Kernelization of matrix updates, when and how?

Manfred K. Warmuth[1] ⋆, Wojciech Kotλowski[2] ⋆⋆, and Shuisheng Zhou[3] ⋆⋆⋆

[1] Department of Computer Science, University of California, Santa Cruz, CA 95064
manfred@cse.ucsc.edu
[2] Institute of Computing Science, Poznań University of Technology, Poland
wkotlowski@cs.put.poznan.pl
[3] School of Science, Xidian University, Xian, China, 710071
sszhou@mail.xidian.edu.cn

Abstract. We define what it means for a learning algorithm to be kernelizable in the case when the instances are vectors, asymmetric matrices and symmetric matrices, respectively. We can characterize kernelizability in terms of an invariance of the algorithm to certain orthogonal transformations. If we assume that the algorithm's action relies on a linear prediction, then we can show that in each case the linear parameter vector must be a certain linear combination of the instances. We give a number of examples of how to apply our methods. In particular we show how to kernelize multiplicative updates for symmetric instance matrices.

Keywords: Kernelization, multiplicative updates, rotational invariance.

1 Introduction

The following kernelization trick was popularized by a paper on support vector machines [**?**] and has become one of the most successful methods in machine learning: Any algorithm that reduces to computing dot products between instance vectors $\boldsymbol{x} \in \mathbb{R}^n$ can be enhanced by a feature map that maps the instances $\boldsymbol{x}$ to $\phi(\boldsymbol{x}) \in \mathbb{R}^N$ as long as there is a kernel function available which efficiently computes the dot products $\phi(\boldsymbol{x})'\phi(\widetilde{\boldsymbol{x}})$ between expanded instances. The dimension N of the expanded instance is typically much larger than the dimension n of the original instances and even may be infinite. Complicated neural nets are often beaten by simple linear models which are enhanced with a carefully chosen problem specific feature map or kernel function. The resulting algorithms only access the expanded instances $\phi(\boldsymbol{x})$ via the kernel function $k(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) = \phi(\boldsymbol{x})'\phi(\widetilde{\boldsymbol{x}})$, i.e. the components of the feature vectors are never accessed.

In this paper we discuss kernel methods in the matrix domain. We begin by considering instances that are outer products $\boldsymbol{x}\boldsymbol{y}'$, where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^m$ (it is easy to generalize from outer products to asymmetric instance matrices

⋆ The author was supported by NSF grants IIS-0325363 and IIS-0917397
⋆⋆ This research was done while visiting UCSC supported by the NSF grant IIS-0917397
⋆⋆⋆ This research was done while visiting UCSC supported by the NNSFC grant61179040

$\boldsymbol{M} \in \mathbb{R}^{n\times m}$). As long as algorithms only rely on dot products between pairs $\boldsymbol{x}, \widetilde{\boldsymbol{x}}$ of *left* instances and dot products between pairs $\boldsymbol{y}, \widetilde{\boldsymbol{y}}$ of *right* instances, then we can expand the left instances $\boldsymbol{x}$ via a feature map $\phi(\boldsymbol{x})$ and the right $\boldsymbol{y}$ instances via a second feature map $\psi(\boldsymbol{y})$. Note that matrix parameters can model all interactions between components, and therefore can take second order information into account. We also consider a case when the instances are symmetric products of the form $\boldsymbol{x}\boldsymbol{x}'$, with a single feature map $\boldsymbol{x}\boldsymbol{x}' \mapsto \phi(\boldsymbol{x})\phi(\boldsymbol{x})'$.

The goal of this paper is to give "if and only if" conditions for kernelizable algorithms. We do this for three cases: vector instances, asymmetric matrix instances and symmetric matrix instances under the assumption that the algorithm is linear and produces a unique solution. The vector case has been largely worked out in [**?**], but we rephrase it here mainly as a reference for comparison. The matrix cases are the main contribution of the paper. We define an algorithm to be *kernelizable* if its output depends on the data only via the kernel matrix (matrices) which contains the dot products between the instance vectors. We next give a simple equivalent characterization in each case in terms of certain geometric invariance properties of the algorithm[4]. In the vector case, multiplying the instance by an orthogonal matrix must essentially keep the algorithm unchanged. In the asymmetric matrix case, the algorithm must produce the same output if the instance matrices are left and right multiplied by two orthogonal matrices. The symmetric matrix case gives the invariance under left and right multiplication by the same orthogonal matrix.

The main point of the paper is to show that in each case, if the output of the algorithm is a linear function of the input, then the algorithm is kernelizable iff the linear parameter vector/matrix is a linear combination of the instances and remains invariant under an appropriate orthogonal transformation. In particular, in the vector case the parameter vector $\boldsymbol{w}$ must be a linear combination of the instance vectors, $\boldsymbol{w} = \sum_i c_i \boldsymbol{x}_i$. When the instances are asymmetric outer products $\boldsymbol{x}_i\boldsymbol{y}_i'$, then the parameter matrix must have the form $\boldsymbol{W} = \sum_{i,j} c_{i,j}\ \boldsymbol{x}_i\boldsymbol{y}_j'$. For the symmetric outer products $\boldsymbol{x}_i\boldsymbol{x}_i'$, the symmetric parameter matrix must have the form $\boldsymbol{W} = c\boldsymbol{I} + \sum_{i,j} c_{i,j}\ \boldsymbol{x}_i\boldsymbol{x}_j'$, where $\boldsymbol{I}$ is the identity matrix in $\mathbb{R}^n$ and $c_{i,j} = c_{j,i}$. The presence of an additional identity term $\boldsymbol{I}$ in the expansion for symmetric matrices stems from the existence of a unique element that is invariant under all orthogonal transformations. Such an element does not exist for asymmetric matrices.

We then prove versions of the Representer Theorem for both asymmetric and symmetric outer products. This helps us to develop a number of methods for building kernelizable algorithms from optimization problems. In particular, we give methods for kernelizing the matrix versions of various "multiplicative" update algorithms [**?**,**?**,**?**]. This family of algorithms is motivated by using the quantum relative entropy as a regularization, and methods from online learning can be used to prove regret bounds that grow logarithmically in the dimensions

[4] Although invariance is with respect to orthogonal transformations, we use the term *rotational invariance* rather than *orthogonal invariance*, as the former term is commonly used in the literature.

of the vectors. The logarithmic dependence lets us use high dimensional feature spaces. Moreover, we show that if the loss function is negative (i.e., we are maximizing gains rather than minimizing losses), then the logarithmic dependence on the dimension can be reduced to the logarithmic dependence on the rank of the kernel matrix. For outer product instances, this rank is at most the number of instances T. Multiplicative algorithms learn well when there is a low-rank matrix that can accurately explain the labels [**?**]. The kernel method greatly enhances the applicability of multiplicative algorithms because now we can expand the instances to outer products of high-dimensional feature vectors and still obtain efficient algorithms as long as the instance matrices have low total rank.

Relationship to previous work: One way to ensure kernelizability in the vector case is to apply the Representer Theorem [**?**,**?**]. It states that whenever the solution minimizes the trade-off between the square Euclidean distance and a loss function that only depends on the dot products between the weight vector and feature vector, then the solution is always a linear combination of the feature vectors. Representer type theorems have recently been generalized to the case of outer product instances [**?**,**?**]. For instance, it is shown in [**?**] that as long as the regularization term is increasing in the spectrum of the parameter matrix and the loss function only depends on the traces of the product of the parameter matrix and the outer product instances, then algorithms that minimize a trade-off between the regularization and the loss can be kernelized. However this is only a necessary condition.

In contrast we give necessary and sufficient conditions for kernelization. Using our results we are able to prove a simple Representer Theorem that holds under conditions incomparable with those from [**?**]: we only assume that the problem is rotationally invariant and the solution is unique. Our proofs are elementary and intuitive. We can also handle the case of symmetric outer product instances, which is the mainstay of multiplicative updates, but was not considered in [**?**,**?**]. In [**?**] it was also shown that the matrix version of the p-norm perceptron can be kernelized. Again kernelizability is easily implied by our methods.

We show in this paper for an algorithm to be kernelizable, it must not even be defined as minimizing the trade-off between a regularization and a loss. Instead we show that kernelizability is characterized by a geometric invariance property. We also went through the painstaking exercise of translating our proofs to the case when instance domains are arbitrary Hilbert spaces instead of real vector spaces. No new insights were gained from this translation and we therefore present our results in the notationally simpler case of real vector spaces.

The question of whether multiplicative update algorithms are kernelizable has been a longstanding open problem in machine learning and we resolve this problem. In previous work [**?**], regret bounds were proven for matrix versions of multiplicative algorithms that grow logarithmically with the feature dimension N. Our work shows that the total rank of the instance matrices (or, equivalently, the rank of the kernel matrix) is the crucial parameter instead of the feature dimension N. Now the regret bounds are logarithmic in the total rank instead of the feature dimension N which can be unbounded.

2 Kernalization via rotational invariance

Vector instances: We begin with the case of vector instances $\boldsymbol{x} \in \mathbb{R}^n$. Examples $(\boldsymbol{x}, \ell)$ are labeled instances where ℓ is in some fixed label domain. A *learning algorithm* $\mathcal{A}$ is any mapping from example sequences $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{x}_t, \ell_t)\}_{t=1}^T$ followed by a next instance $\boldsymbol{x}$ to some fixed output range. Informally, the output of the algorithm is the "action" that $\mathcal{A}$ takes after receiving the $\boldsymbol{\mathcal{S}}$ and an unlabeled instance $\boldsymbol{x}$. We denote with $\widehat{\boldsymbol{X}}$ the matrix with the $T+1$ instances as columns and call $\widehat{\boldsymbol{X}}'\widehat{\boldsymbol{X}}$, the *augmented kernel matrix*, where "augmented" hints at the fact that we included the unlabeled instance $\boldsymbol{x}$ as the $(T+1)$st instance. Note that $[\widehat{\boldsymbol{X}}'\widehat{\boldsymbol{X}}]_{pq}$ is the dot product $\boldsymbol{x}_p'\boldsymbol{x}_q$ for $1 \le p, q \le T+1$.

We define algorithm $\mathcal{A}$ for vector instances to be *kernelizable* if for any two input sequences $\boldsymbol{\mathcal{S}}, \boldsymbol{x}$ and $\widetilde{\boldsymbol{\mathcal{S}}}, \widetilde{\boldsymbol{x}}$ with the same labels and the same augmented kernel matrix, algorithm $\mathcal{A}$ maps to the same output, i.e. $\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{x}) = \mathcal{A}(\widetilde{\boldsymbol{\mathcal{S}}}, \widetilde{\boldsymbol{x}})$. We next rewrite this characterization using the following elementary lemma:

Lemma 1. *Two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times t}$ are orthogonal transformations of each other (i.e. there is an orthogonal matrix $\boldsymbol{U}$, such that $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{A}$) iff the kernel matrices $\boldsymbol{A}'\boldsymbol{A}$ and $\boldsymbol{B}'\boldsymbol{B}$ are the same.*

For any orthogonal matrix $\boldsymbol{U} \in \mathbb{R}^{n\times n}$, let $\boldsymbol{U}\boldsymbol{\mathcal{S}}$ denote the transformed sequence $\{(\boldsymbol{U}\boldsymbol{x}_t, \ell_t)\}_{t=1}^T$. Note that the labels remain unchanged. The above lemma implies the following:

Theorem 1. *An algorithm $\mathcal{A}$ is kernelizable iff for all sequences $\boldsymbol{\mathcal{S}}$, next instance $\boldsymbol{x}$ and orthogonal matrix $\boldsymbol{U}$, $\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{x}) = \mathcal{A}(\boldsymbol{U}\boldsymbol{\mathcal{S}}, \boldsymbol{U}\boldsymbol{x})$.*

Proof. The sequences $\boldsymbol{\mathcal{S}}, \boldsymbol{x}$ and $\boldsymbol{U}\boldsymbol{\mathcal{S}}, \boldsymbol{U}\boldsymbol{x}$ have the same labels and augmented kernel matrix. Therefore, $\mathcal{A}$ kernelizable implies that $\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{x}) = \mathcal{A}(\boldsymbol{U}\boldsymbol{\mathcal{S}}, \boldsymbol{U}\boldsymbol{x})$ for all suitable $\boldsymbol{\mathcal{S}}$, $\boldsymbol{x}$ and $\boldsymbol{U}$. To prove the contrapositive of the opposite implication we assume there are two sequences $\boldsymbol{\mathcal{S}}, \boldsymbol{x}$ and $\widetilde{\boldsymbol{\mathcal{S}}}, \widetilde{\boldsymbol{x}}$ with the same augmented kernel matrix for which $\mathcal{A}$ produces a different output (witnessing that $\mathcal{A}$ is not kernelizable). Then by the above lemma there is an orthogonal matrix $\boldsymbol{U}$ for which $\widetilde{\boldsymbol{\mathcal{S}}} = \boldsymbol{U}\boldsymbol{\mathcal{S}}$, $\widetilde{\boldsymbol{x}} = \boldsymbol{U}\boldsymbol{x}$, and therefore $\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{x}) \neq \mathcal{A}(\boldsymbol{U}\boldsymbol{\mathcal{S}}, \boldsymbol{U}\boldsymbol{x})$. □

We now make an additional assumption which assures that the algorithm predicts with a linear combination of the instances: An *algorithm $\mathcal{A}$ is linear*, if upon receiving input sequence $\boldsymbol{\mathcal{S}}$ and an unlabeled instance $\boldsymbol{x}$, $\mathcal{A}$ first computes a weight vector $\boldsymbol{w} \in \mathbb{R}^n$ from the input sequence $\boldsymbol{\mathcal{S}}$ and then outputs the dot product $\boldsymbol{w}'\boldsymbol{x}$. In short, the algorithm learns a linear function. Clearly the produced $\boldsymbol{w}$ may be nonlinear in $\boldsymbol{\mathcal{S}}$.

Theorem 2. *A linear algorithm $\mathcal{A}$ is kernelizable iff for every input sequence $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{x}_t, \ell_t)\}_{t=1}^T$ the weight vector $\boldsymbol{w}$ is a linear combination of the instances of $\boldsymbol{\mathcal{S}}$, and the coefficients of the linear combination depend on $\boldsymbol{\mathcal{S}}$ only via the kernel matrix $\boldsymbol{X}'\boldsymbol{X}$, where $\boldsymbol{X}$ contains the instances $\{\boldsymbol{x}_t\}_{t=1}^T$ as columns.*

This can be proven by essentially repackaging a theorem given in [**?**]. The key contribution of this paper is that we will develop analogous theorems for the case when the instances are matrices.

Asymmetric matrix instances: We first consider the case of asymmetric matrices. In this case the instances are outer products $\boldsymbol{xy'}$, where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^m$. Examples have the form $(\boldsymbol{xy'}, \ell)$, where ℓ is from some labeling domain. A learning algorithm $\mathcal{A}$ is again, any mapping from example sequences $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{x}_t\boldsymbol{y}'_t, \ell_t)\}_{t=1}^T$, followed by a next instance $\boldsymbol{xy'}$ to some fixed output range.[5] Now we have two augmented kernel matrices, $\widehat{\boldsymbol{X}}'\widehat{\boldsymbol{X}}$ and $\widehat{\boldsymbol{Y}}'\widehat{\boldsymbol{Y}}$, where $\widehat{\boldsymbol{X}}$ contains the T instances $\{\boldsymbol{x}_t\}_{t=1}^T$ plus $\boldsymbol{x}$ as columns and $\widehat{\boldsymbol{Y}}$ is defined similarly.

Analogous to the vector case, an algorithm $\mathcal{A}$ for asymmetric outer product instances is *kernelizable* if for any two input sequences $\boldsymbol{\mathcal{S}}, \boldsymbol{xy'}$ and $\widetilde{\boldsymbol{\mathcal{S}}}, \widetilde{\boldsymbol{x}}\widetilde{\boldsymbol{y}}'$ with the same labels and the same augmented kernel matrices, algorithm $\mathcal{A}$ maps to the same output, i.e. $\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{xy'}) = \mathcal{A}(\widetilde{\boldsymbol{\mathcal{S}}}, \widetilde{\boldsymbol{x}}\widetilde{\boldsymbol{y}}')$. In the asymmetric case, we need two orthogonal matrices. For any orthogonal matrices $\boldsymbol{U} \in \mathbb{R}^{n\times n}$, and $\boldsymbol{V} \in \mathbb{R}^{m\times m}$, we let $\boldsymbol{U\mathcal{S}V'}$ denote the transformed sequence $\{(\boldsymbol{U}\boldsymbol{x}_t\boldsymbol{y}'_t\boldsymbol{V}', \ell_t)\}_{t=1}^T$. By applying Lemma **??** twice (to the left vectors $\boldsymbol{x}_t$ and the right vectors $\boldsymbol{y}_t$), it follows that algorithm $\mathcal{A}$ is kernelizable iff for all sequences $\boldsymbol{\mathcal{S}}$, next instances $\boldsymbol{xy'}$ and orthogonal matrices $\boldsymbol{U}, \boldsymbol{V}$,

$$\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{xy'}) = \mathcal{A}(\boldsymbol{U\mathcal{S}V'}, \boldsymbol{Uxy'V'}). \tag{1}$$

The generalization of the linearity of algorithms to the matrix domain is straightforward: An algorithm $\mathcal{A}$ is *linear* if $\mathcal{A}$, upon input $\boldsymbol{\mathcal{S}}, \boldsymbol{xy'}$, first computes a weight matrix $\boldsymbol{W} \in \mathbb{R}^{n\times m}$ from the input sequence $\boldsymbol{\mathcal{S}}$ and then outputs the trace $\operatorname{tr}(\boldsymbol{W'xy'})$. As we shall prove now, the linearity of the algorithm has the consequence that the algorithm maintains a weight vector that is a linear combination of the instances.

Theorem 3. *A linear algorithm $\mathcal{A}$ is kernelizable iff for every input sequence $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{x}_t\boldsymbol{y}'_t, \ell_t)\}_{t=1}^T$ the weight matrix of $\mathcal{A}$ can be written as $\boldsymbol{W} = \boldsymbol{XCY'}$, where $\boldsymbol{X}$ contains the instances $\{\boldsymbol{x}_t\}_{t=1}^T$ as columns, $\boldsymbol{Y}$ contains the $\{\boldsymbol{y}_t\}_{t=1}^T$ as columns, and the coefficient matrix $\boldsymbol{C} \in \mathbb{R}^{T\times T}$ depends on $\boldsymbol{\mathcal{S}}$ only via the kernel matrices $\boldsymbol{X'X}$ and $\boldsymbol{Y'Y}$.*

Note that the expression $\boldsymbol{W} = \boldsymbol{XCY'}$ is just a concise way of expressing the linear combination of instances $\sum_{i=1}^T \sum_{j=1}^T \boldsymbol{C}_{ij}\, \boldsymbol{x}_i\boldsymbol{y}'_j$.

Proof. Let $\boldsymbol{W}(\boldsymbol{\mathcal{S}})$ denote the weight matrix produced by algorithm $\mathcal{A}$ from the sequence $\boldsymbol{\mathcal{S}}$. Since $\mathcal{A}$ is kernelizable and outputs the trace $\operatorname{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})'\boldsymbol{xy'})$ we have

$$\operatorname{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})'\, \boldsymbol{xy'}) = \operatorname{tr}(\boldsymbol{W}(\boldsymbol{U\mathcal{S}V'})'\, \boldsymbol{Uxy'V'}), \tag{2}$$

[5] For conciseness we use outer products $\boldsymbol{xy'}$ as instances instead of the longer notation $(\boldsymbol{x}, \boldsymbol{y})$. Technically this means that the kernel matrices are only determined up to sign patterns but this is immaterial.

for all sequences $\boldsymbol{\mathcal{S}}$, orthogonal matrices $\boldsymbol{U}, \boldsymbol{V}$ of dimensions $n \times n$ and $m \times m$, and instances $\boldsymbol{x}\boldsymbol{y}'$, for $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^m$. In Part 1, we first show that (**??**) implies that for any $\boldsymbol{\mathcal{S}}$, $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'$ for some $\boldsymbol{C} \in \mathbb{R}^{T\times T}$. In Part 2, we show that (**??**) implies that for any $\boldsymbol{\mathcal{S}}$ and orthogonal matrices $\boldsymbol{U}, \boldsymbol{V}$ of dimensions $n \times n$ and $m \times m$, respectively, $\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}') = \boldsymbol{U}\boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'\boldsymbol{V}'$. This means that $\boldsymbol{C}$ is invariant under left and right orthogonal transformations $\boldsymbol{U}$ and $\boldsymbol{V}$ of the example sequence $\boldsymbol{\mathcal{S}}$, and thus by Lemma **??** this is equivalent to stating that $\boldsymbol{C}$ depends on $\boldsymbol{\mathcal{S}}$ only via the kernel matrices $\boldsymbol{X}'\boldsymbol{X}$ and $\boldsymbol{Y}'\boldsymbol{Y}$.

The opposite direction is easy: Since $\boldsymbol{C}$ depends on $\boldsymbol{\mathcal{S}}$ only via the kernel matrices, $\boldsymbol{C}$ is invariant under orthogonal transformation $\boldsymbol{\mathcal{S}} \mapsto \boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}'$ (which leaves the kernel matrices unchanged), and thus $\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}') = \boldsymbol{U}\boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'\boldsymbol{V}' = \boldsymbol{U}\boldsymbol{W}(\boldsymbol{\mathcal{S}})\boldsymbol{V}'$. This implies (**??**) and kernelizability:

$$\operatorname{tr}(\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}')'\boldsymbol{U}\boldsymbol{x}\boldsymbol{y}'\boldsymbol{V}') = \operatorname{tr}((\boldsymbol{U}\boldsymbol{W}(\boldsymbol{\mathcal{S}})\boldsymbol{V}')'\boldsymbol{U}\boldsymbol{x}\boldsymbol{y}'\boldsymbol{V}') = \operatorname{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})'\boldsymbol{x}\boldsymbol{y}').$$

Proof of Part 1: Let $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^{r_1}$ be an orthonormal basis for *Span* $\left(\{\boldsymbol{x}_t\}_{t=1}^{T}\right)$ and $\{\widehat{\boldsymbol{y}}_j\}_{j=1}^{r_2}$ be an orthonormal basis for *Span* $\left(\{\boldsymbol{y}_t\}_{t=1}^{T}\right)$, where r_1 and r_2 are the ranks of the corresponding spaces. Complete these two bases to orthonormal bases for $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively, and denote these bases as $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^{n}$ and $\{\widehat{\boldsymbol{y}}_j\}_{j=1}^{m}$. Since $\{\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{y}}_j' \mid i = 1\ldots n, j = 1\ldots m\}$ is an orthonormal basis for $\mathbb{R}^{n\times m}$ we can rewrite the matrix $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) \in \mathbb{R}^{n\times m}$ as

$$\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i=1}^{n}\sum_{j=1}^{m} \hat{c}_{i,j}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{y}}_j'.$$

Choose any index $r_1 < p \le n$ and any index $1 \le q \le m$, and we now show that $\hat{c}_{p,q} = 0$ (the case $r_2 < q \le m$ and $1 \le p \le n$ is proven similarly). We use the notion of transformation invariance (**??**). We choose $\boldsymbol{x} = \widehat{\boldsymbol{x}}_p$ and $\boldsymbol{y} = \widehat{\boldsymbol{y}}_q$. Furthermore, choose $\boldsymbol{U}$ as the *Hauseholder reflection matrix* $\boldsymbol{I} - 2\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_p'$ and $\boldsymbol{V} = \boldsymbol{I}$. Since $\widehat{\boldsymbol{x}}_p \perp \boldsymbol{x}_t$ for any $t = 1, \ldots, T$ (because $p > r_1$), $\boldsymbol{U}\boldsymbol{x}_t = \boldsymbol{x}_t - 2\widehat{\boldsymbol{x}}_p(\widehat{\boldsymbol{x}}_p'\boldsymbol{x}_t) = \boldsymbol{x}_t$. Also $\boldsymbol{V}\boldsymbol{y}_t = \boldsymbol{I}\boldsymbol{y}_t = \boldsymbol{y}_t$. It thus follows that the transformed sample $\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}'$ is same as the original sample $\boldsymbol{\mathcal{S}}$, and therefore $\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}') = \boldsymbol{W}(\boldsymbol{\mathcal{S}})$. Thus the l.h.s. of Equation (**??**) becomes:

$$\operatorname{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})'\boldsymbol{x}\boldsymbol{y}') = \operatorname{tr}\Big((\sum_{i,j}\hat{c}_{i,j}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{y}}_j')'\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{y}}_q'\Big) = \sum_{i,j}\hat{c}_{i,j}\widehat{\boldsymbol{x}}_i'\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{y}}_q'\widehat{\boldsymbol{y}}_j = \hat{c}_{p,q}\widehat{\boldsymbol{x}}_p'\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{y}}_q'\widehat{\boldsymbol{y}}_q = \hat{c}_{p,q}.$$

However since $\widehat{\boldsymbol{x}}_p'\widehat{\boldsymbol{x}}_p = 1$, $\boldsymbol{U}\widehat{\boldsymbol{x}}_p = \widehat{\boldsymbol{x}}_p - 2\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_p'\widehat{\boldsymbol{x}}_p = -\widehat{\boldsymbol{x}}_p$ and therefore the r.h.s. of Equation (**??**) has the opposite sign:

$$\operatorname{tr}(\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V})'\boldsymbol{U}\boldsymbol{x}\boldsymbol{y}'\boldsymbol{V}') = -\operatorname{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})'\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{y}}_q') = -\hat{c}_{p,q}.$$

We conclude that the transformation invariance (**??**) implies $\hat{c}_{p,q} = 0$ if $p > r_1$ (and similarly $\hat{c}_{p,q} = 0$ if $q > r_2$). Since for any $p \le r_1$, $\widehat{\boldsymbol{x}}_p$ is a linear combination of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$, and for any $q \le r_2$, $\widehat{\boldsymbol{y}}_q$ is a linear combination of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T$, it follows that

$$\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i=1}^{r_1}\sum_{j=1}^{r_2}\hat{c}_{i,j}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{y}}_j' = \sum_{i=1}^{T}\sum_{j=1}^{T}\boldsymbol{C}_{i,j}\boldsymbol{x}_i\boldsymbol{y}_j',$$

for some coefficient matrix $\boldsymbol{C} \in \mathbb{R}^{T\times T}$.

Proof of Part 2: By Part 1, $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i=1}^{r_1}\sum_{j=1}^{r_2} \hat{c}_{i,j}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{y}}_j'$. By applying Part 1 to the sequence $\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}'$ we get $\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}') = \sum_{i=1}^{r_1}\sum_{j=1}^{r_2} \hat{d}_{i,j}\boldsymbol{U}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{y}}_j'\boldsymbol{V}'$ for some coefficients $\hat{d}_{i,j}$, because if $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^{r_1}$ is an orthonormal basis for $\boldsymbol{X}$, $\{\boldsymbol{U}\widehat{\boldsymbol{x}}_i\}_{i=1}^{r_1}$ is an orthonormal basis for $\boldsymbol{U}\boldsymbol{X}$ (and similarly for $\boldsymbol{Y}$ and $\boldsymbol{V}\boldsymbol{Y}$). To prove the Part 2, it suffices to show that $\hat{c}_{p,q} = \hat{d}_{p,q}$ for any $1 \le p \le n$ and $1 \le q \le m$. By (**??**),

$$\begin{aligned}\hat{c}_{p,q} &= \operatorname{tr}\Big(\Big(\sum_{i,j}\hat{c}_{i,j}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{y}}_j'\Big)'\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{y}}_q'\Big) = \operatorname{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})'\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{y}}_q') = \operatorname{tr}(\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}')'\boldsymbol{U}\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{y}}_q\boldsymbol{V}')\\ &= \operatorname{tr}\Big(\Big(\sum_{i,j}\hat{d}_{i,j}\boldsymbol{U}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{y}}_j'\boldsymbol{V}'\Big)'\boldsymbol{U}\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{y}}_q'\boldsymbol{V}'\Big) = \operatorname{tr}\Big(\Big(\sum_{i,j}\hat{d}_{i,j}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{y}}_j'\Big)'\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{y}}_q'\Big) = \hat{d}_{p,q}.\square\end{aligned}$$

Note that the size of the coefficient matrix $\boldsymbol{C}$ is quadratic in the number of instances T. Actually $r_1 \times r_2$ non-zero coefficients suffice, where r_1, r_2 is the rank of the kernel matrices $\boldsymbol{X}, \boldsymbol{Y}$, respectively. The reason for the quadratic size is that transformation invariance for asymmetric matrices involves two orthogonal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$. If we viewed the outer products $\boldsymbol{x}_t\boldsymbol{y}_t'$ in $\mathbb{R}^{n\times m}$ as vectors in $\mathbb{R}^{nm}$ and assumed rotational invariance with respect to a single orthogonal matrix of dimension $k = nm$, then $\boldsymbol{\mathcal{S}}$ would have the form $\sum_t c_t\, \boldsymbol{x}_t\boldsymbol{y}_t'$, i.e. only one coefficient per outer product instance.

There are straightforward generalizations of the above theorem to the case when the instances are general matrices of a given rank s. Using the SVD decomposition, the instances then can be written as sums of a fixed number of outer products. That is, now the instances have the form

$$\underset{n\times s}{\boldsymbol{X}_t}\ \underset{s\times m}{\boldsymbol{Y}_t'} = \sum_{q=1}^{s}\boldsymbol{x}_t^q\boldsymbol{y}_t^{q\prime}.$$

In other words the vectors $\{\boldsymbol{x}_t^q\}_{q=1}^s$ and $\{\boldsymbol{y}_t^q\}_{q=1}^s$ are the columns of $\boldsymbol{X}_t$ and $\boldsymbol{Y}_t$, respectively. The above theorem remains essentially unchanged, but for a sequence $\{\boldsymbol{X}_t\boldsymbol{Y}_t'\}_{t=1}^T$ of T instances, the kernel matrix $\boldsymbol{X}\boldsymbol{X}'$ is formed by letting $\boldsymbol{X}$ contain the columns of all $\boldsymbol{X}_t$, which adds up to sT columns in total. Similarly, $\boldsymbol{Y}$ contains the sT columns of all $\boldsymbol{Y}_t$ and both indices in the sums in the proof of Theorem **??** range from one to sT.

Symmetric matrix instances: Let us now consider the case of symmetric outer product instances. A broad set of applications falls into this framework, including Principal Component Analysis, Fisher Discriminant Function, or Quantum Information Theory. In this case, the instances are $\boldsymbol{x}\boldsymbol{x}'$ for $\boldsymbol{x} \in \mathbb{R}^n$, and the learning algorithm $\mathcal{A}$ is any mapping from example sequences $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{x}_t\boldsymbol{x}_t', \ell_t)\}_{t=1}^T$ followed by a next instance $\boldsymbol{x}\boldsymbol{x}'$ to some fixed output range. Contrary to asymmetric instances, we now have a single augmented kernel matrix $\widehat{\boldsymbol{X}}'\widehat{\boldsymbol{X}}$, which contains the T instances $\{\boldsymbol{x}_t\}_{t=1}^T$ plus $\boldsymbol{x}$ as columns.

An algorithm $\mathcal{A}$ for symmetric outer product instances is *kernelizable* if for any two input sequences $\boldsymbol{\mathcal{S}}, \boldsymbol{x}\boldsymbol{x}'$ and $\widetilde{\boldsymbol{\mathcal{S}}}, \widetilde{\boldsymbol{x}}\widetilde{\boldsymbol{x}}'$ with the same labels and the same augmented kernel matrix, algorithm $\mathcal{A}$ maps to the same output, i.e. $\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{x}\boldsymbol{x}') = \mathcal{A}(\widetilde{\boldsymbol{\mathcal{S}}}, \widetilde{\boldsymbol{x}}\widetilde{\boldsymbol{x}}')$. By applying Lemma **??** it follows that $\mathcal{A}$ is kernelizable iff for all $\boldsymbol{\mathcal{S}}$, $\boldsymbol{x}\boldsymbol{x}'$ and orthogonal matrices $\boldsymbol{U}$,

$$\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{x}\boldsymbol{x}') = \mathcal{A}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}', \boldsymbol{U}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{U}').$$

Note that contrary to the asymmetric case, the same matrix $\boldsymbol{U}$ is applied on both sides. An algorithm $\mathcal{A}$ is linear, if upon input $\boldsymbol{\mathcal{S}}, \boldsymbol{x}\boldsymbol{x}'$, $\mathcal{A}$ first computes a *symmetric* weight matrix $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ from the input sequence $\boldsymbol{\mathcal{S}}$ and then outputs the trace $\mathrm{tr}(\boldsymbol{W}'\boldsymbol{x}\boldsymbol{x}')$.[6]

Theorem 4. *A linear algorithm $\mathcal{A}$ is kernelizable iff for every input sequence $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{x}_t\boldsymbol{x}_t', \ell_t)\}_{t=1}^T$ the weight matrix of $\mathcal{A}$ can be written as $\boldsymbol{W} = \boldsymbol{X}\boldsymbol{C}\boldsymbol{X}' + c\boldsymbol{I}$, where $\boldsymbol{X}$ contains the instances $\{\boldsymbol{x}_t\}_{t=1}^T$ as columns, $\boldsymbol{C} \in \mathbb{R}^{T\times T}$ is a symmetric coefficient matrix, c is a real number, $\boldsymbol{I}$ is the identity matrix in $\mathbb{R}^n$, and $\boldsymbol{C}$ and c depend on $\boldsymbol{\mathcal{S}}$ only via the kernel matrix $\boldsymbol{X}'\boldsymbol{X}$.*

Proof. We only show the part of the proof which corresponds to "Part 1" of the proof of Theorem **??** (the rest of the proof follows closely the proof of Theorem **??**). Since $\mathcal{A}$ is kernelizable, we have

$$\mathrm{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})'\,\boldsymbol{x}\boldsymbol{x}') = \mathrm{tr}(\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}')'\,\boldsymbol{U}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{U}'), \tag{3}$$

for all $\boldsymbol{\mathcal{S}}, \boldsymbol{U}, \boldsymbol{x}\boldsymbol{x}'$. We want to show that (**??**) implies that for any $\boldsymbol{\mathcal{S}}$, $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \boldsymbol{X}\boldsymbol{C}\boldsymbol{X}' + c\boldsymbol{I}$ for some symmetric $\boldsymbol{C} \in \mathbb{R}^{T\times T}$ and $c \in \mathbb{R}$. Let $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^r$ be an orthonormal basis for *Span* $\left(\{\boldsymbol{x}_t\}_{t=1}^T\right)$. Complete this basis to an orthonormal basis $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^n$ for $\mathbb{R}^n$, We decompose $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i,j} \hat{c}_{i,j}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{x}}_j'$, and due to symmetry of $\boldsymbol{W}(\boldsymbol{\mathcal{S}})$, $\hat{c}_{i,j} = \hat{c}_{j,i}$ for all i, j.

We need to show that $\hat{c}_{p,q} = 0$ if $p \neq q$ and either $p > r$ or $q > r$, and that $\hat{c}_{p,p} = \hat{c}$ for some constant $\hat{c}$, for $p > r$. We show the former first. Due to the symmetry of $\boldsymbol{W}(\boldsymbol{\mathcal{S}})$, it suffices to show that that $\hat{c}_{p,q} = 0$ for any $q > r$ and any p. Choose $\boldsymbol{x} = \widehat{\boldsymbol{x}}_p + \widehat{\boldsymbol{x}}_q$ and $\boldsymbol{U}$ as the Hauseholder reflection $\boldsymbol{I} - 2\widehat{\boldsymbol{x}}_q\widehat{\boldsymbol{x}}_q'$, so that $\boldsymbol{U}\boldsymbol{x}_t = \boldsymbol{x}_t$ for $1 \le t \le T$, $\boldsymbol{U}\widehat{\boldsymbol{x}}_p = \widehat{\boldsymbol{x}}_p$, and $\boldsymbol{U}\widehat{\boldsymbol{x}}_q = -\widehat{\boldsymbol{x}}_q$. Then, the transformed sample $\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}'$ is the same as the original sample $\boldsymbol{\mathcal{S}}$, and $\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}') = \boldsymbol{W}(\boldsymbol{\mathcal{S}})$. Therefore, the l.h.s. and r.h.s. of (**??**) become

$$\mathrm{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})'\boldsymbol{x}\boldsymbol{x}') = \sum_{i,j} \hat{c}_{i,j}\;\widehat{\boldsymbol{x}}_i'(\widehat{\boldsymbol{x}}_p + \widehat{\boldsymbol{x}}_q)(\widehat{\boldsymbol{x}}_p' + \widehat{\boldsymbol{x}}_q')\widehat{\boldsymbol{x}}_j = \hat{c}_{p,p} + \hat{c}_{q,q} + \hat{c}_{p,q} + \hat{c}_{q,p}$$

$$\mathrm{tr}(\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U})'\boldsymbol{U}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{U}') = \sum_{i,j} \hat{c}_{i,j}\;\widehat{\boldsymbol{x}}_i'(\widehat{\boldsymbol{x}}_p - \widehat{\boldsymbol{x}}_q)(\widehat{\boldsymbol{x}}_p' - \widehat{\boldsymbol{x}}_q')\widehat{\boldsymbol{x}}_j = \hat{c}_{p,p} + \hat{c}_{q,q} - \hat{c}_{p,q} - \hat{c}_{q,p},$$

which along with $\hat{c}_{p,q} = \hat{c}_{q,p}$ implies $\hat{c}_{p,q} = 0$.

[6] The assumption on the symmetry of $\boldsymbol{W}$ comes without loss of generality: Given any matrix $\boldsymbol{W}$, we can always take a symmetrized version $\boldsymbol{W}_{\mathrm{sym}} = \frac{\boldsymbol{W}+\boldsymbol{W}'}{2}$, and for any $\boldsymbol{x}\boldsymbol{x}'$, it holds $\mathrm{tr}(\boldsymbol{W}_{\mathrm{sym}}'\boldsymbol{x}\boldsymbol{x}') = \mathrm{tr}(\boldsymbol{W}'\boldsymbol{x}\boldsymbol{x}')$.

To show that $\hat{c}_{p,p} = \hat{c}$ for some constant $\hat{c}$, for all $p > r$, we choose $\boldsymbol{x} = \widehat{\boldsymbol{x}}_p$, and $\boldsymbol{U}$ to be a permutation matrix that swaps the basis vectors $\widehat{\boldsymbol{x}}_p$ and $\widehat{\boldsymbol{x}}_q$ for some $q > r$, while leaving all other basis vectors unchanged, i.e.:

$$\boldsymbol{U} = \boldsymbol{I} - \widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_p' - \widehat{\boldsymbol{x}}_q\widehat{\boldsymbol{x}}_q' + \widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_q' + \widehat{\boldsymbol{x}}_q\widehat{\boldsymbol{x}}_p'.$$

For this choice of $\boldsymbol{U}$, $\boldsymbol{U}\boldsymbol{U}' = \boldsymbol{I}$, $\boldsymbol{U}\widehat{\boldsymbol{x}}_p = \widehat{\boldsymbol{x}}_q$, $\boldsymbol{U}\widehat{\boldsymbol{x}}_q = \widehat{\boldsymbol{x}}_p$, and $\boldsymbol{U}\boldsymbol{x}_t = \boldsymbol{x}_t$ for all $1 \leq t \leq T$ (because $p, q > r$). Thus, the transformed sample $\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}'$ is the same as the original sample $\boldsymbol{\mathcal{S}}$, so that $\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}') = \boldsymbol{W}(\boldsymbol{\mathcal{S}})$. On the other hand, $\boldsymbol{U}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{U}' = \boldsymbol{U}\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_p'\boldsymbol{U}' = \widehat{\boldsymbol{x}}_q\widehat{\boldsymbol{x}}_q'$. The l.h.s. and r.h.s. (**??**) become

$$\begin{aligned} \operatorname{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})'\boldsymbol{x}\boldsymbol{x}') &= \sum_{i,j} \hat{c}_{i,j}\ \widehat{\boldsymbol{x}}_i'\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_p'\widehat{\boldsymbol{x}}_j = \hat{c}_{p,p} \\ \operatorname{tr}(\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U})'\boldsymbol{U}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{U}') &= \sum_{i,j} \hat{c}_{i,j}\ \widehat{\boldsymbol{x}}_i'\widehat{\boldsymbol{x}}_q\widehat{\boldsymbol{x}}_q'\widehat{\boldsymbol{x}}_j = \hat{c}_{q,q}, \end{aligned}$$

which implies $\hat{c}_{p,p} = \hat{c}_{q,q}$. Since q was an arbitrary index such that $q > r$, we conclude that $\hat{c}_{p,p} = \hat{c}$ for some constant $\hat{c}$, for all $p > r$.

We conclude that the transformation invariance (**??**) implies that

$$\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i=1}^{r}\sum_{j=1}^{r} \hat{c}_{i,j}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{x}}_j' + \hat{c}\sum_{i=r+1}^{n} \widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{x}}_i' = \sum_{i,j} \boldsymbol{C}_{i,j}\boldsymbol{x}_i\boldsymbol{x}_j' + c\boldsymbol{I} = \boldsymbol{X}\boldsymbol{C}\boldsymbol{X}' + c\boldsymbol{I}$$

for some coefficient matrix $\boldsymbol{C}$ and real number c, where the second equality follows from the fact that $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^{r}$ is an orthonormal basis for $\mathit{Span}\left(\{\boldsymbol{x}_t\}_{t=1}^{T}\right)$, and $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^{n}$ an orthonormal basis for $\mathbb{R}^n$. W.l.o.g. $\boldsymbol{C}$ is symmetric, because if $\boldsymbol{C}_{i,j} \neq \boldsymbol{C}_{j,i}$, then changing both to $\frac{\boldsymbol{C}_{i,j}+\boldsymbol{C}_{j,i}}{2}$ does not change $\boldsymbol{W}(\boldsymbol{\mathcal{S}})$. $\square$

Comparing Theorem **??** with Theorem **??**, an additional term $c\boldsymbol{I}$ entered the expansion. The term was absent for asymmetric matrices as there is no identity matrix in this case. The term $c\boldsymbol{I}$ can easily be dealt with when the instances are expanded via a feature map $\boldsymbol{x} \mapsto \phi(\boldsymbol{x})$, as it leads to the expression of the form $\operatorname{tr}(c\boldsymbol{I}\phi(\boldsymbol{x})\phi(\boldsymbol{x})') = c\,k(\boldsymbol{x}, \boldsymbol{x})$. We note that Theorem **??** also generalizes easily from symmetric outer product instances to symmetric matrix instances with fixed rank s.

3 Kernelization via a Representer Theorem

The following Representer Theorem for asymmetric outer product instances was proven in [?]: Given a penalty function $\mathit{\Omega}(\boldsymbol{W}) = \sum_{i=1}^{d} s_i(\sigma_i(\boldsymbol{W}))$, where $\{\sigma_1, \ldots, \sigma_d\}$ is the set of singular values of $\boldsymbol{W}$ in decreasing order, and s_i are non-decreasing functions satisfying $s(0) = 0$, then there exists a solution to the minimization problem

$$\min_{\boldsymbol{W}} \quad \mathit{\Omega}(\boldsymbol{W}) \ + \ \eta\sum_t \operatorname{loss}_t(\operatorname{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{y}_t')), \tag{4}$$

which can be written as $\boldsymbol{W} = \sum_{i,j} \boldsymbol{C}_{i,j}\boldsymbol{x}_i\boldsymbol{y}_j' = \boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'$. Using our results, we are able to prove a version of the Representer Theorem with different, not directly comparable assumptions:

Theorem 5. *Consider the minimization problem* $\min_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}, \boldsymbol{\mathcal{S}})$*, which for all* $\boldsymbol{\mathcal{S}}$ *has a unique solution and is* rotationally invariant*, i.e. for any* $\boldsymbol{\mathcal{S}}$ *and any orthogonal matrices* $\boldsymbol{U}$ *and* $\boldsymbol{V}$*,* $\mathcal{L}(\boldsymbol{W}, \boldsymbol{\mathcal{S}}) = \mathcal{L}(\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}', \boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}')$*. In this case the solution* $\boldsymbol{W}^*(\boldsymbol{\mathcal{S}})$ *can be written as* $\boldsymbol{W}^*(\boldsymbol{\mathcal{S}}) = \boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'$ *where* $\boldsymbol{C}$ *depends on* $\boldsymbol{\mathcal{S}}$ *only via the kernel matrices* $\boldsymbol{X}'\boldsymbol{X}$*,* $\boldsymbol{Y}'\boldsymbol{Y}$*.*

Proof. Let algorithm $\mathcal{A}$ produce the matrix $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) := \boldsymbol{W}^*(\boldsymbol{\mathcal{S}})$. The algorithm satisfies our definition of linearity. fIt is also kernelizable, because due to rotational invariance of $\mathcal{L}$ and uniqueness of the solution, $\boldsymbol{W}^*(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}') = \boldsymbol{U}\boldsymbol{W}^*(\boldsymbol{\mathcal{S}})\boldsymbol{V}'$ and thus $\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}') = \boldsymbol{U}\boldsymbol{W}(\boldsymbol{\mathcal{S}})\boldsymbol{V}'$, so that for any $\boldsymbol{x}\boldsymbol{y}'$,

$$\mathrm{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})'\boldsymbol{x}\boldsymbol{y}') = \mathrm{tr}(\boldsymbol{V}'\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}')'\boldsymbol{U}\boldsymbol{x}\boldsymbol{y}') = \mathrm{tr}(\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}')'\boldsymbol{U}\boldsymbol{x}\boldsymbol{y}\boldsymbol{V}').$$

The theorem now follows from the forward direction of Theorem **??**. □

We also note that with some more effort, it is possible to prove Theorem **??** under the weaker assumption that $\min_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}, \boldsymbol{\mathcal{S}})$ has a unique solution only for a particular sequence $\boldsymbol{\mathcal{S}}$, rather than for all sequences $\boldsymbol{\mathcal{S}}$.

The problem (**??**) is rotationally invariant (because Ω is a function of the singular values only), so the Theorem **??** applies as long as the solution is unique. Note that [**?**] specify different conditions: no uniqueness assumption is needed, rather some structure of the penalty function is imposed. Therefore our conditions are not directly comparable with theirs. Our proof of the Representer Theorem is however much simpler than the proof in [**?**]. Also, our conditions apply to a much wider class of algorithms, which does not need be defined as solution to the optimization problem above. Moreover, using our approach it is straightforward to generalize the Representer Theorem to the optimization problem with constraints, as long as the constraints are rotationally invariant and the solution is unique. Finally, we easily get the version of the Representer Theorem for the case of symmetric outer products, which has not be considered elsewhere:

Theorem 6. *Consider the problem* $\min_{\mathrm{sym.}\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}, \boldsymbol{\mathcal{S}})$*, which for all* $\boldsymbol{\mathcal{S}}$ *has a unique solution and is* rotationally invariant*, i.e. for any* $\boldsymbol{\mathcal{S}}$ *and any orthogonal matrix* $\boldsymbol{U}$*,* $\mathcal{L}(\boldsymbol{W}, \boldsymbol{\mathcal{S}}) = \mathcal{L}(\boldsymbol{U}\boldsymbol{W}\boldsymbol{U}', \boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}')$*. In this case the solution* $\boldsymbol{W}^*(\boldsymbol{\mathcal{S}})$ *can be written as* $\boldsymbol{W}^*(\boldsymbol{\mathcal{S}}) = \boldsymbol{X}\boldsymbol{C}\boldsymbol{X}' + c\boldsymbol{I}$*, where* $\boldsymbol{C}$ *and* c *depend on* $\boldsymbol{\mathcal{S}}$ *only via the kernel matrix* $\boldsymbol{X}'\boldsymbol{X}$*.*

4 Example applications

We provide a few examples of how the arguments given in this paper can shed light on the kernelization of algorithms for particular learning problems. We focus on the online setting, i.e. when the instances are revealed sequentially to the learner. We also give algorithms only for the matrix case (both asymmetric and symmetric), as the vector case has been much exploited in the last decades, mostly in connection to support vector machines.

The algorithms of this section require the use of the singular value decomposition of the matrix $\boldsymbol{XCV}'$, or the eigenvalue decomposition (in the symmetric case) of the symmetric matrix $\boldsymbol{XCX}'$. As discussed in the introduction, the dimensions n and m of the left instances $\boldsymbol{x}_i$ and the right instances $\boldsymbol{y}_i$, respectively, are typically much larger than the number of instances T. Thus the dimension of the matrix $\boldsymbol{XCY}' \in \mathbb{R}^{n\times m}$ (or $\boldsymbol{XCX}' \in \mathbb{R}^{n\times n}$) is too large. The key is to obtain its decomposition in terms of the smaller kernel matrices $\boldsymbol{X}'\boldsymbol{X}, \boldsymbol{Y}'\boldsymbol{Y} \in \mathbb{R}^{T\times T}$:

Lemma 2. *For any left instance set* $\boldsymbol{X} \in \mathbb{R}^{n\times T}$, *right instance set* $\boldsymbol{Y} \in \mathbb{R}^{m\times T}$ *and square matrix* $\boldsymbol{C} \in \mathbb{R}^{T\times T}$, *if* $\boldsymbol{U\Sigma V}'$ *is a compact SVD of* $\sqrt{\boldsymbol{X}'\boldsymbol{X}}\boldsymbol{C}\sqrt{\boldsymbol{Y}'\boldsymbol{Y}}$, *where* $\boldsymbol{\Sigma} = diag(\sigma_1, \cdots, \sigma_r)$, *then the compact SVD of* $\boldsymbol{XCY}'$ *is* $\tilde{\boldsymbol{U}}\boldsymbol{\Sigma}\tilde{\boldsymbol{V}}$ *with* $\tilde{\boldsymbol{U}} = \boldsymbol{XC}\sqrt{\boldsymbol{Y}'\boldsymbol{Y}}\boldsymbol{V\Sigma}^{-1}$ *and* $\tilde{\boldsymbol{V}} = \boldsymbol{YC}'\sqrt{\boldsymbol{X}'\boldsymbol{X}}\boldsymbol{U\Sigma}^{-1}$. *Similarly, for any* $\boldsymbol{X} \in \mathbb{R}^{n\times T}$ *and symmetric matrix* $\boldsymbol{C} \in \mathbb{R}^{T\times T}$, *if* $\boldsymbol{U\Sigma U}'$ *is a compact eigendecomposition of* $\sqrt{\boldsymbol{X}'\boldsymbol{X}}\boldsymbol{C}\sqrt{\boldsymbol{X}'\boldsymbol{X}}$, *where* $\boldsymbol{\Sigma} = diag(\sigma_1, \cdots, \sigma_r)$, *then the compact eigendecomposition of* $\boldsymbol{XCX}'$ *is* $\tilde{\boldsymbol{U}}\boldsymbol{\Sigma}\tilde{\boldsymbol{U}}'$ *with* $\tilde{\boldsymbol{U}} = \boldsymbol{XC}\sqrt{\boldsymbol{X}'\boldsymbol{X}}\boldsymbol{U\Sigma}^{-1}$.

The proof (omitted) consists of checking the orthogonality of $\tilde{\boldsymbol{U}}$ and $\tilde{\boldsymbol{V}}$ and showing that $\boldsymbol{XCY}' = \tilde{\boldsymbol{U}}\boldsymbol{\Sigma}\tilde{\boldsymbol{V}}$. For symmetric instances, a particularly simple case is obtained when $\boldsymbol{C} = \boldsymbol{I}$:

Corollary 1. *For any* $\boldsymbol{X} \in \mathbb{R}^{n\times T}$, *if* $\boldsymbol{U\Sigma U}'$ *is a compact eigendecomposition of* $\boldsymbol{X}'\boldsymbol{X}$, *then* $\boldsymbol{XX}'$ *has the compact eigendecomposition* $\widetilde{\boldsymbol{U}}\boldsymbol{\Sigma}\widetilde{\boldsymbol{U}}'$, *where* $\widetilde{\boldsymbol{U}} = \boldsymbol{XU\Sigma}^{-1/2}$.

This known fact was key to the kernelization of PCA and Fisher Linear Discriminant Functions [**?**,**?**].

Asymmetric case and additive updates: Consider the following online learning problem: The data $\{(\boldsymbol{x}_t\boldsymbol{y}_t', \ell_t)\}_{t=1}^T$ is revealed to the learner sequentially. The learner predicts at trial t with a matrix $\boldsymbol{W}_t \in \mathcal{W}$ from some convex set $\mathcal{W}$, and suffers a convex loss denoted as $\text{loss}(\text{tr}(\boldsymbol{W}_t'\boldsymbol{x}_t\boldsymbol{y}_t'), \ell_t)$. The goal of the learner is to have total loss in trials $t = 1, \ldots, T$ not much higher then the total loss of the best matrix $\boldsymbol{W}^* \in \mathcal{W}$ chosen in hindsight, i.e. to have small regret

$$\text{Reg}(\boldsymbol{\mathcal{S}}) = \sum_t \text{loss}(\text{tr}(\boldsymbol{W}_t'\boldsymbol{x}_t\boldsymbol{y}_t'), \ell_t) - \min_{\boldsymbol{W}\in\mathcal{W}} \sum_t \text{loss}(\text{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{y}_t'), \ell_t).$$

Assume that $\mathcal{W} = \{\boldsymbol{W} \colon \|\boldsymbol{W}\| \le B\}$, where $\|\boldsymbol{W}\|$ is a rotationally invariant norm, i.e. depends on $\boldsymbol{W}$ only via its singular values. A typical choice, used e.g. in collaborative filtering, would be the trace norm $\|\boldsymbol{W}\|_1$. Let us also assume for simplicity that $\|\boldsymbol{x}_t\|_2 \le 1$ and $\|\boldsymbol{y}_t\|_2 \le 1$ for all t, where $\|\cdot\|_2$ is the Euclidean norm. A popular approach to solve the minimization problem is the online gradient descent (GD) [**?**]: Let $\partial_t(\boldsymbol{W})$ denote the subgradient $\partial_{\hat{\ell}_t}\text{loss}(\hat{\ell}_t, \ell_t)$ at $\hat{\ell}_t = \text{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{y}_t')$. The GD step can be derived as the solution to the following optimization problem:

$$\boldsymbol{W}_{t+1} = \operatorname*{argmin}_{\boldsymbol{W}\in\mathcal{W}} \; \|\boldsymbol{W} - \boldsymbol{W}_t\|_F^2 + \eta\partial_t(\boldsymbol{W}_t)\,\text{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{y}_t'), \tag{5}$$

where $\|\cdot\|_F$ is the Frobenious norm. Solving (**??**) leads to the *additive update*:

$$\boldsymbol{W}_{t+1} = \mathrm{proj}\left(\boldsymbol{W}_t - \eta \partial_t(\boldsymbol{W}_t)\boldsymbol{x}_t\boldsymbol{y}_t'\right),$$

where the projection operation is defined as $\mathrm{proj}(\boldsymbol{W}) = \mathrm{argmin}_{\|\widetilde{\boldsymbol{W}}\| \le B} \|\boldsymbol{W} - \widetilde{\boldsymbol{W}}\|_F^2$. Since the norm $\|\cdot\|$ is rotationally invariant, the projection becomes a projection on the singular values $\{\sigma_1, \ldots, \sigma_{\min\{n,m\}}\}$ of $\boldsymbol{W}$. In particular, for $\|\cdot\|$ being the trace norm, the projection leads to $\sigma_i \mapsto (\sigma_i - \tau)_+$, where τ is the smallest value for which $\sum_i (\sigma_i - \tau)_+ \le B$. When $\boldsymbol{W}_1 = \boldsymbol{0}$, one can show by a simple induction that the problem (**??**) is rotationally invariant for all t. Due to the strictly convex objective function, (**??**) has a unique solution, and we conclude from Theorem **??** that $\boldsymbol{W}_t$ is in the span of the data, i.e. has the form $\boldsymbol{XCY'}$. Thus the algorithm can be kernelized by calculating the SVD of the matrices $\boldsymbol{XCY'}$ i.t.o. of the kernel matrices using Lemma **??**. Also the output $\mathrm{tr}(\boldsymbol{XCY'xy'}) = x'\boldsymbol{XCY'y}$ only relies on the kernel matrices. For the trace norm, it can be shown using a standard analysis of GD, that given $|\partial_t(\boldsymbol{W})| \le G$, $\mathrm{Reg}(\boldsymbol{\mathcal{S}}) \le BG\sqrt{T}$, and is independent of the dimension of the feature space[7] [**?**].

Symmetric case and multiplicative updates: In the symmetric case, the data sequence becomes $\{(\boldsymbol{x}_t\boldsymbol{x}_t', \ell_t)\}_{t=1}^T$. Let us assume for simplicity that $\|\boldsymbol{x}_t\|_2 = 1$ for all t. The learner predicts at trial t with the symmetric matrix $\boldsymbol{W}_t \in \mathcal{W}$, and suffers loss $\mathrm{loss}(\mathrm{tr}(\boldsymbol{W}_t'\boldsymbol{x}_t\boldsymbol{x}_t'), \ell_t)$. We focus on the interesting case when $\mathcal{W}$ is a set of positive-semidefinite matrices with unit trace (*density* matrices), a generalization of the probability simplex to symmetric matrices. A choice of the algorithm is the Matrix Exponentiated Gradient [**?**], defined as a trade-off between minimization of the quantum relative entropy and the negative gradient of the loss:

$$\boldsymbol{W}_{t+1} = \underset{\boldsymbol{W} \in \mathcal{W}}{\mathrm{argmin}} \;\; \mathrm{tr}\left(\boldsymbol{W}\left(\mathbf{log}\,\boldsymbol{W} - \mathbf{log}\,\boldsymbol{W}_t\right)\right) + \eta \partial_t(\boldsymbol{W}_t)\,\mathrm{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{x}_t'), \qquad (6)$$

which leads to to the following *multiplicative update* [**?**]:

$$\boldsymbol{W}_{t+1} = \frac{\mathbf{exp}\left(\mathbf{log}\,\boldsymbol{W}_t - \eta\partial_t(\boldsymbol{W}_t)\boldsymbol{x}_t\boldsymbol{x}_t'\right)}{Z_t}, \qquad (7)$$

where $Z_t = \mathrm{tr}\left(\mathbf{exp}\left(\mathbf{log}\,\boldsymbol{W}_t - \eta\partial_t(\boldsymbol{W}_t)\boldsymbol{x}_t\boldsymbol{x}_t'\right)\right)$ is the normalization factor. When $\boldsymbol{W}_1 = \boldsymbol{I}/n$, a simple inductive argument proves rotational invariance of (**??**) for all t. Due to the strictly convex objective function, (**??**) has a unique solution, and we conclude from Theorem **??** that the algorithm can be kernelized (we note that the standard representer theorems do not cover this case). The main challenge in the update (**??**) is to do the **exp** operation, but it can be done by eigendecomposition of $\mathbf{log}\,\boldsymbol{W}_t - \eta\partial_t(\boldsymbol{W}_t)\boldsymbol{x}_t\boldsymbol{x}_t'$, which by Lemma **??** only requires to calculate the kernel matrix.

[7] In practical applications the choice of B may still depend on the dimension.

A particularly interesting case is when $\operatorname{loss}(\operatorname{tr}(\boldsymbol{W}_t'\boldsymbol{x}_t\boldsymbol{x}_t'), \ell_t) = -\operatorname{tr}(\boldsymbol{W}_t'\boldsymbol{x}_t\boldsymbol{x}_t')$. In other words, the game is the *gain game* with a linear gain function $\operatorname{tr}(\boldsymbol{W}_t\boldsymbol{x}_t\boldsymbol{x}_t')$. Then, the offline solution to the problem $\boldsymbol{W}^*$ is a one-dimensional projector to the subspace that captures the most of the variance of the data, i.e. the subspace associated with the largest eigenvalue of $\sum_t \boldsymbol{x}_t\boldsymbol{x}_t'$. This is exactly the problem of single-component PCA[8] [?]. In this case, the EG update (**??**) simplifies to $\boldsymbol{W}_{t+1} = Z_t^{-1}\,\mathbf{exp}\left(\eta\sum_{i=1}^t \boldsymbol{x}_i\boldsymbol{x}_i'\right)$, and the eigendecomposition can be handled using Corollary **??**.

By modifying the EG analysis of [?,?], we can show shown that $\operatorname{Reg}(\boldsymbol{\mathcal{S}}) \le \sqrt{2L^*\ln n} + \ln n$, where L^* is the *approximation error*, i.e. part of the variance in the data not captured by $\boldsymbol{W}^*$, $L^* = \min_{\boldsymbol{W}\in\mathcal{W}}\left\{\sum_{t=1}^T (1 - \operatorname{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{x}_t'))\right\}$. Unfortunately, this bound (which essentially appears in [?]) is not satisfactory, as it depends on the feature space dimension n. When the instances $\boldsymbol{x}\boldsymbol{x}'$ are replaced by $\phi(\boldsymbol{x})\phi(\boldsymbol{x})'$ then the $\ln n$ term can become unbounded. Below we sketch a new method for replacing $\ln n$ by $\ln r$, where are is the total rank of the instances. So for the first time, we obtain a bound for a Matrix EG algorithm that does not depend on the feature dimension.

We observe that the best density matrix in hindsight $\boldsymbol{W}^*$ projects into the span of the data. If we knew the span in hindsight, we could disregard the other dimensions and play EG within this subspace, achieving the bound $\sqrt{2L^*\ln r} + \ln r$, where r is the dimension of the subspace, i.e. the rank of the kernel matrix $\boldsymbol{X}'\boldsymbol{X}$. This bound is independent on n, as $r \le T$. Of course, the data span is unknown to the learner, but we can slightly modify the EG algorithm (let us call the modification EG$^+$) to obtain the bound $\sqrt{2L^*\ln r} + \ln r + 1 \le \sqrt{2L^*\ln T} + \ln T + 1$ without any prior knowledge of the span. The EG$^+$ algorithm is defined by modifying the update (**??**) to $\boldsymbol{W}_t^+ = \left(Z_t^+\right)^{-1}\mathbf{exp}^+\left(\eta\sum_{i=1}^{t-1}\boldsymbol{x}_i\boldsymbol{x}_i'\right)$, where $Z_t^+ = \operatorname{tr}\left(\mathbf{exp}^+\left(\eta\sum_{i=1}^{t-1}\boldsymbol{x}_i\boldsymbol{x}_i'\right)\right)$, and $\mathbf{exp}^+(\boldsymbol{A})$ is a function that exponentiate the positive eigenvalues of $\boldsymbol{A}$ only, and leaves the zero eigenvalues unchanged.[9] In other words if $\boldsymbol{A}$ has a compact eigenvalue decomposition $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}'$, then, $\mathbf{exp}^+(\boldsymbol{A}) = \boldsymbol{U}\,\mathbf{exp}(\boldsymbol{\Sigma})\boldsymbol{U}'$. To prove the regret bound $\sqrt{2L^*\ln r} + \ln r + 1$ for EG$^+$, it suffices to show that given a feature space with dimension n, the total loss of EG$^+$ (which does not know n) is by at most one larger than the total loss of EG (which knows n):

Lemma 3. *Let $\boldsymbol{W}_t$ and $\boldsymbol{W}_t^+$ be the matrices produced by the EG and EG$^+$ algorithms, respectively. Then $\sum_{t=1}^T -\operatorname{tr}(\boldsymbol{W}_t^{+\prime}\boldsymbol{x}_t\boldsymbol{x}_t') - \sum_{t=1}^T -\operatorname{tr}(\boldsymbol{W}_t'\boldsymbol{x}_t\boldsymbol{x}_t') \le 1$.*

Proof. Fix iteration t and let $\boldsymbol{S}_{t-1} := \sum_{i=1}^{t-1}\boldsymbol{x}_i\boldsymbol{x}_i'$. If $\boldsymbol{x}_t$ is a linear combination of past instances $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}$, then the loss incurred by EG$^+$ is smaller than the loss incurred by EG. Indeed, $\operatorname{tr}\left(\mathbf{exp}^+\left(\eta\boldsymbol{S}_{t-1}\right)\boldsymbol{x}_t\boldsymbol{x}_t'\right) = \operatorname{tr}\left(\mathbf{exp}\left(\eta\boldsymbol{S}_{t-1}\right)\boldsymbol{x}_t\boldsymbol{x}_t'\right)$ (because $\boldsymbol{x}_t$ belongs to the subspace associated with non-zero eigenvalues of

8 By capping the eigenvalues to $\frac{1}{k}$ (as done in [?]) we can generalize this algorithm to k-component PCA where one seeks a k-dimensional subspace with maximal variance.

9 The initial weight matrix $\boldsymbol{W}_1$ is set arbitrarily.

$\boldsymbol{S}_{t-1})$, but $Z_t^+ \le Z_t$ (because $\mathbf{exp}^+(\boldsymbol{A}) \preceq \mathbf{exp}(\boldsymbol{A})$ for any positive matrix $\boldsymbol{A}$). If $\boldsymbol{x}_t$ is linearly independent of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}$, then the loss incurred by EG$^+$ in any trial $t > 1$ is larger by at most $\frac{1}{n}$ (and $t = 1$ can be handled seperately):

$$\begin{aligned}
-\mathrm{tr}(\boldsymbol{W}_t^{+'} \boldsymbol{x}_t \boldsymbol{x}_t') &= -(Z_t^+)^{-1} \mathrm{tr}\left(\mathbf{exp}^+\left(\eta \boldsymbol{S}_{t-1}\right) \boldsymbol{x}_t \boldsymbol{x}_t'\right) \\
&\le -Z_t^{-1} \mathrm{tr}\left(\mathbf{exp}^+\left(\eta \boldsymbol{S}_{t-1}\right) \boldsymbol{x}_t \boldsymbol{x}_t'\right) \\
&\le -Z_t^{-1} \mathrm{tr}\left(\left(\mathbf{exp}\left(\eta \boldsymbol{S}_{t-1}\right) - \boldsymbol{I}\right) \boldsymbol{x}_t \boldsymbol{x}_t'\right) \\
&= -\mathrm{tr}(\boldsymbol{W}_t' \boldsymbol{x}_t \boldsymbol{x}_t') + Z_t^{-1} \\
&\le -\mathrm{tr}(\boldsymbol{W}_t' \boldsymbol{x}_t \boldsymbol{x}_t') + 1/n,
\end{aligned}$$

where we used the fact that $\mathbf{exp}^+(\boldsymbol{A}) \succeq \mathbf{exp}(\boldsymbol{A}) - \boldsymbol{I}$ for any positive matrix $\boldsymbol{A}$, and that $Z_t = \mathrm{tr}\left(\mathbf{exp}\left(\eta \sum_{i=1}^{t-1} \boldsymbol{x}_i \boldsymbol{x}_i'\right)\right) \ge \mathrm{tr}(\boldsymbol{I}) = n$. □

Note that the EG$^+$ is as easy to kernelize as the EG, because they differ only in the update of the eigenvalues. We can also easily handle the case when the instances are positive symmetric matrices of rank at most s. Since the EG bound does not depend on the sparsity of the instances, we immediately get the same regret bound $\sqrt{2L^* \log r} + \ln r$, where $r \le Ts$.

We finally note that one can also use an additive update (GD) algorithm in the symmetric case, and obtain the bound $\sqrt{T}$ for outer product instances, and $\sqrt{Ts}$ for matrix instances. The bounds for the GD and the EG$^+$ algorithms are not directly comparable: EG$^+$ has an additional $\log r$ factor, but GD scales worse with the rank s of matrix instances. Moreover, the EG$^+$ bound is especially useful for low-noise conditions, when the approximation error L^* is small. There is no corresponding bound known for the GD in this case.

5 Conclusion

We gave necessary and sufficient conditions for kernelizability for the case of vector, asymmetric matrix, and symmetric matrix instances, under the assumption that the algorithm is linear, produces a unique solution and satisfies a certain rotational invariance. We also proved simple representer theorems for both asymmetric and symmetric matrix instances, and gave a number of examples of our methods, including the kernelization of multiplicative updates. In some sense our approach resembles how the models in Physics are built, where the equations of motion follow from certain invariance properties of physical laws.

We conclude with a subtle open problem. A new family of so called "Forward" algorithms was developed [**?**] whose predictions may depend on the current unlabeled instance for which the algorithm is to produce a label. In particular in the case of linear regression [**?**,**?**], better regret bounds were proven for the Forward algorithm than for the standard Ridge Regression algorithm. Therefore a natural open problem is whether our characterization of kernelizability can be generalized to algorithms that may predict with linear combinations of the labeled as well as the last unlabeled instance.

References

1. Abernethy, J., Bach, F., Evgeniou, T., Vert, J.P.: A new approach to collaborative filtering: Operator estimation with spectral regularization. Journal of Machine Learning 10, 803–826 (2009)
2. Argyriou, A., Micchelli, C.A., Pontil, M.: When is there a representer theorem? vector versus matrix regularizers. Journal of Machine Learning Research 10, 2507–2529 (2009)
3. Azoury, K., Warmuth, M.K.: Relative loss bounds for on-line density estimation with the exponential family of distributions. Journal of Machine Learning 43(3), 211–246 (June 2001)
4. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proc. 5th Annual ACM Workshop on Comput. Learning Theory. pp. 144–152. ACM Press, New York, NY (1992)
5. Cavallanti, G., Cesa-Bianchi, N., Gentile, C.: Linear algorithms for online multitask classification. In: Proceedings of the 21st Annual Conference on Learning Theory (COLT 08). pp. 251–262 (July 2008)
6. Forster, J.: On relative loss bounds in generalized linear regression. In: 12th International Symposium on Fundamentals of Computation Theory. pp. 269–280 (1999)
7. Herbster, M., Warmuth, M.K.: Tracking the best linear predictor. Journal of Machine Learning Research 1, 281–309 (2001)
8. Kimeldorf, G.S., Wahba, G.: Some results on Tchebycheffian spline functions. J. Math. Anal. Applic. 33, 82–95 (1971)
9. Kuzmin, D., Warmuth, M.K.: Online Kernel PCA with entropic matrix updates. In: Proceedings of the 24rd international conference on Machine learning (ICML '07). pp. 465–471. ACM International Conference Proceedings Series (June 2007)
10. Mika, S., Ratsch, G., Weston, J., Schölkopf, B., Mullers, K.R.: Fisher discriminant analysis with kernels. In: Proc. NNSP'99. IEEE Signal Processing Society Workshop. pp. 41–48 (1999)
11. Schölkopf, B., Herbrich, R., Smola, A.J.: A generalized representer theorem. In: Helmbold, D.P., Williamson, B. (eds.) Proceedings of the 14th Annual Conference on Computational Learning Theory. pp. 416–426. No. 2111 in Lecture Notes in Computer Science, Springer-Verlag, London, UK (2001)
12. Schölkopf, B., Smola, A.J., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation 10(5), 1299–1319 (1998)
13. Srebro, N., Sridharan, K., Tewari, A.: On the universality of online mirror descent. In: Advances in Neural Information Processing Systems 23 (NIPS '11). pp. 2645–2653 (2011)
14. Tsuda, K., Rätsch, G., Warmuth, M.K.: Matrix exponentiated gradient updates for on-line learning and Bregman projections. Journal of Machine Learning Research 6, 995–1018 (June 2005)
15. Vovk, V.: Competitive on-line statistics. International Statistical Review 69, 213–248 (2001)
16. Warmuth, M.K.: Winnowing subspaces. In: Proceedings of the 24rd international conference on Machine learning (ICML '07). ACM Press (June 2007)
17. Warmuth, M.K., Kuzmin, D.: Randomized PCA algorithms with regret bounds that are logarithmic in the dimension. Journal of Machine Learning Research 9, 2217–2250 (2008)
18. Warmuth, M.K., Vishwanathan, S.: Leaving the span. In: Proceedings of the 18th Annual Conference on Learning Theory (COLT '05). Springer-Verlag, Bertinoro, Italy (June 2005), journal version in progress