ORCCA: Optimal Randomized Canonical Correlation Analysis

Yinsong Wang and Shahin Shahrampour, *Senior Member, IEEE*

***Abstract*— Random features approach has been widely used for kernel approximation in large-scale machine learning. A number of recent studies have explored data-dependent sampling of features, modifying the stochastic oracle from which random features are sampled. While proposed techniques in this realm improve the approximation, their suitability is often verified on a single learning task. In this article, we propose a task-specific scoring rule for selecting random features, which can be employed for different applications with some adjustments. We restrict our attention to canonical correlation analysis (CCA) and provide a novel, principled guide for finding the score function maximizing the canonical correlations. We prove that this method, called optimal randomized CCA (ORCCA), can outperform (in expectation) the corresponding kernel CCA with a default kernel. Numerical experiments verify that ORCCA is significantly superior to other approximation techniques in the CCA task.**

***Index Terms*— Canonical correlation analysis (CCA), kernel approximation, kernel methods, random features.**

I. Introduction

KERNEL methods are powerful tools to capture the nonlinear representation of data by mapping the dataset to a high-dimensional feature space. Despite their tremendous success in various machine learning problems, kernel methods suffer from massive computational cost on large datasets. The time cost of computing the kernel matrix alone scales quadratically with data, and if the learning method involves inverting the matrix (e.g., kernel ridge regression), the cost would increase to cubic. This computational bottleneck motivated a great deal of research on kernel approximation, where the seminal work of [1] on random features is a prominent point in case. For the class of shift-invariant kernels, they showed that one can approximate the kernel by the Monte Carlo sampling from the inverse Fourier transform of the kernel. This idea has been used in solving many machine learning problems, including distributed learning [2], [3], online learning [4], [5], and deep learning [6].

Due to the practical success of random features, the idea was later used for one of the ubiquitous problems in statistics and machine learning, namely, canonical correlation analysis (CCA). CCA derives a pair of linear mappings of two datasets such that the correlation between the projected datasets is maximized. Similar to other machine learning methods, CCA also has a nonlinear counterpart called kernel CCA (KCCA) [7], which provides a more flexible framework for maximizing the correlation. Due to the prohibitive computational cost of KCCA, randomized CCA (RCCA) was introduced [8], [9] to serve as a surrogate for KCCA. RCCA uses random features for transformation of the two datasets. Therefore, it provides the flexibility of nonlinear mappings with a moderate computational cost.

On the other hand, more recently, data-dependent sampling of random features has been an intense focus of research in the machine learning community. The main objective is to modify the stochastic oracle from which random features are sampled to improve a certain performance metric. Examples include [10], [11] with a focus only on kernel approximation as well as [12]–[14] with the goal of better generalization in supervised learning. While the proposed techniques in this realm improve their respective learning tasks, they are not necessarily suitable for other learning tasks, such as CCA, which is the focus of this work.

In this article, we propose a task-specific scoring rule for reweighting random features, which can be employed for various applications with some adjustments. In particular, our scoring rule depends on a matrix that can be adjusted based on the application. We first observe that a number of data-dependent sampling methods (e.g., leverage scores (LSs) in [13] and energy-based sampling in [15]) can be recovered by our scoring rule using specific choices of the matrix. Then, we draw a connection between the scoring rule and correlation analysis/dimension reduction problems that deal with trace optimization objectives. As an important case study, we focus on CCA and provide a principled guide for finding the score function maximizing the canonical correlations. Our result reveals a novel data-dependent method for selecting features, called optimal randomized CCA (ORCCA). This suggests that prior data-dependent methods are not necessarily optimal for the CCA task. We also prove that ORCCA achieves a better performance compared to KCCA (in expectation) with a default kernel. We conduct extensive numerical experiments verifying that ORCCA indeed introduces significant improvement over the state of the art in random features for CCA.

The rest of this article is organized as follows. In Section II, we provide the preliminaries on random features, CCA, and formally define the problem we will address in this article.

Manuscript received December 16, 2020; revised July 16, 2021; accepted October 22, 2021. This work was supported by the NSF Award #2038625 as part of the NSF/DHS/DOT/NIH/USDA-NIFA Cyber-Physical Systems Program, as well as Texas A&M Triads for Transformation (T3) Program. *(Corresponding author: Shahin Shahrampour.)*

Yinsong Wang and Shahin Shahrampour are with the Department of Mechanical and Industrial Engineering, Northeastern University, Boston, MA 02115 USA (e-mail: wang.yinso@northeastern.edu; s.shahrampour@northeastern.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3124868.

Digital Object Identifier 10.1109/TNNLS.2021.3124868

This section also includes the related literature. In Section III, we propose our score function, discuss its connection with existing score functions in supervised learning, and show a class of problems that is compatible with our score function. In Section IV, we present our theoretical results. We illustrate the effectiveness of ORCCA on benchmark datasets in Section V and conclude in Section VI.

II. Preliminaries and Problem Setting

Notation: We denote by $[n]$ the set of positive integers $\{1, \ldots, n\}$, by $\mathrm{Tr}[\cdot]$ the trace operator, by $\langle \cdot, \cdot \rangle$ the standard inner product, by $\|\cdot\|$ the spectral (respectively, Euclidean) norm of a matrix (respectively, vector), and by $\mathbb{E}[\cdot]$ the expectation operator. Boldface lowercase variables (e.g., $\mathbf{a}$) are used for vectors, and boldface uppercase variables (e.g., $\mathbf{A}$) are used for matrices. $[\mathbf{A}]_{ij}$ denotes the ijth entry of matrix $\mathbf{A}$. The vectors are all in column form.

A. Random Features and Kernel Approximation

Kernel methods are powerful tools for data representation, commonly used in various machine learning problems. Let $\{\mathbf{x}_i\}_{i=1}^n$ be a set of given points where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ for any $i \in [n]$, and consider a symmetric positive-definite function $k(\cdot, \cdot)$ such that $\sum_{i,j=1}^n \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ for $\boldsymbol{\alpha} \in \mathbb{R}^n$. Then, $k(\cdot, \cdot)$ is called a positive (semi)definite kernel, serving as a similarity measure between any pair of vectors $(\mathbf{x}_i, \mathbf{x}_j)$. This class of kernels can be thought as inner product of two vectors that map the points from a d_x-dimensional space to a higher dimensional space (and potentially infinite-dimensional space).

Despite the widespread use of kernel methods in machine learning, they have an evident computational issue. Computing the kernel for every pair of points costs $O(n^2)$, and if the learning method requires inverting that matrix (e.g., kernel ridge regression), the cost would increase to $O(n^3)$. This particular disadvantage makes the kernel method impractical for large-scale machine learning.

An elegant method to address this issue was the use of random Fourier features (RFFs) for kernel approximation [1]. Let $p(\boldsymbol{\omega})$ be a probability density with support $\Omega \subseteq \mathbb{R}^{d_x}$. Consider any kernel function in the following form with a corresponding feature map $\phi(\mathbf{x}, \boldsymbol{\omega})$ such that:

$$k(\mathbf{x}, \mathbf{x}') = \int_\Omega \phi(\mathbf{x}, \boldsymbol{\omega}) \phi(\mathbf{x}', \boldsymbol{\omega}) p(\boldsymbol{\omega}) d\boldsymbol{\omega} \approx \frac{1}{M} \sum_{m=1}^M \phi(\mathbf{x}, \boldsymbol{\omega}_m) \phi(\mathbf{x}', \boldsymbol{\omega}_m) \tag{1}$$

where $\{\boldsymbol{\omega}_m\}_{m=1}^M$ are independent samples from $p(\boldsymbol{\omega})$, called random features. Examples of kernels taking the form (1) include shift-invariant kernels [1] or dot-product (e.g., polynomial) kernels [16] (see [17, Table I] for an exhaustive list). Let us now define

$$\mathbf{z}(\boldsymbol{\omega}) \triangleq [\phi(\mathbf{x}_1, \boldsymbol{\omega}), \ldots, \phi(\mathbf{x}_n, \boldsymbol{\omega})]^\top. \tag{2}$$

Then, the kernel matrix $[\mathbf{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ can be approximated with $\mathbf{Z}\mathbf{Z}^\top$ where $\mathbf{Z} \in \mathbb{R}^{n \times M}$ is defined as

$$\mathbf{Z} \triangleq \frac{1}{\sqrt{M}} [\mathbf{z}(\boldsymbol{\omega}_1), \ldots, \mathbf{z}(\boldsymbol{\omega}_M)]. \tag{3}$$

The low-rank approximation above can save significant computational cost when $M \ll n$. As an example, for kernel ridge regression, the time cost would reduce from $O(n^3)$ to $O(nM^2)$. Since the main motivation of using randomized features is to reduce the computational cost of kernel methods (with $M \ll n$), this observation will naturally raise the following question:

Problem 1 (Informal): Can we develop a sampling (or selection) mechanism for random features that takes into account the "learning task" to improve the performance compared to plain sampling?

Section III sheds light on Problem 1. First, in Section III-A, we will propose a score function with a potential to be adapted to different learning tasks. Section III-B shows the connection between the proposed score function and two existing score functions for random features in supervised learning. Then, Section III-C provides another class of problems (i.e., dimensionality reduction/correlation analysis) for which the proposed score function can prove useful. In this article, we focus on kernel CCA (as one potential application), introduce our main question in Problem 2, and provide our theoretical results.

B. Overview of Canonical Correlation Analysis

Linear CCA was introduced in [18] as a method of correlating linear relationships between two multidimensional random variables $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d_x}$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^\top \in \mathbb{R}^{n \times d_y}$. This problem is often formulated as finding a pair of canonical bases $\boldsymbol{\Pi}_x$ and $\boldsymbol{\Pi}_y$ such that $\|\mathrm{corr}(\mathbf{X}\boldsymbol{\Pi}_x, \mathbf{Y}\boldsymbol{\Pi}_y) - \mathbf{I}_r\|_F$ is minimized, where $r = \max(\mathrm{rank}(\mathbf{X}), \mathrm{rank}(\mathbf{Y}))$ and $\|\cdot\|_F$ is the Frobenius norm.

The problem has a well-known closed-form solution (see, e.g., [19]), relating canonical correlations and canonical pairs to the eigensystem of the following matrix:

$$\begin{bmatrix} (\boldsymbol{\Sigma}_{xx} + \mu_x \mathbf{I})^{-1} & \mathbf{0} \\ \mathbf{0} & (\boldsymbol{\Sigma}_{yy} + \mu_y \mathbf{I})^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \mathbf{0} \end{bmatrix} \tag{4}$$

where $\boldsymbol{\Sigma}_{xx} = \mathbf{X}^\top \mathbf{X}$, $\boldsymbol{\Sigma}_{yy} = \mathbf{Y}^\top \mathbf{Y}$, $\boldsymbol{\Sigma}_{xy} = \mathbf{X}^\top \mathbf{Y}$, and μ_x, μ_y are regularization parameters to avoid singularity. In particular, the eigenvalues correspond to the canonical correlations and the eigenvectors correspond to the canonical pairs.

The kernel version of CCA, called KCCA [7], [20], investigates the correlation analysis using the eigensystem of the following matrix:

$$\begin{bmatrix} (\mathbf{K}_x + \mu_x \mathbf{I})^{-1} & \mathbf{0} \\ \mathbf{0} & (\mathbf{K}_y + \mu_y \mathbf{I})^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{K}_y \\ \mathbf{K}_x & \mathbf{0} \end{bmatrix} \tag{5}$$

where $[\mathbf{K}_x]_{ij} = k_x(\mathbf{x}_i, \mathbf{x}_j)$ and $[\mathbf{K}_y]_{ij} = k_y(\mathbf{y}_i, \mathbf{y}_j)$.

As the inversion of kernel matrices involves $O(n^3)$ time cost, Lopez-Paz *et al.* [9] adopted the idea of kernel approximation with random features, introducing randomized CCA (RCCA). RCCA uses approximations $\mathbf{K}_x \approx \mathbf{Z}_x \mathbf{Z}_x^\top$ and $\mathbf{K}_y \approx \mathbf{Z}_y \mathbf{Z}_y^\top$ in (5), where $\mathbf{Z}_x$ and $\mathbf{Z}_y$ are the transformed matrices using random features as in (3). In other words, $\mathrm{RCCA}(\mathbf{X}, \mathbf{Y}) = \mathrm{CCA}(\mathbf{Z}_x, \mathbf{Z}_y) \approx \mathrm{KCCA}(\mathbf{X}, \mathbf{Y})$. Now, the question we would like to answer in this article is as follows.

Problem 2: If we want to maximize the total canonical correlations, i.e., the trace of matrix (5), what is the corresponding score function in the form of (6) to reweight (or select) the random features?

Note that the term "maximize" makes sense here due to the approximation using random features. In other words, we are interested in finding the features providing more correlation (or maximize the correlation) between $\mathbf{X}$ and $\mathbf{Y}$. In the next section, we will derive the desired score function and show its performance advantage in theory. We will see that the features that we select based on the score function change the kernel in a way that improves the correlation.

C. Related Literature

1) Random Features: As discussed in Section II-A, kernels of form (1) can be approximated using random features (e.g., shift-invariant kernels using Monte Carlo [1] or quasi Monte Carlo [21] sampling and dot-product kernels [16]). A number of methods have been proposed to improve the time cost, decreasing it by a linear factor of the input dimension (see, e.g., fast food [10], [22]). The generalization properties of random features have been studied for ℓ_1-regularized risk minimization [23] and ridge regression [24], both improving the early generalization bound of [25]. Also, Felix *et al.* [26] developed orthogonal random features (ORFs) to improve kernel approximation variance. It turns out that ORF provides an optimal kernel estimator in terms of mean-squared error [27]. May *et al.* [28] presented an iterative gradient method for selecting the best set of random features for supervised learning with acoustic model. A number of recent works have focused on kernel approximation techniques based on data-dependent sampling of random features. Examples include [29] on compact nonlinear feature maps, [10], [30] on approximation of shift-invariant/translation-invariant kernels, [11] on the Stein effect in kernel approximation, and [31] on data-dependent approximation using greedy approaches (e.g., Frank–Wolfe). On the other hand, another line of research has focused on generalization properties of data-dependent sampling. In addition to works mentioned in Section III-B, the work [14] also studied data-dependent approximation of translation-invariant/rotation-invariant kernels for improving generalization in SVM. Li *et al.* [32] recently proposed a hybrid approach (based on importance sampling) to reweight random features with application to both kernel approximation and supervised learning.

2) Canonical Correlation Analysis: As discussed in Section II-B, the computational cost of KCCA [7] motivated a great deal of research on kernel approximation for CCA in large-scale learning. Several methods tackle this issue by explicitly transforming datasets (e.g., randomized CCA (RCCA) [8], [9], fix-sized kernel CCA (FSCCA) [33], and deep CCA (DCCA) [34]). RCCA and FSCCA tackle the computation issue of KCCA with two well-known kernel approximation methods, random features and Nystrom method, respectively. RCCA focuses on transformation using randomized one-hidden layer neural network, whereas DCCA considers deep neural networks. Perhaps not surprisingly, the time cost of RCCA is significantly smaller than DCCA [9]. There exist other nonparametric approaches such as nonparametric CCA (NCCA) [35], which estimates the density of training data to provide a practical solution to Lancaster's theory for CCA [36]. Also, more recently, a method is proposed in [37] for sparsifying KCCA through the ℓ_1 regularization. A different (but relevant) literature has focused on addressing the optimization problem in CCA. Wang *et al.* [38] and Arora *et al.* [39] [38], [39] discussed this problem by developing novel techniques, such as alternating least squares, shift-and-invert preconditioning, and inexact matrix stochastic gradient. In a similar spirit is [40], which presents a memory-efficient stochastic optimization algorithm for RCCA.

The main novelty of our approach is proposing an optimized scoring rule for random features selection, which can be adopted for different learning tasks, including various correlation analysis techniques, e.g., CCA.

III. Objective-Based Score Function

A. Task-Specific Scoring Rule for Random Features Selection

Several recent works have answered to Problem 1 in the affirmative; however, quite interestingly, there is so much difference in adopted strategies given the learning task. For example, a sampling scheme that improves kernel approximation (e.g., ORFs [26]) will not necessarily be competitive for supervised learning [15]. In other words, Problem 1 has been addressed in a task-specific fashion. In this article, we propose a scoring rule for selecting features that lends itself to several important tasks in machine learning. Let $\mathbf{B}$ be a real matrix and define the following score function for any $\boldsymbol{\omega} \in \boldsymbol{\Omega}$:

$$q(\boldsymbol{\omega}) \triangleq p(\boldsymbol{\omega})\mathbf{z}^\top(\boldsymbol{\omega})\mathbf{B}\,\mathbf{z}(\boldsymbol{\omega}) \tag{6}$$

where $p(\boldsymbol{\omega})$ is the original probability density of random features. $p(\boldsymbol{\omega})$ can be thought as an easy prior to sample from. The score function $q(\boldsymbol{\omega})$ can then serve as the metric to reweight the random features from prior $p(\boldsymbol{\omega})$. The key advantage of the score function is that $\mathbf{B}$ can be selected based on the learning task to improve the performance. We will elaborate on this choice in Sections III-B–III-C.

B. Relation to Supervised Learning Scoring Rules

A number of recent works have proposed the idea of sampling random features based on data-dependent distributions, mostly focusing on improving generalization in supervised learning. In this section, we show that the score function (6) will bring some of these methods under the same umbrella. More specifically, given a particular choice of the center matrix $\mathbf{B}$, we can recover a number of data-dependent sampling schemes, such as LSs [13], [24], [41], [42] and energy-based exploration of random features (EERFs) [15].

1) Leverage Scores: Following the framework mentioned in [13], LS sampling is according to the following probability density function:

$$q_{LS}(\boldsymbol{\omega}) \propto p(\boldsymbol{\omega})\mathbf{z}^\top(\boldsymbol{\omega})(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{z}(\boldsymbol{\omega}) \tag{7}$$

which can be recovered precisely when $\mathbf{B} = (\mathbf{K}+\lambda\mathbf{I})^{-1}$ in (6). A practical implementation of LS was proposed in [41] and

later used in the experiments mentioned in [42] for SVM. The generalization properties of (a variant of) LS algorithm was also studied in [43] for the case of ridge regression.

2) Energy-Based Exploration of Random Features: The EERF algorithm was proposed in [15] for improving generalization. In supervised learning, the goal is to map input vectors $\{\mathbf{x}_i\}_{i=1}^n$ to output variables $\{y_i\}_{i=1}^n$, where $y_i \in \mathbb{R}$ for $i \in [n]$. The EERF algorithm employs the following scoring rule for random features

$$q_{\text{EERF}}(\boldsymbol{\omega}) \propto \left| \frac{1}{n} \sum_{i=1}^{n} y_i \phi(\mathbf{x}_i, \boldsymbol{\omega}) \right| \tag{8}$$

where the score is calculated for a large pool of random features and a subset with the largest score will be used for the supervised learning problem. Now, if we let $\mathbf{y} = [y_1, \ldots, y_n]^\top$, we can observe that $q_{\text{EERF}}(\boldsymbol{\omega})$ is equivalent to (6) with the center matrix $\mathbf{B} = \mathbf{y}\mathbf{y}^\top$ because ordering the pool of features according to $(\mathbf{y}^\top \mathbf{z}(\boldsymbol{\omega}))^2 = (\sum_{i=1}^n y_i \phi(\mathbf{x}_i, \boldsymbol{\omega}))^2$ is equivalent to $|(1/n)\sum_{i=1}^n y_i \phi(\mathbf{x}_i, \boldsymbol{\omega})|$ given above. Shahrampour *et al.* [15] showed in their numerical experiments that EERF consistently outperforms plain random features and other data-independent methods in terms of generalization. We remark that the kernel alignment method in [12] is also in a similar spirit. Instead of choosing features with largest scores, an optimization algorithm is proposed to reweight the features such that the transformed input is correlated enough with output variable.

Given the success of algorithms such as LS and EERF, we can hope that the scoring rule (6) has the potential to be adopted in various learning tasks. Indeed, the center matrix $\mathbf{B}$ should be chosen based on the objective function that needs to be optimized in the learning task at hand.

C. Relation to Dimension Reduction and Correlation Analysis

We now show another potential of the score function (6) by establishing that

$$\begin{aligned} \text{Tr}[\mathbf{KB}] &= \int_\Omega q(\boldsymbol{\omega}) d\boldsymbol{\omega} = \int_\Omega p(\boldsymbol{\omega}) \mathbf{z}^\top(\boldsymbol{\omega}) \mathbf{B}\ \mathbf{z}(\boldsymbol{\omega}) \\ &\approx \frac{1}{M} \sum_{m=1}^{M} \mathbf{z}^\top(\boldsymbol{\omega}_m) \mathbf{B}\ \mathbf{z}(\boldsymbol{\omega}_m) \end{aligned} \tag{9}$$

where $\{\boldsymbol{\omega}_m\}_{m=1}^M$ are independent samples from $p(\boldsymbol{\omega})$. The above relationship reveals the connection of the score function with a class of trace maximization problems dealing with kernelized objective functions. Several dimension reduction/correlation analysis methods fall into this category. For example, kernel principal component analysis (KPCA) and kernel orthogonal neighborhood preserving projections (KONPPs) have the exact form of the objective function in (9) (see, e.g., [44]). In each case, we can identify the matrix $\mathbf{B}$ by looking at the corresponding eigenvalue problem. For KPCA, the eigenvalue problem implies that the matrix $\mathbf{B} = (\mathbf{I} - (1/n)\mathbf{1}\mathbf{1}^\top)$, where $\mathbf{1}$ is the vector of all ones. In KONPP, the corresponding eigenvalue problem entails that $\mathbf{B} = (\mathbf{I} - \mathbf{W}^\top)(\mathbf{I} - \mathbf{W})$, where $\mathbf{W}$ is the affinity matrix in the feature space. Now, instead of covering more high level formulations, in Section IV, we will carefully study CCA, which is also formulated as a trace optimization (9).

Remark 1: The scoring rule (6) offers a principled way to select random features that promise good performance for a specific objective. In this section, we identify a class of dimension reduction and correlation analysis problems that are compatible with the proposed scoring rule. However, its adaptation to other problems still requires efforts in the identification and derivation of the score function. In Section IV, we will focus on nonlinear CCA, as an important problem in machine learning and statistics, to derive the respective score function and use it for algorithm implementation.

IV. Canonical Correlation Analysis With Score-Based Random Features Selection

We now show the application of the scoring rule (6) to nonlinear CCA.

A. Optimal Randomized Canonical Correlation Analysis

We now propose the adaptations of the scoring rule (6) for CCA, where the center matrix $\mathbf{B}$ is selected particularly for maximizing the total canonical correlation. We start with an important special case of $d_y = 1$ due to the natural connection to supervised learning. We will use index x for any quantity in relation to $\mathbf{X}$ and y for any quantity in relation to $\mathbf{Y}$.

1) Optimal Randomized Canonical Correlation Analysis 1 ($\mathbf{d_y} = 1$ and Linear $\mathbf{K}_y$): We consider the scenario where $\mathbf{X} \in \mathbb{R}^{n \times d_x}$ is mapped into a nonlinear space $\mathbf{Z}_x \in \mathbb{R}^{n \times M}$ (using random features) following (3). On the other hand, $\mathbf{Y} = \mathbf{y} \in \mathbb{R}^n$ remains in its original space (with $d_y = 1$ and $\mathbf{K}_y = \mathbf{y}\mathbf{y}^\top$). It is well known that if $\mathbf{y} = \mathbf{Z}_x \boldsymbol{\alpha}$ for some $\boldsymbol{\alpha} \in \mathbb{R}^M$, perfect (linear) correlation is achieved between $\mathbf{y}$ and $\mathbf{Z}_x$ (with $\mu_x = \mu_y = 0$ and $n > d_x$), simply because $\mathbf{y}$ is a linear combination of the columns of $\mathbf{Z}_x$. This motivates the idea that sampling schemes that are good for supervised learning may be natural candidates for CCA in that with $\mathbf{y} = \mathbf{Z}_x \boldsymbol{\alpha}$, we can achieve perfect correlation. The following proposition finds the optimal scoring rule of form (6) that maximizes the total canonical correlation.

Proposition 1: Consider KCCA in (5) with $\mu_x = \mu_y = \mu$, a nonlinear kernel matrix $\mathbf{K}_x$ and a linear kernel $\mathbf{K}_y = \mathbf{y}\mathbf{y}^\top$. If we approximate $\mathbf{K}_x \approx \mathbf{Z}_x \mathbf{Z}_x^\top$ only in the right block matrix of (5), the optimal scoring rule maximizing the total canonical correlation can be expressed as

$$q(\boldsymbol{\omega}) = p(\boldsymbol{\omega}) \mathbf{z}_x^\top(\boldsymbol{\omega}) (\mathbf{K}_x + \mu \mathbf{I})^{-1} \mathbf{y}\mathbf{y}^\top \mathbf{z}_x(\boldsymbol{\omega}) \tag{10}$$

for any $\boldsymbol{\omega} \in \Omega_x \subseteq \mathbb{R}^{d_x}$. The scoring rule above corresponds to (6) with $\mathbf{B} = (\mathbf{K}_x + \mu \mathbf{I})^{-1} \mathbf{y}\mathbf{y}^\top$.

Interestingly, the principled way of choosing $\mathbf{B}$ that maximizes total CCA leads to a feature selection rule that was not previously investigated. It is clear that the score function in (10) is different from LS (7) and EERF (8). While the scoring rule (10) optimizes canonical correlations in view of Proposition 1, calculating $\mathbf{B}$ would cost $O(n^3)$, which is not scalable to large datasets. The following corollary offers an approximated solution to avoid this issue.

Corollary 1: For any finite pool of random features $\{\boldsymbol{\omega}_m\}_{m=1}^{M_0}$, instead of selecting according to the scoring rule (10), we can approximate the scoring rule with the following empirical score:

$$q(\boldsymbol{\omega}_i) \approx \widehat{q}(\boldsymbol{\omega}_i) \triangleq \left[\left(\mathbf{Z}_x^\top \mathbf{Z}_x + \mu \mathbf{I}\right)^{-1} \mathbf{Z}_x^\top \mathbf{y}\mathbf{y}^\top \mathbf{Z}_x\right]_{ii} \tag{11}$$

for any $\boldsymbol{\omega}_{x,i} \in \Omega_x \subseteq \mathbb{R}^{d_x}$ and $i \in [M_0]$, where $\mathbf{Z}_x$ is formed with M_0 random features as in (3) and $\widehat{q}(\boldsymbol{\omega}_i)$ denotes the empirical score of the ith random features in the pool of M_0 features.

Observe that selecting according to the score rule above will reduce the computational cost from $O(n^3)$ to $O(nM_0^2 + M_0^3)$, which is a significant improvement when $M_0 \ll n$. After constructing (11), we can select the M features with the highest empirical scores. This algorithm is called ORCCA1 presented in Algorithm 1.

Algorithm 1 ORCCA1

Input: $\mathbf{X} \in \mathbb{R}^{n\times d_x}$,$\mathbf{y} \in \mathbb{R}^n$, the feature map $\phi(\cdot,\cdot)$, an integer M_0, an integer M, the prior distribution $p(\boldsymbol{\omega})$, the parameter $\mu > 0$.

1: Draw M_0 independent samples $\{\boldsymbol{\omega}_m\}_{m=1}^{M_0}$ from $p(\boldsymbol{\omega})$.

2: Construct the matrix

$$\mathbf{Q} = \left(\mathbf{Z}_x^\top \mathbf{Z}_x + \mu \mathbf{I}\right)^{-1} \mathbf{Z}_x^\top \mathbf{y}\mathbf{y}^\top \mathbf{Z}_x,$$

where $\mathbf{Z}_x$ is defined in (3).

3: Let for $i \in [M_0]$

$$\widehat{q}(\boldsymbol{\omega}_i) = [\mathbf{Q}]_{ii}.$$

The new weights $\widehat{\mathbf{q}} = [\widehat{q}(\boldsymbol{\omega}_1), \ldots, \widehat{q}(\boldsymbol{\omega}_{M_0})]^\top$.

4: Sort $\widehat{\mathbf{q}}$ and select top M features with highest scores from the pool to construct the transformed matrix $\widehat{\mathbf{Z}}_x$ following (3).

Output: Linear canonical correlations between $\widehat{\mathbf{Z}}_x$ and $\mathbf{y}$ (with regularization parameter μ).

2) Optimal Randomized Canonical Correlation Analysis 2 (Nonlinear $\mathbf{K}_y$): We now follow the idea of KCCA with both views of data mapped to a nonlinear space. More specifically, $\mathbf{X} \in \mathbb{R}^{n\times d_x}$ is mapped to $\mathbf{Z}_x \in \mathbb{R}^{n\times M}$ and $\mathbf{Y} \in \mathbb{R}^{n\times d_y}$ is mapped to $\mathbf{Z}_y \in \mathbb{R}^{n\times M}$ following (3). For this set up, we provide below the optimal scoring rule of form (6) that maximizes the total canonical correlation.

Theorem 1: Consider KCCA in (5) with $\mu_x = \mu_y = \mu$, a nonlinear kernel matrix $\mathbf{K}_x$, and a nonlinear kernel $\mathbf{K}_y$. If we alternatively approximate $\mathbf{K}_x \approx \mathbf{Z}_x\mathbf{Z}_x^\top$ and $\mathbf{K}_y \approx \mathbf{Z}_y\mathbf{Z}_y^\top$ only in the right block matrix of (5), the optimal scoring rule maximizing the total canonical correlation can be expressed as

$$\begin{aligned} q_x(\boldsymbol{\omega}) &= p_x(\boldsymbol{\omega})\mathbf{z}_x^\top(\boldsymbol{\omega})(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{K}_y\left(\mathbf{K}_y + \mu\mathbf{I}\right)^{-1}\mathbf{z}_x(\boldsymbol{\omega}) \\ q_y\left(\boldsymbol{\omega}'\right) &= p_y\left(\boldsymbol{\omega}'\right)\mathbf{z}_y^\top\left(\boldsymbol{\omega}'\right)\left(\mathbf{K}_y + \mu\mathbf{I}\right)^{-1}\mathbf{K}_x(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{z}_y\left(\boldsymbol{\omega}'\right) \end{aligned} \tag{12}$$

for any $\boldsymbol{\omega} \in \Omega_x \subseteq \mathbb{R}^{d_x}$ and any $\boldsymbol{\omega}' \in \Omega_y \subseteq \mathbb{R}^{d_y}$, respectively. The probability densities $p_x(\boldsymbol{\omega})$ and $p_y(\boldsymbol{\omega}')$ are the priors defining the default kernel functions in the space of $\mathcal{X}$ and $\mathcal{Y}$ according to (1).

We can associate the scoring rules above to the task-specific scoring rule (6) as well. Indeed, for choosing the random features from Ω_x to transform $\mathbf{X}$, the center matrix is $\mathbf{B} = (\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{K}_y(\mathbf{K}_y + \mu\mathbf{I})^{-1}$, and for choosing the random features from Ω_y to transform $\mathbf{Y}$, the center matrix is $\mathbf{B} = (\mathbf{K}_y + \mu\mathbf{I})^{-1}\mathbf{K}_x(\mathbf{K}_x + \mu\mathbf{I})^{-1}$. While the scoring rule (12) optimizes canonical correlations in view of (5), calculating $\mathbf{B}$ would cost $O(n^3)$, which is not scalable to large datasets. The following corollary offers an approximated solution to avoid this issue.

Corollary 2: For any finite pool of random features $\{\boldsymbol{\omega}_{x,m}\}_{m=1}^{M_0}$ and $\{\boldsymbol{\omega}_{y,m}\}_{m=1}^{M_0}$ (sampled from priors $p_x(\boldsymbol{\omega})$ and $p_y(\boldsymbol{\omega})$, respectively), instead of selecting according to the scoring rules (12), we can approximate them using the following empirical versions:

$$\begin{aligned} \hat{q}_x\left(\boldsymbol{\omega}_{x,i}\right) &= \left[\left(\mathbf{Z}_x^\top\mathbf{Z}_x + \mu\mathbf{I}\right)^{-1}\mathbf{Z}_x^\top\mathbf{Z}_y\left(\mathbf{Z}_y^\top\mathbf{Z}_y + \mu\mathbf{I}\right)^{-1}\mathbf{Z}_y^\top\mathbf{Z}_x\right]_{ii} \\ \hat{q}_y\left(\boldsymbol{\omega}_{y,i}\right) &= \left[\left(\mathbf{Z}_y^\top\mathbf{Z}_y + \mu\mathbf{I}\right)^{-1}\mathbf{Z}_y^\top\mathbf{Z}_x\left(\mathbf{Z}_x^\top\mathbf{Z}_x + \mu\mathbf{I}\right)^{-1}\mathbf{Z}_x^\top\mathbf{Z}_y\right]_{ii} \end{aligned}$$

for any $\boldsymbol{\omega}_{x,i} \in \Omega_x \subseteq \mathbb{R}^{d_x}$ and any $\boldsymbol{\omega}_{y,i} \in \Omega_y \subseteq \mathbb{R}^{d_y}$, respectively. $\mathbf{Z}_x$ and $\mathbf{Z}_y$ are the transformed matrices of $\mathbf{X}$ and $\mathbf{Y}$ as in (3) using M_0 random features. $\widehat{q}_x(\boldsymbol{\omega}_{x,i})$ and $\widehat{q}_y(\boldsymbol{\omega}_{y,i})$ denote the scores of the ith random features in the pools corresponding to $\mathbf{X}$ and $\mathbf{Y}$, respectively.

As we observe, the computational cost in both views of the data is reduced from $O(n^3)$ to $O(nM_0^2+M_0^3)$. The justification is provided in the appendix (Section VI-F). We use the above empirical scores for the implementation of ORCCA2, described in Algorithm 2. We also prove below that the theoretical score functions in (10) and (12) always provide improvement over RCCA and KCCA.

Proposition 2: The total canonical correlation obtained with M features selected from an M_0 features pool ($M < M_0 < \infty$) using nonempirical scores (10) and (12) provides a theoretical upper bound of the total canonical correlation obtained with M plain RFFs in expectation. Let us denote by ρ(KCCA) the total canonical correlation obtained by KCCA and by $\rho^{(M)}$(RCCA) the total canonical correlation obtained by RCCA with M plain RFFs. Let us also represent by $\rho^{(M,M_0)}$(ORCCA) the total canonical correlation obtained by ORCCA (1 and 2), where top M features are selected from a pool of $M_0 > M$ plain RFFs according to score (12). Then, the following relationship holds:

$$\mathbb{E}\left[\rho^{(M,M_0)}(\text{ORCCA})\right] \geq \mathbb{E}\left[\rho^{(M)}(\text{RCCA})\right] = \rho(\text{KCCA}) \tag{13}$$

where the expectation is taken over random features.

The intuition is that the selected features via the proposed score obtain a sup value over the subsets of the feature pool, and therefore, the expectation of the sup will be greater than the sup of expectation. This logic applies to other adaptations as well, meaning that proper adaptation of score function (6) will provide uniformly better results than RFFs in any suitable tasks. The proof of our results is given in the appendix.

Remark 2: Notice that ORCCA1 is a special case of ORCCA2 where $d_y = 1$ and $\mathbf{K}_y = \mathbf{y}\mathbf{y}^\top$, and we are presenting

Algorithm 2 ORCCA2

Input: $\mathbf{X} \in \mathbb{R}^{n\times d_x}$,$\mathbf{Y} \in \mathbb{R}^{n\times d_y}$, the feature map $\phi(\cdot,\cdot)$, an integer M_0, an integer M, the prior densities $p_x(\boldsymbol{\omega})$ and $p_y(\boldsymbol{\omega})$, parameter $\mu > 0$.

1: Draw samples $\{\boldsymbol{\omega}_{x,m}\}_{m=1}^{M_0}$ and $\{\boldsymbol{\omega}_{y,m}\}_{m=1}^{M_0}$ according to $p_x(\boldsymbol{\omega})$ and $p_y(\boldsymbol{\omega})$, respectively.

2: Construct the matrices

$$\mathbf{Q} = \left(\mathbf{Z}_x^\top \mathbf{Z}_x + \mu \mathbf{I}\right)^{-1} \mathbf{Z}_x^\top \mathbf{Z}_y$$
$$\mathbf{P} = \left(\mathbf{Z}_y^\top \mathbf{Z}_y + \mu \mathbf{I}\right)^{-1} \mathbf{Z}_y^\top \mathbf{Z}_x.$$

where $\mathbf{Z}_x$ and $\mathbf{Z}_y$ are defined in (3).

3: Let for $i \in [M_0]$

$$\widehat{q}_x\left(\boldsymbol{\omega}_{x,i}\right) = [\mathbf{QP}]_{ii}.$$

The new weights $\widehat{\mathbf{q}}_x = [\widehat{q}_x(\boldsymbol{\omega}_{x,1}), \ldots, \widehat{q}_x(\boldsymbol{\omega}_{x,M_0})]^\top$.

4: Let for $i \in [M_0]$

$$\widehat{q}_y\left(\boldsymbol{\omega}_{y,i}\right) = [\mathbf{PQ}]_{ii}.$$

The new weights $\widehat{\mathbf{q}}_y = [\widehat{q}_y(\boldsymbol{\omega}_{y,1}), \ldots, \widehat{q}_y(\boldsymbol{\omega}_{y,M_0})]^\top$.

5: Select top M features with the highest scores from each of the pools $\{\boldsymbol{\omega}_{x,i}\}_{i=1}^{M_0}$ and $\{\boldsymbol{\omega}_{y,i}\}_{i=1}^{M_0}$, according to the new scores $\widehat{\mathbf{q}}_x$ and $\widehat{\mathbf{q}}_y$ to construct the transformed matrices $\widehat{\mathbf{Z}}_x \in \mathbb{R}^{n\times M}$ and $\widehat{\mathbf{Z}}_y \in \mathbb{R}^{n\times M}$, respectively, as in (3).

Output: Linear canonical correlations between $\widehat{\mathbf{Z}}_x$ and $\widehat{\mathbf{Z}}_y$ (with parameter μ).

it as a separate algorithm to highlight its connection with supervised learning.

V. Numerical Experiments

A. Approximated KCCA Comparison

We now investigate the empirical performance of ORCCA1 and ORCCA2 against other approximated versions of KCCA using six datasets from the UCI Machine Learning Repository.

1) Benchmark Algorithms: We compare our work to four random features based benchmark algorithms that have shown good performance in supervised learning and/or kernel approximation. All four algorithms approximate KCCA by randomized low-rank kernel approximation. The first one is plain RFFs [1]. Next is ORFs [26], which improves the variance of kernel approximation. We also include two data-dependent sampling methods, LS [13], [41] and EERF [15], due to their success in supervised learning as mentioned in Section III-B.

1) RFFs [1] With $\phi = \cos(\mathbf{x}^\top\boldsymbol{\omega} + b)$ as the Feature Map to Approximate the Gaussian Kernel: $\{\boldsymbol{\omega}_m\}_{m=1}^{M}$ are sampled from the Gaussian distribution $\mathcal{N}(0, \sigma^2\mathbf{I})$ and $\{b_m\}_{m=1}^{M}$ are sampled from uniform distribution $\mathcal{U}(0, 2\pi)$. We use this to transform $\mathbf{X} \in \mathbb{R}^{n\times d_x}$ to $\mathbf{Z}_x \in \mathbb{R}^{n\times M}$. The same procedure applies to $\mathbf{y} \in \mathbb{R}^n$ to map it to $\mathbf{Z}_y \in \mathbb{R}^{n\times M}$. This algorithm corresponds to the aforementioned RCCA [9], and we use RFF instead of its original name to emphasize the role of plain random features method here.

2) Orthogonal Random Features [26] With $\phi = [\cos(\mathbf{x}^\top\boldsymbol{\omega}), \sin(\mathbf{x}^\top\boldsymbol{\omega})]$ as the Feature Map: $\{\boldsymbol{\omega}_m\}_{m=1}^{M}$ are sampled from a Gaussian distribution $\mathcal{N}(0, \sigma^2\mathbf{I})$ and then modified based on a QR decomposition step. The transformed matrices for ORF are $\mathbf{Z}_x \in \mathbb{R}^{n\times 2M}$ and $\mathbf{Z}_y \in \mathbb{R}^{n\times 2M}$. Given that the feature map is 2-D here, to keep the comparison fair, the number of random features used for ORF will be half of other algorithms.

3) Leverage Score Sampling [13], [41] With $\phi = \cos(\mathbf{x}^\top\boldsymbol{\omega} + b)$ as the Feature Map: $\{\boldsymbol{\omega}_m\}_{m=1}^{M_0}$ are sampled from the Gaussian distribution $\mathcal{N}(0, \sigma^2\mathbf{I})$ and $\{b_m\}_{m=1}^{M_0}$ are sampled from the uniform distribution $\mathcal{U}(0, 2\pi)$. M features are sampled from the pool of M_0 RFFs according to the scoring rule of LS (7). Note that the transformed matrices $\widetilde{\mathbf{Z}}_x \in \mathbb{R}^{n\times M}$ and $\widetilde{\mathbf{Z}}_y \in \mathbb{R}^{n\times M}$ correspond to (3) with the jth column normalized by a factor of $(p(\boldsymbol{\omega}_j)/q_{LS}(\boldsymbol{\omega}_j))^{1/2}$ due to importance sampling.

4) Energy-Based Exploration of Random Features [15] With $\phi = \cos(\mathbf{x}^\top\boldsymbol{\omega} + b)$ as the Feature Map to Approximate the Gaussian Kernel: $\{\boldsymbol{\omega}_m\}_{m=1}^{M_0}$ are sampled from the Gaussian distribution $\mathcal{N}(0, \sigma^2\mathbf{I})$ and $\{b_m\}_{m=1}^{M_0}$ are sampled from the uniform distribution $\mathcal{U}(0, 2\pi)$. M features are selected using the scoring rule in (8) from the M_0 feature pool. We use the sampled M features to transform $\mathbf{X} \in \mathbb{R}^{n\times d_x}$ to $\mathbf{Z}_x \in \mathbb{R}^{n\times M}$.

1) *Numerical Experiments for ORCCA1:*
Practical Considerations: Following [8], we work with empirical copula transformation of datasets to achieve invariance with respect to marginal distributions. For $\mathbf{X}$ domain, the variance of random features σ_x is set to be the inverse of mean distance of the 50th nearest neighbor (in Euclidean distance), following [26]. We use the corresponding Gaussian kernel width for KCCA. The label information y has remained in its original space after coupula transform. For LS, EERF, and ORCCA1, the pool size is $M_0 = 10M$ when M random features are used in the CCA calculation. The regularization parameter λ for LS is chosen through grid search. The regularization parameter $\mu = 10^{-6}$ is set to be small enough to make its effect on CCA negligible while avoiding numerical errors caused by singularity. The feature map for ORCCA1 is set to be $\phi = \cos(\mathbf{x}^\top\boldsymbol{\omega}+b)$.
Performance: The empirical results for ORCCA1 are reported in Fig. 1. The results are averaged over 30 simulations and error bars are presented in the plots. There is only one canonical correlation to report due to $d_y = 1$. We can clearly observe that ORCCA1 shows dominance over the other benchmark algorithms except for two occasions: 1) at $M = 100$ features, ORF performs on par with ORCCA1 in the Adult dataset and 2) at 60+ features, EERF performs on par with ORCCA1 in the Energy dataset. Another interesting observation here is that other than ORCCA1, all the benchmark algorithms do not show any clear hierarchy in performance given their established empirical performance hierarchy in supervised learning [45].

2) *Numerical Experiments for ORCCA2:*
Practical Considerations: The variance of random features σ_x is set to be the inverse of mean distance of the 50th nearest neighbor (same procedure as ORCCA1 experiments). After performing a grid search, the variance of random features σ_y for $\mathbf{Y}$ is set to be the same

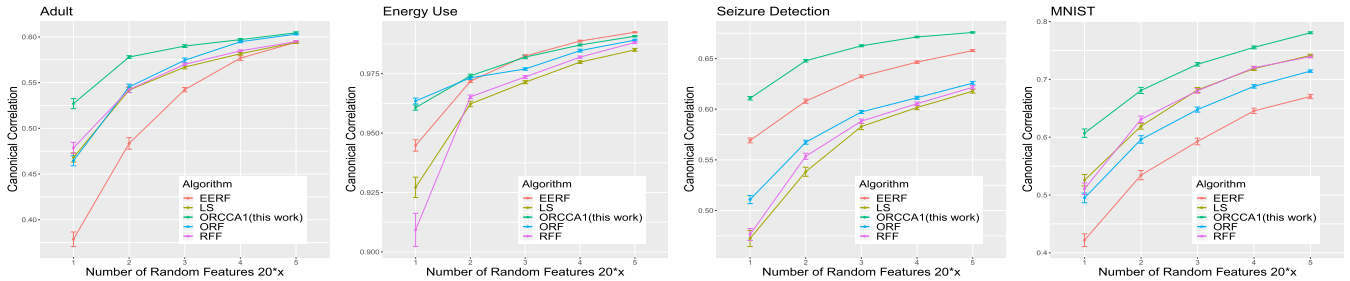

Fig. 1. Plot of canonical correlations versus the number of features obtained by different algorithms (for ORCCA1 comparison). The error bars are obtained with 30 Monte Carlo simulations.

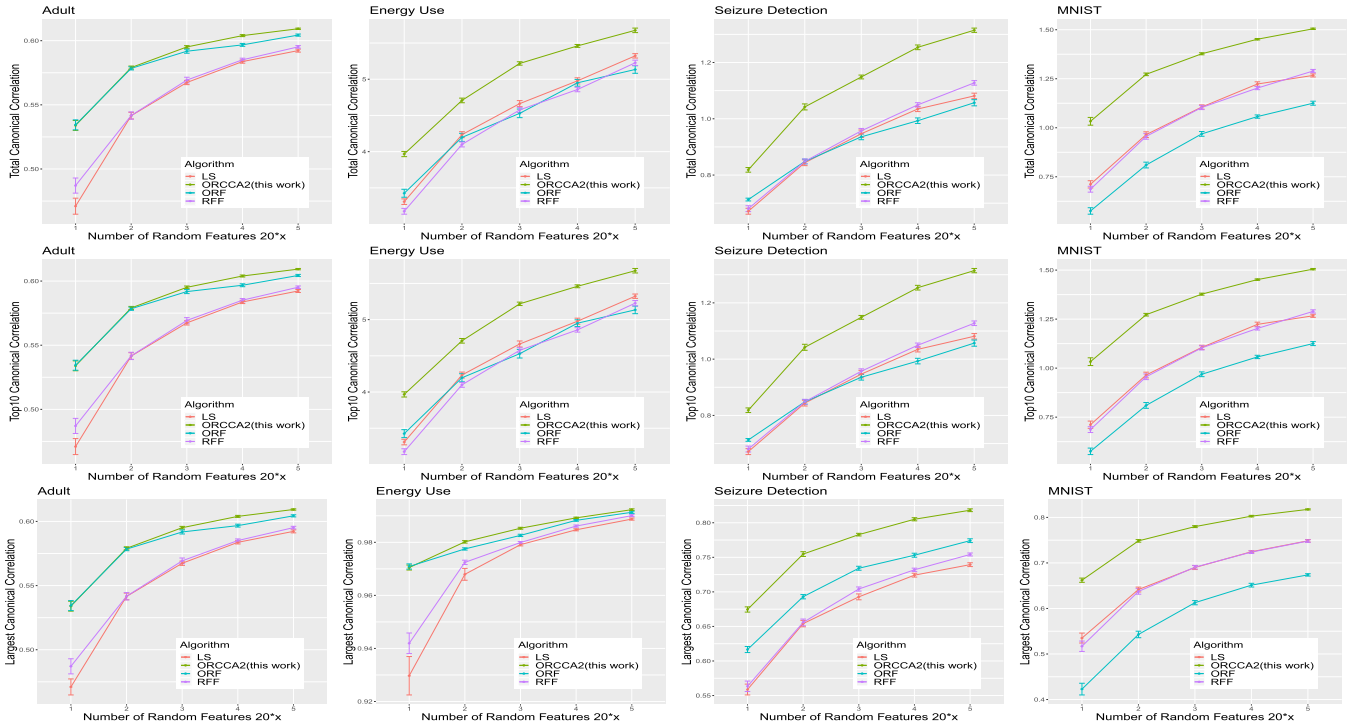

Fig. 2. Plot of total canonical correlation (first row), top-10 canonical correlations (second row), and the largest canonical correlation (third row) versus the number of features obtained by different algorithms. The error bars are obtained with 30 Monte Carlo simulations.

as σ_x, producing the best results for all algorithms. For LS and ORCCA2, the pool size is $M_0 = 10M$ when M random features are used in the CCA calculation. The regularization parameter λ for LS is chosen through grid search. The regularization parameter in CCA calculation remains at $\mu = 10^{-6}$. The feature map for ORCCA2 is also $\phi = \cos(\mathbf{x}^\top \boldsymbol{\omega} + b)$.

Performance: Our empirical results on four datasets are reported in Fig. 2. The results are averaged over 30 simulations and error bars are presented in the plots. The first row of Fig. 2 represents the total canonical correlation versus the number of random features M for ORCCA2 (this work), RFF, ORF, and LS. We observe that ORCCA2 is superior compared to other benchmarks and only for the Adult dataset ORF is initially on par with our algorithm. The second row of Fig. 2 represents the top-10 canonical correlations, where we observe the exact same trend. This result shows that the total canonical correlation is mostly explained by their leading correlations. The third row of Fig. 2 represents the largest canonical correlation. Although ORCCA2 is developed for the total canonical correlation objective function, we can still achieve performance boost in identifying the largest canonical correlation, which is also a popular objective in CCA. The theoretical time complexity and practical time cost are tabulated in Table I. Our cost is comparable to LS as both algorithms calculate a new score function for feature selection. The run time is obtained on a desktop with an eight-core, 3.6-GHz Ryzen 3700X processor and 32 GB of RAM (3000 MHz). Given the dominance of ORCCA1 over LS and EERF and ORCCA2 over LS, we can clearly conclude that data-dependent sampling methods that improve supervised learning are not necessarily best choices for CCA. Finally, in Fig. 4, we compare ORCCA2 and KCCA. The datasets used for this part are

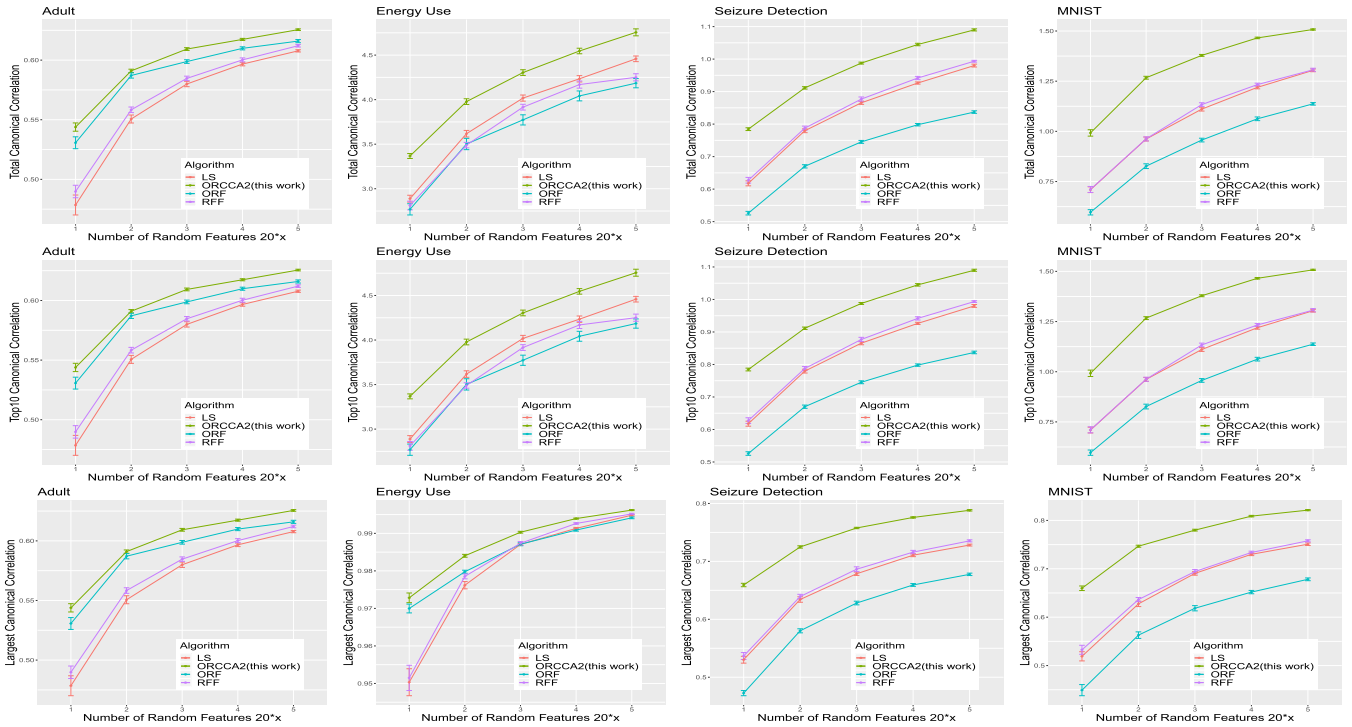

Fig. 3. Plot of total canonical correlation (first row), top-10 canonical correlations (second row), and the largest canonical correlation (third row) versus the number of features obtained by different algorithms on test sets. The error bars are obtained with 30 Monte Carlo simulations.

smaller than the previous one due to prohibitive cost of KCCA. On these datasets, ORCCA2 can gradually outperform KCCA in total CCA value as the computation time increases (due to increasing the number of random features), while it is also more efficient in terms of time cost. The main reason is that ORCCA2 approximates a "better" kernel than Gaussian by choosing good features.

3) *Performance Comparison With Training and Testing Sets (ORCCA2):*
Due to the sample and select nature of ORCCA algorithms, one might wonder the necessity of cross validation during the implementation of ORCCA. In this experiment, we randomly choose 80% of the data as the training set and use the rest as the testing set. For ORCCA2 and LS, the random features are reselected according to the score calculated with the training set, and canonical correlations are calculated with these features for the testing set. RFF and ORF algorithms are only implemented on the test set as their choice is independent of data. All the parameters, including the random features variances σ_x^2 and σ_y^2, feature map ϕ, pool size M_0, the original distribution for random features, the regularization parameter μ for CCA, and the regularization parameter λ for LS, are kept the same as the previous ORCCA2 numerical experiment. The results are shown in Fig. 3 following the same layout as Fig. 2. We can observe that the performance of algorithms is almost identical to the results of Fig. 2, which shows the robustness of ORCCA2 to test data as well. In another word, we can avoid the loss of information due to cross validation when implementing ORCCA algorithms.

B. Deep Learning CCA Comparison

In addition to variants of kernel approximation for CCA, recent studies have been utilizing deep learning to further enhance CCA. The recent deep learning algorithms include deep CCA (DCCA) [34], deep variational CCA (DVCCA) [46], deep generalized CCA (DGCCA) [47], and deep tensor CCA (DTCCA) [48]. In this section, we will compare ORCCA2 with the above-mentioned state-of-the-art CCA methods.

1) Benchmark Algorithms:

1) *DCCA:* DCCA uses deep neural networks for the nonlinear transformation of both views of data. The parameters of the network are updated using gradient flow, calculated with negative total canonical correlation used as the loss function. We identically construct the neural networks for both views of the data. The neural network consists of a hidden layer with 100 neurons and ReLU activations, with an output of dimension 20.
2) *DTCCA:* Deep tensor CCA uses a deep neural network for nonlinear transformation, similar to DCCA. However, the loss function is replaced by the tensor formulation of total canonical correlations from tensor CCA to extend DCCA to more than two views of data. We construct the same neural network embedding as DCCA. We must note that the main goal of DTCCA is to extend CCA to more than two views of data with tensor formulation. The main reason to include DTCCA in this comparison is to show the impact of multiview CCA on the correlation extraction.
3) *DGCCA:* Deep generalized CCA uses deep neural networks for nonlinear transformation, similar to DCCA. It forces the two views of data to transform into a shared

TABLE I

THEORETICAL TIME COMPLEXITY OF ORCCA2 AND BENCHMARK ALGORITHMS AND THEIR PRACTICAL TIME COST (IN SEC). FOR REPORTING TIME COST (ONLY), WE USED $n = 5000$ DATA POINTS, $M = 100$ RANDOM FEATURES, AND $M_0 = 1000$ FEATURE POOL. FOR ENERGY USE, WE USED THE FULL DATASET $n = 768$

ALGORITHMS	COMPLEXITY	MNIST	ENERGY USE	SEIZURE DETECTION	ADULT
ORCCA2	$O(nM_0^2 + M_0^3)$	21.03	2.49	17.39	15.70
LS	$O(nM_0^2 + M_0^3)$	18.73	2.44	16.32	14.92
RFF	$O(nM^2)$	2.86	0.40	2.59	2.53
ORF	$O(nM^2)$	2.78	0.40	2.62	2.58

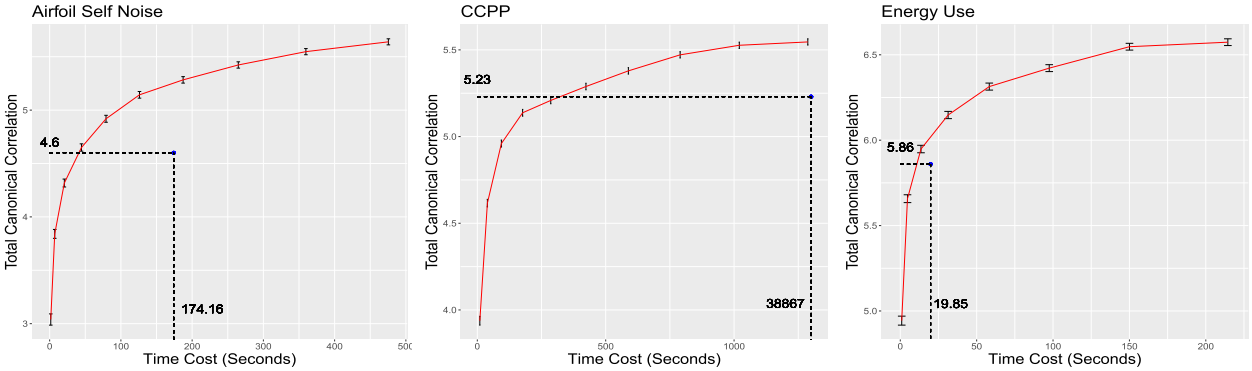

Fig. 4. ORCCA2 versus KCCA comparison. The plots represent the total canonical correlations obtained by ORCCA2 with different time costs (by varying the number of random features). The dot represents the total canonical correlation obtained by KCCA and its time cost. The total canonical correlations and time cost of KCCA are also marked by text on the axis. The error bars are obtained with 30 Monte Carlo simulations. CCPP is "Combined Cycle Power Plant" data.

TABLE II

ORCCA2 PERFORMANCE COMPARISON WITH OTHER STATE-OF-THE-ART DEEP LEARNING VARIANTS OF CCA. ALL DEEP LEARNING ALGORITHMS USE 20 LATENT DIMENSIONS, AND RFF AND ORCCA2 USE 20 RANDOM FEATURES FOR A FAIR COMPARISON. THE STANDARD ERRORS ARE ESTIMATED WITH 30 MONTE CARLO SIMULATIONS

	FASHIONMNIST				MNIST			
ALGORITHMS	TIME(S)	LARGEST	TOP-10	TOTAL	TIME(S)	LARGEST	TOP-10	TOTAL
RFF	$\mathbf{0.016 \pm 0.001}$	0.559 ± 0.010	3.126 ± 0.027	3.971 ± 0.034	$\mathbf{0.016 \pm 0.001}$	0.405 ± 0.005	2.773 ± 0.020	3.586 ± 0.027
DCCA	34.983 ± 0.020	0.517 ± 0.026	1.993 ± 0.039	2.360 ± 0.047	33.023 ± 0.459	0.261 ± 0.018	1.355 ± 0.056	1.585 ± 0.072
DTCCA	7.266 ± 0.422	0.535 ± 0.008	1.372 ± 0.028	1.194 ± 0.042	6.911 ± 0.348	0.173 ± 0.008	0.731 ± 0.032	0.555 ± 0.48
DGCCA	53.941 ± 0.171	$\mathbf{0.698 \pm 0.014}$	1.952 ± 0.040	2.132 ± 0.065	55.732 ± 0.150	0.221 ± 0.012	1.155 ± 0.050	1.274 ± 0.043
DVCCA	4.756 ± 0.018	0.401 ± 0.006	3.226 ± 0.055	4.168 ± 0.122	4.615 ± 0.010	0.184 ± 0.004	1.329 ± 0.034	1.600 ± 0.042
ORCCA2	0.083 ± 0.003	0.638 ± 0.008	$\mathbf{3.509 \pm 0.021}$	$\mathbf{4.483 \pm 0.028}$	0.090 ± 0.003	$\mathbf{0.452 \pm 0.005}$	$\mathbf{3.077 \pm 0.017}$	$\mathbf{4.016 \pm 0.024}$

view that will be learned through gradient flow. We use a similar neural network structure for DGCCA, where we feed the data into a hidden layer with 160 neurons and ReLU activations. It will then be transformed into a latent variable of dimension 20.

4) *DVCCA:* Deep variational CCA replaces the total canonical correlation loss function with the expected log likelihood, derived from the evidence lower bound given a latent variable with shared information from both views of data. We use the same network structure as DCCA for DVCCA.

2) Datasets: Instead of using low-dimensional datasets, we focus on high-dimensional image data correlation extraction, following the convention in deep learning variants of CCA. We use two datasets, Fashion MNIST, a 28×28 grayscale image dataset for clothes and shoes, and MNIST, a 28×28 grayscale image dataset for handwritten digits. For both datasets, we randomly sample 1500 images, 500 for training, 500 for validation, and 500 for testing. We rotate these images for a degree uniformly distributed in $[-(\pi/4), (\pi/4)]$ to form the first view of the data. Then, for each image in View 1, we randomly sample a new image with the same label from the original dataset and add centered Gaussian noise to form View 2. This experimental setting is identical to that of [46]. The training set in ORCCA2 is used for feature selection, whereas in all deep learning algorithms, it is used for neural network training. The validation set is used in all deep learning algorithms for training epochs' validation. The test set is then used in all the algorithms for recording canonical correlations. Some sample images for the two views are shown in Fig. 5.

3) Practical Consideration: The latent dimension for deep learning algorithms as well as the number of random features for ORCCA2 and RFF are set to be 20 for a fair comparison. Note that any deep learning algorithm is able to achieve perfect correlation between two views of training data for large enough latent dimension (overfitting). However, this is not the intent in applications of nonlinear CCA, and all the deep learning CCA works [34], [46]–[48] conduct their experiments in a low latent dimension, as we do here. We use the "Adam"

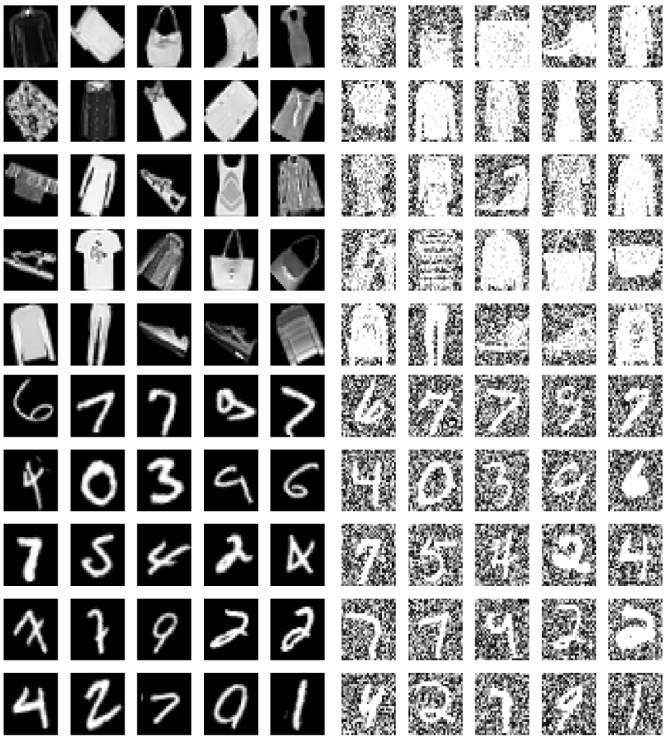

Fig. 5. Rotated View 1 (left) and Noisy View 2 (right) of FashionMNIST dataset and MNIST dataset.

optimizer for all deep learning algorithms, and the number of training epochs is selected using the validation set. DCCA is trained for 50 epochs. DTCCA is trained for 100 epochs. DGCCA is trained for 20 epochs. DVCCA is trained for 300 epochs.

4) Performance: All the results are tabulated in Table II, where we report the time cost, largest correlation value, the sum of top-10 correlation values, and the sum of total correlation values. The standard errors are calculated using 30 Monte Carlo simulations. The reported time for deep learning algorithms and ORCCA2 includes the model training/feature selection time in addition to the CCA time. The validation time for deep learning algorithms is not included. As we can observe, ORCCA2 dominates all the benchmark algorithms in correlation extraction while being significantly faster than the deep learning algorithms.

VI. Conclusion

Random features have been widely used for various machine learning tasks but often times they are sampled from a preset distribution, approximating a fixed kernel. In this work, we highlight the role of the objective function in the learning task at hand. We propose a score function for selecting random features, which depends on a parameter (matrix) chosen based on the specific objective function to improve the performance. We start by drawing connections to score functions for random features in supervised learning. We first show the potential of our score function through a class of dimension reduction and correlation analysis models, which involves a trace operator in their objective functions. We then focus on CCA and derive the optimal score function for maximizing the total CCA. Empirical results verify that random features selected using our score function significantly outperform other state-of-the-art methods for random features. It would be interesting to explore the potential of this score function for other learning tasks as a future direction.

Appendix

A. Proof of Proposition 1

To find canonical correlations in KCCA (5), we deal with the following eigenvalue problem (see, e.g., [49, Sec. 4.1]):

$$(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{K}_y\big(\mathbf{K}_y + \mu\mathbf{I}\big)^{-1}\mathbf{K}_x\boldsymbol{\pi}_x = \delta^2\boldsymbol{\pi}_x \tag{14}$$

where the eigenvalues δ^2 are the kernel canonical correlations and their corresponding eigenvectors are the kernel canonical pairs of $\mathbf{X}$. The solutions to $\mathbf{y}$ can be obtained by switching the indices of x and y. When $\mathbf{K}_y$ is a linear kernel, we can rewrite above as follows:

$$(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{y}\mathbf{y}^\top\big(\mathbf{y}\mathbf{y}^\top + \mu\mathbf{I}\big)^{-1}\mathbf{K}_x\boldsymbol{\pi}_x = \delta^2\boldsymbol{\pi}_x.$$

Then, maximizing the total canonical correlation (over random features) will be equivalent to maximizing the approximated objective $\mathrm{Tr}[(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{y}\mathbf{y}^\top(\mathbf{y}\mathbf{y}^\top + \mu\mathbf{I})^{-1}\mathbf{K}_x]$. Here, we first approximate $\mathbf{K}_x$ at the end of the left-hand side with $\mathbf{Z}_x\mathbf{Z}_x^\top$ such that

$$\begin{aligned}
&\mathrm{Tr}[(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{y}\mathbf{y}^\top\big(\mathbf{y}\mathbf{y}^\top + \mu\mathbf{I}\big)^{-1}\mathbf{K}_x] \\
&\quad\approx \mathrm{Tr}\Big[(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{y}\mathbf{y}^\top\big(\mathbf{y}\mathbf{y}^\top + \mu\mathbf{I}\big)^{-1}\mathbf{Z}_x\mathbf{Z}_x^\top\Big] \\
&\quad= \mathrm{Tr}\big[(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{Z}_x\mathbf{Z}_x^\top\big]\frac{1}{\mu + \mathbf{y}^\top\mathbf{y}} \\
&\quad\propto \mathrm{Tr}\big[(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{Z}_x\mathbf{Z}_x^\top\big] \\
&\quad= \mathrm{Tr}\big[\mathbf{Z}_x^\top(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{Z}_x\big] \\
&\quad= \sum_{i=1}^{M_0}\big[\mathbf{Z}_x^\top(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{Z}_x\big]_{ii} \\
&\quad= \frac{1}{M_0}\sum_{i=1}^{M_0}\mathbf{z}_x^\top(\boldsymbol{\omega}_i)(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{z}_x(\boldsymbol{\omega}_i)
\end{aligned} \tag{15}$$

where the third line follows by the Woodbury inversion lemma (push-through identity).

Given the closed form above, we can immediately see that good random features are ones that maximize the objective above. Hence, we can select random features according to the (unnormalized) $\mathbf{z}_x^\top(\boldsymbol{\omega})(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{z}_x(\boldsymbol{\omega})$. Given that random features are sampled from the prior $p(\boldsymbol{\omega})$, we can then write the score function as

$$q(\boldsymbol{\omega}) \triangleq p(\boldsymbol{\omega})\mathbf{z}_x^\top(\boldsymbol{\omega})(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{z}_x(\boldsymbol{\omega}) \tag{16}$$

completing the proof of Proposition 1.

B. Proof of Corollary 1

In Corollary 1, we further approximate (10) and achieve an improved time cost.

Notice that when we sample M_0 random features as a pool, the probability $p(\boldsymbol{\omega})$ is already incorporated in the score. Therefore, we just approximate the kernel matrix $\mathbf{K}_x$ with $\mathbf{Z}_x\mathbf{Z}_x^\top$ [according to (3)] such that

$$q_x(\boldsymbol{\omega}_m) \approx \mathbf{z}_x^\top(\boldsymbol{\omega}_m)\big(\mathbf{Z}_x\mathbf{Z}_x^\top + \mu\mathbf{I}\big)^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{z}_x(\boldsymbol{\omega}_m). \tag{17}$$

Then, using the push-through identity again, we derive

$$\begin{aligned}&\mathbf{z}_x^\top(\boldsymbol{\omega}_m)\big(\mathbf{Z}_x\mathbf{Z}_x^\top + \mu\mathbf{I}\big)^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{z}_x(\boldsymbol{\omega}_m)\\ &\quad= \Big[\mathbf{Z}_x^\top\big(\mathbf{Z}_x\mathbf{Z}_x^\top + \mu\mathbf{I}\big)^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{Z}_x\Big]_{mm}\\ &\quad= \Big[\big(\mathbf{Z}_x^\top\mathbf{Z}_x + \mu\mathbf{I}\big)^{-1}\mathbf{Z}_x^\top\mathbf{y}\mathbf{y}^\top\mathbf{Z}_x\Big]_{mm}.\end{aligned} \tag{18}$$

Therefore, for a specific $\boldsymbol{\omega}_m$, the RHS of (17) can be rewritten in the following form:

$$\text{RHS} = \Big[\big(\mathbf{Z}_x^\top\mathbf{Z}_x + \mu\mathbf{I}\big)^{-1}\mathbf{Z}_x^\top\mathbf{y}\mathbf{y}^\top\mathbf{Z}_x\Big]_{mm} = \widehat{q}_x(\boldsymbol{\omega}_m).$$

Then, Corollary 1 is proven.

C. Proof of Theorem 1

When $\mathbf{Y}$ is also mapped into a nonlinear space, we need to work with the eigensystem in (14). As discussed before, the objective is then maximizing the approximated version of $\text{Tr}[(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{K}_y(\mathbf{K}_y + \mu\mathbf{I})^{-1}\mathbf{K}_x]$. Following the same idea of (15), we have the following:

$$\begin{aligned}&\text{Tr}[(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{K}_y\big(\mathbf{K}_y + \mu\mathbf{I}\big)^{-1}\mathbf{K}_x]\\ &\quad\approx \text{Tr}\Big[(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{K}_y\big(\mathbf{K}_y + \mu\mathbf{I}\big)^{-1}\mathbf{Z}_x\mathbf{Z}_x^\top\Big]\\ &\quad= \text{Tr}\Big[\mathbf{Z}_x^\top(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{K}_y\big(\mathbf{K}_y + \mu\mathbf{I}\big)^{-1}\mathbf{Z}_x\Big]\\ &\quad= \frac{1}{M_0}\sum_{m=1}^{M_0}\mathbf{z}_x^\top(\boldsymbol{\omega}_m)(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{K}_y\big(\mathbf{K}_y + \mu\mathbf{I}\big)^{-1}\mathbf{z}_x(\boldsymbol{\omega}_m).\end{aligned} \tag{19}$$

We can then follow the exact same lines in the proof of Proposition 1 to arrive at the following score function:

$$q_x(\boldsymbol{\omega}) = p_x(\boldsymbol{\omega})\mathbf{z}_x^\top(\boldsymbol{\omega})(\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{K}_y\big(\mathbf{K}_y + \mu\mathbf{I}\big)^{-1}\mathbf{z}_x(\boldsymbol{\omega}).$$

The proof for $q_y(\boldsymbol{\omega})$ follows in a similar fashion.

D. Proof of Corollary 2

Similar to approximation ideas in the proof of Corollary 1, we can use the approximations $\mathbf{K}_x \approx \mathbf{Z}_x\mathbf{Z}_x^\top$ and $\mathbf{K}_y \approx \mathbf{Z}_y\mathbf{Z}_y^\top$ to get

$$\begin{aligned}&q_x(\boldsymbol{\omega}_m)\\ &\quad\approx \mathbf{z}_x^\top(\boldsymbol{\omega}_m)\big(\mathbf{Z}_x\mathbf{Z}_x^\top + \mu\mathbf{I}\big)^{-1}\mathbf{Z}_y\mathbf{Z}_y^\top\big(\mathbf{Z}_y\mathbf{Z}_y^\top + \mu\mathbf{I}\big)^{-1}\mathbf{z}_x(\boldsymbol{\omega}_m).\end{aligned} \tag{20}$$

Using the push-through identity twice in the following, we have:

$$\begin{aligned}&\mathbf{z}_x^\top(\boldsymbol{\omega}_m)\big(\mathbf{Z}_x\mathbf{Z}_x^\top + \mu\mathbf{I}\big)^{-1}\mathbf{Z}_y\mathbf{Z}_y^\top\big(\mathbf{Z}_y\mathbf{Z}_y^\top + \mu\mathbf{I}\big)^{-1}\mathbf{z}_x(\boldsymbol{\omega}_m)\\ &\quad= \Big[\mathbf{Z}_x^\top\big(\mathbf{Z}_x\mathbf{Z}_x^\top + \mu\mathbf{I}\big)^{-1}\mathbf{Z}_y\mathbf{Z}_y^\top\big(\mathbf{Z}_y\mathbf{Z}_y^\top + \mu\mathbf{I}\big)^{-1}\mathbf{Z}_x\Big]_{mm}\\ &\quad= \Big[\big(\mathbf{Z}_x^\top\mathbf{Z}_x + \mu\mathbf{I}\big)^{-1}\mathbf{Z}_x^\top\mathbf{Z}_y\big(\mathbf{Z}_y^\top\mathbf{Z}_y + \mu\mathbf{I}\big)^{-1}\mathbf{Z}_y^\top\mathbf{Z}_x\Big]_{mm}\end{aligned}$$

which provides the approximate scoring rule

$$\widehat{q}_x(\boldsymbol{\omega}_m) = \Big[\big(\mathbf{Z}_x^\top\mathbf{Z}_x + \mu\mathbf{I}\big)^{-1}\mathbf{Z}_x^\top\mathbf{Z}_y\big(\mathbf{Z}_y^\top\mathbf{Z}_y + \mu\mathbf{I}\big)^{-1}\mathbf{Z}_y^\top\mathbf{Z}_x\Big]_{mm}. \tag{21}$$

The proof for $\widehat{q}_y(\boldsymbol{\omega}_m)$ follows in a similar fashion.

E. Proof of Proposition 2

We will demonstrate the proof using the ORCCA2 formulation. Recall the total canonical correlation decomposition derived in (19). We will now formally define two quantities in the following:

$$\begin{aligned}\rho^{(M)}(\text{RCCA}) &\triangleq \frac{1}{M}\sum_{m=1}^{M}\mathbf{z}_x^\top(\boldsymbol{\omega}_m)\mathbf{G}\mathbf{z}_x(\boldsymbol{\omega}_m)\\ \rho^{(M,M_0)}(\text{ORCCA}) &\triangleq \sup_{\mathcal{N}\subseteq[M_0],|\mathcal{N}|=M}\frac{1}{M}\sum_{m\in\mathcal{N}}\mathbf{z}_x^\top(\boldsymbol{\omega}_m)\mathbf{G}\mathbf{z}_x(\boldsymbol{\omega}_m)\end{aligned} \tag{22}$$

where

$$\mathbf{G} \triangleq (\mathbf{K}_x + \mu\mathbf{I})^{-1}\mathbf{K}_y\big(\mathbf{K}_y + \mu\mathbf{I}\big)^{-1}.$$

We can see that $\rho^{(M)}(\text{RCCA})$ corresponds to the total canonical correlation obtained by M plain RFFs. Similarly, we can see that $\rho^{(M,M_0)}(\text{ORCCA})$ corresponds to the total canonical correlation obtained by ORCCA2, where top M features are selected from a pool of $M_0 > M$ plain RFFs according to score (12).

Taking expectation over random features, we have that

$$\begin{aligned}&\mathbb{E}\big[\rho^{(M,M_0)}(\text{ORCCA})\big]\\ &\quad= \frac{1}{M}\mathbb{E}\left[\sup_{\mathcal{N}\subseteq[M_0],|\mathcal{N}|=M}\sum_{m\in\mathcal{N}}\mathbf{z}_x^\top(\boldsymbol{\omega}_m)\mathbf{G}\mathbf{z}_x(\boldsymbol{\omega}_m)\right]\\ &\quad\geq \frac{1}{M}\sup_{\mathcal{N}\subseteq[M_0],|\mathcal{N}|=M}\mathbb{E}\left[\sum_{m\in\mathcal{N}}\mathbf{z}_x^\top(\boldsymbol{\omega}_m)\mathbf{G}\mathbf{z}_x(\boldsymbol{\omega}_m)\right]\\ &\quad\geq \frac{1}{M}\mathbb{E}\left[\sum_{m\in[M]}\mathbf{z}_x^\top(\boldsymbol{\omega}_m)\mathbf{G}\mathbf{z}_x(\boldsymbol{\omega}_m)\right]\\ &\quad= \mathbb{E}\big[\rho^{(M)}(\text{RCCA})\big] = \rho(\text{KCCA}).\end{aligned} \tag{23}$$

The above equation shows that the total canonical correlation obtained by ORCCA2 provides an upper bound for the total canonical correlation obtained by RFF in the view of $\mathbf{X}$. We can conclude the same thing in the view of $\mathbf{Y}$ following the same procedure. Setting $\mathbf{K}_y = \mathbf{y}\mathbf{y}^\top$, we can also have the same property for ORCCA1. Therefore, Corollary 2 is proven.

F. Computation Complexity

To highlight the computational advantage, we examine the computational cost of the dominating components in KCCA formulation and ORCCA formulation.

1) For KCCA, the dominating component in terms of computational cost is the matrix inversion of the full rank matrices $\mathbf{K}_x + \mu\mathbf{I}$ and $\mathbf{K}_y + \mu\mathbf{I}$, as we can see in (5). Both of them have the time complexity of $\mathcal{O}(n^3)$ as $\mathbf{K}_x, \mathbf{K}_y \in \mathbb{R}^{n\times n}$. The matrix multiplication will also induce a theoretical time complexity of $\mathcal{O}(n^3)$. Therefore, the computational cost of KCCA is $\mathcal{O}(n^3)$.
2) For ORCCA1 and ORCCA2, the dominating component in terms of computational cost comes from calculating the empirical score function (21). The score function needs to be computed once for ORCCA1 and twice for ORCCA2. The matrix inversion of $\mathbf{Z}_x^\top\mathbf{Z}_x + \mu\mathbf{I}$ induces a cost of $\mathcal{O}(M_0^3)$, where M_0 is the number of random features in the pool. The matrix multiplication $\mathbf{Z}_x^\top\mathbf{Z}_x$ induces a cost of $\mathcal{O}(nM_o^2)$. Therefore, the computation cost of ORCCA1 and ORCCA2 is dominated by this term and the cost is $\mathcal{O}(nM_0^2 + M_0^3)$.

ACKNOWLEDGMENT

The authors submitted the first version of this article at Texas A&M University. The authors would like to thank the support of Texas A&M University as well as Texas A&M Triads for Transformation (T3) Program.

REFERENCES

[1] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1177–1184.
[2] D. Richards, P. Rebeschini, and L. Rosasco, "Decentralised learning with random features and distributed gradient descent," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 8105–8115.
[3] Y. Wang and S. Shahrampour, "Distributed parameter estimation in randomized one-hidden-layer neural networks," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2020, pp. 737–742.
[4] Z. Hu, M. Lin, and C. Zhang, "Dependent online kernel learning with constant number of random Fourier features," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2464–2476, Oct. 2015.
[5] T. D. Nguyen, T. Le, H. Bui, and D. Phung, "Large-scale online kernel learning with random feature reparameterization," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 2543–2549.
[6] P.-S. Huang, L. Deng, M. Hasegawa-Johnson, and X. He, "Random features for kernel deep convex network," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 3143–3147.
[7] P. L. Lai and C. Fyfe, "Kernel and nonlinear canonical correlation analysis," *Int. J. Neural Syst.*, vol. 10, pp. 365–377, Oct. 2000.
[8] D. Lopez-Paz, P. Hennig, and B. Schölkopf, "The randomized dependence coefficient," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 1–9.
[9] D. Lopez-Paz, S. Sra, A. Smola, Z. Ghahramani, and B. Schölkopf, "Randomized nonlinear component analysis," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1359–1367.
[10] Z. Yang, A. Wilson, A. Smola, and L. Song, "A la carte—Learning fast kernels," in *Proc. Artif. Intell. Statist.*, 2015, pp. 1098–1106.
[11] W.-C. Chang, C.-L. Li, Y. Yang, and B. Póczos, "Data-driven random Fourier features using stein effect," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 1–7.
[12] A. Sinha and J. C. Duchi, "Learning kernels with random features," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1298–1306.
[13] H. Avron, M. Kapralov, C. Musco, C. Musco, A. Velingker, and A. Zandieh, "Random Fourier features for kernel ridge regression: Approximation bounds and statistical guarantees," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 253–262.
[14] B. Bullins, C. Zhang, and Y. Zhang, "Not-so-random features," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–19.
[15] S. Shahrampour, A. Beirami, and V. Tarokh, "On data-dependent random features for improved generalization in supervised learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 4026–4033.
[16] P. Kar and H. Karnick, "Random feature maps for dot product kernels," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2012, pp. 583–591.
[17] J. Yang, V. Sindhwani, Q. Fan, H. Avron, and M. W. Mahoney, "Random Laplace feature maps for semigroup kernels on histograms," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 971–978.
[18] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, nos. 3–4, p. 321, Dec. 1936.
[19] T. De Bie, N. Cristianini, and R. Rosipal, "Eigenproblems in pattern recognition," in *Handbook Geometric Computing*. Berlin, Germany: Springer, 2005, pp. 129–167.
[20] F. R. Bach and M. I. Jordan, "Kernel independent component analysis," *J. Mach. Learn. Res.*, vol. 3, pp. 1–48, Jan. 2002.
[21] J. Yang, V. Sindhwani, H. Avron, and M. Mahoney, "Quasi-Monte Carlo feature maps for shift-invariant kernels," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 485–493.
[22] Q. Le, T. Sarlós, and A. Smola, "Fastfood-approximating kernel expansions in loglinear time," in *Proc. Int. Conf. Mach. Learn.*, vol. 85, 2013, pp. 1–11.
[23] I. E.-H. Yen, T.-W. Lin, S.-D. Lin, P. K. Ravikumar, and I. S. Dhillon, "Sparse random feature algorithm as coordinate descent in Hilbert space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2456–2464.
[24] A. Rudi and L. Rosasco, "Generalization properties of learning with random features," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3218–3228.
[25] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1313–1320.
[26] X. Y. Felix, A. T. Suresh, K. M. Choromanski, D. N. Holtmann-Rice, and S. Kumar, "Orthogonal random features," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1975–1983.
[27] K. Choromanski, M. Rowland, T. Sarlós, V. Sindhwani, R. Turner, and A. Weller, "The geometry of random features," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 1–9.
[28] A. May *et al.*, "Kernel approximation methods for speech recognition," *J. Mach. Learn. Res.*, vol. 20, no. 59, pp. 1–36, 2019.
[29] F. X. Yu, S. Kumar, H. Rowley, and S.-F. Chang, "Compact nonlinear maps and circulant extensions," 2015, *arXiv:1503.03893*.
[30] J. B. Oliva, A. Dubey, A. G. Wilson, B. Póczos, J. Schneider, and E. P. Xing, "Bayesian nonparametric kernel-learning," in *Proc. Artif. Intell. Statist.*, 2016, pp. 1078–1086.
[31] R. Agrawal, T. Campbell, J. Huggins, and T. Broderick, "Data-dependent compression of random features for large-scale kernel approximation," in *Proc. 22nd Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2019, pp. 1822–1831.
[32] Y. Li, K. Zhang, J. Wang, and S. Kumar, "Learning adaptive random features," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 4229–4236.
[33] S. Mehrkanoon and J. A. K. Suykens, "Regularized semipaired kernel CCA for domain adaptation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3199–3213, Jul. 2018.
[34] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, "Deep canonical correlation analysis," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1247–1255.
[35] T. Michaeli, W. Wang, and K. Livescu, "Nonparametric canonical correlation analysis," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 1967–1976.
[36] H. Lancaster, "The structure of bivariate distributions," *Ann. Math. Statist.*, vol. 29, no. 3, pp. 719–736, 1958.
[37] V. Uurtio, S. Bhadra, and J. Rousu, "Large-scale sparse kernel canonical correlation analysis," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6383–6391.
[38] W. Wang, J. Wang, D. Garber, and N. Srebro, "Efficient globally convergent stochastic optimization for canonical correlation analysis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 766–774.
[39] R. Arora, T. V. Marinov, P. Mianjy, and N. Srebro, "Stochastic approximation for canonical correlation analysis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4775–4784.
[40] W. Wang and K. Livescu, "Large-scale approximate kernel canonical correlation analysis," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–14.

[41] F. Bach, "On the equivalence between kernel quadrature rules and random feature expansions," *J. Mach. Learn. Res.*, vol. 18, no. 21, pp. 1–38, 2017.

[42] Y. Sun, A. Gilbert, and A. Tewari, "But how does it work in theory? Linear SVM with random features," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3383–3392.

[43] Z. Li, J.-F. Ton, D. Oglic, and D. Sejdinovic, "Towards a unified analysis of random Fourier features," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3905–3914.

[44] E. Kokiopoulou, J. Chen, and Y. Saad, "Trace optimization and eigenproblems in dimension reduction methods," *Numer. Linear Algebra Appl.*, vol. 18, no. 3, pp. 565–602, 2011.

[45] S. Shahrampour and S. Kolouri, "On sampling random features from empirical leverage scores: Implementation and theoretical guarantees," 2019, *arXiv:1903.08329*.

[46] W. Wang, X. Yan, H. Lee, and K. Livescu, "Deep variational canonical correlation analysis," 2016, *arXiv:1610.03454*.

[47] A. Benton, H. Khayrallah, B. Gujral, D. A. Reisinger, S. Zhang, and R. Arora, "Deep generalized canonical correlation analysis," in *Proc. 4th Workshop Represent. Learn.*, 2019, pp. 1–14.

[48] H. S. Wong, L. Wang, R. Chan, and T. Zeng, "Deep tensor CCA for multi-view learning," *IEEE Trans. Big Data*, early access, May 11, 2021, doi: 10.1109/TBDATA.2021.3079234.

[49] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neural Comput.*, vol. 16, no. 12, pp. 2639–2664, 2004.

Yinsong Wang received the B.S. degree in mechanical engineering from Shandong University, Jinan, China, in 2017, and the M.S. degree in manufacturing system engineering and management from The Hong Kong Polytechnic University, Hong Kong, in 2019. He is currently pursuing the Ph.D. degree in industrial engineering with Northeastern University, Boston, MA, USA.

His research interests include machine learning, data science, and kernel methods.

Shahin Shahrampour (Senior Member, IEEE) received the Ph.D. degree in electrical and systems engineering, the M.A. degree in statistics (The Wharton School), and the M.S.E. degree in electrical engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2015, 2014, and 2012, respectively.

He is currently an Assistant Professor with the Department of Mechanical and Industrial Engineering, Northeastern University, Boston, MA, USA. His research interests include machine learning, optimization, sequential decision-making, and distributed learning, with a focus on developing computationally efficient methods for data analytics.