyper-parameters $(\alpha_\lambda, \beta_\lambda, \alpha_\gamma, \beta_\gamma, \alpha_\omega, \beta_\omega) = (1, 1, 1, 1, 1, 1)$ as suggested in (Gonen, 2012). For our proposed method, we set hyper-parameters $(\kappa_0, \theta_0, \mu_0, \sigma_0) = (1, 1, 1, 1)$.

We measure the performance in terms of total training time, accuracy, precision, specificity and $F-$score. We also note that we report the total time including grid search instead of the time of one parameter set. The results are reported in Table 1. Here we observe

Methods	**Training time (h)**	**Accuracy (%)**	***F*−score (%)**	**Precision (%)**	**Recall (%)**	**Specificity (%)**
flowers17 ($F = 1{,}400; M = 17; N = 680$)						
MKL-DA	**0.51**	75.23	74.95	78.80	75.23	98.45
BEMKL	1.30	82.06	81.37	83.60	82.06	98.88
UFO-MKL	6.31	75.29	75.25	81.46	75.29	98.46
protein ($F = 20; M = 3; N = 17{,}666; D = 357$)						
MKL-DA	**6.36**	68.52	52.00	67.55	54.13	77.84
BEMKL	8.74	70.29	66.16	67.72	65.26	83.66
UFO-MKL	176.89	53.00	51.52	40.68	55.59	75.82
gisette ($F = 300; M = 2; N = 6{,}000; D = 5{,}000$)						
MKL-DA	**1.37**	96.88	96.86	96.88	96.86	96.86
BEMKL	6.15	97.90	97.90	97.91	97.90	97.90
UFO-MKL	33.39	97.74	97.74	97.74	97.74	97.74
mushrooms ($F = 300; M = 2; N = 6{,}500; D = 112$)						
MKL-DA	**0.06**	99.99	99.99	99.99	99.99	99.99
BEMKL	12.62	100	100	100	100	100
UFO-MKL	3.33	100	100	100	100	100
pendigits ($F = 200; M = 3; N = 2{,}000; D = 180$)						
MKL-DA	**1.96**	93.65	92.96	93.63	93.06	99.22
BEMKL	1,250.80	97.45	97.47	97.48	97.48	99.71
UFO-MKL	2,134.26	96.60	96.62	96.64	96.65	99.62
german ($F = 300; M = 2; N = 800; D = 24$)						
MKL-DA	**0.01**	75.10	65.68	73.72	67.54	67.54
BEMKL	0.05	77.00	68.92	71.46	67.66	67.66
UFO-MKL	0.41	75.20	66.47	68.57	65.65	65.65
ijcnn1 ($F = 300; M = 2; N = 49{,}990; D = 22$)						
MKL-DA	**16.59**	96.50	88.59	94.16	84.50	84.50
BEMKL	NA	NA	NA	NA	NA	NA
UFO-MKL	461.33	96.86	89.68	95.97	85.17	85.17
shuttle ($F = 300; M = 7; N = 43{,}500; D = 9$)						
MKL-DA	**5.14**	99.73	70.87	79.54	66.86	99.86
BEMKL	NA	NA	NA	NA	NA	NA
UFO-MKL	424.11	99.19	67.50	91.64	62.56	99.75

Table 1: Performance comparison on benchmark datasets. F denotes the number of kernel functions, M represents the number of classes, N is the training size, D is the dimension of data, and NA means the result is not available. Our proposed method reduces significantly in the total training time (as shown in the second column) while still yields comparable accuracy in comparison with other state-of-the-art baselines (as shown gray columns).

that the total training time of UFO-MKL is much higher than MKL-DA and BEMKL. It is because UFO-MKL needs to find the optimal parameters (i.e., the regularation parameter λ and sparsity tuning parameter α) by grid search, and this affects significantly the overall training time. Conversely, both MKL-DA and BEMKL can tune the trade-off parameter,

Datasets	**Running time (h)**			**Accuracy (%)**		
	UFO-MKL	**MKL-DAO1**	**MKL-DAO2**	**UFO-MKL**	**MKL-DAO1**	**MKL-DAO2**
mushrooms	7.90	**0.03**	**0.03**	95.76	97.25	**98.11**
gisette	271.92	**0.20**	0.21	87.25	**88.83**	88.71
pendigits	18.28	**0.23**	0.43	**88.25**	79.85	83.41
ijcnn1	130.40	**1.65**	**1.65**	**91.42**	88.97	88.97

Table 2: Running time and accuracy comparison on benchmark datasets under online setting. MKL-DAO1 and MKL-DAO2 denote MKL-DAO method with $dt_{update} = 100$ and $dt_{update} = 200$ respectively.

and thus do not need grid search strategy. However, BEMKL takes $\mathcal{O}\left(N^3 + F^3 + N^2F^2\right)$ running time for matrix inversion. In addition, the BEMKL system consumes a large amount of memory for loading kernel matrix. Consequently, BEMKL is suitable only for small datasets which contain no more than ten thousands data points. For example, in our experiments, BEMKL cannot run with ijcnn1 and shuttle datasets, thus the results are not available as we denote NA in Table 1. In contrast, the training time of MKL-DA is significantly less than BEMKL and UFO-MKL, while the accuracy performances are still comparable with other baselines.

5.3. Online mode comparison

As mentioned above, the extended version of our proposed method (named MKL-DAO) can learn under online setting. Meanwhile, BEMKL cannot be applied for online learning because BEMKL needs to load full kernel matrix for matrix inversion, which is not available in online learning context. For UFO-MKL, we modified the code to enable it to work under online setting. We investigate our proposed method under two different settings: $dt_{update} = 100$ (named MKL-DAO1) and $dt_{update} = 200$ (named MKL-DAO2). The performance of compared methods is presented in Table 2. As observed from experimental results, the running time of MKL-DAO is much less than UFO-MKL. Due to the online setting, we cannot do grid search for UFO-MKL which needs to store the models corresponding to the parameter sets to obtain the optimal accuracy, which means that if we have 100 pairs of λ and α, we need to store 100 models. In contrast, MKL-DAO can periodically choose an optimal trade-off parameters after learning some data points, and hence yields to the optimal solution faster than UFO-MKL. In our experiments, for UFO-MKL, we consider the parameter λ in the grid $\left\{2^{-5}, 2^{-3}, \ldots, 2^{5}\right\}$ and the parameter α in the grid $\{0.0001, \ldots, 0.02\}$. However, this range is not suitable for all datasets, e.g., the accuracy of UFO-MKL in mushrooms and gisette datasets are lower than MKL-DAO. This shows the limitation of UFO-MKL when the range of parameters is unknown in advance..

To have a better comparison in terms of running time and accuracy between our proposed method and UFO-MKL, we use Quadrant Score (QS) (Nguyen et al., 2016b) which shows the difference among learning methods. The best learning method not only obtains higher accuracy but also costs less time than others. Mathematically, QS can be calculated by $QS = \text{sqrt}\left((100 - acc)^2 + (100 \times time/limit)\right)$ where $limit$ is the maximum value on the time axis. This formula shows that the best learning method has running time 0 and accuracy 100%. We visualize QS comparison between MKL-DAO and UFO-MKL on several datasets in Figure 2. Intuitively, for each dataset, our proposed methods (the square and

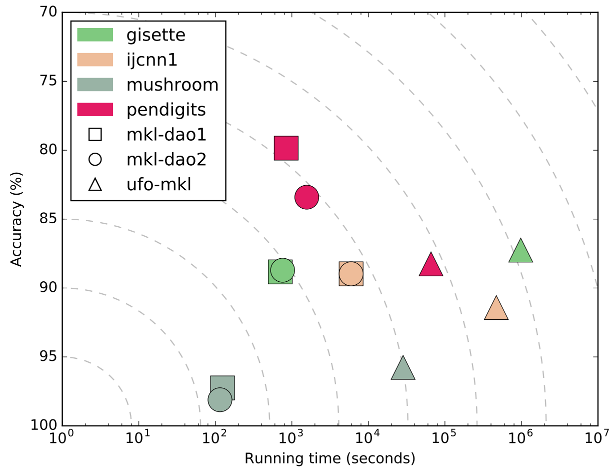

Figure 2: Quadrant Visualization plotting the performance under the online setting. The markers locating near the left-bottom corner of the graph show the better performance in term of running time and accuracy.

the circle ones) locate nearer the left-bottom corner of the graph in comparison with UFO-MKL, i.e. the QS of MKL-DAO is always less than UFO-MKL. This observation shows that MKL-DAO is more efficient than UFO-MKL when views two aspects of performance (running time and accuracy) simultaneously.

6. Conclusion

In this paper, we have considered the multiple kernel learning problem under probabilistic perspective to avoid the grid search. We then have utilized the data augmentation technique to make the posterior optimization problem tractable. Conveniently, the $\mathcal{L}_{2,1}$ optimization problem can be reduced into the $\mathcal{L}_{2,2}$ optimization problem which is much easier to solve. Consequently, we can apply stochastic gradient descent framework resulting in the proposed methods that can automatically learn the parameters in both batch and online settings. We validate the proposed method on benchmark datasets and compare with the state-of-the-art methods. Experimental results indicate that our proposed method is effectively scalable and can be extended to run under online setting.

References

D.J. Newman A. Asuncion. UCI machine learning repository, 2007. URL http://mlearn.ics.uci.edu/MLRepository.html.

James H Albert and Siddhartha Chib. Bayesian analysis of binary and polychotomous response data. *Journal of the American statistical Association*, 88(422):669–679, 1993.

Fevzi Alimoglu and Ethem Alpaydin. Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition. In *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN 96*. Citeseer, 1996.

Erling D Andersen and Knud D Andersen. The mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High performance optimization*, pages 197–232. Springer, 2000.

David F Andrews and Colin L Mallows. Scale mixtures of normal distributions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 99–102, 1974.

Francis R Bach, Gert RG Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM, 2004.

Ning Chen, Jun Zhu, Jianfei Chen, and Ting Chen. Dropout training for svms with data augmentation. *arXiv preprint arXiv:1508.02268*, 2015.

Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

Koby Crammer, Joseph Keshet, and Yoram Singer. Kernel design using boosting. In *Advances in neural information processing systems*, pages 537–544, 2002.

Nello Cristianini, Andre Elisseeff, John Shawe-Taylor, and Jaz Kandola. On kernel-target alignment. 2001.

Theodoros Damoulas and Mark A Girolami. Pattern recognition with a bayesian kernel combination machine. *Pattern Recognition Letters*, 30(1):46–54, 2009.

Sylvia Frühwirth-Schnatter, Rudolf Frühwirth, Leonhard Held, and Håvard Rue. Improved auxiliary mixture sampling for hierarchical models of non-gaussian data. *Statistics and Computing*, 19(4): 479–492, 2009.

Mark Girolami and Simon Rogers. Hierarchic bayesian models for kernel learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 241–248. ACM, 2005.

Mark Girolami and Mingjun Zhong. Data integration for classification problems employing gaussian process priors. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19, page 465. MIT Press, 2007.

Mehmet Gonen. Bayesian efficient multiple kernel learning. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 1–8, New York, NY, USA, July 2012. Omnipress. ISBN 978-1-4503-1285-1.

Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in neural information processing systems*, pages 545–552, 2004.

Chris C Holmes, Leonhard Held, et al. Bayesian auxiliary variable models for binary and multinomial regression. *Bayesian Analysis*, 1(1):145–168, 2006.

Marius Kloft, Ulf Brefeld, Pavel Laskov, Klaus-Robert Müller, Alexander Zien, and Sören Sonnenburg. Efficient and accurate lp-norm multiple kernel learning. In *Advances in neural information processing systems*, pages 997–1005, 2009.

Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.

Xiao-Li Meng and David A Van Dyk. Seeking efficient data augmentation schemes via conditional and marginal augmentation. *Biometrika*, 86(2):301–320, 1999.

Tu Dinh Nguyen, Vu Nguyen, Trung Le, and Dinh Phung. Distributed data augmented support vector machine on spark. In *Proceedings of the 23rd International Conference on Pattern Recognition (ICPR 2016)*, Dec. 2016a.

Vu Nguyen, Tu Dinh Nguyen, Trung Le, Dinh Phung, and Svetha Venkatesh. One-pass logistic regression for label-drift and large-scale classification on distributed systems. In *Proceedings of the IEEE International Conference on Data Mining (ICDM) 2016*, Dec. 2016b. Accepted.

Maria-Elena Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1447–1454. IEEE, 2006.

Francesco Orabona and Luo Jie. Ultra-fast optimization algorithm for sparse multi kernel learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 249–256, 2011.

Francesco Orabona, Luo Jie, and Barbara Caputo. Online-batch strongly convex multi kernel learning. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 787–794. IEEE, 2010.

Hugh Perkins, Minjie Xu, Jun Zhu, and Bo Zhang. Fast parallel svm using data augmentation. *arXiv preprint arXiv:1512.07716*, 2015.

Nicholas G Polson, Steven L Scott, et al. Data augmentation for support vector machines. *Bayesian Analysis*, 6(1):1–23, 2011.

D Prokhorov. Ijcnn 2001 neural network competition. slide presentation in ijcnn'01, ford research laboratory, 2001.

Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

Jeff Schlimmer. Mushroom records drawn from the audubon society field guide to north american mushrooms. *GH Lincoff (Pres), New York*, 1981.

S. Shalev-Shwartz and S. M Kakade. Mind the duality gap: Logarithmic regret algorithms for online optimization. In *Advances in Neural Information Processing Systems*, pages 1457–1464, 2009.

Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *The Journal of Machine Learning Research*, 7:1531–1565, 2006.

Zhaonan Sun, Nawanol Ampornpunt, Manik Varma, and Svn Vishwanathan. Multiple kernel learning and the smo algorithm. In *Advances in neural information processing systems*, pages 2361–2369, 2010.

Jung-Ying Wang. *Application of support vector machines in bioinformatics.* PhD thesis, National Taiwan University, 2002.

Zenglin Xu, Rong Jin, Irwin King, and Michael Lyu. An extended level method for efficient multiple kernel learning. In *Advances in neural information processing systems*, pages 1825–1832, 2009.

Zhihua Zhang, Guang Dai, and Michael I Jordan. Bayesian generalized kernel mixed models. *The Journal of Machine Learning Research*, 12:111–139, 2011.

Alexander Zien and Cheng Soon Ong. Multiclass multiple kernel learning. In *Proceedings of the 24th international conference on Machine learning*, pages 1191–1198. ACM, 2007.