Trace and Determinant Kernels between Matrices

S. Kevin Zhou[*]
Center for Automation Research and ECE Department
University of Maryland, College Park MD 20742

Abstract

Kernels between ensembles (or a collection of entities) have recently attracted growing interests in the literature on machine learning . In this paper, we focus on the 'ensemble' that is defined as a collection of vectors. One natural way to interpret such an ensemble is through the notion of matrix. We present two basic reproducing kernels between matrices: namely trace and determinant kernels that can be interpreted using a 'vector' viewpoint as they are in an inner product form between two vectors, explaining the required positive definiteness for a reproducing kernel. Using the 'vector' viewpoint and generic kernel construction rules, we are able to construct more kernels between matrices based on the basic trace and determinant kernels. Further, we also consider column space matrices, possibly arising from matrices of different column sizes, and 'kernerlized' matrices whose columns are mapped to a reproducing kernel Hilbert space.

1 Introduction

Kernel methods play increasingly important roles in the machine learning literature. While developing novel kernel methods is of theoretical interest, constructing kernel functions that are dependent on the nature of the problem at hand has a practical impact.

Real applications call for different data representations. While vector is a very conventional way to represent data, alternative representations include ensembles. Here, the term 'ensemble' can be loosely understood as a collect of entities with additional structures or attributes. Depending on the nature of the structure or attribute imposed on the data, examples of ensembles arising in the literature include strings [11], graphs [10], statistical manifolds [6, 16, 15], probability distributions [7, 9, 12], and so on [3, 19].

In this paper, we consider an ensemble that is a collection of vectors. Roughly speaking, there are three distinct methods to handle such an ensemble. The first is simply to summarize an ensemble by a vector so that any kernel function between vectors can be used. The second is to treat an ensemble as a matrix as in reported in [19] and in this paper. The third is to treat an ensemble as realizations generated from a certain underlying probabilistic model. Kernel functions can be constructed between probabilistic distributions learned from the ensemble. In [7, 9], Bhattacharyya and expected likelihood kernels

[*]Email: shaohua@cfar.umd.edu.

are constructed. In [12], the famous Kullback-Leibler distance has been converted in to a kernel function.

However, the matrix interpretation of ensemble has not yet been fully explored in the machine learning literature. We attempt to bridge the gap by presenting in Section 2 two basic reproducing kernels between matrices: namely trace and determinant kernels. Interestingly, both of these kernels can be interpreted using a 'vector' viewpoint since they are in the form of an inner product between two vectors, explaining the required positive definiteness for a reproducing kernel.

Utilizing the trace and determinant kernels between matrices as building blocks, we are able to construct more kernels between matrices, exploiting the following two modalities:
(a) *'Vector' viewpoint* (Section 3). This viewpoint offers flexible ways to construct new kernels between matrices. For example, the so-called dot product kernels (see Section 3.1 for definition) and isotropic kernels [5] (see Section 3.2 for definition) can be generalized to handle matrix inputs.
(b) *Matrix structure* (Section 4). Matrix can be thought as a structured vector, i.e., the elements are arranged in a rectangular array. One important concept in matrix theory is column space that uniquely characterizes an equivalent class of matrices. In other words, for any two matrices belonging to this equivalent class, they share a common column space. Thus, we can deal with matrices with different sizes in terms of their column spaces. In addition, we also process a 'kernelized' matrix that nonlinearly maps each of its column vector to a reproducing kernel Hilbert space (RKHS). The problem here is to avoid the need to explicitly know the nonlinear mapping by casting the computation as a dot product and then using the so-called 'kernel trick'.

We proceed by reviewing conditions that a reproducing kernel function must satisfy. Let $\mathcal{E}$ be a set. A two variable function $k(x, y)$ on $\mathcal{E} \times \mathcal{E}$ is a *reproducing kernel* if for any finite point set $\{x_1, x_2, ..., x_n\}$ and for any corresponding real numbers $\{a_1, a_2, ..., a_n\}$,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j k(x_i, x_j) \geq 0. \tag{1}$$

2 Trace and Determinant Kernels between Matrices

Suppose X and Y are two $d \times c$ matrices belonging to a certain matrix space $\Omega_{d\times c}$ (so here $\mathcal{E} = \Omega_{d\times c}$). We are interested in defining a kernel function $k(\mathsf{X}, \mathsf{Y})$.

The proposed kernels, namely trace and determinant kernels, are based on the so-called dot product matrix (or Gram matrix). The dot product matrix between two matrices X and Y is given by $\mathsf{X}^{\mathsf{T}}\mathsf{Y}$.

2.1 Trace kernel between matrices

Suppose $\mathsf{X} = [x_{ij}]_{d\times c}$ and $\mathsf{Y} = [y_{ij}]_{d\times c}$, the trace kernel between matrices X and Y (or matrix trace kernel or trace kernel) is defined as

$$k_{\bullet}(\mathsf{X}, \mathsf{Y}) = \mathtt{tr}(\mathsf{X}^{\mathsf{T}}\mathsf{Y}) = \sum_{i=1}^{d}\sum_{j=1}^{c} x_{ij} y_{ij} = \mathtt{tvec}(\mathsf{X})^{\mathsf{T}}\mathtt{tvec}(\mathsf{Y}), \tag{2}$$

where the operator $\mathtt{tvec} : \Omega_{d\times c} \to \mathcal{R}^{dc}$ converts a $d \times c$ matrix to a $dc \times 1$ vector by arranging all the elements of the matrix according to a fixed order, say the lexicographic order.

The function $k_\bullet(\mathsf{X},\mathsf{Y})$ is a kernel function because

$$\sum_{i,j=1}^{n} a_i a_j k_\bullet(\mathsf{X}_i,\mathsf{X}_j) = \sum_{i,j=1}^{n} a_i a_j \mathtt{tvec}(\mathsf{X}_i)^{\mathsf{T}}\mathtt{tvec}(\mathsf{X}_j) = \{\sum_{i=1}^{n} a_i \mathtt{tvec}(\mathsf{X}_i)\}^{\mathsf{T}}\{\sum_{j=1}^{n} a_j \mathtt{tvec}(\mathsf{X}_j)\} \geq 0. \tag{3}$$

The 'vector' viewpoint raises the question about the originality of this kernel function. However, the operation of 'vectorization' damages the structure of a matrix. In order to differentiate the matrix trace kernel from a conventional kernel for vectors, we will bring back the matrix structure in Section 4, offering additional advantages. Incidentally, the trace of the dot product matrix has been used in [4] to perform an alignment of the Gram matrix to an ideal case.

2.2 Determinant kernel between matrices

Further suppose that the matrix dimension d and c satisfying $d \geq c$, the determinant kernel between matrices X and Y (or matrix determinant kernel or determinant kernel) is defined as

$$k_\star(\mathsf{X},\mathsf{Y}) = \mathtt{det}(\mathsf{X}^{\mathsf{T}}\mathsf{Y}) = \mathtt{dvec}(\mathsf{X})^{\mathsf{T}}\mathtt{dvec}(\mathsf{Y}). \tag{4}$$

where the operator $\mathtt{dvec} : \Omega_{d\times c} \rightarrow \mathcal{R}^p$ with $p = \left(\begin{array}{c} d \\ c \end{array}\right)$ defines the Grassman vector of a matrix of size $d \times c$ whose elements are the c^{th}-order minors of X constructed in a lexicographic order. The Grassman vector is also known as the c^{th} compound of the matrix. The second equation in (4) is guaranteed by a special case of the Binet-Cauchy theorem [1, 19]. Using the same argument as in (3), the function $k_\star(\mathsf{X},\mathsf{Y})$ is positive definite.

3 Generalized Trace and Determinant Kernels between Matrices

To generalize the trace and determinant kernels to arrive at more kernels between matrices, we exploit their common 'vector' viewpoint and several generic kernel construction rules [5], among which the followings are of particular interest:
R_1 *(summation)*. If $k_1(x,y)$ and $k_2(x,y)$ are two kernels defined on $\mathcal{E}\times\mathcal{E}$, and a_1 and a_2 are nonnegative real numbers, then $k(x,y) = a_1k_1(x,y) + a_2k_2(x,y)$ is a kernel function.
R_2 *(product)*. If $k_1(x,y)$ and $k_2(x,y)$ are two kernels defined on $\mathcal{E}\times\mathcal{E}$, then $k(x,y) = k_1(x,y)k_2(x,y)$ is a kernel function.
R_3. If $g : \mathcal{E} \rightarrow \mathcal{E}'$ and $k_0(x,y)$ is a kernel function defined on $\mathcal{E}'\times\mathcal{E}'$, then $k(x,y) = k_0(g(x),g(y))$ is a kernel function.

3.1 Dot product kernels

There are many kernel functions of the following form:

$$k(x,y) = k_{\mathsf{D}}(x^{\mathsf{T}}y). \tag{5}$$

Since the input to k_D is a dot product, we call this type of kernels as dot product kernels. A list of k_D functions is provided in Table 1. The positive definiteness of these kernels [1] can be verified by applying the aforementioned kernel construction rules.

By letting g be a mapping $g : \Omega_{d\times c} \rightarrow \mathcal{R}^{dc}$ such that $g(\mathsf{X}) = \mathtt{tvec}(\mathsf{X})$ and k_{D} be an $\mathcal{R}$-valued function satisfying (5), we know from rule R_3 that the function $k_{\mathsf{D}\bullet}(\mathsf{X},\mathsf{Y})$ defined below is a kernel function:

$$k_{\mathsf{D}\bullet}(\mathsf{X},\mathsf{Y}) = k_{\mathsf{D}}(\mathtt{tvec}(\mathsf{x})^{\mathsf{T}}\mathtt{tvec}(\mathsf{Y})) = k_{\mathsf{D}}(k_\bullet(\mathsf{X},\mathsf{Y})). \tag{6}$$

[1]The neural network kernel, albeit widely used, is in fact not a positive kernel.

Name of kernel	The $k_{\mathtt{D}}$ function	Name of kernel	The $k_{\mathtt{D}}$ function
Dot product	$k_{\mathtt{D}}(a) = a$	Neural network	$k_{\mathtt{D}}(a) = tanh(a + \theta)$
Polynomial	$k_{\mathtt{D}}(a) = (a + \theta)^p;$	Polynomial (generic)	$k_{\mathtt{D}}(a) = \sum_{n=1}^{N} \alpha_n a^n, \ \ \alpha_n \geq 0;$

Table 1: A list of dot product kernels.

Name of kernel	The $k_{\mathtt{I}}$ function	Name of kernel	The $k_{\mathtt{I}}$ function
Exponential	$k_{\mathtt{I}}(a) = exp(-\frac{a}{\theta})$	Gaussian (RBF)	$k_{\mathtt{I}}(a) = exp(-\frac{a^2}{2\theta^2})$
Inverse multiquadratic	$k_{\mathtt{I}}(a) = 1/\sqrt{a^2 + \theta^2}$	Thin plate splines	$k_{\mathtt{I}}(a) = a^2 \log a$

Table 2: A list of isotropic kernels.

Likewise, we can construct the following kernel function:

$$k_{\mathtt{D}\star}(\mathsf{X}, \mathsf{Y}) = k_{\mathtt{D}}(\mathtt{dvec}(\mathsf{X})^{\mathsf{T}}\mathtt{dvec}(\mathsf{Y})) = k_{\mathtt{D}}(k_{\star}(\mathsf{X}, \mathsf{Y})). \tag{7}$$

3.2 Isotropic kernels

An important class of kernels is the stationary kernels [5], which are characterized by the translation invariant property: $k(x, y) = k_{\mathtt{S}}(x - y)$. Stationary kernels can be elegantly analyzed using a spectral representation [5]. One special subclass of the stationary kernel is isotropic kernel (or homogeneous kernel), which is only a function of the distance:

$$k(x, y) = k_{\mathtt{I}}(||x - y||). \tag{8}$$

A list of $k_{\mathtt{I}}$ functions is provided in Table 2.

By utilizing the following fact that

$$\{\mathtt{tvec}(\mathsf{X}) - \mathtt{tvec}(\mathsf{Y})\}^{\mathsf{T}}\{\mathtt{tvec}(\mathsf{X}) - \mathtt{tvec}(\mathsf{Y})\} = k_{\bullet}(\mathsf{X}, \mathsf{X}) + k_{\bullet}(\mathsf{Y}, \mathsf{Y}) - 2k_{\bullet}(\mathsf{X}, \mathsf{Y}), \tag{9}$$

we generalize the isotropic kernels to handle matrix inputs. By letting g be a mapping $g : \Omega_{d\times c} \rightarrow \mathcal{R}^{dc}$ such that $g(\mathsf{X}) = \mathtt{tvec}(\mathsf{X})$ and $k_{\mathtt{I}}$ an $\mathcal{R}$-valued function satisfying (8), we know from rule R_3 that the function $k_{\mathtt{I}\bullet}(\mathsf{X}, \mathsf{Y})$ defined below is a kernel function:

$$k_{\mathtt{I}\bullet}(\mathsf{X}, \mathsf{Y}) = k_{\mathtt{I}}(||\mathtt{tvec}(\mathsf{x}) - \mathtt{tvec}(\mathsf{Y})||) = k_{\mathtt{I}}(\sqrt{k_{\bullet}(\mathsf{X}, \mathsf{X}) + k_{\bullet}(\mathsf{Y}, \mathsf{Y}) - 2k_{\bullet}(\mathsf{X}, \mathsf{Y})}\,). \tag{10}$$

Similarly, we can construct the following kernel function:

$$k_{\mathtt{I}\star}(\mathsf{X}, \mathsf{Y}) = k_{\mathtt{I}}(\sqrt{k_{\star}(\mathsf{X}, \mathsf{X}) + k_{\star}(\mathsf{Y}, \mathsf{Y}) - 2k_{\star}(\mathsf{X}, \mathsf{Y})}\,) \tag{11}$$

4 Trace and Determinant Kernels between Column Space Matrices and 'Kernelized' Matrices

4.1 Column space matrix

In real applications, it is often the case that we have heterogeneous matrices with different column sizes. Typically, the matrices $\mathsf{X}_{d\times c_x}$ and $\mathsf{Y}_{d\times c_y}$ have the same number of rows but different number of columns, i.e., $c_x \neq c_y$. To compare such matrices, we need to summarize them using meaningful quantities of the same size. For example, a naive way is to summarize a matrix by its column mean. We propose to use the column space matrix since matrices are deemed equivalent if they span the same column space.

Therefore, for matrices of different column sizes, we can just use their column spaces. To find the column space, we invoke the singular value decomposition (SVD). Suppose that the

SVD for $\mathsf{X}_{d\times c_x}$ is given by $\mathsf{X} = \mathsf{UDV}^{\mathsf{T}}$, the matrix U characterizes the column space of the X matrix and is a unique representation up to a diagonal matrix $\mathsf{R}_{c\times c} = \mathsf{D}[r_1, r_2, \ldots, r_c]$ whose diagonal elements are either one or minus one. In other words, UR is also a valid matrix from the SVD. In practice, we keep only those column vectors (or left singular vectors) in U corresponding to the top c ($c \le c_x$) singular values and store them in U_x called as the column space matrix. Because $\mathsf{XX}^{\mathsf{T}} = \mathsf{UD}^2\mathsf{U}^{\mathsf{T}}$, U is also the matrix encoding the eigenvectors for XX^{T}. In addition, even for matrices of the same column sizes, we can still summarize them using their column space matrices.

Using U_x for $\mathsf{X}_{d\times c_x}$ and U_y for $\mathsf{Y}_{d\times c_y}$, where U_x and U_y are of the same dimension $d \times c$ with $c \le min(c_x, c_y)$, we define the following kernels:

$$k_{\mathsf{U}\bullet}(\mathsf{X},\mathsf{Y}) = \mathtt{tr}(\mathsf{U}_x^{\mathsf{T}}\mathsf{U}_y);\ \ k_{\mathsf{U}\star}(\mathsf{X},\mathsf{Y}) = \mathtt{det}(\mathsf{U}_x^{\mathsf{T}}\mathsf{U}_y). \tag{12}$$

We still need to remove the ambiguity in the U matrix since this ambiguity causes different $k_{\mathsf{U}\bullet}$ and $k_{\mathsf{U}\star}$ values in different implementations. For $k_{\mathsf{U}\bullet}(\mathsf{X},\mathsf{Y})$,

$$k_{\mathsf{U}\bullet}(\mathsf{X},\mathsf{Y}) = \mathtt{tr}(\mathsf{R}_x^{\mathsf{T}}\mathsf{U}_x^{\mathsf{T}}\mathsf{U}_y\mathsf{R}_y) = \mathtt{tr}(\mathsf{RU}_x^{\mathsf{T}}\mathsf{U}_y); \tag{13}$$

where $\mathsf{R} = \mathsf{R}_y\mathsf{R}_x^{\mathsf{T}}$ and it is hence not unique. To make $k_{\mathsf{U}\bullet}(\mathsf{X},\mathsf{Y})$ unique, we redefine it, using the fact that $\mathsf{RR} = \mathtt{I}$, as

$$k_{\mathsf{U}\bullet}(\mathsf{X},\mathsf{Y}) = \mathtt{tr}(\{\mathsf{U}_x^{\mathsf{T}}\mathsf{U}_y\}.^{\wedge 2}), \tag{14}$$

where $\{\circ\}.^{\wedge 2}$ is a MATLAB operator that squares every element[2]. It is easy to show that the above function $k_{\mathsf{U}\bullet}(\mathsf{X},\mathsf{Y})$ is positive definite.

Similarly for $k_{\mathsf{U}\star}(\mathsf{X},\mathsf{Y})$,

$$k_{\mathsf{U}\star}(\mathsf{X},\mathsf{Y}) = \mathtt{det}(\mathsf{R}_x^{\mathsf{T}}\mathsf{U}_x^{\mathsf{T}}\mathsf{U}_y\mathsf{R}_y) = (-1)^q\mathtt{det}(\mathsf{U}_x^{\mathsf{T}}\mathsf{U}_y) \tag{15}$$

where q is an arbitrary integer. To make $k_{\mathsf{U}\bullet}(\mathsf{X},\mathsf{Y})$ unique, we redefine it as

$$k_{\mathsf{U}\star}(\mathsf{X},\mathsf{Y}) = \{\mathtt{det}(\mathsf{U}_x^{\mathsf{T}}\mathsf{U}_y)\}^2. \tag{16}$$

Again, it is easy to show that the above function $k_{\mathsf{U}\star}(\mathsf{X},\mathsf{Y})$ is positive definite.

4.2 'Kernelized' matrix

Directly applying the 'kernel trick' to $\mathtt{tvec}(.)$ and $\mathtt{dvec}(.)$ has been investigated in Section 3. We now explore another possibility of applying the 'kernel trick' to the matrix (rather columns of the matrix).

Suppose that the mapping $\varphi : \mathcal{R}^d \to \mathcal{H}_l$ maps a vector in $\mathcal{R}^d$ to a RKHS induced by a kernel function l defined on $\mathcal{R}^d \times \mathcal{R}^d$ and the cardinality of the RKHS is f (f could be ∞). In other words, for any two vectors $\mathsf{x},\mathsf{y} \in \mathcal{R}^d$, we have $l(\mathsf{x},\mathsf{y}) = \varphi(\mathsf{x})^{\mathsf{T}}\varphi(\mathsf{y})$.

Define a 'kernelized' matrix as

$$\varphi(\mathsf{X}_{d\times c}) = [\varphi(\mathsf{x}_1), \varphi(\mathsf{x}_2), \ldots, \varphi(\mathsf{x}_c)]_{f\times c}, \tag{17}$$

where x_i's are column vectors of the matrix X. The Gram matrix between two 'kernelized' matrices is given by

$$\mathsf{L}_\varphi(\mathsf{X},\mathsf{Y}) = \varphi(\mathsf{X})^{\mathsf{T}}\varphi(\mathsf{Y}) = [l(\mathsf{x}_i,\mathsf{y}_j)]_{c\times c}. \tag{18}$$

So, the kernels between 'kernelized' matrices are given by

$$k_{\varphi\bullet}(\mathsf{X},\mathsf{Y}) = \mathtt{tr}(\mathsf{L}_\varphi(\mathsf{X},\mathsf{Y})) = \sum_{i=1}^{c} l(\mathsf{x}_i,\mathsf{y}_i);\ \ k_{\varphi\star}(\mathsf{X},\mathsf{Y}) = \mathtt{det}(\mathsf{L}_\varphi(\mathsf{X},\mathsf{Y})). \tag{19}$$

Here we are defining the kernel function k based on another kernel function l. In the literature, hyperkernels [13] and Bhattacharyya kernels [9] are defined on kernels.

[2]In fact, to compute $k_{\mathsf{U}\bullet}(\mathsf{X},\mathsf{Y})$, we only need to square the diagonal elements and add them up.

	Dot product kernel	Isotropic kernel
Column space matrix	$k_{\mathtt{D}\Sigma\bullet}$; $k_{\mathtt{DU}\bullet}$; $k_{\mathtt{D}\Sigma\star}$; $k_{\mathtt{DU}\star}$	$k_{\mathtt{I}\Sigma\bullet}$; $k_{\mathtt{IU}\bullet}$; $k_{\mathtt{I}\Sigma\star}$; $k_{\mathtt{IU}\star}$
'Kernelized' matrix	$k_{\mathtt{D}\varphi\bullet}$; $k_{\mathtt{D}\varphi\star}$	$k_{\mathtt{I}\varphi\bullet}$; $k_{\mathtt{I}\varphi\star}$
Column space matrix of 'Kernelized' matrix	$k_{\mathtt{D}\Sigma\varphi\bullet}$; $k_{\mathtt{DU}\varphi\bullet}$; $k_{\mathtt{D}\Sigma\varphi\star}$; $k_{\mathtt{DU}\varphi\star}$	$k_{\mathtt{I}\Sigma\varphi\bullet}$; $k_{\mathtt{IU}\varphi\bullet}$; $k_{\mathtt{I}\Sigma\varphi\star}$; $k_{\mathtt{IU}\varphi\star}$

Table 3: Various combinations of constructing kernels between matrices.

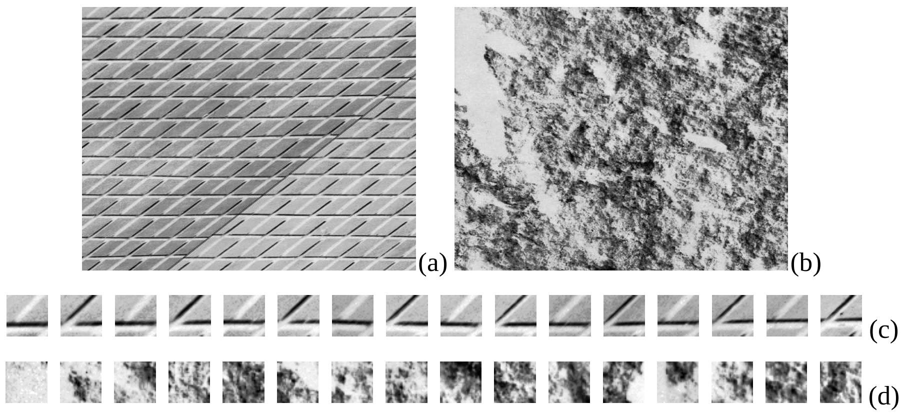

Figure 1: Original texture images of (a) 'D1'and (b) 'D2' with size 512×512. Small patches of 'D1' and 'D2' of size 32×32 drawn from the original texture images.

4.3 Column space matrix of 'kernelized' matrix

The key is to cast the involved computation into dot products. We summarize in Appendix a standard technique [14, 8, 17, 19] to obtain the column space matrix for a 'kernelized' matrix $\varphi(\mathsf{X})$. The resulting $\mathsf{U}_{\varphi x}$ is $\mathsf{U}_{\varphi x} = \varphi(\mathsf{X})\mathsf{V}_x\Lambda_x^{-1/2}$ with V_x encoding the top eigenvectors of $\mathsf{L}_\varphi(\mathsf{X},\mathsf{X})$ corresponding to its top c eigenvalues that are further encoded in the diagonal Λ_x matrix.

Therefore, the trace and determinant kernels are

$$k_{\mathsf{U}\varphi\bullet}(\mathsf{X},\mathsf{Y}) = \mathtt{tr}(\{\mathsf{U}_{\varphi x}^{\mathsf{T}}\mathsf{U}_{\varphi y}\}.\wedge^2) = \mathtt{tr}(\{\Lambda_x^{-1/2}\mathsf{V}_x^{\mathsf{T}}\mathsf{L}_\varphi(\mathsf{X},\mathsf{Y})\mathsf{V}_y\Lambda_y^{-1/2}\}.\wedge^2). \tag{20}$$

$$k_{\mathsf{U}\varphi\star}(\mathsf{X},\mathsf{Y}) = \{\mathtt{det}(\{\mathsf{U}_{\varphi x}^{\mathsf{T}}\mathsf{U}_{\varphi y}\})\} = \{\mathtt{det}(\Lambda_x^{-1/2}\mathsf{V}_x^{\mathsf{T}}\mathsf{L}_\varphi(\mathsf{X},\mathsf{Y})\mathsf{V}_y\Lambda_y^{-1/2})\}^2. \tag{21}$$

The kernel function $k_{\mathsf{U}\varphi\star}(\mathsf{X},\mathsf{Y})$ is the choice proposed in [19].

Finally, we can use the same construction techniques presented in Section 3 to construct more kernels between matrices of different column sizes, 'kernelized' matrices, and 'kernelized' matrices of different column sizes. For example, the isotropic trace kernel between 'kernelized' matrices of different column sizes using the column space representation is defined as

$$k_{\mathtt{IU}\varphi\bullet}(\mathsf{X},\mathsf{Y}) = k_{\mathtt{I}}(\sqrt{k_{\mathsf{U}\varphi\bullet}(\mathsf{X},\mathsf{X}) + k_{\mathsf{U}\varphi\bullet}(\mathsf{Y},\mathsf{Y}) - 2k_{\mathsf{U}\varphi\bullet}(\mathsf{X},\mathsf{Y})}\,). \tag{22}$$

Table 3 lists all possible ways of constructing a variety of kernels between matrices.

5 Experiments

We conducted preliminary texture classification experiments using the two textures 'D1' and 'D2' (as shown in Figure 1) extracted from the Brodatz's album [2]. Each original

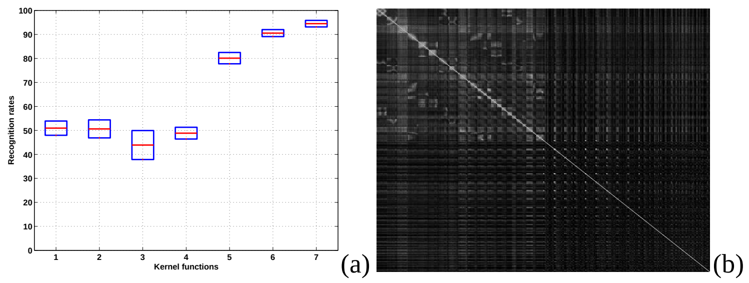

Figure 2: (a) The recognition accuracy obtained by SVM with the following kernel: (1) $k_\bullet$ as in Eq. (2); (2) $k_\star$ as in Eq. (4); (3) $k_{\mathtt{U}_\bullet}$ as in Eq. (14); (4) $k_{\mathtt{U}_\star}$ as in Eq. (16); (5) $k_{\mathtt{I}_\bullet}$ as in Eq. (10); (6) $k_{\mathtt{D}_{\varphi\bullet}}$ as in Table 3; (7) $k_{\mathtt{I}_{\varphi\bullet}}$ as in Table 3. (b) The Gram matrix for the 7^{th} kernel $k_{\mathtt{I}_{\varphi\bullet}}$.

texture image of size 512×512 is divided into 256 small non-overlapping patches of size 32×32. The image intensities are normalized to [0,1]. We randomly selected 128 data points per class for training and the remaining 128 for testing. We trained a support vector machine (SVM) [18] using the training data and used it to classify the testing data. For comparison, different kernel functions are used in the SVM implementation. We repeated this random division 20 times and Figure 2(a) reports the average recognition rates with their standard deviations.

The parameters associated with each kernel are just empirical choices. To derive optimal choices, a systematic study e.g. cross-validation can be employed. The actual choices used are the following. There is no parameter involved in the 1^{st} and 2^{nd} kernels; For the 3^{rd} and 4^{th} kernels, we set $c = 4$; For the 5^{th} kernel $k_{\mathtt{I}_\bullet}$, we set $k_{\mathtt{I}}$ as the Gaussian kernel with kernel width $\theta = 32$. This is equivalent to the RBF kernel based on vectors of size 1024×1 by 'vectorizing' the images of 32×32; For the 6^{th} kernel $k_{\mathtt{D}_{\varphi\bullet}}$, we set $k_{\mathtt{D}}$ as the polynomial kernel $k_{\mathtt{D}}(a) = a^6$ and the kernel function $l(\mathsf{x}, \mathsf{y})$ (where $\mathsf{x}, \mathsf{y} \in \mathcal{R}^{32}$) associated with the nonlinear function φ as the RBF kernel with $\theta = 4$; For the 7^{th} kernel $k_{\mathtt{I}_{\varphi\bullet}}$, we set $k_{\mathtt{I}}$ as the RBF kernel with $\theta = 1$ and the kernel function l as the RBF kernel with $\theta = 4$.

Using the raw matrix kernels (the 1^{st} and 2^{nd} kernels) yields classification accuracy equivalent to equiprobable choice. Using the 3^{rd} and 4^{th} kernels that involve summarization by the column space matrix does not increase accuracy. By using the RBF kernel between vectors, i.e. the 5^{th} kernel, we increase the accuracy to above 80%. However, the 'vectorization' operator loses the spatial distribution of pixels. Using the 6^{th} and 7^{th} kernels, i.e., kernels between the 'kernelized' matrices, the classification accuracy is above 90% (95% for the 7^{th} kernel). The Gram matrix of all data points for the the 7^{th} kernel is given in Figure 2(b). It is possible to increase the accuracy by tuning the parameters. Since the 'kernelized' matrix keeps the spatial distribution of pixels, we conjectur that *kernel functions between 'kernelized' matrices capture the spatial statistics that are necessary for texture analysis*.

We are now pursuing substantial investigations on other recognition tasks such as gait recognition using the proposed matrix kernels. Gait here refers to the pattern presented when human walks and a walking cycle in gait exhibits the transition between stances (see Figure 3). Automatic synchronization of the stances of gait video sequences belonging to different people is crucial for accurate gait recognition but is very difficult. Using the matrix whose columns encode stances exhibited by the video as input, its column space matrix naturally presents a 'stance-free' representation appropriate for gait analysis.

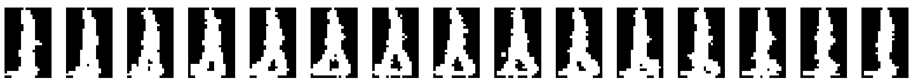

Figure 3: A sequence of 15 stances forming a half walking cycle.

6 Conclusions

We have proposed two kernels between matrices, the trace and determinant kernels, and subsequently constructed a family of kernels between matrices using the 'vector' viewpoint of two basic matrix kernels and the matrix structure. This family of matrix kernels is very flexible: (i) a variety of dot product and isotropic kernels can be used; (ii) various methods of summarizing matrices of different column sizes in a sensible way can be readily adopted to generate new matrix kernels; and (iii) the kernel functions for the 'kernelized' matrix can be chosen freely.

References

[1] A. Aitken. *Determinants and marices*. Interscience Publisher, New York, 1956.

[2] P. Brodatz. *Textures: a photographic album for artists and designers*. Dover, New York, 1966.

[3] M. Collins and N. Duffy. Convolution kernels for natural language. *NIPS*, 14:625–632, 2002.

[4] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. *NIPS*, 2002.

[5] M. Gentor. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2001.

[6] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. *NIPS*, 11, 1999.

[7] T. Jebara and R. Kondor. Bhattarcharyya and expected likelihood kernels. *COLT*, 2003.

[8] M. Kirby and L. Sirovich. Application of karhunen-loéve procedure of the characterization of human faces. *IEEE Trans. PAMI*, 12:103–108, 1990.

[9] R. Kondor and T. Jebara. A kernel between sets of vectors. *ICML*, 2003.

[10] R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. *ICML*, 2002.

[11] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

[12] P. Moreno, P. Ho, and N. Vasconcelos. A Kullback-Leibler divergence based kernel for svm classfication in multimedia applications. *NIPS*, 2003.

[13] C. S. Ong, A. Smola, and R. Williams. Hyperkernels. *NIPS*, 16, 2004.

[14] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[15] M. Seeger. Covariances kernel from Bayesian generative models. *NIPS*, 14:905–912, 2002.

[16] K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K. Müller. A new discriminative kernel from probabilistic models. *NIPS*, 14, 2002.

[17] M. Turk and A. Pentland. Eigenfaces for recognition. *J. of Cogn. Neuroscience*, 3:72–86, 1991.

[18] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[19] L. Wolf and A. Shashua. Learning over sets using kernel principal angles. *Journal of Machine Learning Research*, 4:895–911, 2003.

Appendix: Deriving column space matrix $\mathsf{U}_{\varphi x}$

Suppose that $(\lambda_x, \mathsf{v}_x)$ with $\mathsf{v}^\mathsf{T}\mathsf{v} = 1$ is an eigenpair for the matrix $\mathsf{L}_\varphi(\mathsf{X},\mathsf{X}) = \varphi(\mathsf{X})^\mathsf{T}\varphi(\mathsf{X})$, i.e., $\varphi(\mathsf{X})^\mathsf{T}\varphi(\mathsf{X})\mathsf{v}_x = \lambda_x\mathsf{v}_x$. It is easy to check that $(\gamma_x = c_x^{-1}\lambda_x, \lambda_x^{-1/2}\varphi(\mathsf{X})\mathsf{v}_x)$ is an eigenpair for $\varphi(\mathsf{X})\varphi(\mathsf{X})^\mathsf{T}$. The matrix $\mathsf{U}_{\varphi x}$ in fact contains the eigenvalues for the matrix $\varphi(\mathsf{X})\varphi(\mathsf{X})^\mathsf{T}$. Hence, If only the top c components are kept,

$$\mathsf{U}_{\varphi x} = \varphi(\mathsf{X})\mathsf{V}_x\Lambda_x^{-1/2},$$

where $\mathsf{V}_x = [\mathsf{v}_{x,1}, \mathsf{v}_{x,2}, ..., \mathsf{v}_{x,c}]_{c_x \times c}$ contains the eigenvectors corresponding to the top c eigenvalues of $\mathsf{L}_\varphi(\mathsf{X},\mathsf{X})$ that are contained in a diagonal matrix $\Lambda_x = \mathsf{D}[\lambda_{x,1}, \lambda_{x,2}, \ldots, \lambda_{x,c}]$ with diagonal elements being $\{\lambda_{x,1}, \lambda_{x,2}, \ldots, \lambda_{x,c}\}$.