Simultaneous Twin Kernel Learning using Polynomial Transformations for Structured Prediction

Chetan Tonde and Ahmed Elgammal
Department of Computer Science
Rutgers, the State University of New Jersey, New Brunswick, NJ, USA
{cjtonde,elgammal}@cs.rutgers.edu

Abstract

Many learning problems in computer vision can be posed as structured prediction problems*, where the input and output instances are structured objects such as trees, graphs or strings rather than, single labels* $\{+1, -1\}$ *or scalars. Kernel methods such as Structured Support Vector Machines , Twin Gaussian Processes (TGP), Structured Gaussian Processes, and vector-valued Reproducing Kernel Hilbert Spaces (RKHS), offer powerful ways to perform learning and inference over these domains. Positive definite kernel functions allow us to quantitatively capture similarity between a pair of instances over these arbitrary domains. A poor choice of the kernel function, which decides the RKHS feature space, often results in poor performance. Automatic kernel selection methods have been developed, but have focused only on kernels on the input domain (*i.e.*'one-way'). In this work, we propose a novel and efficient algorithm for learning kernel functions simultaneously, on both input and output domains. We introduce the idea of learning* polynomial kernel transformations*, and call this method Simultaneous Twin Kernel Learning (STKL). STKL can learn arbitrary, but continuous kernel functions, including 'one-way' kernel learning as a special case. We formulate this problem for learning covariances kernels of Twin Gaussian Processes. Our experimental evaluation using learned kernels on synthetic and several real-world datasets demonstrate consistent improvement in performance of TGP's.*

1. Introduction

Kernel methods for structured prediction [29, 22, 30, 25, 6, 7, 12] have recently gathered a great deal of interest for solving problems in computer vision [23]. Many learning problems in vision *e.g.* Image Segmentation [19], Handwritten Digit Reconstruction [16], Human Pose Estimation [6] and others, can be posed as structured prediction problems. These techniques deal with complex inputs and outputs, which are governed by inherent structure and strong dependencies between different parts of a single input and output. These methods rely on the *kernel trick* [18] to solve these problems efficiently, and the accuracy depends on the choice of kernel and kernel parameters.

Previous work on automatic learning of kernels splits roughly into three groups based on whether the kernels are learned on input or output: 1) learning a *kernel function* only on the input domain $\mathcal{X}$, 2) learning a *kernel function* on both input domain ($\mathcal{X}$) and output domain ($\mathcal{Y}$) and 3) learning explicit *feature representations*, say $\mathcal{X}'$ of the input so as to use a simple untuned kernel (*e.g.* RBF kernel) on them.

In the first group of work, supervised learning of target kernel matrix $\mathbf{K}'$ is done given an input kernel matrix $\mathbf{K}$. In this case, the target kernel is either known or infered from the label information $\mathcal{Y}$ [20]. In cases, such as [26, 10, 21, 9, 24], they are restricted to belong to a family of kernels and are learned by maximizing the likelihood of observed data. In other words, given an input domain $\mathcal{X}$ with kernel matrix $\mathbf{K}$, we would like to compute a kernel matrix $\mathbf{K}'$ representing the feature space $\mathcal{K}'$, such that the mapped features correlate with the target output/label. These methods are very general and can learn any target kernel [20], but they are computationally intensive and do not give an analytic form for the kernel function. Others methods are restrictive in terms of the family/class of kernel that they can represent [26, 10, 21, 9, 24].

In the second group of work, both input and output domains $\mathcal{X}$ and $\mathcal{Y}$, with input kernel matrices $\mathbf{K}$ and $\mathbf{G}$ respectively, are mapped to feature spaces $\mathcal{K}'$ and $\mathcal{G}'$, such that, correlation between the mapped features is maximized. Most of these techniques are similar in spirit to Kernel Canonical Correlation Analysis (Kernel CCA) [2]. In Kernel CCA, we learn feature mappings $\phi(\cdot)$ and $\psi(\cdot)$ on both the inputs and outputs, such that, their correlations after mapping to spaces ($\mathcal{K}'$, $\mathcal{G}'$) is maximized. In the KCCA case, these mapping are nonlinear but restricted to belong

to their corresponding input and output RKHS spaces, that is, $\phi \in \mathcal{K}$ and $\psi \in \mathcal{G}$. This makes the mapping restrictive and dependent on the choice of the kernel functions $k(\cdot,\cdot)$ and $g(\cdot,\cdot)$. Other techniques are similar to KCCA and optimize different criteria such as Hilbert Schmidt Independence Criterion (HSIC) [8, 15] or Kernel Target Alignment (KTA) [8, 10]) but have similar drawbacks.

In the third group of work, the problem of learning the kernel focusses on learning better *feature representations* so that standard untuned kernels can be used on them [17]. These methods learn a nonlinear mapping on input features $\mathcal{X}$ (*e.g.* using Deep Belief Networks) to learn new features ($\mathcal{X}'$) such that applying a simple known kernel $k'(\cdot,\cdot)$ computes the target kernel matrix $\mathbf{K}'$. In this approach the mappings are nonlinear and are dependent on the design of the DBN architecture, and are computationally expensive.

We propose a novel approach for learning kernels which we refer to as *kernel transformations* or *kernel mappings*. The first contribution of our work is that these *kernel transformations* (referred to as $\phi(\cdot)$ and $\psi(\cdot)$), simultaneously map both input and output feature spaces ($\mathcal{K}$ and $\mathcal{G}$), to new RKHS feature spaces ($\mathcal{K}'$ and $\mathcal{G}'$), such that, correlation between these new features is maximized. The second contribution is that these mappings are general and have explicit analytical forms in terms of special polynomials, which map inner products between feature spaces. These polynomials have a power series form with non-negative coefficients and correspond to concatenating or dropping of polynomial combinations of features from the initial input vector. Also, their analytical form makes it easy to incorporate them into kernel-based algorithms. We propose a simple and efficient algorithm to compute these mappings and to demonstrate the benefits of learning them in the structured prediction setting, where simultaneous learning of kernels on input and output is important and beneficial. We test on several synthetic and real world datasets for the problem of structured prediction.

The outline of this paper is as follows. In Section 2, we present relevant background material on kernel methods for structured prediction, and the proposed framework for learning kernel transformations for structured prediction. In Section 3 and 4 we describe the theory for learning kernel transformations. In Section 5 we describe algorithms for structured prediction. In Section 6, experimental results on synthetic and real-world datasets. Finally, Section 7 and Section 8 present a discussion and conclusion, respectively.

2. Background

2.1. Kernel Methods for Structured Prediction

In structured prediction problems, we learn a prediction function $f\colon \mathcal{X} \to \mathcal{Y}$, from an input domain $\mathcal{X}$ to an output domain $\mathcal{Y}$. We do this by formulating a meaningful, auxiliary evaluation function $h\colon \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ which is maximized during prediction given the training data, for all possible elements in $\mathcal{Y}$, such that,

$$y^* = f(x) = \arg\max_{y\in\mathcal{Y}} h(x,y) \tag{1}$$

Kernel methods for structured prediction [29, 22, 30, 6] can be seen as solving the above problem for some appropriate choice of auxiliary function $h(\cdot,\cdot)$ and kernel functions $k(x,\cdot)\colon \mathcal{X} \to \mathcal{K}$ and $g(y,\cdot)\colon \mathcal{Y} \to \mathcal{G}$ which map input and output elements, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, to RKHS feature spaces.

Twin Gaussian Processes [6] are a recent and popular form of structured prediction methods, which model input-output domains using Gaussian processes with covariance functions, represented by $\mathbf{K}$ and $\mathbf{G}$. These covariance matrices encode prior knowledge about the underlying process that is being modeled. In many real world applications data is high dimensional and highly structured, and the choice of kernel functions is not obvious. In our work, we aim to learn kernel covariance matrices simultaneously. We use TGP as an example to demonstrate the benefits of learning them, although the framework is not limited only to the use of TGP.

2.2. Kernel Transformations

Traditionally, kernel learning can be seen as *one-way* kernel learning, that is, trying to learn a nonlinear mapping from the input domain ($\mathcal{X}$) to the feature space $\mathcal{K}$, by estimating the kernel function $k(x,\cdot)\colon \mathcal{X} \to \mathcal{K}$. This can be seen as a nonlinear mapping from the *linear data space* $\mathcal{X}$, with inner product $k(x_i,x_j) = \langle x_i, x_j\rangle_{\mathcal{K}}$, to a feature space with $k'(x_i,x_j) = \langle x_i, x_j\rangle_{\mathcal{K}'}$. In our proposed idea of kernel transformation, we do the same but by directly operating on the kernel function $k(\cdot,\cdot)$, such that, $\phi\colon \mathcal{K} \to \mathcal{K}'$, where $\mathcal{K}'$ represents the feature space corresponding to some appropriate target kernel function $k'(\cdot,\cdot)$. The learned transformation function $\phi(\cdot)$ needs to satisfy the *positive definiteness preserving property*, which means that the resulting kernel matrix $\mathbf{K}'$ is positive definite and hence is a valid feature space with an inner product. As we will see, this transformation, assuming it exists and is continuous, should be a polynomial with non-negative coefficients (See Eq. 2).

Learning of kernel transformations can also be done on both input and output domains, simultaneously (*Twin*). In this case, both input and outputs are mapped simultaneously to new feature spaces represented by RKHS's $\mathcal{K}'$ and $\mathcal{G}'$. We learn two *kernel transformations* $\phi\colon \mathcal{K} \to \mathcal{K}'$, on input domain $\mathcal{X}$ and $\psi\colon \mathcal{G} \to \mathcal{G}'$, on the output domain $\mathcal{Y}$. Also, both maps $\phi(\cdot)$ and $\psi(\cdot)$, should satisfy the above mentioned *positive definiteness preserving property*. All this is done in a way such that these two new feature spaces $\mathcal{K}'$ and $\mathcal{G}'$, are more correlated with each other, giving better

prediction. Several different criteria have been proposed in the literature [15, 8, 28, 10]. We choose Hilbert Schmidt Independence Criterion (HSIC) [14] for reasons explained later in Section 3.2.

3. Framework

3.1. Polynomial Kernel Transformations

Kernel functions maintain their symmetric positive definite (SPD) property under certain types of operations [18]; these operations have proven to be useful in designing algorithms for kernel based learning. If $\mathbf{K}_{n\times n}$ is a positive definite kernel matrix, then its combinations $\mathbf{K}+\mathbf{K}$ (direct sum), $\alpha\mathbf{K}$ (scaling, $\alpha > 0$), $\mathbf{K}\mathbf{K} = \mathbf{K}^2$ (matrix multiplication), $\mathbf{K} \otimes \mathbf{K}$ (direct product), and $\mathbf{K} \odot \mathbf{K} = \mathbf{K}^{(2)}$(Hadamard/Schur product [1]) are also kernel matrices. These operations are equivalent to applying functions of the form $\phi(\cdot) : \mathbb{R} \to \mathbb{R}$ element wise, on entries $[\mathbf{K}]_{i,j}$ of the kernel matrix $\mathbf{K}$, for example $\phi(t) = 2t$ (addition), $\phi(t) = t^2$ (power) and $\phi(t) = \alpha t$ (scaling) on a single kernel [2]. In general, the following statement is known from mathematical literature,

Theorem 3.1 (FtizGerald et. al. 1995 [13])**.** *If there exists a continuous function $\phi\colon \mathbb{R} \to \mathbb{R}$, such that, $\mathbf{K}' = \phi([\mathbf{K}]_{i,j})$ then $\mathbf{K}'$ is positive definite for any SPD matrix $\mathbf{K}$ if and only if $\phi(\cdot)$ it is real entire and of the form below,*

$$\phi(t) = \sum_{i=0}^{\infty} \alpha_i t^i \tag{2}$$

with $\alpha_i \geq 0$ for all $i \geq 0$.

The above form indicates that $\phi(\cdot)$ is an infinitely differentiable ($\phi(\cdot) \in C^{\infty}$) power series, and all coefficients α_j are non-negative (*e.g.* $\phi(t) = e^t$). Now, by Mercer's theorem we know that there is a one to one correspondence between kernel matrices, the RKHS's they represent, and the kernel function which represents it. Using this fact, and for simplicity of exposition, we write ϕ in three different but equivalent forms $\phi : \mathcal{K} \to \mathcal{K}'$, or $\phi : \mathbf{K} \to \mathbf{K}'$, or else $\phi : \mathbb{R} \to \mathbb{R}$. In short, the polynomial $\phi(\cdot)$ gives us a nonlinear map between any two RKHS spaces represented by $\mathcal{K}$ and $\mathcal{K}'$, and with the inner products $k(\cdot,\cdot)$ and $k'(\cdot,\cdot)$, respectively.

So given any input kernel $\mathbf{K}$ on input data and a given target kernel $\mathbf{K}'$, if we can compute (or approximate) $\phi(\cdot)$, then we can use this explicit $\phi(\cdot)$ function to map our data from an initial RKHS $\mathcal{K}$ to $\mathcal{K}'$. This unique form of the polynomial mapping above has been known before but has never been used in this form for learning, to the best our knowledge. In the next section, we derive an approach to learn/approximate these kernel transformations and later show how these mappings could potentially help.

[1] $\mathbf{K}^{(i)} = \mathbf{K} \odot \mathbf{K} \ldots \mathbf{K}$ i-times

[2] A similar generalization exists for operations on multiple kernels, see [13].

3.2. Twin Kernel Transformations (TKT)

Let $\phi(\cdot) : \mathcal{K} \to \mathcal{K}'$ be a *kernel transformation*, such that it maps the kernel matrix $\mathbf{K}$ to the matrix $\mathbf{K}'$, and simultaneously, we assume $\psi(\cdot)\colon \mathcal{G} \to \mathcal{G}'$ is a *kernel transformation* that maps the kernel matrix $\mathbf{G}$ to $\mathbf{G}'$. Using Eq. 2 on the output kernel we have for the mapping $\psi\colon \mathcal{G} \to \mathcal{G}'$,

$$\psi(t) = \sum_{j=0}^{\infty} \beta_j t^j. \tag{3}$$

We are interested in applying these kernel transformations to both the input and output matrices such that, after application, the correlation between the new mapped RKHS features is maximized. To measure this correlation we propose to use the Hilbert Schmidt Independence Criterion (HSIC) given in [14], which measures this naturally. We restate the definition of HSIC for convenience below.

Definition 3.1. If we have two RKHS's $\mathcal{K}$ and $\mathcal{G}$, then a measure of statistical dependence between $\mathcal{X}$ and $\mathcal{Y}$ is given by the norm of the cross-covariance operator $\mathbf{M}_{xy} : \mathcal{G} \to \mathcal{K}$, which is defined as,

$$\mathbf{M}_{xy} := \mathbf{E}_{x,y}[(k(x,\cdot) - \mu_x) \otimes (g(y,\cdot) - \mu_y)] \tag{4}$$

$$= \mathbf{E}_{x,y}[(k(x,\cdot) \otimes g(y,\cdot))] - \mu_x \otimes \mu_y \tag{5}$$

and the measure is given by the Hilbert-Schmidt norm of $\mathbf{M}_{xy}$ which is,

$$HSIC(p_{xy}, \mathcal{K}, \mathcal{G}) := ||\mathbf{M}_{xy}||^2_{HS} \tag{6}$$

The larger the above norm, the higher the statistical dependence between $\mathcal{X}$ and $\mathcal{Y}$. The advantages of using HSIC for measuring statistical dependence, as stated in [14] are as follows: first, it has good uniform convergence guarantees; second, it has very little bias even in high dimensions; and third a number of algorithms can be viewed as maximizing HSIC subject to constraints on the labels/outputs. Empirically, in terms of kernel matrices, this is defined below,

Definition 3.2. Let $Z := \{(x_1, y_1), \ldots (x_m, y_m)\} \subseteq \mathcal{X} \times \mathcal{Y}$ be a series of m independent observations drawn from p_{xy}. An unbiased estimator of $HSIC(Z, \mathcal{K}, \mathcal{G})$ is given by,

$$HSIC(Z, \mathcal{K}, \mathcal{G}) = (m-1)^{-2} trace(\mathbf{KHGH}) \tag{7}$$

where $\mathbf{K}, \mathbf{H}, \mathbf{G} \in \mathbb{R}^{m\times m}$, $[\mathbf{K}]_{i,j} := k(x_i, x_j)$, $[\mathbf{G}]_{i,j} := g(y_i, y_j)$ and $[\mathbf{H}]_{i,j} := \delta_{ij} - m^{-1}$

Now, for well defined and *bounded kernels* $\mathbf{K}$ and $\mathbf{G}$ we have $HSIC(Z, \mathcal{K}, \mathcal{G}) \geq 0$. For the sake of simplicity,

we denote $HSIC(Z, \mathcal{K}, \mathcal{G})$ by $\overline{HSIC}(\mathbf{K}, \mathbf{G})$. Using the above empirical version of the HSIC, we finally define the following objective function for our problem,

$$\mathbf{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}} \overline{HSIC}(\phi(\mathbf{K}), \psi(\mathbf{G})) \tag{8}$$

$$\text{subject to } \alpha_i \geq 0, \beta_j \geq 0, \forall i, j \geq 0 \tag{9}$$

Now, solving the above problem for functions $\phi(\cdot)$ and $\psi(\cdot)$, which have infinitely many unknown coefficient's α_i's and β_j's, is obviously intractable. Hence, we approximate our solution by truncation of the input and output mapping degrees to d_1 and d_2, respectively. We give a detailed justification for this for this in Section 4.

4. Learning Twin Kernel Transformations

4.1. Algorithm

In this subsection, we propose a linear algebraic solution to approximate the nonlinear *kernel transformations* $\phi(\cdot)$ and $\psi(\cdot)$. Using Eqs. 2 and 3 on kernel matrices $\mathbf{K}$ and $\mathbf{G}$ respectively, we get the following Eqs. for $\phi : \mathbf{K} \rightarrow \mathbf{K}'$ and $\psi : \mathbf{G} \rightarrow \mathbf{G}'$,

$$\phi(\mathbf{K}) = \sum_{i=0}^{\infty} \alpha_i \mathbf{K}^{(i)}, \alpha_i \geq 0, \ \forall i \geq 0 \tag{10}$$

$$\phi(\mathbf{G}) = \sum_{j=0}^{\infty} \beta_j \mathbf{G}^{(j)}, \beta_j \geq 0, \ \forall j \geq 0 \tag{11}$$

Assuming that our input and output kernels are bounded ($\mathbf{K} \leq M < \infty$ and $\mathbf{G} \leq M' < \infty$), we would like our new mapped kernels $\phi(\mathbf{K})$ and $\psi(\mathbf{G})$ to be finite and bounded; for this to happen, we impose a regularization constraint on the magnitude of α_i's and β_j's by setting $\boldsymbol{\alpha} \in \boldsymbol{\Delta}^{d_1}$ and $\boldsymbol{\beta} \in \boldsymbol{\Delta}^{d_2}$, where $\boldsymbol{\Delta}^d = \{(x_0, x_1, \ldots, x_d) | x_i \geq 0 \text{ and } ||\boldsymbol{x}||_2 = 1\}$. This regularization helps generalization to unseen test data and avoid scaled increase in the coefficient values so as to maximize the objective function.

Substituting Eqs. 10 and 11 in Eq. 9, along with the regularization constraint and defining a $\mathbf{C}$ - matrix such that, $[\mathbf{C}]_{i,j} = \overline{HSIC}(\mathbf{K}^{(i)}, \mathbf{G}^{(j)})$, we get the final optimization problem as,

$$\text{maximize} \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \alpha_i \beta_j \mathbf{C}_{i,j} \tag{12}$$

$$\text{subject to, } \boldsymbol{\alpha} \in \boldsymbol{\Delta}^{d_1}, \boldsymbol{\beta} \in \boldsymbol{\Delta}^{d_2}$$

The $\mathbf{C}$-matrix above is finite-dimensional approximation of the true infinite-dimensional matrix, and hence, can be solved using methods from linear algebra. We should also note that every entry $[\mathbf{C}]_{i,j}$ represents a form of higher order cross-covariance between kernel matrices. This form of $\mathbf{C}$-matrix and has not been used before in kernel learning literature, to the best of our knowledge.

The justification for approximation to finite degrees is as follows. Choosing the degree of the mapping functions to be $deg(\phi) = d_1$ and $deg(\psi) = d_2$ amounts to approximating the $\mathbf{C}$-matrix to have finite dimensions ($(d_1+1) \times (d_2+1)$). Every entry $[\mathbf{C}]_{i,j} = \overline{HSIC}(\mathbf{K}^{(i)}, \mathbf{G}^{(j)})$ represents higher order correlations among the polynomial features of order i and j of input and output, respectively. Therefore, we are maximizing these higher order correlations among features by appropriately choosing the coefficient's α_i and β_j. So, the higher the degree we choose, the better it is.

We solve for the unknown's α_i's and β_j's by using Singular Value Decomposition (SVD) and choosing $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to be the first left and right singular vectors of the $\mathbf{C}$-matrix (See Theorem 4.2). We note that the solutions vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ need to be non-negative and have a unit $\mathbf{L}2$-norm. We show that this is in fact true and our proposed solution satisfies these constraints.

Note. In the above formation, if we set $d_2 = 1$, then the solution for the $\boldsymbol{\beta}$ vector is $\boldsymbol{\beta} = [0, 1]^T$. This is because α_0 and β_0 are zero (See Theorem 4.3). Hence, the output kernel mapping corresponds to the identity mapping, $\psi(t) = t$, which is equivalent to using *no mapping* on the output kernel. A similar property holds if we set $d_1 = 1$, then we have $\phi(t) = t$, which uses *no mapping* on the input kernel.

Note. Our proposed model is similar to Kernel CCA using HSIC criterion [8]. In KCCA, we find two nonlinear mappings $\phi(\cdot) \in \mathcal{K}$ and $\psi(\cdot) \in \mathcal{G}$, such that, the statistical correlations are maximized. In our approach, we are also looking for nonlinear kernel transformations $\phi(\cdot)$ and $\psi(\cdot)$ between kernel matrices, but that are more general, have an explicit analytical form, and do not belong to a vector space $(\mathcal{K}, \mathcal{G})$.

4.2. Theoretical Analysis

4.2.1 Some properties of $\phi(\cdot), \psi(\cdot)$ and C-matrix

In this section, we characterize some properties of mappings $\phi(\cdot)$ and $\psi(\cdot)$. We begin by stating the following lemma about the $\mathbf{C}$ - matrix in Eqn. 12 ,

Lemma 4.1. $[\mathbf{C}]_{0,j} = [\mathbf{C}]_{i,0} = 0$ *and* $[\mathbf{C}]_{i,j} \geq 0$.

Proof. See supplementary material. □

We have the following theorem regarding the solution of optimization problem 12,

Theorem 4.2. *The solution* $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ *to the optimization problem in 12 is given by the first left and right singular vector of the* $\mathbf{C}$*-matrix*

Proof. Using the Perron-Frobenius theorem to square nonnegative matrices $\mathbf{C}^T\mathbf{C}$ and $\mathbf{C}\mathbf{C}^T$ which are nonnegative,

we claim that both $\mathbf{C}^T\mathbf{C}$ and $\mathbf{C}\mathbf{C}^T$ have Perron vectors $\boldsymbol{\alpha}^*$ and $\boldsymbol{\beta}^*$, respectively. Both $\boldsymbol{\alpha}^*$ and $\boldsymbol{\beta}^*$ are the left and right singular vectors of $\mathbf{C}$ and also maximize Eq. 12. □

This gives us our required solution to the problem. We now make an important statement regarding the mapping coefficients α_i's and β_j's.

Theorem 4.3. *If $\boldsymbol{\alpha}^* \in \boldsymbol{\Delta}^{d_1}$ and $\boldsymbol{\beta}^* \in \boldsymbol{\Delta}^{d_2}$ are maximizers of optimization problem 12, then the following statements hold,*

1. $\alpha_0^* = 0$ *and* $\beta_0^* = 0$
2. *If $[\mathbf{C}]_{i,k}$ decreases/increases with respect to i for all k then α_i^* also correspondingly decreases/increases.*
3. *If $[\mathbf{C}]_{k,j}$ decreases/increases with respect to j for all k then β_j^* also correspondingly decreases/increases.*

Proof. See supplementary material. □

From the above theorem we observe that for the setting $d_1 = d_2 = 1$ corresponds to identity the mapping, $\phi(t) = t$ and $\psi(t) = t$. It also characterizes trends in the coefficients of $\phi(\cdot)$ and $\psi(\cdot)$ in terms of the correlations which we observe empirically.

4.2.2 Computational Complexity

If we ignore the initial kernel evaluation of matrix $\mathbf{K}$ and $\mathbf{G}$ which take $O(n^2)$ time. The runtime complexity of our algorithm mainly depends on construction of our $\mathbf{C}$-matrix. To compute matrices $\mathbf{K}^{(i)}$ and $\mathbf{G}^{(j)}$ it takes $O(n^2(d_1+d_2))$ multiplications. To construct the entries of the $\mathbf{C}$-matrix, we need $O(n^2 d_1 d_2)$ multiplications. These computations can be heavily parallelized. Computing the left and right eigenvectors of the $\mathbf{C}$-matrix can be done in $O(d_1 d_2)$ time which is linear in both d_1 and d_2, and hence is also optimal. Typically we have $d_1, d_2 << n$ therefore doing it is efficient. The space requirement for our algorithm is $O(n^2(d_1+d_2))$ to store powers of matrices $\mathbf{K}$ and $\mathbf{G}$, which can be cached. If we compare this to its most closely related algorithm, KCCA [8] which has time complexity $O(n^3)$ this algorithm is one order of magnitude faster in terms of the number of data points n.

5. Structured Prediction with STKL

In this section, we devise algorithms for structured prediction using Twin Gaussian Processes with *Simultaneous Twin Kernel Learning*. An important benefit of learning these *Kernel Transformations* is that they have explicit and analytic forms making it straightforward to integrate them into kernel based methods by simply replacing with transformed kernels $(\phi(\boldsymbol{K}), \psi(\boldsymbol{G}))$.

5.1. Modified Twin Gaussian Processes (TGP)

In Twin Gaussian Processes the choice of the auxiliary evaluation function is typically some form of information measure, *e.g.* KL-Divergence or HSIC which are known to be special cases of Bregman divergences [3]. KL-Divergence is an asymmetric measure of information, while HSIC is symmetric in its arguments. We also investigate both of these criteria against the degree of mapping. This allows us to show how each information measure is affected as the mapping degrees d_1 and d_2 are increased. The relationship we observe is straightforward and direct, allowing the choice of d_1 and d_2 to be made easily. We refer to these modified TGP's as Higher Order TGP with KL-Divergence (HOTGP) and Higher Order HSIC (HOHSIC) for TGP using HSIC.

TGP with KL-Divergence: In this version of TGP, we minimize the KL-divergence between the transformed kernels, $\phi(\boldsymbol{K})$ and $\psi(\boldsymbol{G})$, given the training data $(X \times Y)$ and test example x^*. The prediction function for HOTGP is,

$$\mathbf{y}^* = \arg\min_y D_{KL}((\psi(\boldsymbol{G}_{Y\cup y})||\phi(\boldsymbol{K}_{X\cup x^*})) \tag{13}$$

The mapping degree of transformations (d_1 and d_2), has direct and straightforward effects on the value of KL-Divergence (increase and decrease respectively); this variation is explained in detail in the experiments section.

TGP with HSIC (HOHSIC): For this version of TGP with HSIC criteria, the prediction function maximizes the HSIC between the transformed kernels $\phi(\boldsymbol{K})$ and $\psi(\boldsymbol{G})$ given the training data $(X \times Y)$ and test example x^*. The prediction function looks as follows,

$$\mathbf{y}^* = \arg\max_y \overline{HSIC}((\psi(\boldsymbol{G}_{Y\cup y}), \phi(\boldsymbol{K}_{X\cup x^*})) \tag{14}$$

In this case, the mapping degrees of the input and output transformations, d_1 and d_2, have a similar impact (increase) on the maximum value of the prediction function.

6. Experiments

We demonstrate empirical results of HOTGP and HOHSIC on a synthetic dataset and three real-world datasets. To measure improvement of performance over the baseline we look at empirical reduction in error which we call *% Gain* , defined as, $\%\ Gain = \left(1 - \frac{Error_{(mapping)}}{Error_{(no\ mapping)}}\right) \times 100$. In all the experiments, we use the RBF kernel which is $k(x_i, x_j) = exp(-\gamma_x||x_i - x_j||^2)$ and $g(y_i, y_j) = exp(-\gamma_y||y_i - y_j||^2)$ as an initial kernel on input and output. The bandwidth parameters(γ_x, γ_y) were chosen from papers relevant to these datasets.

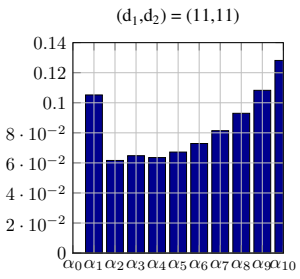
6.1. S Shape Dataset

Consider the *S-shape* synthetic dataset from [4] which is a 1D input/output regression problem. In this dataset,

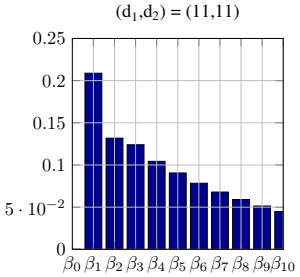
500 values of inputs (x) are sampled uniformly in $(0, 1)$ and then evaluated for $r = x + 0.3sin(2\pi x) + \epsilon$, with ϵ drawn from a zero mean Gaussian noise with standard deviation $\sigma = 0.05$. The goal here is to solve the inverse problem, which is to predict x, given r. The dataset is challenging in the sense that it is multivalued (in the middle of the S-shape), discontinuous (at the boundary of univalued and multivalued region) and noisy ($\epsilon = \mathcal{N}(0, \sigma)$).

We use the RBF kernel with $\gamma_x = 0.2$ and $\gamma_y = 20$ from [6] and learn the input and output mapping's $\phi(\cdot)$ and $\psi(\cdot)$ using our proposed approach. A sample plot of the learned coefficient's (α_i's , β_j's) is given in Figs. 1a and 1b. We clearly observe that $\alpha_0 = \beta_0 = 0$, and the variation of magnitude of the coefficients with mapping degree d_1 and d_2 (See Theorem 4.3).

Figures 2a and 2b show the plot of *% Gain* vs input degree (d_1) vs output degree (d_2). In the case of HOTGP, the *% Gain* degrades with an increase in input mapping degree (d_1), while improving with output mapping degree (d_2). This is because of the asymmetric nature of the KL-Divergence criterion. We are minimizing KL-Divergence in our prediction function which increases with an increase in d_1 and decreases with an increase in d_2, which explains this. In the other case of HOHSIC, the criterion is symmetric: there is an increase in $\%Gain$ when both input and output degrees are increased. This correlates well with the formulation of our proposed STKL problem objective. Finally, Figures 3c-3d and 3a-3b, show the initial and improved regression estimates against the ground truth data (GT) for HOTGP and HOHSIC, respectively. We can clearly observe improvements in both cases.



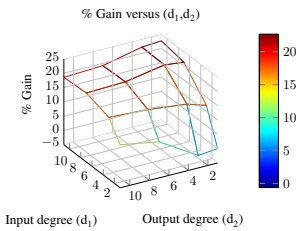
(a) Coefficient α_i's



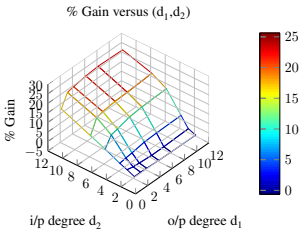
(b) Coefficient β_j's

6.2. USPS Handwritten Digits Reconstruction

In the problem of handwritten digit reconstruction from [30], the goal is to predict 16 pixel values in the center of an image, given the outer pixels. We use 7425 examples for training (without labels) and 2475 examples (roughly 1/4th for each digit) for testing. Table 2 shows results for the Mean Absolute Error and compares our approach with other kernel-based methods. The mapping degrees chosen for HOTGP are $(d_1, d_2) = (1, 11)$ and for HOHSIC are



(a) *% Gain* in Mean Abs. Error for S-shape dataset with KL-Div. criterion. $\%Gain$ improves with d_1 only.



(b) *% Gain* in Mean Abs. Error for S-shape dataset with HSIC criterion. $\%Gain$ increases with d_1 and d_2.

$(d_1, d_2) = (11, 11)$. This was done empirically based on the prediction criteria as described in subsection 6.1 for the S-Shape dataset, and then increasing the degrees (d_1, d_2) until the $\%Gain$ is saturated.

Crit. / Mean Abs. Er	(no mapping)	(mapping)	Gain %
KL-Div	0.2151	0.21078	**1.9924** %
HSIC	0.3399	0.3314	**2.4842** %

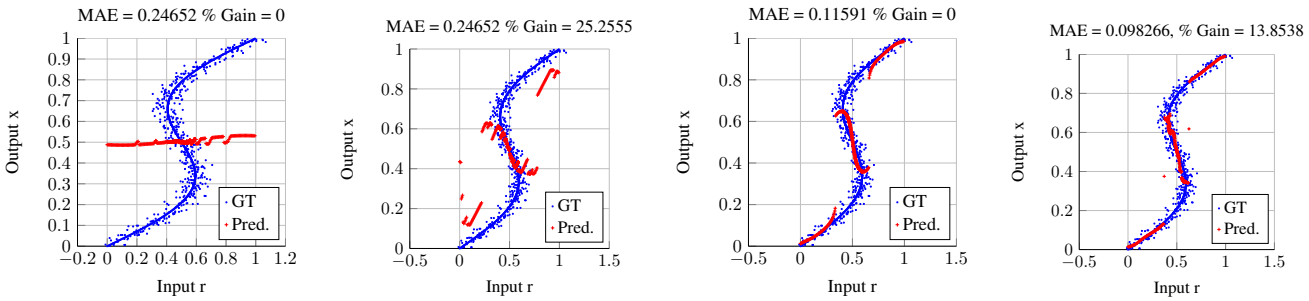
Table 1: Mean Absolute Error for *USPS Handwritten digits* dataset for the two criteria, with and without mapping.

Approach	MAE	Approach	MAE
NN	0.341	KRR	0.250
SVR	0.250	KDE	0.260
$SOAR_{krr}$	0.233	$SOAR_{svr}$	0.230
HSIC (wo/map)	0.3399	KL-Div (wo/map)	*0.21508*
HSIC (w/map)	0.3327	KL-Div (w/map)	*0.21084*
% Gain	**2.4842** %	*% Gain*	**1.9924** %

Table 2: Comparison with others models from [5] for *USPS digits* dataset. The two lowest errors are *emphasized*. NN means nearest neighbor regression, KDE means kernel dependency estimation [30] with 16d latent space obtained by kernel principal component analysis. SOAR means Structured Output Associative regression [5]. The last row shows the *% Gain* for KL-Div and HSIC criteria, with and without mapping.

6.3. Poser Dataset

The *Poser* dataset contains synthetic images of human motion capture sequences from the *Poser 7* software [1]. The motion sequences includes 8 categories: walk, run, dance, fall, prone, sit, transitions and misc. There are 1927 training examples coming from different sequences of varying lengths and the test set is a continuous sequence of 418 time steps. The input feature vectors are 100d silhouette shape descriptors while the output feature vectors are 54d vectors with the x, y and z rotation of joint angles. The final results are shown in Table 4.



(a) HSIC $(d_1, d_2) = (1, 1)$ (b) HSIC $(d_1, d_2) = (11, 11)$ (c) KL-Div $(d_1, d_2) = (1, 1)$ (d) KL-Div $(d_1, d_2) = (1, 11)$

Figure 3: Regression on S-Shape dataset with HOTGP (Fig.3c-3d) and HOHSIC (Fig.3a-3b)

6.4. HumanEva-I Pose Dataset

We demonstrate the performance of HOTGP and HOHSIC on the challenging *HumanEva-I* dataset from [27]. This dataset contains real motion capture sequences from three different subjects (S1,S2,S3) performing five different actions (Walking, Jogging, Box, Throw/Catch, Gestures). Our models are trained on data from all subjects. We have input images from three different cameras; C1,C2 and C3 and HoG features [11] from them. The output vectors are 60d with the x, y, z joint positions in mm.

We report results using concatenated features from all three cameras (C1+C2+C2) and also using features from each individual camera (C1,C2 or C3. See Table 3). We use an RBF kernel with $\gamma_x = \gamma_y = 10^{-4}$ and the $(d_1, d_2) = (1, 11)$ for HOTGP and $(d_1, d_2) = (11, 11)$ for HOHSIC. The $\%$ *Gain* for each criteria is shown in bold. In the case of concatenated features (C1+C2+C3), we observe a *% Gain* of 5.0796% using HOTGP and 0.2% for symmetric HOHSIC. In all cases we show consistent improvement in performance. For detailed results on all subjects and all actions for both criteria, see supplementary material.

Features	Crit.	w/map	wo/ map	Gain %
HoG (C1C2C3)	KL-Div	45.1729	42.8783	**5.0796** %
	HSIC	171.4085	171.3766	**0.018613** %
HoG (C1)	KL-Div	34.2885	33.4262	**2.5147** %
	HSIC	171.4085	171.3769	**0.018427** %
HoG (C2)	KL-Div	31.9928	31.5792	**1.2928** %
	HSIC	171.4085	171.3755	**0.019237** %
HoG (C3)	KL-Div	30.9279	30.4928	**1.4067** %
	HSIC	171.4085	171.3762	**0.018835** %

Table 3: Mean Absolute Error for *HumanEva-I* dataset for the two criteria, with and without mapping. The mapping degrees for KL-Divergence are $(d_1, d_2) = (1, 11)$ and for HSIC are $(d_1, d_2) = (11, 11)$

7. Discussion

In Table 4 we summarize all of our results. In general, we observe the following. Firstly, for HOTGP, we observe that the higher the mapping degree of $\psi(\cdot)$, the more the decrease in error, while in the case of mapping function $\phi(\cdot)$, mapping to higher degrees does not help. This is predictable because of the asymmetric nature of KL-Divergence. In the other case of HOHSIC, increasing the degree of both $\phi(\cdot)$ and $\psi(\cdot)$ is helpful as it is a symmetric criteria. The improvement in HOHSIC is greater because it is more of an alignment measure, which is unlike KL-Divergence, and it is also a part of our STKL objective. This proves that the choice of whether to map input, or output, or both, is entirely dependent on the criteria of the prediction function. Regardless, after a choice is made, as a general rule, having higher mapping degree(s) gives better the optimization (minimization/maximization) of the involved prediction function. We also note that, in case of TGP, there is also a tradeoff between solving a more nonlinear optimization problem and using higher degrees for mapping.

8. Conclusions and Future Work

In this paper, we introduce a novel method of *Simultaneous Twin Kernel Learning* which simultaneously learns polynomial kernel transformations on both input and output so as to maximize their statistical dependence. We show that our proposed approach is general and includes *one-way* kernel learning as special case. These transformations are analytical and explicit, making it easy to incorporate them into existing kernel-based frameworks. We empirically demonstrate on a synthetic and several real-world datasets that we consistently improve accuracy over baseline results. In future work, we intend to further investigate by looking into multiple kernel extensions of this work and to provide theoretical and statistical guarantees of our learned kernels.

% Gain Criterion - (d_1, d_2)	*S-shape*	*Poser*	USPS Digits	*HumaEva-I* (C1+C2+C3)	*HumanEva-I* (C1,C2,C3)
KL-Divergence - $(1, 11)$	13.8538 %.	6.39 %	1.9924 %	5.0796 %	(2.5147 %,1.2928 %,1.4067 %)
HSIC - $(11, 11)$	25.2555 %.	1.2613 %	2.4842 %	0.0186 %	(0.0184 %, 0.0192 %, 0.0188 %)

Table 4: Summary: $\%Gain$ for all three datasets with two different criteria with and without mapping (baseline).

9. Acknowledgements

This work has been partially funded by a Google Research Award.

References

[1] Poser 3D Animation & Character Creation Software - Official Website. 6

[2] F. R. Bach and M. I. Jordan. Kernel independent component analysis. *The Journal of Machine Learning Research*, 3(6):1–48, Mar. 2003. 1

[3] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005. 5

[4] C. M. Bishop and M. Svenskn. Bayesian hierarchical mixtures of experts. In *UAI'03: Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 57–64, Aug. 2002. 5

[5] L. Bo and C. Sminchisescu. Structured output-associative regression. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2403–2410. IEEE, 2009. 6

[6] L. Bo and C. Sminchisescu. Twin Gaussian Processes for Structured Prediction. *International Journal of Computer Vision*, 87(1-2):28–52, 2010. 1, 2, 6

[7] S. Bratieres, N. Quadrianto, and Z. Ghahramani. Bayesian Structured Prediction Using Gaussian Processes. July 2013. 1

[8] B. Chang, U. Kruger, R. Kustra, and J. Zhang. Canonical Correlation Analysis based on Hilbert-Schmidt Independence Criterion and Centered Kernel Target Alignment. *jmlr.org*, pages 316–324, 2013. 2, 3, 4, 5

[9] O. Chapelle and A. Rakotomamonjy. Second order optimization of kernel parameters. In *Proc. of the NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, page 87, 2008. 1

[10] C. Cortes, M. Mohri, and A. Rostamizadeh. Algorithms for Learning Kernels Based on Centered Alignment. *Journal of Machine Learning Research*, (1):550–828, Mar. 2012. 1, 2, 3

[11] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 886–893. IEEE, 2005. 7

[12] T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning Multiple Tasks with Kernel Methods. *Journal of Machine Learning Research*, 2005. 1

[13] C. H. FitzGerald, C. A. Micchelli, and A. Pinkus. Functions that preserve families of positive semidefinite matrices. *Linear Algebra and its Applications*, 1995. 3

[14] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. *Algorithmic Learning Theory*, 2005. 3

[15] A. Gretton, R. Herbrich, A. Smola, O. Bousquet, and B. Schölkopf. Kernel Methods for Measuring Independence. *The Journal of Machine Learning Research*, 6:2075–2129, Dec. 2005. 2, 3

[16] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 263–270, 2011. 1

[17] G. E. Hinton and R. Salakhutdinov. Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes. *Advances in Neural Information Processing (NIPS)*, 2007. 2

[18] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 2008. 1, 3

[19] A. Ion, J. Carreira, and C. Sminchisescu. Image segmentation by figure-ground composition into maximal cliques. *Computer Vision (ICCV)*, pages 2110–2117, 2011. 1

[20] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the Kernel Matrix with Semidefinite Programming. *The Journal of Machine Learning Research*, 5:27–72, Dec. 2004. 1

[21] C. A. Micchelli and M. Pontil. Learning the kernel function via regularization. In *Journal of Machine Learning Research*, pages 1099–1125, 2005. 1

[22] C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17(1):177–204, 2005. 1, 2

[23] S. Nowozin and C. H. Lampert. *Structured Learning and Prediction in Computer Vision*. Now Publishers Inc, May 2011. 1

[24] C. S. Ong, A. J. Smola, and R. C. Williamson. Learning the Kernel with Hyperkernels. *The Journal of Machine Learning Research*, 6, Dec. 2005. 1

[25] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. 1

[26] N. Shawe-Taylor and A. Kandola. On kernel target alignment. *Advances in Neural information Processing Systems*, 14:367, 2002. 1

[27] L. Sigal and M. J. Black. Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. *Brown Univertsity Technical Report*, 2006. 7

[28] L. Song, A. Smola, A. Gretton, and K. M. Borgwardt. A dependence maximization view of clustering. In *The 24th International Conference on Machine Learning*, pages 815–822, New York, New York, USA, 2007. ACM Press. 3

[29] I. Tsochantaridis and T. Joachims. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 2005. 1, 2

[30] J. Weston, O. Chapelle, and V. Vapnik. Kernel dependency estimation. *Advances in Neural Information Processing (NIPS)*, 2002. 1, 2, 6