Proceedings of Machine Learning Research 95:129-144, 2018 ACML 2018

Clustering Induced Kernel Learning

Khanh Nguyen NKHANH@DEAKIN.EDU.AU
Deakin University, Australia

Nhan Dam NHAN.DAM@MONASH.EDU
Trung Le TRUNGLM@MONASH.EDU
Tu Dinh Nguyen TU.DINH.NGUYEN@MONASH.EDU
Dinh Phung DINH.PHUNG@MONASH.EDU
Monash University, Australia

Editors: Jun Zhu and Ichiro Takeuchi

Abstract

Learning rich and expressive kernel functions is a challenging task in kernel-based supervised learning. Multiple kernel learning (MKL) approach addresses this problem by combining a mixed variety of kernels and letting the optimization solver choose the most appropriate combination. However, most of existing methods are parametric in the sense that they require a predefined list of kernels. Hence, there appears a substantial trade-off between computation and the modeling risk of not being able to explore more expressive and suitable kernel functions. Moreover, current existing approaches to combine kernels cannot exploit clustering structure carried in data, especially when data are heterogeneous. In this work, we present a new framework that leverages Bayesian nonparametric models (i.e, automatically grow kernel functions) with multiple kernel learning to develop a new framework that enjoys the nonparametric flavor in the context of multiple kernel learning. In particular, we propose *Clustering Induced Kernel Learning* (CIK) method that can automatically discover clustering structure from the data and train a single kernel machine to fit data in each discovered cluster simultaneously. The outcome of our proposed method includes both clustering analysis and multiple kernel classifier for a given dataset. We conduct extensive experiments on several benchmark datasets. The experimental results show that our method can improve classification and clustering performance when datasets have complex clustering structure with different preferred kernels.

1. Introduction

A crucial question in machine learning is how to choose a suitable representation for input data. Under the umbrella of kernel methods, this question is translated to the task of selecting appropriate kernel functions, which are adequate and rich enough for the tasks at hand. In particular, to perform satisfactorily on real-world datasets, kernel methods require to choose a proper kernel function and its relevant kernel parameters. To address the problem of choosing a suitable kernel, a typical, classical approach is to do grid search on a set of single kernels (Hsu et al., 2003). Although this approach is simple, efficient and can be distributed to work on multiple machines, it requires to scan over an exponential

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

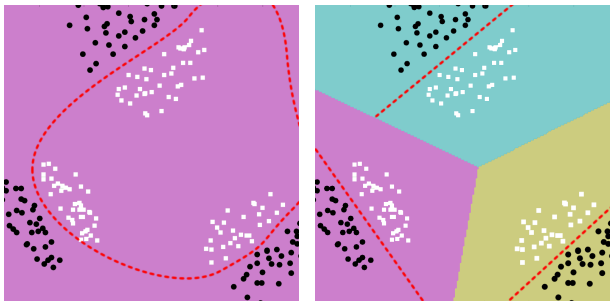


Figure 1: With the help of clustering structure, the classification problem becomes simpler. The region in color (yellow, cyan, and magenta) denotes cluster boundary. The red line represents the decision boundary for classification. On the left, without clustering, the decision boundary should be nonlinear to accurately classify data. On the right, after clustering into 3 groups, data in each group are clearly separated, thus a linear decision boundary for each cluster is good enough.

(a) (b)

amount of parameters and more importantly, determining the ranges of kernel parameters in this approach is known to be challenging. Another notable approach is to utilize multiple kernel learning (MKL), in which a wide spectrum of kernels is combined instead of a single kernel and an optimizer chooses the most appropriate combination. In literature, there have been many methods proposed to combine multiple kernels or to infer mixing proportions of individual kernels (Bach et al., 2004; Girolami and Zhong, 2007; Bach, 2009; Gönen and Alpaydın, 2011; Gonen, 2012; Orabona and Jie, 2011; Oliva et al., 2016). Nevertheless, most of these aforementioned methods require predefining a specific list of kernels. To increase the representation ability of the mixture kernel function and avoid missing any promising candidate, one usually tries to aggregate as many kernels as possible. This strategy, in return, demands a huge amount of computational resources and might lead to overfitting in many cases.

Data have their own intrinsic clustering structure where each cluster contains similar data instances. Since data instances in a cluster are highly similar and correlated, it is often more convenient and advantageous to learn a single model that fits data in this individual cluster than data in the entire dataset. Therefore, it is desirable to exploit cluster-based information carried in data by devising a mixture-model learning method which can be aware of cluster structure inside data and train its individual models to fit data in the discovered clusters. Under the context of multiple kernel learning, this principle translates into the task of simultaneous discovering the cluster structure inside data and training single kernel machines to fit data in the discovered clusters. For instance, we illustrate a toy setting in Fig. 1 to show that learning involving the whole dataset is more complicated and requires more complex kernel functions than learning in each cluster separately. Specifically, it can be observed that if we can recognize the cluster structure inside data, a linear kernel function is sufficiently good to classify data in each cluster.

It is certain that to utilize the cluster-based information carried in the data, a simple and naive approach is to first run clustering method and then separately train single kernel machines according to the discovered clusters. This approach, however, suffers from two key limitations. Firstly, the clustering method cannot take advantage of the classifiers (i.e., the current label assignments) which are subsequently trained. Secondly, since the classifiers cannot interact with the clustering method, an unqualified clustering solution can severely affect the performance of these classifiers. We thus prefer a strategy that enables the interaction of the clustering method and the classifier. Particularly, the current cluster-assignment information from the clustering method would enhance the classifiers and the

current label-assignment information from the classifiers would boost the clustering method to converge faster and be more accurate.

To encourage cluster-based multiple kernel learning in a nonparametric setting wherein the cluster structure is automatically detected and the single kernel machines that fit data in the clusters are trained accordingly, we are in need of the clustering methods that have nonparametric flavor, that is, the number of clusters is automatically adapted along with the data growth. To this end, Dirichlet Process Mixture (DPM) model (Antoniak, 1974) and its variants (Neal, 2000; Kurihara et al., 2006), which are based on the sound foundations of random processes (e.g., Dirichlet process (Ferguson, 1973), Pitman-Yor process (Pitman and Yor, 1997)) and Bayesian inference paradigm, allow us to have the flexibility to learn the cluster structure and density information inside the data in a nonparametric manner. However, these aforementioned methods belong to be the unsupervised learning and hence cannot utilize the label information inside data. To utilize the strong nonparametric capability of Bayesian inference paradigm in supervised learning context, (Shahbaba and Neal, 2009) proposed using DPM to generate linear models and distribution parameters. In particular, the linear multinomial logit models and distribution parameters are shared accross the collected data instances due to the nature of DPM. The MCMC using Hamiltonian dynamics and slice sampling were then used to infer both linear models and density parameters. Since a mixture of linear models can sufficiently represent a nonlinear model, this method can capture non-linear corellations between data and lables. However the very high computational cost due to a slow convergence of its sampling scheme makes this method applicable to only small-scale datasets.

Putting it all together, we propose in this paper the *Clustering Induced Kernel Learning* (CIK) algorithm which simultaneously discovers the cluster structure inside data and trains single kernel machines to fit the data in the discovered clusters. Our proposed CIK leverages the well-known DPM model with multiple kernel learning paradigm and take advantage of the nonparametric ability of DPM to infer the number of single kernels that fits to the cluster structure of data. Not only the classifiers take advantage of the clustering method, but also the clustering method gets benefit from the current label assignments to converge faster. To enable kernel learning in each cluster, we employ the random feature technique (Rahimi and Recht, 2008) and further use the reparameterization technique to shift the source of randomness to a fixed distribution and expose the kernel parameters as learnable variables. It is worth noting that in our CIK the kernel parameters are automatically inferred without any prior specification (e.g., the list of kernels to inspect). In summary, our contributions include the following points:

- We formulate classification problem with learning multiple kernels and exploiting clustering structure simultaneously under a generative process which is encoded in a graphical model. Unlike most existing work, our approach is readily amendable to deal with different banks of kernels.
- Our method is fully nonparametric in the sense that the number of kernels is adapted automatically with the growth of data.
- We validate our proposed method on 6 benchmark datasets. The experimental results demonstrate that our proposed method has the ability of classification and clustering simultaneously. For classification, our method gains comparable to better accuracy than other state-of-the-art baselines. For clustering, our method can improve Dunn index and Davies–Bouldin index compared with the original DPM.

2. Related Background

This section presents important related background for our framework. For discussion of other related work to us, see Section Further Related Work.

2.1. Fourier Random Feature Representation

From Mercer's theorem (Mercer, 1909), if a kernel function is a positive definite kernel, then there exists a function $\boldsymbol{\phi}$ mapping $\mathbf{x} \in \mathbb{R}^d$ to $\mathbb{R}^{d_{\mathcal{H}}}$ where $d_{\mathcal{H}}$ is the dimension of the Hilbert space $\mathcal{H}$ induced by that kernel function. However, in many cases, $\mathcal{H}$ is an infinite-dimensional space and is only evaluated through the inner product $\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle = \kappa(\mathbf{x}, \mathbf{x}')$. To construct an explicit representation of $\boldsymbol{\phi}(\mathbf{x})$, (Rahimi and Recht, 2008) proposed a random finite-dimensional feature map $\widetilde{\boldsymbol{\phi}} : \mathbb{R}^d \to \mathbb{R}^{2D}$. For the case of Gaussian kernel $\kappa_k(\mathbf{x}, \mathbf{x}') = \exp\{-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^{\mathsf{T}} \operatorname{diag}(\boldsymbol{\sigma}_k)(\mathbf{x} - \mathbf{x}')\}$, it is of the form:

$$\widetilde{\boldsymbol{\phi}}(\mathbf{x}) = D^{-1/2} \left[\cos\left(\boldsymbol{\omega}_i^{\top}\mathbf{x}\right), \sin\left(\boldsymbol{\omega}_i^{\top}\mathbf{x}\right)\right]_{i=1}^{D} \tag{1}$$

Consequently, the induced kernel $\widetilde{\kappa}(\mathbf{x}, \mathbf{x}') = \widetilde{\boldsymbol{\phi}}(\mathbf{x})^{\top} \widetilde{\boldsymbol{\phi}}(\mathbf{x}')$ that can accurately and efficiently approximate the original kernel: $\widetilde{\kappa}(\mathbf{x}, \mathbf{x}') \approx \kappa(\mathbf{x}, \mathbf{x}')$.

2.2. Random Feature Reparameterization

Using random feature transformation, the feature map function $\boldsymbol{\phi}(\mathbf{x})$ can be accurately approximated by a mapping function $\widetilde{\boldsymbol{\phi}}(\mathbf{x})$ as described in Eq. (1). However, it is still not a deterministic function of kernel parameter $\boldsymbol{\sigma}$ since $\{\boldsymbol{\omega}_i\}_{d=1}^{D}$ are independent and identically distributed drawn from the distribution parameterized by the kernel parameter $\boldsymbol{\sigma}$. Thus, it is infeasible to learn these kernel parameters. To this end, (Nguyen et al., 2017) proposed a principle to learn kernel via a reparameterization trick. The main idea is to shift the source of randomness to an auxiliary space of noise variable $\boldsymbol{\epsilon} \in \mathbb{R}^d$ using the reparameterization trick as $\boldsymbol{\omega}_i = \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}_i$ where $\boldsymbol{\epsilon}_i \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{\epsilon} \mid \mathbf{0}, \boldsymbol{I}_d)$ and $\odot$ is element-wise multiplication operator. As a result, the gradient of the kernel parameters can analytically be derived, then we can incrementally learn these parameters via a proper optimizer.

2.3. Dirichlet Process Mixture Model

A notable approach to cluster data is to solve the problem of mixture model density estimation wherein we assume that data are generated from a mixture model and each data instance $\mathbf{x}$ is generated from a mixture component indexed by a hidden variable z.

To enable nonparametric formality, i.e. discovering the number of mixtures or clusters automatically, infinite mixture model is formed by using a Dirichlet Process (Ferguson, 1973; Rasmussen, 1999). With the Stick-breaking construction (Sethuraman, 1994), the generative process for the infinite mixture model can be represented as follows:

$$\begin{aligned} \boldsymbol{\pi} &\sim \text{GEM}(\alpha) \text{ and } \boldsymbol{\theta}_k \sim H(\boldsymbol{\theta} \mid \boldsymbol{\psi}) \\ z_n &\sim \text{Cat}(\boldsymbol{\pi}) \quad \text{and } \mathbf{x}_n \sim p(\mathbf{x}, \boldsymbol{\theta}_{z_n}) \end{aligned}$$

where $\text{GEM}(\alpha)$ (Ewens, 1990) stands for the *Griffiths-Engen-McCloskey* distribution. To infer the latent variables, we apply inference techniques, for example Gibbs sampling (Neal, 2000) or variational approximation (Kurihara et al., 2006).

3. Clustering Induced Kernel Learning

In this section, we present our proposed framework. We depart with the notions of feature map and space, then describe the generative process and graphical model, followed by the details of the model and kernel parameters inference.

3.1. The Joint Feature Map and Space

We begin with the description of the notions of feature map and space using in our approach. Let $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots$ where $\boldsymbol{\phi}_k : \mathcal{X} \to \mathcal{H}_k, k = 1, 2, \ldots$ be a sequence of feature maps whose incurred kernel functions are $\kappa_k(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{\phi}_k(\mathbf{x}), \boldsymbol{\phi}_k(\mathbf{x}') \rangle$, where $\mathcal{H}_k$(s) are the Reproducing Kernel Hilbert Spaces (RKHS). Define $\mathcal{H} = \text{stack}([\mathcal{H}_k]_k)$ as the joint feature space constituted by stacking the individual feature spaces $\mathcal{H}_k$, $k = 1, 2, ...$

Given a feature vector $\boldsymbol{\phi}_k(\mathbf{x})$, we now define its extended vector $\boldsymbol{\Phi}_k(\mathbf{x})$ in the joint feature space $\mathcal{H}$ by first mapping $\mathbf{x}$ to $\mathcal{H}_k$ using $\boldsymbol{\phi}_k$ and then padding $\boldsymbol{\phi}_k(\mathbf{x})$ with zeros as:

$$\boldsymbol{\Phi}_k^\mathsf{T}(\mathbf{x}) = \left[\mathbf{0}, \ldots, \boldsymbol{\phi}_k^\mathsf{T}(\mathbf{x}), \ldots, \mathbf{0}\right]$$

Given a data sample $\mathbf{x} \in \mathcal{X}$, we assume that there exists a latent variable $z \in \mathbb{N}^*$ indexing the individual feature space that the observation $\mathbf{x}$ should be mapped to. We further define: $\boldsymbol{\Phi}(\mathbf{x}; z) = \boldsymbol{\Phi}_z(\mathbf{x})$. It then follows that:

$$\mathbb{E}_z[\boldsymbol{\Phi}(\mathbf{x}; z)] = \sum_k p(z = k)\, \boldsymbol{\Phi}_z(\mathbf{x}) = [..., \varsigma_k \boldsymbol{\phi}_k^\mathsf{T}(\mathbf{x}), ...]^\mathsf{T}$$

where $\varsigma_k = p(z = k)$.

It turns out that the expectation of a joint feature vector $\boldsymbol{\Phi}_z(\mathbf{x})$ is in the form of multiple kernel representation involving the individual feature vectors $\boldsymbol{\phi}_k(\mathbf{x})$. Referring to the definition of the latent variable z, the task of multiple kernel learning now translates into inferring the latent variable z for each data instance $\mathbf{x}$. Driven by the motive idea that data instances in the same cluster are homogeneous, thus should be projected onto the same individual feature space. Consequently, we overload the latent variable z with the role of an indicator that specifies the cluster of the instance $\mathbf{x}$. This idea will be further explained in the next section.

To make computation efficient, we employ the Fourier random feature technique. Given a Gaussian kernel $\kappa_k(\mathbf{x}, \mathbf{x}') = \exp\{-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\mathsf{T} \text{diag}(\boldsymbol{\sigma}_k)(\mathbf{x} - \mathbf{x}')\}$, we can efficiently approximate the feature vector $\boldsymbol{\phi}_k(\mathbf{x})$ by a random feature vector $\widetilde{\boldsymbol{\phi}}_k(\mathbf{x}) = D_k^{-1/2}\left[\cos\left(\boldsymbol{\omega}_k^\mathsf{T}\mathbf{x}\right), \sin\left(\boldsymbol{\omega}_k^\mathsf{T}\mathbf{x}\right)\right] \in \mathbb{R}^{2D_k}$ where $\boldsymbol{\omega}_k = [\boldsymbol{\omega}_{k,1}, \ldots, \boldsymbol{\omega}_{k,D_k}]$ and $\boldsymbol{\omega}_{k,l} \sim \mathcal{N}(\boldsymbol{\omega} \mid \mathbf{0}, \text{diag}(\boldsymbol{\sigma}_k)) \in \mathbb{R}^d$. In particular, we can approximate $\boldsymbol{\Phi}_z(\mathbf{x})$ by a vector $\widetilde{\boldsymbol{\Phi}}_z(\mathbf{x})$ in $\mathbb{R}^{\widetilde{U}}$ where $\widetilde{U} = \sum_k 2D_k$ as follows

$$\boldsymbol{\Phi}_z^\mathsf{T}(\mathbf{x}) \approx \widetilde{\boldsymbol{\Phi}}_z^\mathsf{T}(\mathbf{x}) = \left[\mathbf{0}, \ldots, \widetilde{\boldsymbol{\phi}}_z^\mathsf{T}(\mathbf{x}), \ldots, \mathbf{0}\right]$$

In other words, $\widetilde{U}$ is an approximate space of the projection from $\mathbf{x}$ onto the joint feature space $\mathcal{H}$.

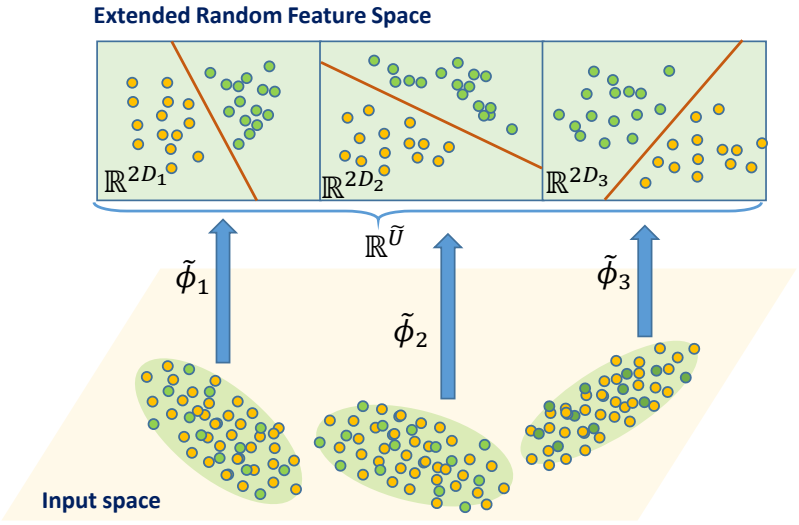
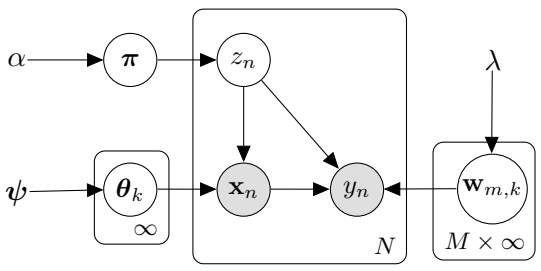


Figure 2: Illustration of our framework where clustering structure is discovered from the data in the input space and mapped into corresponding random subspace via Fourier random feature map ϕ_k. Each random subspace is sufficiently rich to perform classification.

3.2. Graphical Model and Generative Process

Given the training set $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, where $\mathbf{x}_n \in \mathcal{X} \subset \mathbb{R}^d$ is the data instance and $y_n \in \mathcal{Y} = \{1, \ldots, M\}$ is the corresponding label. Each data instance $\mathbf{x}_n$ is associated with the latent variable z_n that indexes the component model that generates $\mathbf{x}_n$. We then project $\mathbf{x}_n$ into the joint random feature space $\mathbb{R}^{\widetilde{U}}$ via $\widetilde{\mathbf{\Phi}}_{z_n}(\mathbf{x}_n)$ (i.e., projecting into the random subspace $\mathbb{R}^{2D_{z_n}}$ of $\mathbb{R}^{\widetilde{U}}$). We then aim to learn M hyperplanes $\mathbf{w}_1, \ldots, \mathbf{w}_M$ in the *approximate* joint feature space where $\mathbf{w}_m^{\mathsf{T}} = [\mathbf{w}_{m,1}^{\mathsf{T}}, \ldots, \mathbf{w}_{m,k}^{\mathsf{T}}, \ldots] \in \mathbb{R}^{\widetilde{U}}$ such that each $\mathbf{w}_m$, $m = 1, ..., M$ can well separate the data instances projected onto the subspace $\mathcal{H}_m$. Visually, we describe our idea in Fig. 2. The motive intuition behind this process is that the data instances generated by the same component model (or cluster) are homogeneous and can be conveniently characterized by a single kernel machine (i.e., a hyperplane in a component feature space). Besides, the information from the single kernel machines supports the clustering process and makes it converge faster. Hence, there is a symbiotic relationship between the single kernel machines and the clustering method (i.e., a DPM model in our case).



$$
\begin{aligned}
\boldsymbol{\pi} &\sim \mathrm{GEM}(\alpha) \\
z_n &\sim \mathrm{Cat}(\boldsymbol{\pi}) \\
\boldsymbol{\theta}_k &\sim H(\boldsymbol{\theta} \mid \boldsymbol{\psi}) \\
\mathbf{x}_n &\sim p(\mathbf{x} \mid \boldsymbol{\theta}_{z_n}) \\
\mathbf{w}_{m,k} &\sim \mathcal{N}(\mathbf{0}, \lambda \mathbf{I}_{2D_k}) \\
y_n &\sim Eq.(X)
\end{aligned}
$$

Figure 3: A graphical model representation for our CIK framework.

For convenience, we stack these hyperplanes into a matrix as $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_M]^\mathsf{T}$. We further denote $\mathbf{W}_{\bullet,k}$ as a $M \times 2D_k$ submatrix of $\mathbf{W}$ such that $\mathbf{W}_{\bullet,k} = [\mathbf{w}_{1,k}, \ldots, \mathbf{w}_{M,k}]^\mathsf{T}$.

The generative process of our model can be interpreted as follows:

- The latent variable z_n specifies the feature space to which the input vector $\mathbf{x}_n$ should map. Then, the label variable y_n is defined based on the weight matrix $\mathbf{W}$ and the extended random feature vector $\widetilde{\mathbf{\Phi}}_{z_n}(\mathbf{x}_n)$.

- Data instances with the same preferable feature map, i.e. the same $z_n = k$, are generated from a component model with the parameter $\boldsymbol{\theta}_k$. We endow the parameter $\boldsymbol{\theta}_k$ by a prior distribution H with a parameter $\boldsymbol{\psi}$. In particular, we define $p(\mathbf{x}_n \mid z_n = k, \boldsymbol{\theta}_k)$ as a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ and $H(\boldsymbol{\theta}_k \mid \boldsymbol{\psi})$ as a Gaussian invert Wishart distribution $\text{GIW}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \mid \mathbf{m}_0, \tau_0, \nu_0, \mathbf{S}_0)$.

- The latent variable z_n is an indicator, hence it should be generated from a categorical distribution with a parameters $\boldsymbol{\pi}$. The parameter $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_k, \ldots]$ is an infinite vector which is drawn from a GEM distribution with a concentration parameter α.

- The hyperplane $\mathbf{w}_{m,k}$ is drawn from a multivariate Gaussian distribution. By choosing this distribution, the maximum a posteriori (MAP) estimation for the posterior distribution of $\mathbf{W}_{\bullet,k}$ is reduced exactly to the optimization problem in MKL.

- For each data point $\mathbf{x}_n$ $(n = 1, \ldots, N)$, the corresponding output label y_n is drawn from pseudo likelihood described as follows

$$p(y_n \mid \mathbf{W}, \mathbf{x}_n, z_n) \propto \exp\left(-l(\mathbf{W}, \mathbf{x}_n, y_n, z_n)\right)$$

  and $l(\mathbf{W}; \mathbf{x}, y, z)$ is the loss function for classification. The loss function in multiclass case is defined as

$$l(\mathbf{W}; \mathbf{x}, y, z) = \max\left\{0, 1 + g_{y,z}(\mathbf{W}, \mathbf{x}) - \mathbf{w}_y^\mathsf{T} \widetilde{\mathbf{\Phi}}_z(\mathbf{x})\right\}$$

  where $g_{y,z}(\mathbf{W}, \mathbf{x}) = \max_{m \in \mathcal{Y} \setminus y}\left(\mathbf{w}_m^\mathsf{T} \widetilde{\mathbf{\Phi}}_z(\mathbf{x})\right)$.

Our generative process is summarized in a graphical model illustrated in Fig. 3. Based on this generative process and the graphical model, we further show how to make inference and learn parameters in the following section.

3.3. Model learning and parameters estimation

In this section, we present our solution to learn model and infer parameters via a MAP step with sampling. Our state space includes $\mathbf{W}$ and $\mathbf{z}$. We note that we will integrate out the parameter $\boldsymbol{\pi}$ because it is an infinite vector. We also integrate out parameters $\boldsymbol{\theta}_k$ to achieve the fast convergence for Gibbs sampling scheme. For convenience, we stack all data instances $\mathbf{x}_n$ into a matrix $\mathbf{X}$. We denote $\mathbf{X}_{-n}$ as a submatrix of $\mathbf{X}$ excluding the data instance $\mathbf{x}_n$. Similarly, $\mathbf{z}_{-n}$ is a vector of all z excluding z_n. The notation $\mathbf{X}^{(k)}$ specifies the submatrix of $\mathbf{X}$ constituted by concatenating all data instances that share the same latent variable $z = k$.

3.3.1. Sample $\mathbf{z}$

We sample every z_n via the posterior distribution of z_n given as follows:

$$\begin{aligned} p\left(z_n = k \mid \text{rest}\right) &= p(z_n = k \mid \mathbf{W}, \mathbf{X}^{(k)}, y_n, \mathbf{z}_{-n}, \boldsymbol{\psi}, \alpha) \\ &\propto p(y_n \mid \mathbf{W}, \mathbf{x}_n, z_n) \times p(z_n \mid \mathbf{z}_{-n}, \alpha) \times p(\mathbf{x}_n \mid \mathbf{X}_{-n}^{(k)}, \boldsymbol{\psi}) \end{aligned} \tag{2}$$

The first term $p\left(y_n \mid \mathbf{W}, \mathbf{x}_n, z_n\right)$ is introduced already in Section 3.2. The second term relates to the Chinese restaurant process (CRP) (Blackwell and MacQueen, 1973), thus being derived as follows:

$$\begin{aligned} p\left(z_n = k_{old} \mid \mathbf{z}_{-n}, \alpha\right) &= {}^{n_k}/_{\alpha+N-1} \\ p\left(z_n = k_{new} \mid \mathbf{z}_{-n}, \alpha\right) &= {}^{\alpha}/_{\alpha+N-1} \end{aligned}$$

where n_k is the number of data instances excluding $\mathbf{x}_n$ that have the same latent variable $z = k$. Using the conjugacy of the likelihood $\mathcal{N}\left(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$ and the prior GIW $\left(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \mid \mathbf{m}_0, \tau_0, \nu_0, \mathbf{S}_0\right)$ for $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, the third term can be derived as a multivariate Student's t-distribution

We note that the formulation to sample z_n differs from that in Dirichlet Process Mixture model in the first term. It shows that the assignment of a data instance to a cluster depends on not only the current number of data instances in that cluster (the second term), and how likely it is drawn by the generator of that cluster (the third term), but also how accurate it is classified by using the current kernel of that cluster.

3.3.2. Sample $\mathbf{W}$ and learn kernel parameters

We sample each $\mathbf{W}_{\bullet,k}$ from the posterior distribution of $\mathbf{W}_{\bullet,k}$ given as follows:

$$\begin{aligned} p\big(\mathbf{W}_{\bullet,k} \mid \text{rest}\big) &= p\big(\mathbf{W}_{\bullet,k} \mid \mathbf{X}^{(k)}, \mathbf{y}^{(k)}, \mathbf{z}^{(k)}, \lambda\big) \\ &\propto p\big(\mathbf{y}^{(k)} \mid \mathbf{W}_{\bullet,k}, \mathbf{X}^{(k)}, \mathbf{z}^{(k)}\big) \times p\big(\mathbf{W}_{\bullet,k} \mid \lambda\big) \end{aligned}$$

To sample W directly, we can utilize variable augmentation technique (Polson et al., 2011) to make posterior tractable. However, it requires to sample from a high dimensional space which is hard and takes long time to converge to an invariant distribution. Thus, to balance the predictive performance and training time, we used MAP to estimate model parameter $\mathbf{W}_{\bullet,k}$, as follows:

$$\mathbf{W}_{\bullet,k}^{*} = \underset{\mathbf{W}_{\bullet,k}}{\operatorname{argmax}} \left\{\log p\left(\mathbf{W}_{\bullet,k} \mid \text{rest}\right)\right\}$$

We then arrive at the following optimization problem:

$$\min_{\mathbf{W}_{\bullet,k}} \left(\frac{\lambda}{2} \left\|\mathbf{W}_{\bullet,k}\right\|_{2,2}^2 + \sum_{z_n = k} l\left(\mathbf{W}_{\bullet,k}; \mathbf{x}_n, y_n, z_n\right) \right)$$

It is similar to the formulation of the original SVM which is appealing to be solved efficiently in the primal form. However, our aim is to further learn kernel parameters automatically for each cluster without any predefined list of kernels. To this end, using the advantage of the reparameterization Fourier random feature (cf. Section 2.2), we can turn the kernel parameters $\boldsymbol{\sigma}_k$ into a variable of the optimization problem, and hence being able to efficiently solve this optimization by alternately updating $\mathbf{W}_{\bullet,k}$ and $\boldsymbol{\sigma}_k$. Recall that $\widetilde{\phi}_k\left(\mathbf{x}\right) = \left[\cos\left(\boldsymbol{\omega}_k^{\mathsf{T}}\mathbf{x}\right), \sin\left(\boldsymbol{\omega}_k^{\mathsf{T}}\mathbf{x}\right)\right]$ where $\boldsymbol{\omega}_k = [\boldsymbol{\omega}_{k,1}, \ldots, \boldsymbol{\omega}_{k,D_k}]$ and $\boldsymbol{\omega}_{k,l} \sim \mathcal{N}\left(\boldsymbol{\omega} \mid \mathbf{0}, \operatorname{diag}\left(\boldsymbol{\sigma}_k\right)\right) \in \mathbb{R}^d$. We note that $\boldsymbol{\sigma}_k$ is a parameter in the normal distribution $\mathcal{N}\left(\boldsymbol{\omega} \mid \mathbf{0}, \operatorname{diag}\left(\boldsymbol{\sigma}_k\right)\right)$, thus it is infeasible to find the optimal value via optimizing the

function w.r.t $\boldsymbol{\sigma}_k$. To resolve this problem, we use the reparameterization trick to shift this source of randomness to an auxiliary space of noise variable $\boldsymbol{\epsilon} \in \mathbb{R}^d$ as $\boldsymbol{\omega}_{k,l} = \boldsymbol{\sigma}_k \odot \boldsymbol{\epsilon}_{k,l}$ where $\boldsymbol{\epsilon}_{k,l} \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{\epsilon} \mid \mathbf{0}, \boldsymbol{I}_d)$ and $\odot$ is element-wise multiplication operator. The mapping $\boldsymbol{\omega}_{k,l}$ becomes a differentiable function w.r.t $\boldsymbol{\sigma}_k$, and hence allows us to find the optimal value of $\boldsymbol{\sigma}_k$ via the optimization. We have the objective function w.r.t $\mathbf{W}_{\bullet,k}$ and $\boldsymbol{\sigma}_k$ as follows

$$\mathcal{J}(\mathbf{W}_{\bullet,k}, \boldsymbol{\sigma}_k) = \frac{\lambda}{2} \|\mathbf{W}_{\bullet,k}\|_{2,2}^2 + \sum_{z_n=k} l(\mathbf{W}_{\bullet,k}; \mathbf{x}_n, y_n, z_n)$$

The update rules for $\mathbf{W}$ and $\boldsymbol{\sigma}_k$ for $k = 1, 2, \ldots, K$ are as:

$$\mathbf{w}_{m,k} = \mathbf{w}_{m,k} - \eta \nabla_{\mathbf{w}_{m,k}} \mathcal{J}_t(\mathbf{W}_{\bullet,k}, \boldsymbol{\sigma}_k)$$
$$\boldsymbol{\sigma}_k = \boldsymbol{\sigma}_k - \eta \nabla_{\boldsymbol{\sigma}_k} \mathcal{J}_t(\mathbf{W}_{\bullet,k}, \boldsymbol{\sigma}_k)$$

where η is a learning rate. $\mathcal{J}_t(\bullet)$ is the instantaneous objective function of $\mathcal{J}(\bullet)$ as follows

$$\mathcal{J}_t(\mathbf{W}_{\bullet,k}, \boldsymbol{\sigma}_k) = \frac{\lambda}{2} \|\mathbf{W}_{\bullet,k}\|_{2,2}^2 + l(\mathbf{W}_{\bullet,k}; \mathbf{x}_{n_t}, y_{n_t}, z_{n_t})$$

where n_t is uniformly sampled from $\{n \in 1, \ldots N \mid z_n = k\}$. For the sake of simplicity, we denote the representation of $\mathbf{x}_{n_t}$ in the random Fourier feature space parameterized by $\boldsymbol{\sigma}_{z_{n_t}}$ as $\widetilde{\boldsymbol{\phi}}_{n_t} = \widetilde{\boldsymbol{\phi}}_{z_{n_t}}(\mathbf{x}_{n_t})$. The derivatives of the objective function $\mathcal{J}_t(\bullet)$ w.r.t $\mathbf{w}_{m,k}$ and $\boldsymbol{\sigma}_k$ are as follows:

$$\nabla_{\mathbf{w}_{m,k}} \mathcal{J}_t(\mathbf{W}_{\bullet,k}, \boldsymbol{\sigma}_k) = \lambda \mathbf{w}_{m,k} + \mathbb{I}_{(m=m_t)\wedge(l_t>0)} \widetilde{\boldsymbol{\phi}}_{n_t} - \mathbb{I}_{(m=y_n)\wedge(l_t>0)} \widetilde{\boldsymbol{\phi}}_{n_t}$$
$$\nabla_{\boldsymbol{\sigma}_k} \mathcal{J}_t(\mathbf{W}_{\bullet,k}, \boldsymbol{\sigma}_k) = \nabla_{\boldsymbol{\sigma}_k} l(\mathbf{W}_{\bullet,k}; \mathbf{x}_{n_t}, y_{n_t}, z_{n_t}) = \frac{\partial l}{\partial \widetilde{\boldsymbol{\phi}}_{n_t}} \frac{\partial \widetilde{\boldsymbol{\phi}}_{n_t}}{\partial \boldsymbol{\omega}_k} \frac{\partial \boldsymbol{\omega}_k}{\partial \boldsymbol{\sigma}_k}$$

where we have

$$\frac{\partial l}{\partial \widetilde{\boldsymbol{\phi}}_{n_t}} = \mathbb{I}_{0<1+\mathbf{w}_{m_t,z_{n_t}}^\mathsf{T} \widetilde{\boldsymbol{\phi}}_{n_t} - \mathbf{w}_{y_{n_t},z_{n_t}}^\mathsf{T} \widetilde{\boldsymbol{\phi}}_{n_t}} \left(\mathbf{w}_{m_t,z_{n_t}}^\mathsf{T} - \mathbf{w}_{y_{n_t},z_{n_t}}^\mathsf{T}\right)$$
$$\frac{\partial \widetilde{\boldsymbol{\phi}}_{n_t}}{\partial \boldsymbol{\omega}_k} \frac{\partial \boldsymbol{\omega}_k}{\partial \boldsymbol{\sigma}_k} = D^{-1/2} \left[-\mathbf{x}_{n_t} \sin\left(\boldsymbol{\omega}_k^\mathsf{T} \mathbf{x}_{n_t}\right), \mathbf{x}_{n_t} \cos\left(\boldsymbol{\omega}_k^\mathsf{T} \mathbf{x}_{n_t}\right)\right]^\mathsf{T} \odot \boldsymbol{\epsilon}_k^\mathsf{T}$$
$$m_t = \max_{m \in \mathcal{Y} \setminus y_{n_t}} \left(\mathbf{w}_{m,z_{n_t}}^\mathsf{T} \widetilde{\boldsymbol{\phi}}_{n_t}\right) \text{ and } l_t = l(\mathbf{W}_{\bullet,k}; \mathbf{x}_{n_t}, y_{n_t}, z_{n_t}) \text{ and } \boldsymbol{\epsilon}_k = [\boldsymbol{\epsilon}_{k,1}, \ldots, \boldsymbol{\epsilon}_{k,D_k}]$$

To summarize, we present the pseudocode of our proposed method in Algorithm 1.

3.4. Predictive distribution

A new data instance $\mathbf{x}^*$ can be predicted by marginalizing out the posterior distributions of y^* and z^* as follows:

$$\begin{aligned} & p(y^* \mid \mathbf{x}^*, \mathbf{W}, \mathbf{X}, \mathbf{y}, \mathbf{z}, \alpha, \boldsymbol{\psi}) \\ = & \sum_k p\big(y^*, z^* = k \mid \mathbf{x}^*, \mathbf{W}, \mathbf{x}^{(k)}, \mathbf{y}^{(k)}, \mathbf{z}, \alpha, \boldsymbol{\psi}\big) \\ \propto & \sum_k p\big(y^* \mid \mathbf{W}, \mathbf{x}^*, z^*\big) \times p\big(\mathbf{x}^* \mid \mathbf{X}^{(k)}, \boldsymbol{\psi}\big) \times p\big(z^* \mid \mathbf{z}, \alpha\big) \end{aligned}$$

The classification decision of a new data instance depends on three terms. The first term is governed by the result of prediction given z^* and the two remaining terms present how well it belongs to that cluster.

Algorithm 1 Clustering Induced Kernel Learning

Input: $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}, \alpha, \lambda, \eta, \mathbf{m}_0, \tau_0, \nu_0, \mathbf{S}_0, T$
Output: $\mathbf{W}, \mathbf{z}$
1: $l \leftarrow 1$ and $\mathbf{W} = \mathbf{0}$
2: **repeat**
3: Sample $\mathbf{z}$ by Eq. 2
4: **for** $t = 1$ **to** T **do**
5: Update $\mathbf{w}_{m,k} = \mathbf{w}_{m,k} - \eta \nabla_{\mathbf{w}_{m,k}} \mathcal{J}\left(\mathbf{W}_{\bullet,k}, \boldsymbol{\sigma}_k\right)$
6: Update $\boldsymbol{\sigma}_k = \boldsymbol{\sigma}_k - \eta \nabla_{\boldsymbol{\sigma}_k} \mathcal{J}\left(\mathbf{W}_{\bullet,k}, \boldsymbol{\sigma}_k\right)$
7: **end for**
8: **until** enough l epochs

4. Experiments

In this section, we present a comprehensive evaluation on the performance of our proposed method CIK compared against other state-of-the-art baselines.

4.1. Synthetic Data

In the first experiment, we validate and investigate the behaviors of our approach on synthetic dataset. To generate dataset, we assume each class contains multiple clusters. The generative process is as follows. We first sample $\boldsymbol{\mu}_{1:K}$ from a mixture of K' different Gaussian distributions with the proportion $\boldsymbol{\pi}'$. We then randomly select $\boldsymbol{\Sigma}_{1:K}$ corresponding with $\boldsymbol{\mu}_{1:K}$. Next, we generate cluster assignments $z_{1:N}$ from categorical distribution with the proportion $\boldsymbol{\pi}$. For each assignment z_n, we sample $\mathbf{x}_n$ from the corresponding Gaussian distribution $\mathcal{N}\left(\boldsymbol{\mu}_{z_n}, \boldsymbol{\Sigma}_{z_n}\right)$. Finally, we label data instances based on which Gaussian distribution they are samples from, i.e. two data instances sampled from the same Gaussian distribution have the same class.

More specifically, we choose $K' = 3, \boldsymbol{\pi}' = (1/3, 1/3, 1/3)$ and $K = 10, \boldsymbol{\pi} = (1/10, \ldots, 1/10)$. For clustering task, our method utilizes the intermediate clustering result with the classification decision boundary, thus it can discover the intrinsic clustering structure in dataset. To examine the clustering capability, which distinguishes our proposed model from traditional MKL methods, we compare our method with DPM - Dirichlet Process Mixture and MKC - multiple kernel clustering approach (Gönen and Margolin, 2014). We use three popular measures: normalized mutual information (NMI), Rand index (RI) and purity (PU) to evaluate clustering results. As shown in Table 1, our method outperforms other baselines in all measures. It indicates that the classification and kernel learning information improve the performance on clustering, hence CIK exploits a better clustering structure in data.

	CIK	**DPM**	**MKC**
NMI	**0.84±0.05**	0.57±0.05	0.55
RI	**0.93±0.02**	0.62±0.09	0.60
PU	**0.56±0.12**	0.28±0.06	0.20

Table 1: Clustering results comparison on synthetic dataset

4.2. Real datasets

We now examine the classification and clustering performance of our proposed method on real-world datasets. We use 6 datasets from a variety of domains. All datasets and their statistics are available at the LIBSVM Repository. We denote the highest value per dataset is in bold and methods in bold text with an asterisk are not statistically different from the highest one using a paired t-test with p-value < 0.05.

4.2.1. Classification task

We employ 3 state-of-the-art baselines as follows:

- UFO-MKL (Orabona and Jie, 2011): a scalable MKL method in which the optimization problem is efficiently solved using the SGD framework. However, it requires a hyperparameter tuning procedure.
- BEMKL (Gonen, 2012): a MKL approach under probabilistic perspective which was proven more efficient and faster than previous works, but its computational complexity still grows cubically with the training size and the number of kernels.
- BaNK (Oliva et al., 2016): Unlike UFO-MKL and BEMKL, it does not require a predefined list of kernels. The idea starts from the random feature space approach (Rahimi and Recht, 2008). This random feature vector is constructed by random frequencies drawn from a certain distribution. Instead of keeping this distribution fixed, BaNK considers it as a mixture of Gaussian distribution following a Dirichlet Process prior distribution. This enables BaNK to have a list of kernels that can grow freely to be adapted from the data. However, BaNK lacks the clustering capability since the model fails to capture the geometry information of the data.

Their implementations are obtained from the corresponding authors. All implementations, including ours, are in Matlab. Here we would like to note that BaNK does not support multiclass classification. For BEMKL, we set hyperparameter $(\alpha_\lambda, \beta_\lambda, \alpha_\gamma, \beta_\gamma, \alpha_\omega, \beta_\omega) = (1, 1, 1, 1, 1, 1)$ as suggested in the original paper (Gonen, 2012). We perform grid search and 5-fold cross-validation to select the best value for regularization parameter λ (UFO-MKL, BaNK, and CIK) and the sparsity tuning parameter μ (UFO-MKL). The considered range for λ is $\{2^{-15}, \ldots, 2^3, 2^5\}$. The parameter μ is selected in $\{0.001, \ldots, 0.02\}$ as suggested in (Orabona and Jie, 2011). For both BaNK and CIK, we set the same dimension in random feature space $D = 384$. We also repeat 5 times and record the corresponding mean value.

The classification results and training time are reported in Table 2 and 3 respectively. UFO-MKL runs faster than others but it will be slowed down quickly when the training size grows as seen in the *susy* dataset. It is because the computation cost is $\mathcal{O}(dN^2)$. For BEMKL, it takes $\mathcal{O}(N^3 + F^3 + N^2F^2)$ running time for matrix inversion where N is the size of training set and F is the number of kernels. In addition, BEMKL requires a large amount of memory for loading kernel matrix, rendering it inapplicable for large-scale datasets such as *cod-rna* and *susy* as denoted N/A. For BaNK, the computation cost is $\mathcal{O}(dND^2)$ where d is the dimension of the input space and D is the dimension of the random feature space which usually larger than d. In the meantime, the complexity of our method is $\mathcal{O}(dNKD)$ which is faster than BaNK and BEMKL.

In terms of predictive performance, UFO-MKL is quite worse than others, especially on *pendigits*, *phishing*, and *susy* datasets. The accuracy of BaNK is slightly better than BEMKL and UFO-MKL. It shows the limitation of using a predefined list of kernels in BEMKL and UFO-MKL as we mentioned in Section 1. However, BaNK's approach does not support multiclass datasets which limits its application in the real world. In contrast, our method can deal with multiclass problem. Furthermore, CIK outperforms others on *pendigits* and *phishing* datasets and obtains comparable accuracy on the remaining datasets. It is because our method is capable of exploiting the clustering structure and learning kernels for individual clusters. This approach makes our method more robust than the others, especially when different clusters in the dataset prefer different kernels.

4.2.2. Clustering Comparison

In the last experiment, we examine the clustering capability, which distinguishes our proposed model from traditional MKL methods. For evaluation, we use two measures: Davies–Bouldin index (DBI) (Davies and Bouldin, 1979) and Dunn index (DI) (Dunn, 1974). DBI is a ratio between low intra-cluster distances and inter-cluster distances. Intuitively, a clustering method is better than others when it produces clusters with low intra-cluster distances and high inter-cluster distances. Thus, the smaller DBI value indicates better performance. In contrast, Dunn index is desired to be high. The essential idea is is to give high score for clustering structure that have small variance between members in a cluster and different clusters are sufficiently far apart.

We compare our method with Dirichlet Process Mixture Model (DPM) and Multiple Kernel Clustering approach (MKC) (Gönen and Margolin, 2014). In MKC, multiple kernel is used to obtain multiple view of data which utilizes the clustering performance. Although MKC is a promising method for clustering, MKC has high demand on memory to deal with multiple kernel matrix computation, thus it is suitable only for small datasets. In our experiments, it cannot run *cod-rna* and *susy* datasets as we denoted O/M (out of memory). For datasets in which results in one cluster (such as *svmguide1* in MKC; *cod-rna* and *susy* datasets in CIK and DPM), we denote N/A in the table since DBI and DI are unspecified when evaluated on a single cluster.

As shown in Table 4, our CIK outperforms DPM in terms of DI score in most of datasets, except for *mushrooms* dataset where the results are comparable. In terms of DBI, our CIK achieves better results than DPM in *svmguide1* and *pendigits* datasets. Comparing with MKC, our methods obtains better results on *pendigits* dataset in both of DBI and DI measures while MKC is a suitable choice for *mushroom* dataset. In *phishing* dataset, DBI

Datasets	CIK	BaNK	BEMKL	UFO-MKL
svmguide1	**96.14±0.64***	**96.85±0.09**	95.66±0.01	**96.72±0.04***
mushrooms	**99.94±0.06***	**100**	**100**	**100**
pendigits	**97.54±0.26**	N/A	**97.14±0.02***	96.37±0.40
phishing	**99.97±0.02**	96.67±0.41	96.52±0.01	94.71±0.25
cod-rna	94.28±0.27	**96.54±0.03**	O/M	95.92±0.32
susy	78.96±0.54	**80.17±0.05**	O/M	74.41±0.13

Table 2: Accuracy (in %) comparison

Datasets	CIK	BaNK	BEMKL	UFO-MKL
svmguide1	291±15	377±152	128±002	2.3±01
mushrooms	611±09	856±027	756±008	38±01
pendigits	1,576±31	N/A	3,615±242	16±01
phishing	650±18	832±045	2,001±114	94±04
cod-rna	2,908±75	5,965±731	O/M	902±21
susy	4,379 ±91	7,719±846	O/M	4,089±91

Table 3: Training time (in seconds) comparison

Datasets	Davies–Bouldin index			Dunn index (×0.1)		
	CIK	DPM	MKC	CIK	DPM	MKC
svmguide1	**1.19±0.01**	1.57±0.04	N/A	**0.03±0.009**	0.02±0.003	N/A
mushrooms	3.67±0.65	3.00±0.50	**2.29**	2.38±0.032	**2.43±0.001**	**2.43**
pendigits	**2.27±0.66**	2.56±0.34	14.71	**0.36±0.080**	**0.33±0.078***	0.19
phishing	4.09±0.93	3.13±0.29	**2.83**	**1.93±0.017**	**1.93±0.024**	0.95
cod-rna	N/A	N/A	O/M	N/A	N/A	O/M
susy	N/A	N/A	O/M	N/A	N/A	O/M

Table 4: Clustering Index comparison on real datasets

indicates MKC is the best one, but, in terms of DI, our methods and DPM have better clustering result than MKC. In general, our method exploits a better clustering structure in data and improve the performance in clustering comparing with DPM and MKC in most of datasets.

5. Further Related Work

Multiple kernel learning (MKL) has been studied intensively due to its advantages over single kernel learning (Gönen and Alpaydın, 2011). (Cristianini et al., 2001) and (Crammer et al., 2002) have placed the first building block for MKL. There are two main approaches to view MKL: as an optimization problem and under Bayesian setting.

In the first approach, many types of optimization problems are formulated and different solutions are proposed to solve such optimization problems (Bach et al., 2004; Xu et al., 2009). A common strategy in these methods is to base on an alternating optimization view consisting of two steps: i) updating the combination of kernels given a current learner-based solution and ii) finding the best learner-based solution for a fixed combination. However, there is no guarantee of the convergence of this method since the number of iterations cannot be bounded. Later, (Sun et al., 2010) proposed a method based on the sequential minimal optimization whose solution is obtained directly without solving intermediate support vector machines (SVMs). Recently, the so-called UFO-MKL method (Orabona and Jie, 2011) efficiently solved the optimization problem directly in the primal form by utilizing the primal-dual framework to minimize the regularized loss functions (Shalev-Shwartz and Kakade, 2009).

The second approach is to view MKL under probabilistic perspective. Parameters of models in this approach are learned by inferring latent variables via Bayesian inference.

Some probabilistic techniques, such as hierarchical probabilistic model (Damoulas and Girolami, 2009), Gaussian Process (Girolami and Zhong, 2007), Markov chain Monte Carlo (Zhang et al., 2011), Data Augmentation (Nguyen et al., 2016), have been integrated into MKL. Among these methods, BEMKL (Gonen, 2012) was proven more efficient and faster than previous work, but its computational complexity still grows cubically with the training size and the number of kernels. It is worthy of noting that Bayesian models proposed in these methods are parametric. These models, however, require a list of kernels that must be predefined, leading to the lack of promising kernel candidates and the demand of high computational complexity. To the best of our knowledge, BaNK (Oliva et al., 2016) is the only nonparametric method in this line. The idea starts from the random feature space approach (Rahimi and Recht, 2008), where the feature vector $\boldsymbol{\phi}(\mathbf{x})$ is approximated by a random feature vector $\widetilde{\boldsymbol{\phi}}(\mathbf{x})$. This random feature vector is constructed by random frequencies drawn from a certain distribution. Instead of keeping this distribution fixed, BaNK considers it as a mixture of Gaussian distribution following a Dirichlet Process prior distribution. This enables BaNK to have a list of kernels that can grow freely to be adapted from the data. However, this method requires applying Metropolis-Hastings sampling scheme whose convergence rate is slower than Gibbs sampling because it does not always accept the current sample value. Another disadvantage of BaNK is the lack of clustering capability since the model fails to capture the geometry information of the data.

To summarize, combining multiple kernels in the aforementioned ways allows MKL to cover almost all types of mapping functions for data. However, real-world data are heterogeneous and complicated since they are collected from multiple sources. They have their own clustering structure and different clusters do not always share a common preferable kernel. Therefore, a universal mapping function cannot be a good option for all clusters. This motivates us to exploit a specific clustering structure for kernel learning in which data instances having the same preferable kernel are grouped into the same cluster.

6. Conclusion

In this paper, we translate multiple kernel learning into the task of exploring clustering structure in data and learning a single kernel machine for each discovered cluster. This approach has two advantages. First, it makes kernel simpler, hence it is easier to learn kernel. Second, both classification and clustering performance are improved because they can make use of the intermediate information of each other during the training process. We evaluate our method on both synthetic and benchmark datasets. For classification, our method gains comparable to better accuracy than other state-of-the-art baselines. For clustering, our method improves clustering quality compared with the original DPM.

References

Charles E Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The annals of statistics*, pages 1152–1174, 1974.

Francis R Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in neural information processing systems*, pages 105–112, 2009.

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.

David Blackwell and James B MacQueen. Ferguson distributions via pólya urn schemes. *The annals of statistics*, pages 353–355, 1973.

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.

David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

Joseph C Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, 4(1):95–104, 1974.

Warren John Ewens. Population genetics theory-the past and the future. In *Mathematical and statistical developments of evolutionary theory*, pages 177–227. Springer, 1990.

Thomas S Ferguson. A Bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230, 1973.

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. *arXiv preprint arXiv:1206.6465*, 2012.

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

Mehmet Gönen and Adam A Margolin. Localized data fusion for kernel k-means clustering with application to cancer biology. In *Advances in Neural Information Processing Systems*, pages 1305–1313, 2014.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. 2003.

Kenichi Kurihara, Max Welling, Nikos Vlassis, et al. Accelerated variational Dirichlet process mixtures. In *NIPS*, volume 6, pages 761–768, 2006.

James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209:415–446, 1909.

Radford M Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

Khanh Nguyen, Trung Le, Vu Nguyen, Tu Nguyen, and Dinh Phung. Multiple kernel learning with data augmentation. In *Asian Conference on Machine Learning*, pages 49–64, 2016.

Tu Dinh Nguyen, Trung Le, Hung Bui, and Dinh Phung. Large-scale online kernel learning with random feature reparameterization. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, IJCAI'16, 2017.

Junier B Oliva, Avinava Dubey, Barnabas Poczos, Jeff Schneider, and Eric P Xing. Bayesian nonparametric kernel-learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 1078–1086, 2016.

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.

Jim Pitman and Marc Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, pages 855–900, 1997.

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

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

Carl Edward Rasmussen. The infinite gaussian mixture model. In *NIPS*, volume 12, pages 554–560, 1999.

Jayaram Sethuraman. A constructive definition of Dirichlet priors. *Statistica sinica*, pages 639–650, 1994.

Babak Shahbaba and Radford Neal. Nonlinear models using Dirichlet process mixtures. *Journal of Machine Learning Research*, 10(Aug):1829–1850, 2009.

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.

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.

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.