Kernelization of matrix updates, when and how?

Manfred K. Warmuth[a,1], Wojciech Kotłowski[b,2], Shuisheng Zhou[c,1]

[a] *Department of Computer Science, University of California, Santa Cruz, CA 95064*
[b] *Institute of Computing Science, Poznań University of Technology, Poznań, Poland*
[c] *School of Science, Xidian University, Xian, China, 710071*

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, Exponentiated Gradient algorithm, Gradient Descent algorithm.

1. Introduction

The following kernelization trick was popularized by a paper on support vector machines [BGV92] 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 each instance $\boldsymbol{x}$ to an expanded instance $\phi(\boldsymbol{x}) \in \mathbb{R}^N$ 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 instances is typically much larger than the dimension n of the original instances and may even 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. Kernelizable algorithms must have the property that they only

Email addresses: manfred@cse.ucsc.edu (Manfred K. Warmuth), wkotlowski@cs.put.poznan.pl (Wojciech Kotłowski), sszhou@mail.xidian.edu.cn (Shuisheng Zhou)

[1] The first author was supported by NSF grant IIS-0917397.

[2] The second author was supported by the Fundation for Polish Science under the Homing Plus Program, co-financed by the European Regional Development Fund. A portion of this research was done while this author was visiting UCSC supported by NSF grant IIS-0917397.

[3] This research was done while the third author was visiting UCSC supported by NNSFC grant 61179040.

[4] A preliminary version of this paper appeared at 24th International Conference on Algorithmic Learning Theory (ALT 2013) [WKZ12]. In this journal version we extended the background discussion, included all the proofs, and added an appendix in which we generalize the results from finite vector spaces in $\boldsymbol{R}^n$ to Hilbert spaces.

entail 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 by first considering instances that are outer products $\boldsymbol{x}\boldsymbol{y}'$, where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^m$, i.e. we first focus on rank 1 instance matrices. It will then be easy to generalize from outer products to asymmetric instance matrices $\boldsymbol{M} \in \mathbb{R}^{n \times m}$. 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. In this case the instances $\boldsymbol{x}\boldsymbol{x}'$ become $\phi(\boldsymbol{x})\phi(\boldsymbol{x})'$.

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 [WV05], 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 rotational invariance properties of the algorithm[5]. 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 prod-

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

ucts. 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 [TRW05, War07, KW07]. 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 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 [War07]. 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 [KW71, SHS01]. It states that whenever the solution minimizes the trade-off between a non-decreasing function of the Euclidean distance and a loss function that only depends on the dot products between the weight vector and expanded instance vectors, then there must be a solution that is a linear combination of the expanded instance vectors.[6] Representer type theorems have recently been generalized to the case of outer product instances [ABEV09, AMP09]. For instance, it is shown in [ABEV09] 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 kernelizability. Our characterization based on rotational invariance immediately leads to a simple Representer Theorem for matrix parameters that holds under the assumption that the objective of the minimization problem is rotationally invariant and the solution of the minimization problem is unique. Our proofs are elementary and intuitive, but the resulting Representer Theorem is incomparable with the one given in [ABEV09] because the latter allows for multiple solutions. Out methods also lead to a Representer Theorem for the case of symmetric matrix parameter, which is the mainstay of multiplicative updates, but was not considered in [ABEV09, AMP09]. In [CCBG08] it was also shown that the matrix version of the p-norm perceptron can be kernelized. Again kernelizability of this algorithm is easily implied by our methods.

[6] Representer Theorems for general Hilbert spaces instead of vector spaces in $\mathbb{R}^N$ are studied in [DS12]. In particular, it is determined which regularization terms guarantee that the existence of a solution that is a linear combination of the expanded instances.

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. For the vector case, invariance properties have been used before for giving necessary conditions for kernelizability [WV05, GBRS09]. Here we essentially repackage the theorem of [WV05] so that is gives us necessary and sufficient conditions for kearnalizability and then generalize this characterization to asymmetric and symmetric matrix instances.

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 in $\mathbb{R}^N$. No new insights were gained from this translation. Therefore in the main body of the paper we present our results in the notationally simpler case of real vector spaces and relegate the generalization to Hilbert spaces to the appendix.

Multiplicative updates and kernelizability

There are two main families of updates in machine learning that are commonly called "additive" and "multiplicative". The algorithms aim to minimize a loss function which is a function of the dot product between the parameter vector $\boldsymbol{w}$ and the expanded instance vectors $\phi(\boldsymbol{x})$. The additive (a.k.a. Gradient Descent) updates subtract a multiple of the gradient of the loss from the current parameter. If the loss is a function of the dot products, then the gradients as well as the entire parameter vector of the additive updates are always linear combinations of the expanded instances. In this case it is easy to see that the linear prediction with a new expanded instance only involves computing dot products, and this entire family of algorithms is kernelizable. Essentially transformation invariance is related to kernelizability and the Gradient Descent family of updates. The reduction to computing dot products often leads to concise algorithms. However, as discussed in [WV05, Ng04], this family of updates does not generalize well when the instances are orthogonal and target weight vectors are sparse.

For the multiplicative (a.k.a. Exponentiated Gradient) family of updates, each component w_i of the parameter vector $\boldsymbol{w}$ is multiplied by a factor that is essentially an exponential of the derivative of the loss with respect to w_i [KW97]. In other words the logarithms of the weights are updated additively, but the weights themselves are updated multiplicatively. Multiplicative updates on vector parameters are interesting because the bounds only depend logarithmically on the feature dimension n [Lit88, LW94]. A natural idea is to expand the instances via a feature map and run a multiplicative update on expanded instances. Now the bounds are logarithmic in the feature dimension N which is typically much larger while $\log N$ might still be tolerable. However this idea is only useful if the resulting update can be implemented efficiently, i.e. can be reduced to computing a small number of kernel computations.

Let us discuss these issues some more in connection with the problem of learning k-term DNF formulas. Efficient learnability of these formulas is a key open problem in machine Learning [Val84]. Consider the following algorithm for predicting well on a sequence of n dimensional bit patterns labeled consistently

with a k term DNF formula. First expand the instances using the feature map $\phi : \{0,1\}^n \to \{0,1\}^{2^n}$ which contains one feature for each of the 2^n monomials. Now run the Winnow algorithm [Lit88] from the multiplicative update family on the expanded examples $(\phi(\boldsymbol{x}_t), y_t)$ which predicts with a linear threshold function. The mistake bound of this algorithm is $O(k \log 2^n)$. Thus by doing repeated passes over the examples, the algorithm finds a consistent threshold function after $O(kn)$ passes. This algorithms has many good properties: It has a low mistake bound, which leads to good bound in the PAC model [FW95]. However we don't know how to implement it efficiently, even though the dot product $\phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}')$ between two expanded 2^n dimensional expanded instances requires only $O(n)$ time to compute [KW97]. In particular, this algorithm cannot be "kernelized", i.e. converted into a polynomial number of kernel computations. It seems that the algorithm must access the individual features and there are too many of them. Nevertheless in this paper we found a way to kernelize the *matrix version* of the multiplicative updates which use a density matrix as their parameter and updates this parameter using matrix logs and exponentials [TRW05, WK06]. Intuitively the updates remain efficient if the dimension of the eigenvectors of the instances is expanded but the number of non-zero eigenvalues of the instances stay polynomial.

In general, if you "lift" a vector problem to the matrix domain, then the straightforward way to do this is to place the instance vector along the diagonal of a corresponding instance matrix (See discussion in [TRW05, WK06, WK08]). However, the update time of the matrix multiplicative updates is typically at least polynomial in the number of non-zero components on the diagonal. Therefore in the sketched DNF learning problem, nothing is gained by placing each expanded instance $\phi(\boldsymbol{x})$ on the diagonal of an instance matrix and running the matrix version of the Winnow algorithm. For more general instance matrices, the number non-zero diagonal entries becomes the number of non-zero eigenvalues or singular values, and it seems that multiplicative updates have to pay for the total number of non-zeros eigenvalues in all the instance matrices (i.e. the total rank of the instance matrices). We have found a case in Section 4 where the regret bounds provable for matrix multiplicative updates are logarithmic in the total rank instances instead of the dimension of eigenvectors. As a matter of fact we show here that matrix multiplicative updates are kernelizable in the sense that the eigenvectors can be expanded (instead of the vector of eigenvalues). The dimension of the eigenvectors can even be unbounded as long as the dot product between such vectors can be computed efficiently via a kernel function.

Outline of the paper

In Section 2 we discuss our characterizations of learnability based on notions of rotational invariance. In Section 3 we give our version of the Representer Theorem and discuss the relationship to the standard versions. This is followed by some example applications of our characterizations in Section 4. We conclude the main body of the paper with a number of open problems in Section 5. This is followed by an appendix where we reformulate our characterization results in the language of Hilbert spaces.

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 $1 \le p, q \le T+1$.

Definition 2.1. *An algorithm $\mathcal{A}$ for vector instances is* 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 2.2. *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 corollary, which gives and equivalent definition of kernelizability:

Corollary 2.3. *An algorithm $\mathcal{A}$ for vector instances 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 focus on a special class of algorithm and then characterize kernelizability for this class.

Definition 2.4. *An algorithm $\mathcal{A}$ for vector instances is* linear*, if upon receiving input sequence $\boldsymbol{\mathcal{S}}$ and an unlabeled instance $\boldsymbol{x}$, the algorithm 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, a linear algorithm is the algorithm which learns a linear function. Note, that we only require the output of the algorithm to be linear with respect to $\boldsymbol{x}$; clearly, the produced $\boldsymbol{w}$ may be nonlinear in $\boldsymbol{\mathcal{S}}$.

Theorem 2.5. *A linear algorithm $\mathcal{A}$ for vector instances 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 [WV05]. 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 $\boldsymbol{x}\boldsymbol{y}'$, where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^m$. Examples have the form $(\boldsymbol{x}\boldsymbol{y}', \ell)$, 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 $\boldsymbol{x}\boldsymbol{y}'$ to some fixed output range.[7] 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.

The structure of this section is analogous to the previous section on vector instances. We begin with our definition of kernelizability:

Definition 2.6. *An algorithm $\mathcal{A}$ for asymmetric outer product instances is* kernelizable*, if for any two input sequences $\boldsymbol{\mathcal{S}}, \boldsymbol{x}\boldsymbol{y}'$ 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. $\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{x}\boldsymbol{y}') = \mathcal{A}(\widetilde{\boldsymbol{\mathcal{S}}}, \widetilde{\boldsymbol{x}}\widetilde{\boldsymbol{y}}')$.*

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 $\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}'$ denote the transformed sequence $\{(\boldsymbol{U}\boldsymbol{x}_t\boldsymbol{y}_t'\boldsymbol{V}', \ell_t)\}_{t=1}^T$. By applying Lemma 2.2 twice (to the left vectors $\boldsymbol{x}_t$ and the right vectors $\boldsymbol{y}_t$), we immediately get the following alternate definition of kernelizability:

Corollary 2.7. *An algorithm $\mathcal{A}$ for asymmetric outer product instances is kernelizable iff for all sequences $\boldsymbol{\mathcal{S}}$, next instances $\boldsymbol{x}\boldsymbol{y}'$ and orthogonal matrices $\boldsymbol{U}, \boldsymbol{V}$,*

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

The generalization of the linearity of algorithms to the matrix domain is straightforward:

Definition 2.8. *An algorithm $\mathcal{A}$ for outer product instances is* linear*, if upon input $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{x}\boldsymbol{y}'$, the algorithm 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* $\mathrm{tr}(\boldsymbol{W}'\boldsymbol{x}\boldsymbol{y}')$.

[7] For conciseness we use outer products $\boldsymbol{x}\boldsymbol{y}'$ 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.

We now give a characterization of the kernerlizability for linear algorithms:

Theorem 2.9. *A linear algorithm $\mathcal{A}$ for asymmetric outer product instances 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 $\boldsymbol{W} = \boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'$, 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 $\boldsymbol{X}'\boldsymbol{X}$ and $\boldsymbol{Y}'\boldsymbol{Y}$.*

Note that the expression $\boldsymbol{W} = \boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'$ 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 $\mathcal{A}$ outputs the trace, $\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{x}\boldsymbol{y}') = \mathrm{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})'\boldsymbol{x}\boldsymbol{y}')$, it follows from Corollary 2.7 that $\mathcal{A}$ is kernelizable iff:

$$\mathrm{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})'\, \boldsymbol{x}\boldsymbol{y}') = \mathrm{tr}(\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}')'\, \boldsymbol{U}\boldsymbol{x}\boldsymbol{y}'\boldsymbol{V}'), \tag{2.1}$$

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 $\boldsymbol{y} \in \mathbb{R}^m$.

The proof of the "if" part 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 (2.1) and kernelizability:

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

The proof of the "only if" part is divided into two parts. In Part 1, we first show that (2.1) 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 (2.1) 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 2.2 this is equivalent to stating that $\boldsymbol{C}$ depends on $\boldsymbol{\mathcal{S}}$ only via the kernel matrices $\boldsymbol{X}'\boldsymbol{X}$ and $\boldsymbol{Y}'\boldsymbol{Y}$.

Proof of Part 1: Let $\{\widehat{\boldsymbol{x}}_p\}_{p=1}^{r_1}$ be an orthonormal basis for $Span\left(\{\boldsymbol{x}_t\}_{t=1}^T\right)$ and $\{\widehat{\boldsymbol{y}}_q\}_{q=1}^{r_2}$ 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. Since $\widehat{\boldsymbol{x}}_p \in Span(\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\})$ and $\widehat{\boldsymbol{y}}_q \in Span(\{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_T\})$, there exist matrices $\boldsymbol{P} \in \mathbb{R}^{T \times r_1}$ and $\boldsymbol{Q} \in \mathbb{R}^{T \times r_2}$ such that

$$\widehat{\boldsymbol{x}}_p = \sum_{i=1}^T \boldsymbol{P}_{i,p}\boldsymbol{x}_i \quad \text{and} \quad \widehat{\boldsymbol{y}}_q = \sum_{j=1}^T \boldsymbol{Q}_{j,q}\boldsymbol{y}_j. \tag{2.2}$$

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 $p \in \{r_1+1, \ldots, n\}$, and any index $q \in \{1, \ldots, m\}$, and we now show that $\hat{c}_{p,q} = 0$ (the case $q \in \{r_2+1, \ldots, m\}$ and $p \in \{1, \ldots, n\}$ is proven similarly). We use the notion of transformation invariance (2.1). We choose the new instance the be $\boldsymbol{x} = \widehat{\boldsymbol{x}}_p$ and $\boldsymbol{y} = \widehat{\boldsymbol{y}}_q$. Furthermore, we choose $\boldsymbol{U}$ as the *Hauseholder reflection matrix*, $\boldsymbol{U} = \boldsymbol{I} - 2\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_p'$ and $\boldsymbol{V} = \boldsymbol{I}$. It holds that $\widehat{\boldsymbol{x}}_p \perp \boldsymbol{x}_t$ for any $t = 1, \ldots, T$, because $p > r_1$ and $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^{r_1}$ is the orthogonal basis for $Span\left(\{\boldsymbol{x}_t\}_{t=1}^{T}\right)$. This implies that $\boldsymbol{U}\boldsymbol{x}_t = \boldsymbol{x}_t - 2\widehat{\boldsymbol{x}}_p(\widehat{\boldsymbol{x}}_p'\boldsymbol{x}_t) = \boldsymbol{x}_t$, i.e. the left instances are not affected by $\boldsymbol{U}$. Also, $\boldsymbol{V}\boldsymbol{y}_t = \boldsymbol{I}\boldsymbol{y}_t = \boldsymbol{y}_t$, i.e. the right instances are not affected by $\boldsymbol{V}$. 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 (2.1) becomes:

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

However since $\widehat{\boldsymbol{x}}_p'\widehat{\boldsymbol{x}}_p = 1$, we have $\boldsymbol{U}\widehat{\boldsymbol{x}}_p = \widehat{\boldsymbol{x}}_p - 2\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_p'\widehat{\boldsymbol{x}}_p = -\widehat{\boldsymbol{x}}_p$ and therefore the r.h.s. of Equation (2.1) has the opposite sign:

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

We conclude that the transformation invariance (2.1) implies $\hat{c}_{p,q} = 0$ if $p > r_1$ (and similarly $\hat{c}_{p,q} = 0$ if $q > r_2$). It follows that

$$\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{p=1}^{r_1}\sum_{q=1}^{r_2}\hat{c}_{p,q}\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{y}}_q'.$$

and we now prove Part 1 by plugging (2.2) into the above:

$$\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{p,q}\hat{c}_{p,q}\sum_i\sum_j\boldsymbol{P}_{i,p}\boldsymbol{Q}_{j,q}\boldsymbol{x}_i\boldsymbol{y}_j' = \sum_{i,j}\underbrace{\sum_{p,q}\boldsymbol{P}_{i,p}\hat{c}_{p,q}\boldsymbol{Q}_{j,q}}_{\boldsymbol{C}_{i,j}}\boldsymbol{x}_i\boldsymbol{y}_j'.$$

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 $Span\left(\{\boldsymbol{x}_t\}_{t=1}^{T}\right)$, $\{\boldsymbol{U}\widehat{\boldsymbol{x}}_i\}_{i=1}^{r_1}$ is an orthonormal basis for $Span\left(\{\boldsymbol{U}\boldsymbol{x}_t\}_{t=1}^{T}\right)$ (and similarly for $Span\left(\{\boldsymbol{y}_t\}_{t=1}^{T}\right)$ and $Span\left(\{\boldsymbol{V}\boldsymbol{y}_t\}_{t=1}^{T}\right)$). To finish the proof, it suffices to show that $\hat{c}_{p,q} = \hat{d}_{p,q}$ for any $p \in \{1, \ldots, r_1\}$ and any $q \in \{1, \ldots, r_2\}$. Then Part 2 follows immediately by plugging (2.2) into the above equality. By (2.1),

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

□

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 2.9 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.

Definition 2.10. *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 2.2, we immediately get the following alternate definition of kernelizability:

Corollary 2.11. *An algorithm $\mathcal{A}$ for symmetric outer product instances 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.

Definition 2.12. *An algorithm $\mathcal{A}$ for symmetric outer product instances is* linear*, if upon input $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{x}\boldsymbol{x}'$, the algorithm 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* $\mathrm{tr}(\boldsymbol{W}'\boldsymbol{x}\boldsymbol{x}')$.[8]

Theorem 2.13. *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. By Corollary (2.11), $\mathcal{A}$ is kernelizable if and only if:

$$\mathrm{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})'\,\boldsymbol{x}\boldsymbol{x}') = \mathrm{tr}(\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}')'\,\boldsymbol{U}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{U}'), \tag{2.3}$$

for all $\boldsymbol{\mathcal{S}}, \boldsymbol{U}, \boldsymbol{x}\boldsymbol{x}'$. The proof of the "if" part is easy and very similar to that of Theorem 2.9: Since $\boldsymbol{C}$ and c depend on $\boldsymbol{\mathcal{S}}$ only via the kernel matrix, $\boldsymbol{C}$ and c are invariant under orthogonal transformation $\boldsymbol{\mathcal{S}} \mapsto \boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}'$, and thus $\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}') = \boldsymbol{U}\boldsymbol{X}\boldsymbol{C}\boldsymbol{X}'\boldsymbol{U}' + c\boldsymbol{I} = \boldsymbol{U}\boldsymbol{W}(\boldsymbol{\mathcal{S}})\boldsymbol{U}'$. This implies (2.3) and kernelizability:

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

The proof of the "only if" part is divided into two parts, as in Theorem 2.9. In Part 1, we show that (2.3) 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 $c \in \mathbb{R}$. Then, in Part 2, we show that (2.3) implies that for any $\boldsymbol{\mathcal{S}}$ and any orthogonal matrix $\boldsymbol{U}$, $\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}') = \boldsymbol{U}\boldsymbol{X}\boldsymbol{C}\boldsymbol{X}'\boldsymbol{U}' + c\boldsymbol{I}$. This means that $\boldsymbol{C}$ and c are invariant under orthogonal transformations of the example sequence $\boldsymbol{\mathcal{S}}$, and thus by Lemma 2.2 this is equivalent to stating that $\boldsymbol{C}$ and c depend on $\boldsymbol{\mathcal{S}}$ only via the kernel matrix $\boldsymbol{X}'\boldsymbol{X}$.

Proof of Part 1: Let $\{\widehat{\boldsymbol{x}}_p\}_{p=1}^r$ be an orthonormal basis for $Span\left(\{\boldsymbol{x}_t\}_{t=1}^T\right)$. Since $\widehat{\boldsymbol{x}}_p \in Span\left(\{\boldsymbol{x}_t\}_{t=1}^T\right)$, there exists matrix $\boldsymbol{P} \in \mathbb{R}^{T\times r}$ such that

$$\widehat{\boldsymbol{x}}_p = \sum_{i=1}^T \boldsymbol{P}_{i,p}\boldsymbol{x}_i. \tag{2.4}$$

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 (a) $\hat{c}_{p,q} = 0$ if $p \neq q$ and either $p > r$ or $q > r$, and that (b) $\hat{c}_{p,p} = \hat{c}$ for some constant $\hat{c}$, for $p > r$. We show (a) 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 $p \geq 1, p \neq q$. Choose $\boldsymbol{x} = \widehat{\boldsymbol{x}}_p + \widehat{\boldsymbol{x}}_q$ and $\boldsymbol{U}$ as the Hauseholder reflection $\boldsymbol{I} - 2\widehat{\boldsymbol{x}}_q\widehat{\boldsymbol{x}}_q'$. Then, for any $1 \leq t \leq T$, we have $\boldsymbol{U}\boldsymbol{x}_t = \boldsymbol{x}_t - 2\widehat{\boldsymbol{x}}_q\widehat{\boldsymbol{x}}_q'\boldsymbol{x}_t = \boldsymbol{x}_t$ (because $q > r$ and thus $\widehat{\boldsymbol{x}}_q$ lies outside $Span\left(\{\boldsymbol{x}_t\}_{t=1}^T\right)$). Thus, the transformed sample $\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}'$ is the same as the original sample $\boldsymbol{\mathcal{S}}$, and

[8] 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 $\mathrm{tr}(\boldsymbol{W}_{\mathrm{sym}}'\boldsymbol{x}\boldsymbol{x}') = \mathrm{tr}(\boldsymbol{W}'\boldsymbol{x}\boldsymbol{x}')$.

$\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}') = \boldsymbol{W}(\boldsymbol{\mathcal{S}})$. On the other hand, $\boldsymbol{U}\widehat{\boldsymbol{x}}_p = \widehat{\boldsymbol{x}}_p - 2\widehat{\boldsymbol{x}}_q\widehat{\boldsymbol{x}}_q'\widehat{\boldsymbol{x}}_p = \widehat{\boldsymbol{x}}_p$, and $\boldsymbol{U}\widehat{\boldsymbol{x}}_q = \widehat{\boldsymbol{x}}_q - 2\widehat{\boldsymbol{x}}_q\widehat{\boldsymbol{x}}_q'\widehat{\boldsymbol{x}}_q = -\widehat{\boldsymbol{x}}_q$. Therefore, the l.h.s. and r.h.s. of (2.3) become

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

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

To show (b), we choose $\boldsymbol{x} = \widehat{\boldsymbol{x}}_p$ (we remind that $p > r$), 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 $1 \le t \le T$ (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. (2.3) 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 (2.3) implies that

$$\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{p=1}^{r}\sum_{q=1}^{r} \hat{c}_{p,q}\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_q' + \hat{c}\sum_{p=r+1}^{n} \widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_p' = \sum_{p=1}^{r}\sum_{q=1}^{r} (\hat{c}_{p,q} - \hat{c}\delta_{pq})\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_q' + \hat{c}\sum_{p=1}^{n} \widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_p'$$

and we now prove Part 1 by plugging (2.4) into the above:

$$\begin{aligned}
\boldsymbol{W}(\boldsymbol{\mathcal{S}}) &= \sum_{p,q}(\hat{c}_{p,q} - \hat{c}\delta_{pq})\sum_{i,j} \boldsymbol{P}_{i,p}\boldsymbol{P}_{j,q}\boldsymbol{x}_i\boldsymbol{x}_j' + \hat{c}\sum_{p=1}^{n} \widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_p' \\
&= \sum_{i,j} \underbrace{\left(\sum_{p,q} \boldsymbol{P}_{i,p}(\hat{c}_{p,q} - \hat{c}\delta_{pq})\boldsymbol{P}_{j,q}\right)}_{\boldsymbol{C}_{i,j}} \boldsymbol{x}_i\boldsymbol{x}_j' + \underbrace{\hat{c}}_{c}\underbrace{\sum_{p=1}^{n} \widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_p'}_{\boldsymbol{I}}.
\end{aligned}$$

Without loss of generality, $\boldsymbol{C}$ 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 $\boldsymbol{W}(\boldsymbol{\mathcal{S}})$. Indeed,

$$\begin{aligned}
\sum_{i,j} \frac{\boldsymbol{C}_{i,j} + \boldsymbol{C}_{j,i}}{2}\boldsymbol{x}_i\boldsymbol{x}_j' + c\boldsymbol{I} &= \frac{1}{2}\left(\sum_{i,j} \boldsymbol{C}_{i,j}\boldsymbol{x}_i\boldsymbol{x}_j' + c\boldsymbol{I}\right) + \frac{1}{2}\left(\sum_{i,j} \boldsymbol{C}_{i,j}\boldsymbol{x}_j\boldsymbol{x}_i' + c\boldsymbol{I}\right) \\
&= \frac{1}{2}\boldsymbol{W} + \frac{1}{2}\boldsymbol{W}' = \boldsymbol{W}.
\end{aligned}$$

Proof of Part 2: By Part 1, $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i=1}^{r}\sum_{j=1}^{r} \hat{c}_{i,j}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{x}}_j' + \hat{c}\sum_{i=r+1}^{n}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{x}}_i'$. By applying Part 1 to the sequence $\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}'$ we get $\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}') = \sum_{i=1}^{r}\sum_{j=1}^{r} \hat{d}_{i,j}\boldsymbol{U}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{x}}_j'\boldsymbol{U}' + \hat{d}\sum_{i=r+1}^{n}\boldsymbol{U}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{x}}_i'\boldsymbol{U}'$ for some coefficients $\hat{d}_{i,j}, \hat{d}$, because if $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^{r}$ is an orthonormal basis for $Span\left(\{\boldsymbol{x}_t\}_{t=1}^{T}\right)$, then $\{\boldsymbol{U}\widehat{\boldsymbol{x}}_i\}_{i=1}^{r}$ is an orthonormal basis for $Span\left(\{\boldsymbol{U}\boldsymbol{x}_t\}_{t=1}^{T}\right)$, and if $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^{n}$ is an orthonormal basis for $\mathbb{R}^n$, then so is $\{\boldsymbol{U}\widehat{\boldsymbol{x}}_i\}_{i=1}^{n}$. To finish the proof, it suffices to show that $\hat{c}_{p,q} = \hat{d}_{p,q}$ for any $p, q \in \{1, \ldots, r\}$, and that $\hat{c} = \hat{d}$. Then Part 2 follows immediately by plugging (2.4) into the above equality. By (2.3), for $1 \le p, q \le r$,

$$\begin{aligned}
\hat{c}_{p,q} &= \mathrm{tr}\Big(\Big(\sum_{i,j}\hat{c}_{i,j}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{x}}_j' + \hat{c}\sum_{i=r+1}^{n}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{x}}_i'\Big)'\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_q'\Big) = \mathrm{tr}((\boldsymbol{W}(\boldsymbol{\mathcal{S}}))'\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_q') \\
&= \mathrm{tr}(\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}')'\boldsymbol{U}\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_q\boldsymbol{U}') = \mathrm{tr}\Big(\Big(\sum_{i,j}\hat{d}_{i,j}\boldsymbol{U}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{x}}_j'\boldsymbol{U}' + \hat{d}\sum_{i=r+1}^{n}\boldsymbol{U}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{x}}_i'\boldsymbol{U}'\Big)'\boldsymbol{U}\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_q'\boldsymbol{U}'\Big) \\
&= \mathrm{tr}\Big(\Big(\sum_{i,j}\hat{d}_{i,j}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{x}}_j' + \hat{d}\sum_{i=r+1}^{n}\widehat{\boldsymbol{x}}_i\widehat{\boldsymbol{x}}_i'\Big)'\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_q'\Big) = \hat{d}_{p,q}.
\end{aligned}$$

Similarly, for any $p > r$,

$$\hat{c} = \mathrm{tr}((\boldsymbol{W}(\boldsymbol{\mathcal{S}}))'\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_p') = \mathrm{tr}(\boldsymbol{W}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}')'\boldsymbol{U}\widehat{\boldsymbol{x}}_p\widehat{\boldsymbol{x}}_p\boldsymbol{U}') = \hat{d}.$$

□

Comparing Theorem 2.13 with Theorem 2.9, an additional term $c\boldsymbol{I}$ entered the expansion. The presence of the 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, i.e. the identity matrix. Such an element does not exist for asymmetric matrices. We note, that the output $\mathrm{tr}(\boldsymbol{W}(\boldsymbol{\mathcal{S}})\boldsymbol{x}\boldsymbol{x}')$ of the algorithms with the parameter matrix $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \boldsymbol{X}\boldsymbol{C}\boldsymbol{X}' + c\boldsymbol{I}$ can be written only be means of coefficient matrix $\boldsymbol{C}$, the single coefficient c and the dot products $\boldsymbol{x}_j'\boldsymbol{x}$ and $\boldsymbol{x}'\boldsymbol{x}$.

Note that when the feature map $\boldsymbol{x} \to \phi(\boldsymbol{x})$ is not the identity, then the dot products become dot products between the expanded instances $\phi(\boldsymbol{x}_j)$ and $\phi(\boldsymbol{x})$, respectively. In particular, the term $c\boldsymbol{I}$ can easily be dealt with when the instances are expanded, as it leads to the expressions of the form $\mathrm{tr}(c\boldsymbol{I}\phi(\boldsymbol{x})\phi(\boldsymbol{x})') = c\,k(\boldsymbol{x}, \boldsymbol{x})$. Also Theorem 2.13 generalizes easily from symmetric outer product instances to symmetric matrix instances with fixed rank s.

An example of algorithm, which falls into the framework of Theorem 2.13, but not of Theorem 2.9, is the aforementioned Matrix Exponentiated Gradient. The weight matrix proportional to $\mathbf{exp}\left(\sum_i \alpha_i\boldsymbol{x}_i\boldsymbol{x}_i'\right)$, for some coefficients α_i, is clearly a full rank matrix. However, eigenvalues of $\boldsymbol{W}$ in the space orthogonal to $Span(\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t\})$ are all equal to each other. This is because $\mathbf{exp}\left(\sum_i \alpha_i\boldsymbol{x}_i\boldsymbol{x}_i'\right)$ acts by exponentiating the eigevalues of $\sum_i \alpha_i\boldsymbol{x}_i\boldsymbol{x}_i'$, which are zero outside the data span. Therefore, MEG is a kernelizable algorithm.

3. Kernelization via Representer Theorems

A standard way to ensure kernelizability of an algorithm is to apply the Representer Theorem [KW71, SHS01]. In the vector case, it states that whenever the solution minimizes the trade-off between essentially a non-decreasing function of the Euclidean distance and a loss function that only depends on the dot products between the weight vector and feature vector, then there always is a solution which is a linear combination of the feature vectors [DS12]. Representer type theorems have recently been generalized to the case of outer product instances [ABEV09, AMP09]. In particular, the following Representer Theorem for asymmetric outer product instances was proven in [ABEV09]: Given a penalty function $\Omega(\boldsymbol{W}) = \sum_{i=1}^{d} s_i(\sigma_i(\boldsymbol{W}))$, 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

$$\min_{\boldsymbol{W}} \quad \Omega(\boldsymbol{W}) \; + \; \eta \sum_{t} \mathrm{loss}_t(\mathrm{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{y}_t')), \tag{3.1}$$

which can be written as $\boldsymbol{W} = \sum_{i,j} \boldsymbol{C}_{i,j}\boldsymbol{x}_i\boldsymbol{y}_j' = \boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'$. Representer theorems rely on the following proof method: If the algorithm is defined as a minimization problem that regularizes with a function that shrinks with the spectrum, and if the weight matrix had an orthogonal component (outside the data span), then removing this component would not change the predictions on the set of instances but would decrease the regularization term.

Note that representer theorems only give necessary conditions for kernelization. In contrast, our geometric characterizations of linear algorithms give necessary and sufficient conditions for kernelization provided that the solution to the optimization problem is always unique. Also, our proof methods have a different flavor: If the weight matrix has an orthogonal component, then we build an instance with this orthogonal component and show that the prediction on this instance and the prediction on a transformed instance do not match. Moreover, the geometric characterization presented in Section 2, immediately leads to the following version of the Representer Theorem:

Theorem 3.1. *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 $\mathcal{A}$ be a linear algorithm which, upon receiving input sequence $\boldsymbol{\mathcal{S}}$, produces the weight matrix $\boldsymbol{W}^*(\boldsymbol{\mathcal{S}})$. In this definition, we used the fact that $\boldsymbol{W}^*(\boldsymbol{\mathcal{S}})$ is uniquely defined for any $\boldsymbol{\mathcal{S}}$. Due to the rotational invariance of $\mathcal{L}$ and the uniqueness of the solution, $\boldsymbol{W}^*(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}') = \boldsymbol{U}\boldsymbol{W}^*(\boldsymbol{\mathcal{S}})\boldsymbol{V}'$. Thus for any $\boldsymbol{x}\boldsymbol{y}'$,

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

and the theorem now follows from the forward direction of Theorem 2.9. □

The problem (3.1) is rotationally invariant (because Ω is a function of the singular values only), so Theorem 3.1 applies as long as the solution is unique. Note that [ABEV09] specify somewhat different conditions: no uniqueness assumption is needed, rather some structure of the penalty function is imposed. However, we can strengthen Theorem 3.1 in such a way that we no longer require that the optimization problem has a unique solution, but we only require that we can always pick a single solution out of the set of optimal solutions in a *rotationally invariant way*, i.e. if $\boldsymbol{W}^*(\boldsymbol{\mathcal{S}})$ is the solution picked for a sequence $\boldsymbol{\mathcal{S}}$, then $\boldsymbol{W}^*(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}') = \boldsymbol{U}\boldsymbol{W}^*(\boldsymbol{\mathcal{S}})\boldsymbol{V}'$ for any orthogonal $\boldsymbol{U}$ and $\boldsymbol{V}$. For example, if there exist multiple optimal solutions, but the *shortest* optimal solution is unique (i.e. the one minimizing some rotation invariant matrix norm $\|\cdot\|$ among all optimal solutions), then the algorithm can choose this solution and our Representer Theorem 3.1 will apply. This is e.g. the case when $\Omega(\boldsymbol{W}) = 0$ (i.e. no regularization) and when the shortest solution is unique.

Most importantly, our conditions apply to a much wider class of linear 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 obtain the version of the Representer Theorem for the case of symmetric outer products, which is the mainstay of multiplicative updates, but which has not been considered elsewhere:

Theorem 3.2. *Consider the problem* $\min_{\text{sym.}\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}, \boldsymbol{\mathcal{S}})$*, 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* $\boldsymbol{X}'\boldsymbol{X}$*.*

Proof. Essentially the same as proof of Theorem 3.1. □

We conclude this section with a discussion of an optimization problem that exemplifies the difference between the Representer Theorem approach and ours. Consider minimizing the following quadric loss:

$$\min_{w \in \boldsymbol{R}} \ (\,(w) \cdot (1)\,)^2 - 1)^2.$$

Thus we are optimizing a 1-dimensional weight vector (w) and there is a single instance vector (1) of dimension 1. Note that there is no regularization term. There are two solutions at ± 1. Even after adding the regularization w^2 to the objective, then there are still two minima. The Representer Theorem for vector instances implies that there is a solution which is a linear combination of the instances. This is clearly the case since both solutions are linear combinations of the instance vector (1). The objective is rotation invariant (which in dimension 1 means invariance under multiplying both w and the instance vector by -1), but since there are multiple solutions, our theorems don't apply. Both solutions have the same norm and

therefore there is no unique shortest solution. As a matter of fact, there is no way to distinguish the two solutions in a rotation invariant way.

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 $\boldsymbol{XCY'}$, 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 (see Figure 4.1). 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 $\boldsymbol{X'X}, \boldsymbol{Y'Y} \in \mathbb{R}^{T\times T}$:

Lemma 4.1. *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 $\sqrt{\boldsymbol{X'X}}\boldsymbol{C}\sqrt{\boldsymbol{Y'Y}}$, 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 $\tilde{\boldsymbol{U}} = \boldsymbol{XC}\sqrt{\boldsymbol{Y'Y}}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}$ and $\tilde{\boldsymbol{V}} = \boldsymbol{YC'}\sqrt{\boldsymbol{X'X}}\boldsymbol{U}\boldsymbol{\Sigma}^{-1}$. 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 $\sqrt{\boldsymbol{X'X}}\boldsymbol{C}\sqrt{\boldsymbol{X'X}}$, 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 $\tilde{\boldsymbol{U}} = \boldsymbol{XC}\sqrt{\boldsymbol{X'X}}\boldsymbol{U}\boldsymbol{\Sigma}^{-1}$.*

Proof. Since $\boldsymbol{\Sigma}$ contains the non-zero singular value of $\sqrt{\boldsymbol{X'X}}\boldsymbol{C}\sqrt{\boldsymbol{Y'Y}}$, $\boldsymbol{\Sigma}^{-1}$ is well defined and the values are non-increasing as you go down the diagonal of $\boldsymbol{\Sigma}$. It is easy to check that the columns of $\tilde{\boldsymbol{U}}$ and $\tilde{\boldsymbol{V}}$ are orthonormal:

$$
\begin{aligned}
\tilde{\boldsymbol{U}}'\tilde{\boldsymbol{U}} &= \boldsymbol{\Sigma}^{-1}\boldsymbol{V'}\ \sqrt{\boldsymbol{Y'Y}}\boldsymbol{C'}\boldsymbol{X'}\ \ \boldsymbol{XC}\sqrt{\boldsymbol{Y'Y}}\ \ \boldsymbol{V}\boldsymbol{\Sigma}^{-1} \\
&= \boldsymbol{\Sigma}^{-1}\boldsymbol{V'}\ \underbrace{\sqrt{\boldsymbol{Y'Y}}\boldsymbol{C'}\sqrt{\boldsymbol{X'X}}}_{\boldsymbol{V\Sigma U'}}\ \underbrace{\sqrt{\boldsymbol{X'X}}\boldsymbol{C}\sqrt{\boldsymbol{Y'Y}}}_{\boldsymbol{U\Sigma V'}}\ \boldsymbol{V}\boldsymbol{\Sigma}^{-1} \\
&= \boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}\ \ \boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1} \ = \ \boldsymbol{I}, \\
\tilde{\boldsymbol{V}}'\tilde{\boldsymbol{V}} &= \boldsymbol{\Sigma}^{-1}\boldsymbol{U'}\ \sqrt{\boldsymbol{X'X}}\boldsymbol{C}\boldsymbol{Y'}\ \ \boldsymbol{YC'}\sqrt{\boldsymbol{X'X}}\ \ \boldsymbol{U}\boldsymbol{\Sigma}^{-1} \\
&= \boldsymbol{\Sigma}^{-1}\boldsymbol{U'}\ \underbrace{\sqrt{\boldsymbol{X'X}}\boldsymbol{C}\sqrt{\boldsymbol{Y'Y}}}_{\boldsymbol{U\Sigma V'}}\ \underbrace{\sqrt{\boldsymbol{Y'Y}}\boldsymbol{C'}\sqrt{\boldsymbol{X'X}}}_{\boldsymbol{V\Sigma U'}}\ \boldsymbol{U}\boldsymbol{\Sigma}^{-1} \\
&= \boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma} \ = \ \boldsymbol{I}.
\end{aligned}
$$

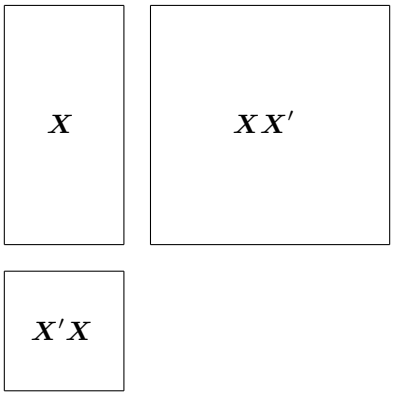

Figure 4.1: Illustrative scheme for Lemma 4.1: Left instance matrix (top left), Gram matrix (top right), kernel matrix (bottom left).

The following completes the proof that $\tilde{\boldsymbol{U}}\boldsymbol{\Sigma}\tilde{\boldsymbol{V}}'$ is the compact SVD decomposition of $\boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'$.

$$\begin{aligned}
\tilde{\boldsymbol{U}}'\,\boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'\,\tilde{\boldsymbol{V}} &= \boldsymbol{\Sigma}^{-1}\boldsymbol{V}'\,\sqrt{\boldsymbol{Y}'\boldsymbol{Y}}\boldsymbol{C}'\boldsymbol{X}'\,\boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'\,\boldsymbol{Y}\boldsymbol{C}'\sqrt{\boldsymbol{X}'\boldsymbol{X}}\;\boldsymbol{U}\boldsymbol{\Sigma}^{-1} \\
&= \boldsymbol{\Sigma}^{-1}\boldsymbol{V}'\,\underbrace{\sqrt{\boldsymbol{Y}'\boldsymbol{Y}}\boldsymbol{C}'\sqrt{\boldsymbol{X}'\boldsymbol{X}}}_{\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}'}\,\underbrace{\sqrt{\boldsymbol{X}'\boldsymbol{X}}\boldsymbol{C}\sqrt{\boldsymbol{Y}'\boldsymbol{Y}}}_{\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'}\,\underbrace{\sqrt{\boldsymbol{Y}'\boldsymbol{Y}}\boldsymbol{C}'\sqrt{\boldsymbol{X}'\boldsymbol{X}}}_{\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}'}\;\boldsymbol{U}\boldsymbol{\Sigma}^{-1} \\
&= \boldsymbol{\Sigma}^{-1}\,\boldsymbol{\Sigma}\boldsymbol{\Sigma}\boldsymbol{\Sigma}\,\boldsymbol{\Sigma}^{-1} \;=\; \boldsymbol{\Sigma}.
\end{aligned}$$

This proves the first part of the lemma. The second part (eigendecomposition) is proven in the same way, by setting $\boldsymbol{V} = \boldsymbol{U}$ and $\boldsymbol{Y} = \boldsymbol{X}$ in the derivation. □

For symmetric instances, a particularly simple case is obtained when $\boldsymbol{C} = \boldsymbol{I}$. Then, $\sqrt{\boldsymbol{X}'\boldsymbol{X}}\,\boldsymbol{C}\sqrt{\boldsymbol{X}'\boldsymbol{X}}$ simplifies to the kernel matrix $\boldsymbol{X}'\boldsymbol{X}$, and the above decomposition lemma simplifies to a derivation of the eigendecomposition of the (symmetric) covariance matrix $\boldsymbol{X}\boldsymbol{X}'$ from the eigendecomposition of the (single) kernel matrix $\boldsymbol{X}'\boldsymbol{X}$:

Corollary 4.2. *For any $\boldsymbol{X} \in \mathbb{R}^{n\times T}$, if $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}'$ is a compact eigendecomposition of $\boldsymbol{X}'\boldsymbol{X}$, then $\boldsymbol{X}\boldsymbol{X}'$ has the compact eigendecomposition $\widetilde{\boldsymbol{U}}\boldsymbol{\Sigma}\widetilde{\boldsymbol{U}}'$, where $\widetilde{\boldsymbol{U}} = \boldsymbol{X}\boldsymbol{U}\boldsymbol{\Sigma}^{-1/2}$.*

This known fact was key to the kernelization of PCA and Fisher Linear Discriminant Functions [SSM98, MRW+99]. Its proof is much simpler than our decomposition lemma based on the SVD decomposition. In the above corollary, the covariance matrix $\boldsymbol{X}\boldsymbol{X}'$ is symmetric positive definite. Curiously enough, when we try to decompose $\boldsymbol{X}\boldsymbol{C}\boldsymbol{X}'$, when $\boldsymbol{C}$ is diagonal with negative entries, then our more intricate version of the lemma based on the SVD decomposition seems to be necessary.

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 $\mathcal{W} = \{\boldsymbol{W} \colon \|\boldsymbol{W}\| \leq B\}$, 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 $\|\boldsymbol{x}_t\|_2 \leq 1$ and $\|\boldsymbol{y}_t\|_2 \leq 1$ for all t, where $\|\cdot\|_2$ is the Euclidean norm. A popular approach to solve the minimization problem is the online gradient descent (GD) [HW01]: 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:

$$\boldsymbol{W}_{t+1} = \operatorname*{argmin}_{\boldsymbol{W}\in\mathcal{W}} \ \|\boldsymbol{W} - \boldsymbol{W}_t\|_F^2 + \eta\partial_t(\boldsymbol{W}_t)\ \text{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{y}_t'), \tag{4.1}$$

where $\|\cdot\|_F$ is the Frobenius norm. Solving (4.1) leads to the *additive update*:

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

where the projection operation is defined as

$$\text{proj}(\boldsymbol{W}) = \operatorname*{argmin}_{\|\widetilde{\boldsymbol{W}}\|\leq B} \|\boldsymbol{W} - \widetilde{\boldsymbol{W}}\|_F^2.$$

Since the norm $\|\cdot\|$ is rotationally invariant, the projection becomes a projection on the singular values $\{\sigma_1, \ldots, \sigma_{\min\{n,m\}}\}$ of $\boldsymbol{W}$ [DSSST10]. For example, if $\|\cdot\|$ is the spectral norm, the projection gives a vector of singular values $\{\sigma_1', \ldots, \sigma_n'\}$, such that $\sigma_i' = \min\{\sigma_i, B\}$. For $\|\cdot\|$ being the trace norm, the projection leads to $\sigma_i \mapsto (\sigma_i - \tau)_+$, where τ is the smallest value for which $\sum_i(\sigma_i - \tau)_+ \leq B$.

When $\boldsymbol{W}_1 = \boldsymbol{0}$, one can show that the problem (4.1) is rotationally invariant for all t; in other words, for any t, for any orthogonal transformation of the instances $\boldsymbol{\mathcal{S}} \mapsto \boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}'$, the weight matrix $\boldsymbol{W}_t$ transforms as $\boldsymbol{W}_t(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}') = \boldsymbol{U}\boldsymbol{W}_t(\boldsymbol{\mathcal{S}})\boldsymbol{V}'$. The proof is by induction: The claim is trivially true for $t = 1$, as $\boldsymbol{W}_1 = \boldsymbol{0}$. Given that $\boldsymbol{W}_t(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}') = \boldsymbol{U}\boldsymbol{W}_t(\boldsymbol{\mathcal{S}})\boldsymbol{V}'$, we have:

$$\begin{aligned}
\boldsymbol{W}_{t+1}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}') &= \operatorname*{argmin}_{\boldsymbol{W}\in\mathcal{W}} \ \|\boldsymbol{W} - \boldsymbol{U}\boldsymbol{W}_t\boldsymbol{V}'\|_F^2 + \eta\partial_t(\boldsymbol{W}_t)\ \text{tr}(\boldsymbol{W}'\boldsymbol{U}\boldsymbol{x}_t\boldsymbol{y}_t'\boldsymbol{V}') \\
&= \operatorname*{argmin}_{\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}'\in\mathcal{W}} \ \|\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}' - \boldsymbol{U}\boldsymbol{W}_t\boldsymbol{V}'\|_F^2 + \eta\partial_t(\boldsymbol{W}_t)\ \text{tr}(\boldsymbol{V}\boldsymbol{W}'\boldsymbol{U}'\boldsymbol{U}\boldsymbol{x}_t\boldsymbol{y}_t'\boldsymbol{V}') \\
&= \operatorname*{argmin}_{\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}'\in\boldsymbol{U}\mathcal{W}\boldsymbol{V}'} \ \|\boldsymbol{W} - \boldsymbol{W}_t\|_F^2 + \eta\partial_t(\boldsymbol{W}_t)\ \text{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{y}_t') \\
&= \boldsymbol{U}\left(\operatorname*{argmin}_{\boldsymbol{W}\in\mathcal{W}} \ \|\boldsymbol{W} - \boldsymbol{W}_t\|_F^2 + \eta\partial_t(\boldsymbol{W}_t)\ \text{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{y}_t')\right)\boldsymbol{V}' \\
&= \boldsymbol{U}\boldsymbol{W}_{t+1}(\boldsymbol{\mathcal{S}})\boldsymbol{V}',
\end{aligned}$$

where in the first line we used the fact that $\partial_t(\boldsymbol{W}_t)$ is invariant under transformation of both $\boldsymbol{\mathcal{S}} \mapsto \boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}'$ and $\boldsymbol{W}_t \mapsto \boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{V}'$ (as it depends on $\boldsymbol{W}_t$ only through the trace $\mathrm{tr}(\boldsymbol{W}_t'\boldsymbol{x}_t\boldsymbol{y}_t')$), in the third line we used $\mathcal{W} = \boldsymbol{U}\mathcal{W}\boldsymbol{V}'$ (norm constraint), and the invariance of Frobenious norm under orthogonal transformation. This proves the rotational invariance of (4.1).

Due to the strictly convex objective function, (4.1) has a unique solution, and we conclude from Theorem 3.1 that $\boldsymbol{W}_t$ is in the span of the data, i.e. has the form $\boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'$. Thus the algorithm can be kernelized by calculating the SVD of the matrices $\boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'$ i.t.o. of the kernel matrices using Lemma 4.1. Also the output $\mathrm{tr}(\boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'\boldsymbol{x}\boldsymbol{y}') = x'\boldsymbol{X}\boldsymbol{C}\boldsymbol{Y}'\boldsymbol{y}$ only relies on the kernel matrices. For the trace norm, it can be shown using a standard analysis of GD, that given $|\partial_t(\boldsymbol{W})| \leq G$, $\mathrm{Reg}(\boldsymbol{\mathcal{S}}) \leq BG\sqrt{T}$, and is independent of the dimension of the feature space[9] [SST11], while for spectral norm $\mathrm{Reg}(\boldsymbol{\mathcal{S}}) \leq BG\sqrt{\min\{n,m\}T}$.

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 (EG) [TRW05], defined as a trade-off between minimization of the quantum relative entropy and the negative gradient of the loss:

$$\boldsymbol{W}_{t+1} = \operatorname*{argmin}_{\boldsymbol{W}\in\mathcal{W}} \ \mathrm{tr}\left(\boldsymbol{W}\left(\log \boldsymbol{W} - \log \boldsymbol{W}_t\right)\right) + \eta\partial_t(\boldsymbol{W}_t)\,\mathrm{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{x}_t'), \tag{4.2}$$

which leads to to the following *multiplicative update* [TRW05]:

$$\boldsymbol{W}_{t+1} = \frac{\exp\left(\log \boldsymbol{W}_t - \eta\partial_t(\boldsymbol{W}_t)\boldsymbol{x}_t\boldsymbol{x}_t'\right)}{Z_t}, \tag{4.3}$$

where $Z_t = \mathrm{tr}\left(\exp\left(\log \boldsymbol{W}_t - \eta\partial_t(\boldsymbol{W}_t)\boldsymbol{x}_t\boldsymbol{x}_t'\right)\right)$ is the normalization factor. When $\boldsymbol{W}_1 = \boldsymbol{I}/n$, a simple inductive argument proves rotational invariance of (4.2) for all t. Indeed, it trivially holds $\boldsymbol{W}_1(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}') = \boldsymbol{U}\boldsymbol{W}_1(\boldsymbol{\mathcal{S}})\boldsymbol{U}'$. Now, given that $\boldsymbol{W}_t(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}') = \boldsymbol{U}W_t(\boldsymbol{\mathcal{S}})\boldsymbol{U}'$, we have:

$$\begin{aligned}
\boldsymbol{W}_{t+1}(\boldsymbol{U}\boldsymbol{\mathcal{S}}\boldsymbol{U}') &= \operatorname*{argmin}_{\boldsymbol{W}\in\mathcal{W}} \ \mathrm{tr}\left(\boldsymbol{W}\left(\log \boldsymbol{W} - \boldsymbol{U}\log \boldsymbol{W}_t\boldsymbol{U}'\right)\right) + \eta\partial_t(\boldsymbol{W}_t)\mathrm{tr}(\boldsymbol{W}'\boldsymbol{U}\boldsymbol{x}_t\boldsymbol{x}_t'\boldsymbol{U}') \\
&= \operatorname*{argmin}_{\boldsymbol{U}\boldsymbol{W}\boldsymbol{U}'\in\mathcal{W}} \ \mathrm{tr}\left(\boldsymbol{U}\boldsymbol{W}\boldsymbol{U}'\boldsymbol{U}\left(\log \boldsymbol{W} - \log \boldsymbol{W}_t\right)\boldsymbol{U}'\right) + \eta\partial_t(\boldsymbol{W}_t)\mathrm{tr}(\boldsymbol{U}\boldsymbol{W}'\boldsymbol{U}'\boldsymbol{U}\boldsymbol{x}_t\boldsymbol{x}_t'\boldsymbol{U}') \\
&= \operatorname*{argmin}_{\boldsymbol{U}\boldsymbol{W}\boldsymbol{U}'\in\boldsymbol{U}\mathcal{W}\boldsymbol{U}'} \ \mathrm{tr}\left(\boldsymbol{W}\left(\log \boldsymbol{W} - \log \boldsymbol{W}_t\right)\right) + \eta\partial_t(\boldsymbol{W}_t)\mathrm{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{x}_t') \\
&= \boldsymbol{U}\left(\operatorname*{argmin}_{\boldsymbol{W}\in\mathcal{W}} \ \mathrm{tr}\left(\boldsymbol{W}\left(\log \boldsymbol{W} - \log \boldsymbol{W}_t\right)\right) + \eta\partial_t(\boldsymbol{W}_t)\mathrm{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{x}_t')\right)\boldsymbol{U}' \\
&= \boldsymbol{U}\boldsymbol{W}_{t+1}(\boldsymbol{\mathcal{S}})\boldsymbol{U}',
\end{aligned}$$

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

which proves rotational invariance of (4.2). Due to the strictly convex objective function, (4.2) has a unique solution, and we conclude from Theorem 3.2 that the algorithm can be kernelized (we note that the standard representer theorems do not cover this case). The main challenge in the update (4.3) 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 4.1 only requires to calculate the kernel matrix.

A new EG algorithm with regret bound logarithmic in the rank of the instances:

A particularly interesting case is when $\mathrm{loss}(\mathrm{tr}(\boldsymbol{W}_t'\boldsymbol{x}_t\boldsymbol{x}_t'), \ell_t) = -\mathrm{tr}(\boldsymbol{W}_t'\boldsymbol{x}_t\boldsymbol{x}_t')$. In other words, the game is the *gain game* with a linear gain function $\mathrm{tr}(\boldsymbol{W}_t\boldsymbol{x}_t\boldsymbol{x}_t')$. 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 $\sum_t \boldsymbol{x}_t\boldsymbol{x}_t'$. Hence, the offline comparator corresponds to the solution of single-component, centered Principal Component Analysis (PCA)[10] [WK08]. The learning algorithm can then be regarded as an online predictive version of PCA. In this case, the EG update (4.3) 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 4.2.

By modifying the EG analysis of [WK08, KW07], we can show shown that $\mathrm{Reg}(\boldsymbol{\mathcal{S}}) \le \sqrt{2L^* \ln n} + \ln n$, where L^* is the *approximation error*, i.e. part of the variance in the data not captured by $\boldsymbol{W}^*$, $L^* = \min_{\boldsymbol{W}\in\mathcal{W}} \left\{\sum_{t=1}^T (1 - \mathrm{tr}(\boldsymbol{W}'\boldsymbol{x}_t\boldsymbol{x}_t'))\right\}$. Unfortunately, this bound (which essentially appears in [KW07]) 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 r 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, because $r \le T$. 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 (4.3) 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 $Z_t^+ = \mathrm{tr}\left(\mathbf{exp}^+\left(\eta \sum_{i=1}^{t-1} \boldsymbol{x}_i\boldsymbol{x}_i'\right)\right)$, and $\mathbf{exp}^+(\boldsymbol{A})$ is a function that exponentiate the positive eigenvalues of $\boldsymbol{A}$ only, and leaves the zero eigenvalues unchanged.[11] 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

[10] By capping the eigenvalues to $\frac{1}{k}$ (as done in [WK08]) we can generalize this algorithm to k-component PCA where one seeks a k-dimensional subspace with maximal variance.

[11] The initial weight matrix $\boldsymbol{W}_1$ is set arbitrarily.

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

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, $\mathrm{tr}\left(\mathbf{exp}^+\left(\eta\boldsymbol{S}_{t-1}\right)\boldsymbol{x}_t\boldsymbol{x}_t'\right) = \mathrm{tr}\left(\mathbf{exp}\left(\eta\boldsymbol{S}_{t-1}\right)\boldsymbol{x}_t\boldsymbol{x}_t'\right)$ (because $\boldsymbol{x}_t$ belongs to the subspace associated with non-zero eigenvalues of $\boldsymbol{S}_{t-1}$), but $Z_t^+ \leq Z_t$ (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):

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

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:

$$Z_t = \mathrm{tr}\left(\mathbf{exp}\left(\eta\sum_{i=1}^{t-1}\boldsymbol{x}_i\boldsymbol{x}_i'\right)\right) \geq \mathrm{tr}(\boldsymbol{I}) = n.$$

□

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 $r \leq Ts$.

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. We conclude that it is of further study, which of the algorithms performs better for typical (real-life) instances of PCA problem.

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 so-

lution and satisfies a certain rotational invariance. We also proved simple representer theorems for both asymmetric and symmetric matrix instances when the solution to the underlying optimization problem is unique. We concluded with 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.

The main open problem is whether our characterization of kernelization based on rotational invariance can be extended to handle the case of non-unique solutions. A second subtle open problem is the following. A new family of so called "Forward" algorithms was developed [AW01] 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 [Vov01, For99], better regret bounds were proven for the Forward algorithm than for the standard Ridge Regression algorithm. It is easy to generalize the Representer Theorem approach to handle this case. However it is an open problem whether our characterization of kernelizability based on rotational invariance can be generalized to algorithms that may predict with linear combinations of the labeled as well as the last unlabeled instance.

Acknowledgements

This work on matrix updates has its firm roots in a joint paper with S.V.N. Vishwanathan on vector updates [WV05]. We thank S.V.N. Vishwanathan for many valuable discussions regarding applications of rotational invariance in Machine Learning.

A. Tensor product instances over Hilbert Spaces

In this part, we generalize our results from finite-dimensional Euclidean space to Hilbert spaces. Throughout this appendix we assume that Hilbert spaces are always real[12] (i.e. they are over the set of reals) and separable (i.e. they admit a countable orthonormal basis).

In the body of the paper, the instances were outer products $\boldsymbol{x}\boldsymbol{y}'$, where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^m$. In the generalization, the instances are tensor products $\boldsymbol{x} \otimes \boldsymbol{y}$, where the left instances $\boldsymbol{x}$ lie in some Hilbert space $\mathcal{X}$ and the right instances $\boldsymbol{y}$ in some Hilbert space $\mathcal{Y}$. In the symmetric case, the two Hilbert spaces coincide. Note that in the original case, the outer product matrix $\boldsymbol{x}\boldsymbol{y}'$ can be written as $\boldsymbol{x} \otimes \boldsymbol{y}$ as well.

We briefly recall the main definitions regarding Hilbert spaces. More details can be found in any general book on Hilbert spaces such as [RY08]. Let $\langle \cdot, \cdot \rangle_{\mathcal{X}}$ and $\langle \cdot, \cdot \rangle_{\mathcal{Y}}$ denote the inner products in $\mathcal{X}$ and $\mathcal{Y}$, respectively. The tensor product $\boldsymbol{x} \otimes \boldsymbol{y}$ can also be interpreted as the following linear operator from $\mathcal{Y}$ to $\mathcal{X}$ [ABEV09]:

$$\forall \widetilde{\boldsymbol{y}} \in \mathcal{Y}: \qquad (\boldsymbol{x} \otimes \boldsymbol{y})\, \widetilde{\boldsymbol{y}} := \langle \boldsymbol{y}, \widetilde{\boldsymbol{y}} \rangle_{\mathcal{Y}}\, \boldsymbol{x}. \tag{A.1}$$

[12] The results of this appendix easily generalize to complex Hilbert spaces.

The tensor product $\mathcal{X} \otimes \mathcal{Y}$ between the Hilbert spaces $\mathcal{X}$ and $\mathcal{Y}$ consists of all linear combinations of instances of the form $\boldsymbol{x} \otimes \boldsymbol{y}$.[13] The inner product between tensor product instances is defined as:

$$\langle \boldsymbol{x} \otimes \boldsymbol{y}, \widetilde{\boldsymbol{x}} \otimes \widetilde{\boldsymbol{y}} \rangle_{\mathcal{X} \otimes \mathcal{Y}} := \langle \boldsymbol{x}, \widetilde{\boldsymbol{x}} \rangle_{\mathcal{X}} \, \langle \boldsymbol{y}, \widetilde{\boldsymbol{y}} \rangle_{\mathcal{Y}}.$$

Since the elements of $\mathcal{X} \otimes \mathcal{Y}$ are linear combinations of tensor product instances, the above dot product between tensor product instances immediately extents to dot products between arbitrary instances of $\mathcal{X} \otimes \mathcal{Y}$. Using the linearity of the inner product and (A.1), respectively, we have:

$$\langle \boldsymbol{x} \otimes \boldsymbol{y}, \widetilde{\boldsymbol{x}} \otimes \widetilde{\boldsymbol{y}} \rangle_{\mathcal{X} \otimes \mathcal{Y}} = \langle \langle \boldsymbol{y}, \widetilde{\boldsymbol{y}} \rangle_{\mathcal{Y}} \, \boldsymbol{x}, \widetilde{\boldsymbol{x}} \rangle_{\mathcal{X}} = \langle (\boldsymbol{x} \otimes \boldsymbol{y}) \widetilde{\boldsymbol{y}}, \widetilde{\boldsymbol{x}} \rangle_{\mathcal{X}}. \tag{A.2}$$

For any linear operator $\boldsymbol{T} : \mathcal{X} \mapsto \mathcal{Y}$, $\boldsymbol{T}\boldsymbol{x}$ is shorthand for $\boldsymbol{T}(\boldsymbol{x})$. Also the linear operator $\boldsymbol{T}^* : \mathcal{Y} \mapsto \mathcal{X}$ is the *adjoint* operator of $\boldsymbol{T}$ iff $\langle \boldsymbol{T}\boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{Y}} = \langle \boldsymbol{x}, \boldsymbol{T}^*\boldsymbol{y} \rangle_{\mathcal{X}}$ for every $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$. If $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y} = \mathbb{R}^m$, then a linear operator $\boldsymbol{T} : \mathcal{X} \mapsto \mathcal{Y}$ is a matrix in $\mathbb{R}^{m \times n}$ and $\boldsymbol{T}^*$ is $\boldsymbol{T}'$. By (A.1) and (A.2), we can prove that

$$(\boldsymbol{x}_i \otimes \boldsymbol{y}_j)^* = \boldsymbol{y}_j \otimes \boldsymbol{x}_i. \tag{A.3}$$

For any linear operators $\boldsymbol{T} : \mathcal{X} \mapsto \mathcal{Y}$ and $\boldsymbol{R} : \mathcal{Y} \mapsto \mathcal{Z}$, $\boldsymbol{R}\boldsymbol{T}$ denotes the linear operator $\mathcal{X} \mapsto \mathcal{Z}$ which is the functional composition of $\boldsymbol{R}$ and $\boldsymbol{T}$. Note that when $\mathcal{X} = \mathbb{R}^n$, $\mathcal{Y} = \mathbb{R}^m$ and $\mathcal{Z} = \mathbb{R}^k$, the composition $\boldsymbol{R}\boldsymbol{T}$ is just the product of two matrices $\boldsymbol{R} \in \mathbb{R}^{k \times m}$ and $\boldsymbol{T} \in \mathbb{R}^{m \times n}$. A linear operator $\boldsymbol{U} : \mathcal{X} \mapsto \mathcal{X}$ is *unitary* iff $\boldsymbol{U}\boldsymbol{U}^* = \boldsymbol{U}^*\boldsymbol{U} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity operator on $\mathcal{X}$ (namely $\boldsymbol{I}\boldsymbol{x} = \boldsymbol{x}$ for very $\boldsymbol{x} \in \mathcal{X}$).

A.1. Asymmetric tensor product instances

We assume the instances are tensor products $\boldsymbol{x} \otimes \boldsymbol{y}$, where $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$ for some Hilbert spaces $\mathcal{X}$ and $\mathcal{Y}$. A learning algorithm $\mathcal{A}$ is any mapping from example sequence $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{x}_t \otimes \boldsymbol{y}_t, \ell_t)\}_{t=1}^T$, followed by a next instance $\boldsymbol{x} \otimes \boldsymbol{y}$ to some fixed output range. Let $\boldsymbol{K}_X \in \mathbb{R}^{T \times T}$ be the kernel matrix for left instances satisfying $(\boldsymbol{K}_X)_{i,j} = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle_{\mathcal{X}}$ and let $\boldsymbol{K}_Y \in \mathbb{R}^{T \times T}$ be the kernel matrix for right instances satisfying $(\boldsymbol{K}_Y)_{i,j} = \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle_{\mathcal{Y}}$. We also define the augmented kernel matrices $\widehat{\boldsymbol{K}}_X \in \mathbb{R}^{(T+1) \times (T+1)}$ and $\widehat{\boldsymbol{K}}_Y \in \mathbb{R}^{(T+1) \times (T+1)}$, by including the unlabeled instance $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively.

Definition A.1. *An algorithm $\mathcal{A}$ for asymmetric tensor product instances is* kernelizable*, if for any two input sequences $\boldsymbol{\mathcal{S}}, \boldsymbol{x} \otimes \boldsymbol{y}$ and $\widetilde{\boldsymbol{\mathcal{S}}}, \widetilde{\boldsymbol{x}} \otimes \widetilde{\boldsymbol{y}}$ with the same labels and the same augmented kernel matrices, algorithm $\mathcal{A}$ maps to the same output, i.e. $\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{x} \otimes \boldsymbol{y}) = \mathcal{A}(\widetilde{\boldsymbol{\mathcal{S}}}, \widetilde{\boldsymbol{x}} \otimes \widetilde{\boldsymbol{y}})$.*

Our next step is to define a notion of transformations for the general Hilbert space setting. For any linear unitary operators $\boldsymbol{U}$ on $\mathcal{X}$ and $\boldsymbol{V}$ on $\mathcal{Y}$, respectively, we define a linear operator $\boldsymbol{U} \otimes \boldsymbol{V}$ on $\mathcal{X} \otimes \mathcal{Y}$ as follows:

$$(\boldsymbol{U} \otimes \boldsymbol{V})(\boldsymbol{x} \otimes \boldsymbol{y}) := \boldsymbol{U}\boldsymbol{x} \otimes \boldsymbol{V}\boldsymbol{y}.$$

[13] In general, not all elements of $\mathcal{X} \otimes \mathcal{Y}$ have the form $\boldsymbol{x} \otimes \boldsymbol{y}$.

Now $(\boldsymbol{U} \otimes \boldsymbol{V})^* = \boldsymbol{U}^* \otimes \boldsymbol{V}^*$ because for all $\boldsymbol{x},\ \widetilde{\boldsymbol{x}} \in \mathcal{X}$ and $\boldsymbol{y},\ \widetilde{\boldsymbol{y}} \in \mathcal{Y}$,

$$\begin{aligned}\langle (\boldsymbol{U} \otimes \boldsymbol{V})(\boldsymbol{x} \otimes \boldsymbol{y}), \widetilde{\boldsymbol{x}} \otimes \widetilde{\boldsymbol{y}} \rangle_{\mathcal{X}\otimes\mathcal{Y}} &= \langle (\boldsymbol{U}\boldsymbol{x} \otimes \boldsymbol{V}\boldsymbol{y}), \widetilde{\boldsymbol{x}} \otimes \widetilde{\boldsymbol{y}} \rangle_{\mathcal{X}\otimes\mathcal{Y}} \\ &= \langle \boldsymbol{U}\boldsymbol{x}, \widetilde{\boldsymbol{x}} \rangle_{\mathcal{X}} \langle \boldsymbol{V}\boldsymbol{y}, \widetilde{\boldsymbol{y}} \rangle_{\mathcal{Y}} = \langle \boldsymbol{x}, \boldsymbol{U}^*\widetilde{\boldsymbol{x}} \rangle_{\mathcal{X}} \langle \boldsymbol{y}, \boldsymbol{V}^*\widetilde{\boldsymbol{y}} \rangle_{\mathcal{Y}} \\ &= \langle \boldsymbol{x} \otimes \boldsymbol{y}, \boldsymbol{U}^*\widetilde{\boldsymbol{x}} \otimes \boldsymbol{V}^*\widetilde{\boldsymbol{y}} \rangle_{\mathcal{X}\otimes\mathcal{Y}} = \langle \boldsymbol{x} \otimes \boldsymbol{y}, (\boldsymbol{U}^* \otimes \boldsymbol{V}^*)(\widetilde{\boldsymbol{x}} \otimes \widetilde{\boldsymbol{y}}) \rangle_{\mathcal{X}\otimes\mathcal{Y}}.\end{aligned}$$

Since for all $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$,

$$(\boldsymbol{U} \otimes \boldsymbol{V})(\boldsymbol{U}^* \otimes \boldsymbol{V}^*)(\boldsymbol{x} \otimes \boldsymbol{y}) = (\boldsymbol{U} \otimes \boldsymbol{V})(\boldsymbol{U}^*\boldsymbol{x} \otimes \boldsymbol{V}^*\boldsymbol{y}) = \boldsymbol{U}\boldsymbol{U}^*\boldsymbol{x} \otimes \boldsymbol{V}\boldsymbol{V}^*\boldsymbol{y} = \boldsymbol{x} \otimes \boldsymbol{y},$$

it follows that the linear operator $\boldsymbol{U} \otimes \boldsymbol{V}$ is a unitary operators on $\mathcal{X} \otimes \mathcal{Y}$ whenever $\boldsymbol{U}, \boldsymbol{V}$ are unitary operators of $\mathcal{X}, \mathcal{Y}$, respectively. However typically not all unitary operators on $\mathcal{X} \otimes \mathcal{Y}$ have this form. Note that in the matrix case, $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices in $\mathbb{R}^{n\times n}$ and $\mathbb{R}^{m\times m}$, respectively, and the matrix $\boldsymbol{U} \otimes \boldsymbol{V}$ is orthogonal in $\mathbb{R}^{n\times n} \otimes \mathbb{R}^{m\times m}$.

We now define the *unitary transformations* of an example sequence $\boldsymbol{\mathcal{S}}$ w.r.t. the linear unitary operator $\boldsymbol{U} \otimes \boldsymbol{V}$ as follows:

$$(\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}} := \{(\boldsymbol{U}\boldsymbol{x}_t \otimes \boldsymbol{V}\boldsymbol{y}_t, \ell_t)\}_{t=1}^T.$$

It is clear that

$$\langle \boldsymbol{U}\boldsymbol{x}_i, \boldsymbol{U}\boldsymbol{x}_j \rangle_{\mathcal{X}} = \langle \boldsymbol{x}_i, \boldsymbol{U}^*\boldsymbol{U}\boldsymbol{x}_j \rangle_{\mathcal{X}} = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle_{\mathcal{X}} \quad \text{and} \quad \langle \boldsymbol{V}\boldsymbol{y}_i, \boldsymbol{V}\boldsymbol{y}_j \rangle_{\mathcal{Y}} = \langle \boldsymbol{y}_i, \boldsymbol{V}^*\boldsymbol{V}\boldsymbol{y}_j \rangle_{\mathcal{Y}} = \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle_{\mathcal{Y}}.$$

Therefore, as in Lemma 2.2, two sequences of examples $\boldsymbol{\mathcal{S}}$ and $\widetilde{\boldsymbol{\mathcal{S}}}$ are unitary transformations of each other iff the kernel matrices associated with the left instances of both sequences and the right instances of both sequences are the same, i.e. $\boldsymbol{K}_X = \widetilde{\boldsymbol{K}}_X$ and $\boldsymbol{K}_Y = \widetilde{\boldsymbol{K}}_Y$. Consequently, we get the following alternate definition of kernelizability:

Corollary A.2. *An algorithm $\mathcal{A}$ for asymmetric tensor product instances is kernelizable iff for all sequences $\boldsymbol{\mathcal{S}}$, next instances $\boldsymbol{x} \otimes \boldsymbol{y}$, linear unitary operators $\boldsymbol{U}$ and $\boldsymbol{V}$,*

$$\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{x} \otimes \boldsymbol{y}) = \mathcal{A}((\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}}, \boldsymbol{U}\boldsymbol{x} \otimes \boldsymbol{V}\boldsymbol{y}).$$

The generalization of the definition of linearity of an algorithm is straightforward:

Definition A.3. *An algorithm $\mathcal{A}$ for asymmetric tensor product instances is* linear*, if upon receiving input $\boldsymbol{\mathcal{S}}, \boldsymbol{x} \otimes \boldsymbol{y}$, the algorithm first computes a tensor $\boldsymbol{W} \in \mathcal{X} \otimes \mathcal{Y}$ (or a linear operator $\boldsymbol{W} : \mathcal{Y} \mapsto \mathcal{X}$) from the input sequence $\boldsymbol{\mathcal{S}}$, and then outputs the inner product $\langle \boldsymbol{W}, \boldsymbol{x} \otimes \boldsymbol{y} \rangle_{\mathcal{X}\otimes\mathcal{Y}} = \langle \boldsymbol{W}\boldsymbol{y}, \boldsymbol{x} \rangle_{\mathcal{X}}$.*

We now give a characterization of the kernerlizability for such linear algorithms:

Theorem A.4. *A linear algorithm $\mathcal{A}$ for asymmetric outer product instances is kernelizable if and only if for every input sequence $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{x}_t \otimes \boldsymbol{y}_t, \ell_t)\}_{t=1}^T$ the linear operator produced by $\mathcal{A}$ can be written as $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i=1}^T \sum_{j=1}^T \boldsymbol{C}_{i,j} \boldsymbol{x}_i \otimes \boldsymbol{y}_j$ and the coefficient matrix $\boldsymbol{C} \in \mathbb{R}^{T \times T}$ depends on $\boldsymbol{\mathcal{S}}$ only via the kernel matrices $\boldsymbol{K}_X$ and $\boldsymbol{K}_Y$.*

Proof. Let $\boldsymbol{W}(\boldsymbol{\mathcal{S}})$ denote the operator produced by algorithm $\mathcal{A}$ from the sequence $\boldsymbol{\mathcal{S}}$. Since $\mathcal{A}$ outputs the dot product, $\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{x} \otimes \boldsymbol{y}) = \langle \boldsymbol{W}(\boldsymbol{\mathcal{S}}), \boldsymbol{x} \otimes \boldsymbol{y} \rangle_{\mathcal{X} \otimes \mathcal{Y}}$, it follows from Corollary A.2 that $\mathcal{A}$ is kernelizable iff:

$$\langle \boldsymbol{W}(\boldsymbol{\mathcal{S}}), \boldsymbol{x} \otimes \boldsymbol{y} \rangle_{\mathcal{X} \otimes \mathcal{Y}} = \langle \boldsymbol{W}((\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}}), \boldsymbol{U}\boldsymbol{x} \otimes \boldsymbol{V}\boldsymbol{y} \rangle_{\mathcal{X} \otimes \mathcal{Y}} \tag{A.4}$$

for all sequences $\boldsymbol{\mathcal{S}}$ and unitary operator $\boldsymbol{U}$ and $\boldsymbol{V}$, and instances $\boldsymbol{x} \otimes \boldsymbol{y}$, for $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$.

The proof of the "if" part is easy: Since the coefficient matrix $\boldsymbol{C}$ depends on $\boldsymbol{\mathcal{S}}$ only via the kernel matrices $\boldsymbol{K}_X$ and $\boldsymbol{K}_Y$, this matrix $\boldsymbol{C}$ is invariant under unitary transformation $\boldsymbol{\mathcal{S}} \mapsto (\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}}$ (which leaves the kernel matrices unchanged), and thus

$$\boldsymbol{W}((\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}}) = \sum_{i=1}^T \sum_{j=1}^T \boldsymbol{C}_{ij} (\boldsymbol{U}\boldsymbol{x}_i) \otimes (\boldsymbol{V}\boldsymbol{y}_j).$$

for the same matrix $\boldsymbol{C}$. This implies (A.4) and kernelizability:

$$\begin{aligned}
\langle \boldsymbol{W}((\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}}), (\boldsymbol{U}\boldsymbol{x}) \otimes (\boldsymbol{V}\boldsymbol{y}) \rangle_{\mathcal{X} \otimes \mathcal{Y}} &= \left\langle \sum_{i,j} \boldsymbol{C}_{i,j} (\boldsymbol{U}\boldsymbol{x}_i) \otimes (\boldsymbol{V}\boldsymbol{y}_j), (\boldsymbol{U}\boldsymbol{x}) \otimes (\boldsymbol{V}\boldsymbol{y}) \right\rangle_{\mathcal{X} \otimes \mathcal{Y}} \\
&= \sum_{i,j} \boldsymbol{C}_{i,j} \langle (\boldsymbol{U}\boldsymbol{x}_i) \otimes (\boldsymbol{V}\boldsymbol{y}_j), (\boldsymbol{U}\boldsymbol{x}) \otimes (\boldsymbol{V}\boldsymbol{y}) \rangle_{\mathcal{X} \otimes \mathcal{Y}} \\
&= \sum_{i,j} \boldsymbol{C}_{i,j} \langle \boldsymbol{U}\boldsymbol{x}_i, \boldsymbol{U}\boldsymbol{x} \rangle_{\mathcal{X}} \langle \boldsymbol{V}\boldsymbol{y}_j, \boldsymbol{V}\boldsymbol{y} \rangle_{\mathcal{Y}} \\
&= \sum_{i,j} \boldsymbol{C}_{i,j} \langle \boldsymbol{x}_i, \boldsymbol{U}^* \boldsymbol{U}\boldsymbol{x} \rangle_{\mathcal{X}} \langle \boldsymbol{y}_j, \boldsymbol{V}^* \boldsymbol{V}\boldsymbol{y} \rangle_{\mathcal{Y}} \\
&= \sum_{i,j} \boldsymbol{C}_{i,j} \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle_{\mathcal{X}} \langle \boldsymbol{y}_j, \boldsymbol{y} \rangle_{\mathcal{Y}} \;=\; \sum_{i,j} \boldsymbol{C}_{i,j} \langle \boldsymbol{x}_i \otimes \boldsymbol{y}_j, \boldsymbol{x} \otimes \boldsymbol{y} \rangle_{\mathcal{X} \otimes \mathcal{Y}} \\
&= \left\langle \sum_{i,j} \boldsymbol{C}_{i,j} \boldsymbol{x}_i \otimes \boldsymbol{y}_j, \boldsymbol{x} \otimes \boldsymbol{y} \right\rangle_{\mathcal{X} \otimes \mathcal{Y}} \;=\; \langle \boldsymbol{W}(\boldsymbol{\mathcal{S}}), \boldsymbol{x} \otimes \boldsymbol{y} \rangle_{\mathcal{X} \otimes \mathcal{Y}}.
\end{aligned}$$

The proof of the "only if" part is divided into two parts. In Part 1, we first show that (A.4) implies that for any $\boldsymbol{\mathcal{S}}$, $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i=1}^T \sum_{j=1}^T \boldsymbol{C}_{ij} \boldsymbol{x}_i \otimes \boldsymbol{y}_j$ for some coefficient matrix $\boldsymbol{C}$. In Part 2, we show that (A.4) implies that for any $\boldsymbol{\mathcal{S}}$ and unitary operators $\boldsymbol{U}$ and $\boldsymbol{V}$, we also have $\boldsymbol{W}((\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}}) = \sum_{i=1}^T \sum_{j=1}^T \boldsymbol{C}_{ij} (\boldsymbol{U}\boldsymbol{x}_i) \otimes (\boldsymbol{V}\boldsymbol{y}_j)$ with the same $\boldsymbol{C}$. This means that $\boldsymbol{C}$ is invariant under unitary transformations of the example sequence $\boldsymbol{\mathcal{S}}$, and thus by a generalization of Lemma 2.2, this is equivalent to stating that $\boldsymbol{C}$ depends on $\boldsymbol{\mathcal{S}}$ only via the kernel matrices $\boldsymbol{K}_X$ and $\boldsymbol{K}_Y$.

Proof of Part 1: Let $\{\widehat{\boldsymbol{x}}_p\}_{p=1}^{r_1}$ be an orthonormal basis for $Span\left(\{\boldsymbol{x}_t\}_{t=1}^T\right)$ and $\{\widehat{\boldsymbol{y}}_q\}_{q=1}^{r_2}$ 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. Since

$\widehat{\boldsymbol{x}}_p \in Span(\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\})$ and $\widehat{\boldsymbol{y}}_q \in Span(\{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_T\})$, there exist matrices $\boldsymbol{P} \in \mathbb{R}^{T \times r_1}$ and $\boldsymbol{Q} \in \mathbb{R}^{T \times r_2}$ such that

$$\widehat{\boldsymbol{x}}_p = \sum_{i=1}^{T} \boldsymbol{P}_{i,p} \boldsymbol{x}_i \quad \text{and} \quad \widehat{\boldsymbol{y}}_q = \sum_{j=1}^{T} \boldsymbol{Q}_{j,q} \boldsymbol{y}_j. \tag{A.5}$$

Since $\mathcal{X}$ and $\mathcal{Y}$ are separable, they have countable orthonormal bases. Thus, we can complete $\{\widehat{\boldsymbol{x}}_p\}_{p=1}^{r_1}$ and $\{\widehat{\boldsymbol{y}}_q\}_{q=1}^{r_2}$ to orthonormal bases for $\mathcal{X}$ and $\mathcal{Y}$, respectively, and denote those bases as $\{\widehat{\boldsymbol{x}}_i\}_{i \in \mathbb{N}}$ and $\{\widehat{\boldsymbol{y}}_j\}_{j \in \mathbb{N}}$. Hence $\{\widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{y}}_j | i, j \in \mathbb{N}\}$ is an orthonormal basis for $\mathcal{X} \otimes \mathcal{Y}$, and we can decompose $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) \in \mathcal{X} \otimes \mathcal{Y}$ as

$$\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i,j \in \mathbb{N}} \hat{c}_{i,j} \widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{y}}_j.$$

Choose any index $p > r_1$, and any index $q \geq 1$, and we now show that $\hat{c}_{p,q} = 0$ (the case $q > r_2$ and and $p \geq 1$ is proven similarly). We use our notion of unitary invariance (A.4). We choose the new instance $\boldsymbol{x} \otimes \boldsymbol{y}$ with $\boldsymbol{x} = \widehat{\boldsymbol{x}}_p$ and $\boldsymbol{y} = \widehat{\boldsymbol{y}}_q$. Furthermore, we choose $\boldsymbol{U}$ as the *unitary Hauseholder operator* $\boldsymbol{H} : \mathcal{X} \mapsto \mathcal{X}$, defined as: $\boldsymbol{H}\boldsymbol{x} := \boldsymbol{x} - 2\langle \boldsymbol{x}, \widehat{\boldsymbol{x}}_p \rangle_{\mathcal{X}} \widehat{\boldsymbol{x}}_p$, and $\boldsymbol{V} = \boldsymbol{I}$, the identity operator on $\mathcal{Y}$.

It holds that $\langle \widehat{\boldsymbol{x}}_p, \boldsymbol{x}_t \rangle_{\mathcal{X}} = 0$ for any $t = 1, \ldots, T$, because $p > r_1$ and $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^{r_1}$ is the orthonormal basis for $Span\left(\{\boldsymbol{x}_t\}_{t=1}^{T}\right)$. This implies that $\boldsymbol{U}\boldsymbol{x}_t = \boldsymbol{H}\boldsymbol{x}_t = \boldsymbol{x}_t - 2\langle \boldsymbol{x}_t, \widehat{\boldsymbol{x}}_p \rangle_{\mathcal{X}} \widehat{\boldsymbol{x}}_p = \boldsymbol{x}_t$, i.e. the left instances are not affected by $\boldsymbol{U}$. Also, $\boldsymbol{V}\boldsymbol{y}_t = \boldsymbol{I}\boldsymbol{y}_t = \boldsymbol{y}_t$, i.e. the right instances are not affected by $\boldsymbol{V}$. It thus follows that the transformed samples in $\widetilde{\boldsymbol{\mathcal{S}}}$ are same as the original samples in $\boldsymbol{\mathcal{S}}$, and therefore $\boldsymbol{W}((\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}}) = \boldsymbol{W}(\boldsymbol{\mathcal{S}})$. Thus, the l.h.s. of Equation (A.4) becomes:

$$\langle \boldsymbol{W}(\boldsymbol{\mathcal{S}}), \boldsymbol{x} \otimes \boldsymbol{y} \rangle_{\mathcal{X} \otimes \mathcal{Y}} = \left\langle \sum_{i,j} \hat{\boldsymbol{c}}_{i,j} \widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{y}}_j, \widehat{\boldsymbol{x}}_p \otimes \widehat{\boldsymbol{y}}_q \right\rangle_{\mathcal{X} \otimes \mathcal{Y}} = \hat{c}_{p,q}.$$

However, we have $\boldsymbol{U}\widehat{\boldsymbol{x}}_p = \boldsymbol{H}\widehat{\boldsymbol{x}}_p = \widehat{\boldsymbol{x}}_p - 2\langle \widehat{\boldsymbol{x}}_p, \widehat{\boldsymbol{x}}_p \rangle_{\mathcal{X}} \widehat{\boldsymbol{x}}_p = -\widehat{\boldsymbol{x}}_p$ and therefore the r.h.s. of Equation (A.4) has the opposite sign:

$$\langle \boldsymbol{W}((\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}}), \boldsymbol{U}\boldsymbol{x} \otimes \boldsymbol{V}\boldsymbol{y} \rangle_{\mathcal{X} \otimes \mathcal{Y}} = \langle \boldsymbol{W}(\boldsymbol{\mathcal{S}}), -\widehat{\boldsymbol{x}}_p \otimes \widehat{\boldsymbol{y}}_p \rangle_{\mathcal{X} \otimes \mathcal{Y}} = -\hat{c}_{p,q}.$$

We conclude that the unitary invariance (A.4) implies $\hat{c}_{p,q} = 0$ if $p > r_1$ (and similarly $\hat{c}_{p,q} = 0$ if $q > r_2$). It thus follows that

$$\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{p=1}^{r_1} \sum_{q=1}^{r_2} \hat{c}_{p,q} \widehat{\boldsymbol{x}}_p \otimes \widehat{\boldsymbol{y}}_q.$$

and we now prove Part 1 by plugging (A.5) into the above:

$$\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{p,q} \hat{c}_{p,q} \sum_i \sum_j \boldsymbol{P}_{i,p} \boldsymbol{Q}_{j,q} \boldsymbol{x}_i \otimes \boldsymbol{y}_j = \sum_{i,j} \underbrace{\sum_{p,q} \boldsymbol{P}_{i,p} \hat{c}_{p,q} \boldsymbol{Q}_{j,q}}_{\boldsymbol{C}_{i,j}} \boldsymbol{x}_i \otimes \boldsymbol{y}_j.$$

Proof of Part 2: By Part 1, $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} \hat{c}_{i,j} \widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{y}}_j$. By applying Part 1 to the sequence $(\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}}$ we get $\boldsymbol{W}((\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}}) = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} \hat{d}_{i,j} (\boldsymbol{U}\widehat{\boldsymbol{x}}_i) \otimes (\boldsymbol{V}\widehat{\boldsymbol{y}}_j)$ for some coefficients $\hat{d}_{i,j}$. By (A.4),

$$\hat{c}_{p,q} = \langle \boldsymbol{W}(\boldsymbol{\mathcal{S}}), \hat{\boldsymbol{x}}_p \otimes \hat{\boldsymbol{y}}_q \rangle_{\mathcal{X} \otimes \mathcal{Y}} = \langle \boldsymbol{W}((\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}}), (\boldsymbol{U}\hat{\boldsymbol{x}}_p) \otimes (\boldsymbol{V}\hat{\boldsymbol{y}}_q) \rangle_{\mathcal{X} \otimes \mathcal{Y}} = \hat{d}_{p,q}.$$

for any $p \in \{1, \dots, r_1\}$ and any $q \in \{1, \dots, r_2\}$. If $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^{r_1}$ is an orthonormal basis for $Span\left(\{\boldsymbol{x}_t\}_{t=1}^{T}\right)$, then $\{\boldsymbol{U}\widehat{\boldsymbol{x}}_i\}_{i=1}^{r_1}$ is an orthonormal basis for $Span\left(\{\boldsymbol{U}\boldsymbol{x}_t\}_{t=1}^{T}\right)$ (and similarly for $Span\left(\{\boldsymbol{y}_t\}_{t=1}^{T}\right)$ and $Span\left(\{\boldsymbol{V}\boldsymbol{y}_t\}_{t=1}^{T}\right)$). Similarly (A.5), we have

$$\boldsymbol{U}\widehat{\boldsymbol{x}}_p = \sum_{i=1}^{T} \boldsymbol{P}_{i,p}(\boldsymbol{U}\boldsymbol{x}_i) \quad \text{and} \quad \boldsymbol{V}\widehat{\boldsymbol{y}}_q = \sum_{j=1}^{T} \boldsymbol{Q}_{j,q}(\boldsymbol{V}\boldsymbol{y}_j). \tag{A.6}$$

Then

$$\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i,j} \underbrace{\sum_{p,q} \boldsymbol{P}_{i,p}\hat{c}_{p,q}\boldsymbol{Q}_{j,q}}_{\boldsymbol{C}_{i,j}} \boldsymbol{x}_i \otimes \boldsymbol{y}_j$$

follows immediately by plugging (A.6) into the equality

$$\boldsymbol{W}((\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}}) = \sum_{i=1}^{r_1}\sum_{j=1}^{r_2} \hat{d}_{i,j}(\boldsymbol{U}\widehat{\boldsymbol{x}}_i) \otimes (\boldsymbol{V}\widehat{\boldsymbol{y}}_j) = \sum_{i=1}^{r_1}\sum_{j=1}^{r_2} \hat{c}_{i,j}(\boldsymbol{U}\widehat{\boldsymbol{x}}_i) \otimes (\boldsymbol{V}\widehat{\boldsymbol{y}}_j).$$

This completes the proof. □

A.2. Symmetric tensor product instances

We assume the instances are tensor products $\boldsymbol{x} \otimes \boldsymbol{x}$, where $\boldsymbol{x} \in \mathcal{X}$ for some Hilbert spaces $\mathcal{X}$. A learning algorithm $\mathcal{A}$ is any mapping from example sequence $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{x}_t \otimes \boldsymbol{x}_t, \ell_t)\}_{t=1}^{T}$, followed by a next instance $\boldsymbol{x} \otimes \boldsymbol{x}$ to some fixed output range. Let $\boldsymbol{K} \in \mathbb{R}^{T \times T}$ be the kernel matrix for instances satisfying $(\boldsymbol{K})_{i,j} = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle_{\mathcal{X}}$. We also define the augmented kernel matrix $\widehat{\boldsymbol{K}} \in \mathbb{R}^{(T+1)\times(T+1)}$ by including the unlabeled instance $\boldsymbol{x}$.

Definition A.5. *An algorithm $\mathcal{A}$ for symmetric tensor product instances is* kernelizable*, if for any two input sequences $\boldsymbol{\mathcal{S}}, \boldsymbol{x} \otimes \boldsymbol{x}$ and $\widetilde{\boldsymbol{\mathcal{S}}}, \widetilde{\boldsymbol{x}} \otimes \widetilde{\boldsymbol{x}}$ with the same labels and the same augmented kernel matrix, algorithm $\mathcal{A}$ maps to the same output, i.e. $\mathcal{A}(\boldsymbol{\mathcal{S}}, \boldsymbol{x} \otimes \boldsymbol{x}) = \mathcal{A}(\widetilde{\boldsymbol{\mathcal{S}}}, \widetilde{\boldsymbol{x}} \otimes \widetilde{\boldsymbol{x}})$.*

For any linear unitary operator $\boldsymbol{U} : \mathcal{X} \mapsto \mathcal{X}$, we let $(\boldsymbol{U} \otimes \boldsymbol{U})\,\boldsymbol{\mathcal{S}} := \{(\boldsymbol{U}\boldsymbol{x}_t \otimes \boldsymbol{U}\boldsymbol{x}_t, \ell_t)\}_{t=1}^{T}$ denote the unitary transformation of sequence $\boldsymbol{\mathcal{S}}$. Similarly as in Lemma 2.2, we can prove that two sequences of examples $\boldsymbol{\mathcal{S}}$ and $\widetilde{\boldsymbol{\mathcal{S}}}$ are unitary transformations of each other iff the kernel matrices associated with both sequences are the same, i.e. $\boldsymbol{K} = \widetilde{\boldsymbol{K}}$. As a consequence of this fact, we get the following alternate definition of kernelizability:

Corollary A.6. *An algorithm $\mathcal{A}$ for symmetric tensor product instances is kernelizable iff for all sequences $\boldsymbol{\mathcal{S}}$, next instances $\boldsymbol{x} \otimes \boldsymbol{x}$ and linear unitary operator $\boldsymbol{U}$,*

$$\mathcal{A}\left(\boldsymbol{\mathcal{S}}, \boldsymbol{x} \otimes \boldsymbol{x}\right) = \mathcal{A}\left((\boldsymbol{U} \otimes \boldsymbol{V})\boldsymbol{\mathcal{S}}, \boldsymbol{U}\boldsymbol{x} \otimes \boldsymbol{U}\boldsymbol{x}\right).$$

The generalization of the linearity of algorithm is straightforward:

Definition A.7. *An algorithm $\mathcal{A}$ for symmetric tensor product instances is* linear *if $\mathcal{A}$, upon input $\boldsymbol{\mathcal{S}}, \boldsymbol{x}\otimes\boldsymbol{x}$, first computes a symmetric tensor $\boldsymbol{W} \in \mathcal{X} \otimes \mathcal{X}$ (or a linear* self-adjoint[14] *operator $\boldsymbol{W} : \mathcal{X} \mapsto \mathcal{X}$) from the input sequence $\boldsymbol{\mathcal{S}}$, and then outputs the inner product $\langle \boldsymbol{W}, \boldsymbol{x} \otimes \boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}}$.*[15]

We now give a characterization of the kernerlizability for linear algorithms:

Theorem A.8. *A linear algorithm $\mathcal{A}$ is kernelizable if and only if for every input sequence $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{x}_t \otimes \boldsymbol{x}_t, \ell_t)\}_{t=1}^T$ the linear operator produced by $\mathcal{A}$ can be written as $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i=1}^T \sum_{j=1}^T \boldsymbol{C}_{i,j}\boldsymbol{x}_i \otimes \boldsymbol{x}_j + c\boldsymbol{I}$, where $\boldsymbol{C} \in \mathbb{R}^{T\times T}$ is a symmetric coefficient matrix, c is a real number, $\boldsymbol{I}$ is the identity operator on $\mathcal{X}$, and $\boldsymbol{C}$ and c depend on $\boldsymbol{\mathcal{S}}$ only via the kernel matrix $\boldsymbol{K}$.*

Proof. By (A.6), $\mathcal{A}$ is kernelizable if and only if

$$\langle \boldsymbol{W}(\boldsymbol{\mathcal{S}}), \boldsymbol{x} \otimes \boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}} = \langle \boldsymbol{W}((\boldsymbol{U} \otimes \boldsymbol{U})\boldsymbol{\mathcal{S}}), \boldsymbol{U}\boldsymbol{x} \otimes \boldsymbol{U}\boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}}, \tag{A.7}$$

for all $\boldsymbol{\mathcal{S}}, \boldsymbol{U}, \boldsymbol{x}\otimes\boldsymbol{x}$. The proof of the "if" part is easy and very similar to that of Theorem A.4: Since $\boldsymbol{C}$ and c depend on $\boldsymbol{\mathcal{S}}$ only via the kernel matrix, $\boldsymbol{C}$ and c are invariant under unitary transformation $\boldsymbol{\mathcal{S}} \mapsto (\boldsymbol{U}\otimes\boldsymbol{U})\boldsymbol{\mathcal{S}}$, and thus $\boldsymbol{W}((\boldsymbol{U} \otimes \boldsymbol{U})\boldsymbol{\mathcal{S}}) = \sum_{i=1}^T \sum_{j=1}^T \boldsymbol{C}_{ij}(\boldsymbol{U}\boldsymbol{x}_i) \otimes (\boldsymbol{U}\boldsymbol{x}_j) + c\boldsymbol{I}$. This implies (A.7) and kernelizability:

$$\begin{aligned}
\langle \boldsymbol{W}((\boldsymbol{U} \otimes \boldsymbol{U})\boldsymbol{\mathcal{S}}), \boldsymbol{U}\boldsymbol{x} \otimes \boldsymbol{U}\boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}} &= \left\langle \sum_{i,j} \boldsymbol{C}_{i,j}(\boldsymbol{U}\boldsymbol{x}_i) \otimes (\boldsymbol{U}\boldsymbol{x}_j) + c\boldsymbol{I}, (\boldsymbol{U}\boldsymbol{x}) \otimes (\boldsymbol{U}\boldsymbol{x}) \right\rangle_{\mathcal{X}\otimes\mathcal{X}} \\
&= \sum_{i,j} \boldsymbol{C}_{i,j}\langle \boldsymbol{U}\boldsymbol{x}_i \otimes \boldsymbol{U}\boldsymbol{x}_j, \boldsymbol{U}\boldsymbol{x} \otimes \boldsymbol{U}\boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}} + c\langle \boldsymbol{I}, \boldsymbol{U}\boldsymbol{x} \otimes \boldsymbol{U}\boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}} \\
&= \sum_{i,j} \boldsymbol{C}_{i,j}\langle \boldsymbol{U}\boldsymbol{x}_i, \boldsymbol{U}\boldsymbol{x}\rangle_{\mathcal{X}}\langle \boldsymbol{U}\boldsymbol{x}_j, \boldsymbol{U}\boldsymbol{x}\rangle_{\mathcal{X}} + c\langle \boldsymbol{I}(\boldsymbol{U}\boldsymbol{x}), \boldsymbol{U}\boldsymbol{x}\rangle_{\mathcal{X}} \\
&= \sum_{i,j} \boldsymbol{C}_{i,j}\langle \boldsymbol{x}_i, \boldsymbol{U}^*\boldsymbol{U}\boldsymbol{x}\rangle_{\mathcal{X}}\langle \boldsymbol{x}_j, \boldsymbol{U}^*\boldsymbol{U}\boldsymbol{x}\rangle_{\mathcal{X}} + c\langle \boldsymbol{U}\boldsymbol{x}, \boldsymbol{U}\boldsymbol{x}\rangle_{\mathcal{X}} \\
&= \sum_{i,j} \boldsymbol{C}_{i,j}\langle \boldsymbol{x}_i, \boldsymbol{x}\rangle_{\mathcal{X}}\langle \boldsymbol{x}_j, \boldsymbol{x}\rangle_{\mathcal{X}} + c\langle \boldsymbol{I}\boldsymbol{x}, \boldsymbol{x}\rangle_{\mathcal{X}} \\
&= \sum_{i,j} \boldsymbol{C}_{i,j}\langle \boldsymbol{x}_i \otimes \boldsymbol{x}_j, \boldsymbol{x} \otimes \boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{Y}} + c\langle \boldsymbol{I}, \boldsymbol{x} \otimes \boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}} \\
&= \left\langle \sum_{i,j} \boldsymbol{C}_{i,j}\boldsymbol{x}_i \otimes \boldsymbol{y}_j + c\boldsymbol{I}, \boldsymbol{x} \otimes \boldsymbol{x} \right\rangle_{\mathcal{X}\otimes\mathcal{X}} = \langle \boldsymbol{W}(\boldsymbol{\mathcal{S}}), \boldsymbol{x} \otimes \boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}}.
\end{aligned}$$

Here we use the property (A.2) for the third and the sixth equality.

[14] Operator $\boldsymbol{W} : \mathcal{X} \mapsto \mathcal{X}$ is self-adjoint if for any $\boldsymbol{x} \in \mathcal{X}$, $\langle \boldsymbol{W}\boldsymbol{x}, \boldsymbol{x}\rangle_{\mathcal{X}} = \langle \boldsymbol{x}, \boldsymbol{W}\boldsymbol{x}\rangle_{\mathcal{X}}$. This assumption on $\boldsymbol{W}$ comes without loss of generality: Given any operator $\boldsymbol{W}$, we can always take a symmetrized version $\boldsymbol{W}_{\text{sym}} = \frac{\boldsymbol{W}+\boldsymbol{W}^*}{2}$, and for any $\boldsymbol{x} \otimes \boldsymbol{x}$, it holds $\langle \boldsymbol{W}_{\text{sym}}, \boldsymbol{x} \otimes \boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}} = \langle \boldsymbol{W}, \boldsymbol{x} \otimes \boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}} = \langle \boldsymbol{W}\boldsymbol{x}, \boldsymbol{x}\rangle_{\mathcal{X}} = \langle \boldsymbol{W}^*\boldsymbol{x}, \boldsymbol{x}\rangle_{\mathcal{X}}$.

[15] We could equivalently define the output of the algorithm as $\langle \boldsymbol{x}, \boldsymbol{W}\boldsymbol{x}\rangle_{\mathcal{X}}$. However, we use $\langle \boldsymbol{W}, \boldsymbol{x} \otimes \boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}}$ to be more in line with the asymmetric instance case.

The proof of the "only if" part is divided into two parts, as in Theorem A.4. In Part 1, we show that (A.7) implies that for any $\boldsymbol{\mathcal{S}}$, $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i=1}^{T}\sum_{j=1}^{T} \boldsymbol{C}_{ij}\boldsymbol{x}_i \otimes \boldsymbol{x}_j + c\boldsymbol{I}$ for some symmetric $\boldsymbol{C} \in \mathbb{R}^{T\times T}$ and $c \in \mathbb{R}$. Then, in Part 2, we show that (2.3) implies that for any $\boldsymbol{\mathcal{S}}$ and any unitary operator $\boldsymbol{U}$, $\boldsymbol{W}((\boldsymbol{U}\otimes\boldsymbol{U})\boldsymbol{\mathcal{S}}) = \sum_{i=1}^{T}\sum_{j=1}^{T} \boldsymbol{C}_{ij}(\boldsymbol{U}\boldsymbol{x}_i)\otimes(\boldsymbol{U}\boldsymbol{x}_j) + c\boldsymbol{I}$. This means that $\boldsymbol{C}$ and c are invariant under unitary transformations of the example sequence $\boldsymbol{\mathcal{S}}$, and thus by a generalization of Lemma 2.2 this is equivalent to stating that $\boldsymbol{C}$ and c depend on $\boldsymbol{\mathcal{S}}$ only via the kernel matrix $\boldsymbol{K}$.

Proof of Part 1: Let $\{\widehat{\boldsymbol{x}}_p\}_{p=1}^{r}$ be an orthonormal basis for $Span\left(\{\boldsymbol{x}_t\}_{t=1}^{T}\right)$. Since $\widehat{\boldsymbol{x}}_p \in Span\left(\{\boldsymbol{x}_t\}_{t=1}^{T}\right)$, there exists matrix $\boldsymbol{P} \in \mathbb{R}^{T\times r}$ such that

$$\widehat{\boldsymbol{x}}_p = \sum_{i=1}^{T} \boldsymbol{P}_{i,p}\boldsymbol{x}_i. \tag{A.8}$$

Complete this basis to an orthonormal basis $\{\widehat{\boldsymbol{x}}_i\}_{i\in\mathbb{N}}$ for $\mathcal{X}$ (note that $\mathcal{X}$ is separable, and hence has a countable basis). We decompose $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i,j} \hat{c}_{i,j}\widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{x}}_j$, and since $\boldsymbol{W}(\boldsymbol{\mathcal{S}})$ is self-adjoint, $\hat{c}_{i,j} = \hat{c}_{j,i}$ for all i, j.

We need to show that (a) $\hat{c}_{p,q} = 0$ if $p \neq q$ and either $p > r$ or $q > r$, and that (b) $\hat{c}_{p,p} = \hat{c}$ for some constant $\hat{c}$, for $p > r$. We show (a) 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 $p \geq 1, p \neq q$. Choose $\boldsymbol{x} = \widehat{\boldsymbol{x}}_p + \widehat{\boldsymbol{x}}_q$ and $\boldsymbol{U}$ as the Hauseholder reflection operator $\boldsymbol{H}$ defined as $\boldsymbol{H}\boldsymbol{x} := \boldsymbol{x} - 2\langle \boldsymbol{x}, \widehat{\boldsymbol{x}}_q\rangle_{\mathcal{X}}\widehat{\boldsymbol{x}}_q$. Then, for any $1 \leq t \leq T$, we have $\boldsymbol{U}\boldsymbol{x}_t = \boldsymbol{H}\boldsymbol{x}_t = \boldsymbol{x}_t - 2\langle \boldsymbol{x}_t, \boldsymbol{x}_q\rangle_{\mathcal{X}}\widehat{\boldsymbol{x}}_q = \boldsymbol{x}_t$ (because $q > r$ and thus $\widehat{\boldsymbol{x}}_q$ lies outside $Span\left(\{\boldsymbol{x}_t\}_{t=1}^{T}\right)$). Thus, the transformed sample $(\boldsymbol{U}\otimes\boldsymbol{U})\boldsymbol{\mathcal{S}}$ is the same as the original sample $\boldsymbol{\mathcal{S}}$, and $\boldsymbol{W}((\boldsymbol{U}\otimes\boldsymbol{U})\boldsymbol{\mathcal{S}}) = \boldsymbol{W}(\boldsymbol{\mathcal{S}})$. On the other hand, $\boldsymbol{U}\widehat{\boldsymbol{x}}_p = \widehat{\boldsymbol{x}}_p - 2\langle \widehat{\boldsymbol{x}}_p, \widehat{\boldsymbol{x}}_q\rangle_{\mathcal{X}}\widehat{\boldsymbol{x}}_q = \widehat{\boldsymbol{x}}_p$, and $\boldsymbol{U}\widehat{\boldsymbol{x}}_q = \widehat{\boldsymbol{x}}_q - 2\langle \widehat{\boldsymbol{x}}_q, \widehat{\boldsymbol{x}}_q\rangle_{\mathcal{X}}\widehat{\boldsymbol{x}}_q = -\widehat{\boldsymbol{x}}_q$. Therefore, the l.h.s. and r.h.s. of (A.7) become

$$\begin{aligned}
\langle \boldsymbol{W}(\boldsymbol{\mathcal{S}}), \boldsymbol{x}\otimes\boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}} &= \hat{c}_{p,p} + \hat{c}_{q,q} + \hat{c}_{p,q} + \hat{c}_{q,p},\\
\langle \boldsymbol{W}((\boldsymbol{U}\otimes\boldsymbol{U})\boldsymbol{\mathcal{S}}), \boldsymbol{U}\boldsymbol{x}\otimes\boldsymbol{U}\boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}} &= \hat{c}_{p,p} + \hat{c}_{q,q} - \hat{c}_{p,q} - \hat{c}_{q,p},
\end{aligned}$$

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

To show (b), we choose $\boldsymbol{x} = \widehat{\boldsymbol{x}}_p$ (we remind that $p > r$), and $\boldsymbol{U}$ to be a permutation operator 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.:

$$\begin{aligned}
&\boldsymbol{U}\boldsymbol{x} = \boldsymbol{x} - \langle \boldsymbol{x}, \widehat{\boldsymbol{x}}_p\rangle_{\mathcal{X}}\widehat{\boldsymbol{x}}_p - \langle \boldsymbol{x}, \widehat{\boldsymbol{x}}_q\rangle_{\mathcal{X}}\widehat{\boldsymbol{x}}_q + \langle \boldsymbol{x}, \widehat{\boldsymbol{x}}_q\rangle_{\mathcal{X}}\widehat{\boldsymbol{x}}_p + \langle \boldsymbol{x}, \widehat{\boldsymbol{x}}_p\rangle_{\mathcal{X}}\widehat{\boldsymbol{x}}_q\\
\text{or}\quad &\boldsymbol{U} = \boldsymbol{I} - \widehat{\boldsymbol{x}}_p\otimes\widehat{\boldsymbol{x}}_p - \widehat{\boldsymbol{x}}_q\otimes\widehat{\boldsymbol{x}}_q + \widehat{\boldsymbol{x}}_p\otimes\widehat{\boldsymbol{x}}_q + \widehat{\boldsymbol{x}}_q\otimes\widehat{\boldsymbol{x}}_p.
\end{aligned}$$

For this choice of $\boldsymbol{U}$, we have $\boldsymbol{U}^* = \boldsymbol{U}$ by (A.3) and $\boldsymbol{U}\widehat{\boldsymbol{x}}_p = \boldsymbol{U}^*\widehat{\boldsymbol{x}}_p = \widehat{\boldsymbol{x}}_q$, $\boldsymbol{U}\widehat{\boldsymbol{x}}_q = \boldsymbol{U}^*\widehat{\boldsymbol{x}}_q = \widehat{\boldsymbol{x}}_p$. Then we can verify that $\boldsymbol{U}\boldsymbol{U}^*\boldsymbol{x} = \boldsymbol{U}^*\boldsymbol{U}\boldsymbol{x} = \boldsymbol{x}$ for any $\boldsymbol{x} \in \mathcal{X}$, namely $\boldsymbol{U}$ is an unitary operator. Since $\boldsymbol{U}\boldsymbol{x}_t = \boldsymbol{x}_t$ for all $1 \leq t \leq T$ (because $p, q > r$), the transformed sample $(\boldsymbol{U}\otimes\boldsymbol{U})\boldsymbol{\mathcal{S}}$ is the same as the original sample $\boldsymbol{\mathcal{S}}$, so that $\boldsymbol{W}((\boldsymbol{U}\otimes\boldsymbol{U})\boldsymbol{\mathcal{S}}) = \boldsymbol{W}(\boldsymbol{\mathcal{S}})$. On the other hand, $(\boldsymbol{U}\boldsymbol{x})\otimes(\boldsymbol{U}\boldsymbol{x}) = (\boldsymbol{U}\widehat{\boldsymbol{x}}_p)\otimes(\boldsymbol{U}\widehat{\boldsymbol{x}}_p) = \widehat{\boldsymbol{x}}_q\otimes\widehat{\boldsymbol{x}}_q$. The l.h.s. and r.h.s. (A.7) become

$$\langle \boldsymbol{W}(\boldsymbol{\mathcal{S}}), \boldsymbol{x}\otimes\boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}} = \hat{c}_{p,p} \quad \text{and} \quad \langle \boldsymbol{W}((\boldsymbol{U}\otimes\boldsymbol{U})\boldsymbol{\mathcal{S}}), \boldsymbol{U}\boldsymbol{x}\otimes\boldsymbol{U}\boldsymbol{x}\rangle_{\mathcal{X}\otimes\mathcal{X}} = \hat{c}_{q,q}, \quad \text{respectively,}$$

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 (A.7) implies that

$$\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{p=1}^{r}\sum_{q=1}^{r} \hat{c}_{p,q}\widehat{\boldsymbol{x}}_p \otimes \widehat{\boldsymbol{x}}_q + \hat{c}\sum_{p>r} \widehat{\boldsymbol{x}}_p \otimes \widehat{\boldsymbol{x}}_p = \sum_{p=1}^{r}\sum_{q=1}^{r} (\hat{c}_{p,q} - \hat{c}\delta_{pq})\widehat{\boldsymbol{x}}_p \otimes \widehat{\boldsymbol{x}}_q + \hat{c}\sum_{i\in\mathbb{N}} \widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{x}}_i$$

and we now prove Part 1 by plugging (A.8) into the above:

$$\begin{aligned}\boldsymbol{W}(\boldsymbol{\mathcal{S}}) &= \sum_{p,q}(\hat{c}_{p,q} - \hat{c}\delta_{pq})\sum_{i,j}\boldsymbol{P}_{i,p}\boldsymbol{P}_{j,q}\boldsymbol{x}_i \otimes \boldsymbol{x}_j + \hat{c}\sum_{i\in\mathbb{N}} \widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{x}}_i \\ &= \sum_{i,j} \underbrace{\left(\sum_{p,q} \boldsymbol{P}_{i,p}(\hat{c}_{p,q} - \hat{c}\delta_{pq})\boldsymbol{P}_{j,q}\right)}_{\boldsymbol{C}_{i,j}} \boldsymbol{x}_i \otimes \boldsymbol{x}_j + \underbrace{\hat{c}}_{c}\underbrace{\sum_{i\in\mathbb{N}} \widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{x}}_i}_{\boldsymbol{I}}.\end{aligned}$$

where $\sum_{i\in\mathbb{N}} \widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{x}}_i$ is just the identity operator $\boldsymbol{I}$ because for any $\boldsymbol{x} = \sum_{i\in\mathbb{N}}\langle \widehat{\boldsymbol{x}}_i, x\rangle \widehat{\boldsymbol{x}}_i \in \mathcal{X}$,

$$\left(\sum_{i\in\mathbb{N}} \widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{x}}_i\right)\boldsymbol{x} = \sum_{i\in\mathbb{N}} (\widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{x}}_i)\,\boldsymbol{x} = \sum_{i\in\mathbb{N}} \langle \widehat{\boldsymbol{x}}_i, \boldsymbol{x}\rangle \widehat{\boldsymbol{x}}_i = \boldsymbol{x}.$$

Without loss of generality, $\boldsymbol{C}$ 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 $\boldsymbol{W}(\boldsymbol{\mathcal{S}})$. Indeed,

$$\begin{aligned}\sum_{i,j} \frac{\boldsymbol{C}_{i,j} + \boldsymbol{C}_{j,i}}{2}\, \boldsymbol{x}_i \otimes \boldsymbol{x}_j + c\boldsymbol{I} &= \frac{1}{2}\left(\sum_{i,j} \boldsymbol{C}_{i,j}\boldsymbol{x}_i \otimes \boldsymbol{x}_j + c\boldsymbol{I}\right) + \frac{1}{2}\left(\sum_{i,j} \boldsymbol{C}_{i,j}\boldsymbol{x}_j \otimes \boldsymbol{x}_i + c\boldsymbol{I}\right) \\ &= \frac{1}{2}\boldsymbol{W} + \frac{1}{2}\boldsymbol{W}^* = \boldsymbol{W}.\end{aligned}$$

Proof of Part 2: By Part 1, $\boldsymbol{W}(\boldsymbol{\mathcal{S}}) = \sum_{i=1}^{r}\sum_{j=1}^{r} \hat{c}_{i,j}\widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{x}}_j + \hat{c}\sum_{i>r} \widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{x}}_i$. Moreover, by applying Part 1 to the sequence $(\boldsymbol{U}\otimes\boldsymbol{U})\boldsymbol{\mathcal{S}}$ we get $\boldsymbol{W}((\boldsymbol{U}\otimes\boldsymbol{U})\boldsymbol{\mathcal{S}}) = \sum_{i=1}^{r}\sum_{j=1}^{r} \hat{d}_{i,j}(\boldsymbol{U}\widehat{\boldsymbol{x}}_i)\otimes(\boldsymbol{U}\widehat{\boldsymbol{x}}_j) + \hat{d}\sum_{i>r}(\boldsymbol{U}\widehat{\boldsymbol{x}}_i)\otimes(\boldsymbol{U}\widehat{\boldsymbol{x}}_i)$ for some coefficients $\hat{d}_{i,j}$ and $\hat{d}$, because if $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^{r}$ is an orthonormal basis for $Span\left(\{\boldsymbol{x}_t\}_{t=1}^{T}\right)$, then $\{\boldsymbol{U}\widehat{\boldsymbol{x}}_i\}_{i=1}^{r}$ is an orthonormal basis for $Span\left(\{\boldsymbol{U}\boldsymbol{x}_t\}_{t=1}^{T}\right)$, and if $\{\widehat{\boldsymbol{x}}_i\}_{i\in\mathbb{N}}$ is an orthonormal basis for $\mathcal{X}$, then so is $\{\boldsymbol{U}\widehat{\boldsymbol{x}}_i\}_{i\in\mathbb{N}}$. To finish the proof, it suffices to show that $\hat{c}_{p,q} = \hat{d}_{p,q}$ for any $p, q \in \{1, \ldots, r\}$, and that $\hat{c} = \hat{d}$. Then Part 2 follows immediately by plugging $\boldsymbol{U}\widehat{\boldsymbol{x}}_p = \sum_{i=1}^{T} \boldsymbol{P}_{i,p}(\boldsymbol{U}\boldsymbol{x}_i)$ into the above equality. By (A.7), for any $1 \leq p, q \leq r$,

$$\hat{c}_{p,q} = \langle \boldsymbol{W}(\boldsymbol{\mathcal{S}}), \boldsymbol{x}_p \otimes \boldsymbol{x}_q\rangle_{\mathcal{X}\otimes\mathcal{X}} = \langle \boldsymbol{W}((\boldsymbol{U}\otimes\boldsymbol{U})\boldsymbol{\mathcal{S}}), \boldsymbol{U}\boldsymbol{x}_p \otimes \boldsymbol{U}\boldsymbol{x}_q\rangle_{\mathcal{X}\otimes\mathcal{X}} = \hat{d}_{p,q}.$$

Similarly, for any $p > r$,

$$\hat{c} = \langle \boldsymbol{W}(\boldsymbol{\mathcal{S}}), \boldsymbol{x}_p \otimes \boldsymbol{x}_p\rangle_{\mathcal{X}\otimes\mathcal{X}} = \langle \boldsymbol{W}((\boldsymbol{U}\otimes\boldsymbol{U})\boldsymbol{\mathcal{S}}), \boldsymbol{U}\boldsymbol{x}_p \otimes \boldsymbol{U}\boldsymbol{x}_p\rangle_{\mathcal{X}\otimes\mathcal{X}} = \hat{d}.$$

□

References

[ABEV09] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. A new approach to Collaborative Filtering: Operator estimation with spectral regularization. *Journal of Machine Learning*, 10:803–826, 2009.

[AMP09] A. Argyriou, C. A. Micchelli, and M. Pontil. When is there a Representer Theorem? Vector versus matrix regularizers. *Journal of Machine Learning Research*, 10:2507–2529, 2009.

[AW01] K. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the Exponential Family of distributions. *Journal of Machine Learning*, 43(3):211–246, June 2001.

[BGV92] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. 5th Annual ACM Workshop on Comput. Learning Theory*, pages 144–152. ACM Press, New York, NY, 1992.

[CCBG08] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear algorithms for online multitask classification. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT 08)*, pages 251–262, July 2008.

[DS12] F. Dinuzzo and B. Schölkopf. The Representer Theorem for Hilbert spaces: a necessary and sufficient condition. In *Advances in Neural Information Processing Systems 24 (NIPS '12)*. MIT Press, 2012.

[DSSST10] J. Duchi, S. Shalev-Shwartz, Yo. Singer, and A. Tewari. Composite objective Mirror Descent. In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT '10)*, 2010.

[For99] J. Forster. On relative loss bounds in generalized linear regression. In *12th International Symposium on Fundamentals of Computation Theory*, pages 269–280, 1999.

[FW95] S. Floyd and M. K. Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):269–304, 1995.

[GBRS09] A. B. A. Graf, O. Bousquet, G. Rätsch, and B. Schölkopf. Prototype classification: Insights from machine learning. *Neural Computation*, 21(1):272–300, January 2009.

[HW01] M. Herbster and M. K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

[KW71] G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian Spline Functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.

[KW97] J. Kivinen and M. K. Warmuth. Additive versus Exponentiated Gradient updates for linear prediction. *Information and Computation*, 132(1):1–64, January 1997.

[KW07] D. Kuzmin and M. K. Warmuth. Online Kernel PCA with entropic matrix updates. In *Proceedings of the 24rd international conference on Machine learning (ICML '07)*, pages 465–471. ACM International Conference Proceedings Series, June 2007.

[Lit88] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[LW94] N. Littlestone and M. K. Warmuth. The Weighted Majority algorithm. *Inform. Comput.*, 108(2):212–261, 1994. Preliminary version in FOCS '89.

[MRW+99] S. Mika, G. Ratsch, J. Weston, B. Schölkopf, and K. R. Müller. Fisher discriminant analysis with kernels. In *Proc. NNSP'99. IEEE Signal Processing Society Workshop*, pages 41–48, 1999.

[Ng04] A. Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML '04)*. ACM, 2004.

[RY08] B. P. Rynne and M. A. Youngson. *Linear Functional Analysis*. Springer, second edition edition, 2008.

[SHS01] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized Representer Theorem. In D. P. Helmbold and B. Williamson, editors, *Proceedings of the 14th Annual Conference on Computational Learning Theory*, number 2111 in Lecture Notes in Computer Science, pages 416–426, London, UK, 2001. Springer-Verlag.

[SSM98] B. Schölkopf, A. J. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[SST11] N. Srebro, K. Sridharan, and A. Tewari. On the universality of online Mirror Descent. In *Advances in Neural Information Processing Systems 23 (NIPS '11)*, pages 2645–2653, 2011.

[TRW05] K. Tsuda, G. Rätsch, and M. K. Warmuth. Matrix Exponentiated Gradient updates for on-line learning and Bregman projections. *Journal of Machine Learning Research*, 6:995–1018, June 2005.

[Val84] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.

[Vov01] V. Vovk. Competitive on-line statistics. *International Statistical Review*, 69:213–248, 2001.

[War07] M. K. Warmuth. Winnowing subspaces. In *Proceedings of the 24rd international conference on Machine learning (ICML '07)*. ACM Press, June 2007.

[WK06] M. K. Warmuth and D. Kuzmin. Online variance minimization. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT '06)*, Pittsburg, June 2006. Springer-Verlag.

[WK08] M. K. Warmuth and D. Kuzmin. Randomized PCA algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research*, 9:2217–2250, 2008.

[WKZ12] M. K. Warmuth, W. Kotłowski, and S. Zhou. Kernelization of matrix updates, when and how? In *Proceeding of the 23rd International Conference on Algorithmic Learning Theory (ALT 12)*, volume 7568 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 2012.

[WV05] M. K. Warmuth and S.V.N. Vishwanathan. Leaving the span. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT '05)*, Bertinoro, Italy, June 2005. Springer-Verlag.