An Efficient Algorithm for Tessellated Kernel Learning

Anonymous Authors[1]

Abstract

The accuracy and complexity of machine learning algorithms based on kernel optimization are determined by the set of kernels over which they are able to optimize. An ideal set of kernels should: admit a linear parameterization (for tractability); be dense in the set of all kernels (for robustness); be universal (for accuracy). The recently proposed set of Tesselated Kernels (TKs) is currently the only known class which meets all three criteria. However, previous algorithms for TK Kernel Learning (TKL) were limited to classification and furthermore relied on computationally complex Semidefinite Programming (SDP) algorithms. In this paper, we pose the TKL problem as a minimax optimization problem and propose a SVD-QCQP primal-dual algorithm which dramatically reduces the computational complexity as compared with previous SDP-based approaches. Furthermore, we provide an efficient implementation of this algorithm for both classification and regression, and which enables us to solve problems with 100 features and up to 30,000 datums. Furthermore, when applied to benchmark data, the algorithm demonstrates significant improvement in accuracy over standard approaches such as Neural Nets, SimpleMKL, and Random Forest with similar or better computation time.

1. Introduction

Kernel methods for classification and regression (and Support Vector Machines (SVMs) in particular) require selection of a kernel. Kernel Learning (KL) algorithms such as those found in (Xu et al., 2010; Sonnenburg et al., 2010; Yang et al., 2011) automate this task by finding the kernel, $k \in \mathcal{K}$ which optimizes an achievable metric such as the soft margin (for classification). The set of kernels, $k \in \mathcal{K}$, over which the algorithm can optimize, however, strongly influences the performance and robustness of the resulting classifier or predictor.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

To understand how the choice of $\mathcal{K}$ influences performance and robustness, three properties were proposed in (Colbert & Peet, 2020) to characterize the set $\mathcal{K}$ - tractability, density, and universality. Specifically, $\mathcal{K}$ is tractable if $\mathcal{K}$ is convex (or, preferably, a linear variety) - implying the KL problem is solvable using, e.g. (Rakotomamonjy et al., 2008; Jain et al., 2012; Lanckriet et al., 2004; Qiu & Lane, 2005; Gönen & Alpaydın, 2011). The set $\mathcal{K}$ has the density property if, for any $\epsilon > 0$ and any positive kernel, k^* there exists a $k \in \mathcal{K}$ where $\|k - k^*\| \leq \epsilon$. The density property implies the kernel will perform well on untrained data (robustness or generalizability). The set $\mathcal{K}$ has the universal property if any $k \in \mathcal{K}$ is universal - ensuring the classifier/predictor will perform arbitrarily well on large sets of training data.

In (Colbert & Peet, 2020), the Tessellated Kernels (TKs) were shown to have all 3 properties, the first known such class of kernels. This work was based on a general framework for using positive matrices to parameterize positive kernels (as opposed to positive kernel matrices as in (Lanckriet et al., 2004; Qiu & Lane, 2005; Ni et al., 2006)). Unfortunately, however, the algorithms proposed in (Colbert & Peet, 2020) were implemented using SemiDefinite Programming (SDP) (thereby limiting the amount of training data) or using SimpleMKL with a randomized linear basis for the kernels (implying loss of density). Thus, while the algorithms in (Colbert & Peet, 2020) outperformed all other methods (including Neural Nets) as measured by Test Set Accuracy (TSA), the computation times were not competitive. Furthermore, the results in (Colbert & Peet, 2020) did not address the problem of regression.

In this paper, we extend the TK framework proposed in (Colbert & Peet, 2020) to the problem of regression. The KL problem in regression has been studied using SDP in (Qiu & Lane, 2005; Ni et al., 2006) and Quadratic Programming (QP) in e.g. (Rakotomamonjy et al., 2008; Jain et al., 2012). However, neither of these previous works considered a set of kernels with both the tractability and the density property. By generalizing the Tessellated KL framework proposed in (Colbert & Peet, 2020) to the regression problem, we demonstrate significant increases in performance, as measured by Mean Square Error (MSE), and when compared to the results in (Rakotomamonjy et al., 2008; Jain et al., 2012; Qiu & Lane, 2005).

In addition, we show that the SDP-based algorithm (Colbert & Peet, 2020) for classification, and extended here to regression, can be decomposed into primal and dual subproblems, OPT_A and OPT_P - similar to the approach taken in (Rakotomamonjy et al., 2008; Jain et al., 2012). Furthermore, we show that OPT_P (an SDP) admits an analytic solution using the Singular Value Decomposition (SVD) - an approach which allows us to consider higher dimensional feature spaces and more complex TKs. In addition, OPT_A is a convex QP and may be solved efficiently with achieved complexity which scales as $O(m^{2.16})$ where m is the number of data points. We use a two-step algorithm on OPT_A and OPT_P and show that termination at $OPT\_A = OPT\_P$ is equivalent to global optimality. The resulting algorithm, then, does not require the use of SDP and, when applied to several standard test cases, is shown to retain the favorable TSA of (Colbert & Peet, 2020) for classification, while offering improved MSE for regression, and competitive computation times as compared to other KL and deep learning algorithms.

2. Properties of Kernel Sets for KL

Consider a generalized representation of the KL problem, which encompasses both classification and regression where (using the representor theorem (Schölkopf et al., 2001)) the learned function is of the form $f_{\alpha,k}(z) = \sum_{i=1}^{m} \alpha_i k(x_i, z)$.

$$\min_{k\in\mathcal{K}} \min_{\alpha\in\mathbb{R}^m, b} \|f_{\alpha,k}\|^2 + C\sum_{i=1}^{m} l(f_{\alpha,k}, b)_{y_i,x_i} \qquad (1)$$

Here $\|f_{\alpha,k}\| = \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j k(x_i, x_j)$ is the norm in the Reproducing Kernel Hilbert Space (RKHS) and $l(f_{\alpha,k}, b)_{y_i,x_i}$ is the loss function defined for SVM binary classification and SVM regression as $l_c(f_{\alpha,k}, b)_{y_i,x_i}$ and $l_r(f_{\alpha,k}, b)_{y_i,x_i}$, respectively, where

$$l_c(f_{\alpha,k}, b)_{y_i,x_i} = \max\{0, 1 - y_i(f_{\alpha,k}(x_i) - b)\},$$

and

$$l_r(f_{\alpha,k}, b)_{y_i,x_i} = \max\{0, |y_i - (f_{\alpha,k}(x_i) - b)| - \epsilon\}.$$

The properties of the classifier/predictor, $f_{\alpha,k}$, resulting from Optimization Problem 1 will depend on the properties of the set $\mathcal{K}$, which is presumed to be a subset of the convex cone of all positive kernels. To understand how $\mathcal{K}$ influences the tractability of the optimization problem and the resulting fit, we consider three properties of the set, $\mathcal{K}$.

2.1. Tractability

We say a set of kernel functions, $\mathcal{K}$, is tractable if it can be represented using a countable basis.

Definition 1. *The set of kernels $\mathcal{K}$ is **tractable** if there exist a countable set $\{G_i(x, y)\}_i$ such that, for any $k \in \mathcal{K}$, there exists $N_G \in \mathbb{N}$ where $k(x, y) = \sum_{i=1}^{N_G} v_i G_i(x, y)$ for some $v \in \mathbb{R}^{N_G}$.*

Note the $G_i(x, y)$ need not be positive kernel functions. The tractable property is required for the KL problem to be tractable using algorithms for convex optimization.

2.2. Universality

Universal kernel functions always have positive definite (full rank) kernel matrices, implying that for arbitrary data $\{y_i, x_i\}_{i=1}^m$, there exists a function $f(z) = \sum_{i=1}^m \alpha_i k(x_i, z)$, such that $f(x_j) = y_j$ for all $j = 1, .., m$. Conversely, if a kernel is not universal, then there exists a data set $\{x_i, y_i\}_{i=1}^m$ such that for any $\alpha \in \mathbb{R}^m$, there exists some $j \in \{1, \cdots, m\}$ such that $f(y_j) \neq \sum_{i=1}^m \alpha_i k(x_i, x_j)$. This ensures that SVMs using universal kernels can always benefit from additional training data, whereas non-universal kernels may saturate.

Definition 2. *A kernel $k : X \times X \to \mathbb{R}$ is said to be universal on the compact metric space X if it is continuous and there exists an inner-product space $\mathcal{W}$ and feature map, $\Phi : X \to \mathcal{W}$ such that $k(x, y) = \langle\Phi(x), \Phi(y)\rangle_{\mathcal{W}}$ and where the unique Reproducing Kernel Hilbert Space (RKHS), $\mathcal{H} := \{f : f(x) = \langle v, \Phi(x)\rangle,\ v \in \mathcal{W}\}$ with associated norm $\|f\|_{\mathcal{H}} := \inf_v\{\|v\|_{\mathcal{W}} : f(x) = \langle v, \Phi(x)\rangle\}$ is dense in $\mathcal{C}(X) := \{f : X \to \mathbb{R} : f \text{ is continuous}\}$ where $\|f\|_{\mathcal{C}} := \sup_{x\in X} |f(x)|$.*

The following definition extends the universal property to a set of kernels.

Definition 3. *A set of kernel functions $\mathcal{K}$ has the universal property if every kernel function $k \in \mathcal{K}$ is universal.*

2.3. Density

The third property is density which distinguishes the TK class from other sets of kernel functions with the universal property. For instance consider a set containing a single Gaussian kernel function - which is clearly not ideal for kernel learning. The set containing a single Gaussian is tractable (it has only one element) and every member of the set is universal. However, it is not dense.

Considering SVM for classification, the KL problem determines the kernel $k \in \mathcal{K}$ for which we may obtain the maximum separation in the kernel-associated feature space. Increasing this separation distance makes the resulting classifier more robust (generalizable) (Boehmke & Greenwell, 2019). The density property, then, ensures that the resulting KL algorithm will be maximally robust (generalizable) in the sense of separation distance.

Likewise, considering SVMs for regression, the KL problem finds the kernel $k \in \mathcal{K}$ which permits the "flattest" (Smola & Schölkopf, 2004) function in feature space. In this case, the density property ensures that the resulting KL algorithm will be maximally robust (generalizable) in the sense of flatness.

These arguments motivate the following definition of the pointwise density property.

Definition 4. *The set of kernels $\mathcal{K}$ is said to be **pointwise dense** if for any positive kernel, k^*, any set of data $\{x_i\}_{i=1}^m$, and any $\epsilon > 0$, there exists $k \in \mathcal{K}$ such that $\|k(x_i, x_j) - k^*(x_i, x_j)\| \leq \epsilon$.*

3. A General Framework for Representation of Tractable Kernel Sets

Here we define a framework for constructing classes of tractable positive kernel functions and illustrate this approach on the class of General Polynomial Kernels.

Lemma 5. *Let N be any bounded measurable function $N : X \times Y \to \mathbb{R}^q$ on compact X and Y. If we define*

$$\mathcal{K} := \left\{ k \mid k(x,y) = \int_X N(z,x)^T P N(z,y) dz, P \geq 0 \right\} \quad (2)$$

then any $k \in \mathcal{K}$ is a positive kernel function and $\mathcal{K}$ is tractable.

For a given N, the map $P \mapsto k$ is linear. Specifically,

$$k(x,y) = \sum_{i=1}^{q} \sum_{j=1}^{q} P_{i,j} G_{i,j}(x,y) \text{ where,}$$

$$G_{i,j}(x,y) = \int_X N_i(z,x) N_j(z,y) dz,$$

and thus by Definition 1 $\mathcal{K}$ is tractable.

In Subsection 3.1 we apply this framework to obtain Generalized Polynomial Kernels. In Subsection 4.1, we use the framework to obtain the TK class.

3.1. The Class of General Polynomial Kernels is Tractable

The class of General Polynomial Kernels (GPKs) is defined as the set of all polynomials ($\mathbb{R}[x,y]$), each of which is a positive kernel.

$$\mathcal{K}_P := \{k \in \mathbb{R}[x,y] \ : \ k \text{ is a positive kernel}\} \quad (3)$$

The GPK class is not universal, but is tractable, as per the following lemma.

Lemma 6. *$\mathcal{K}_P$ is tractable.*

Proof. See supplementary material for the proof. □

This lemma implies that a representation of the form of Equation (2) is necessary and sufficient for a GPK to be positive. For convenience, we denote the set of GPK kernels of degree d or less as follows (Recht, 2006).

$$\mathcal{K}_P^d := \{k \ : \ k(x,y) = Z_d(x)^T P Z_d(y) \ : \ P \geq 0\} \quad (4)$$

where $Z_d : \mathbb{R}^n \to \mathbb{R}^q$ is the vector of monomials of degree d or less where $q = \binom{d+n}{d}$.

4. TKs: Tractable, Dense and Universal

In this section, we define the class of TK kernels and show it is tractable, dense, and universal.

4.1. Tessellated Kernels (TKs)

Again, let $Z_d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^q$ be the vector of monomials of degree d. Define $\mathbf{I}$, the indicator function for the positive orthant, and the following choice of $N : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{2q}$ as

$$\mathbf{I}(z) = \begin{cases} 1 & z \geq 0 \\ 0 & \text{otherwise,} \end{cases} \quad \text{and}$$

$$N_T^d(z,x) = \begin{bmatrix} Z_d(z,x)\mathbf{I}(z-x) \\ Z_d(z,x)\mathbf{I}(x-z) \end{bmatrix} \quad (5)$$

where $z \geq 0$ means $z_i \geq 0$ for all i.

We now define the set of TK kernels for $a < b \in \mathbb{R}^n$ as

$$\mathcal{K}_T^d := \left\{ k : k(x,y) = \int_a^b N_T^d(z,x)^T P N_T^d(z,y) dz, P \geq 0 \right\}, \quad (6)$$

and where $\mathcal{K}_T := \{k \ : \ k \in \mathcal{K}_T^d, \ d \in \mathbb{N}\}$ and P is a symmetric matrix of size $2\binom{d+n}{d}$.

Kernels in the TK class are "Tessellated" in the sense that each datapoint defines a vertex which bisects each dimension of the domain of the resulting classifier/predictor - resulting in a tessellated partition of the feature space.

4.2. The Set of TK Kernels is Tractable

The class of TK kernels is prima facie in the form of Eqn. (2) in Lemma 5 and hence is tractable.

However, we will expand on this result by specifying the basis for the set of TK kernels, which will then be used in Section 5.

Corollary 7. *Suppose that $a < b \in \mathbb{R}^n$, and $d \in \mathbb{N}$. We define the finite set $D_d := \{(\delta, \lambda) \in \mathbb{N}^{2n} : \|(\delta,\lambda)\|_1 \leq d\}$. Let $\{[\delta_i, \gamma_i]\}_{i=1}^q \subseteq D_d$ be some ordering of D_d and define $Z_d(x,z)_j = x^{\delta_j} z^{\gamma_j}$ where $z^{\delta_j} x^{\gamma_j} := \prod_{i=1}^n z_i^{\delta_j,i} x_i^{\gamma_j,i}$. Now let k be as defined in Eqn. (2) for some $P > 0$ and where N is as defined in Eqn. (5). If we partition $P = \begin{bmatrix} Q & R \\ R^T & S \end{bmatrix}$ then we have,*

$$k(x,y) = \sum_{i,j=1}^{q} Q_{i,j} g_{i,j}(x,y) + R_{i,j} t_{i,j}(x,y) + R_{i,j}^T t_{i,j}(y,x) + S_{i,j} h_{i,j}(x,y)$$

where $g_{i,j}, t_{i,j}, h_{i,j} : \mathbb{R}^{2n} \to \mathbb{R}$ are defined as

$$g_{i,j}(x,y) := x^{\delta_i} y^{\delta_j} T(p^*(x,y), b, \gamma_{i,j} + \mathbf{1}),$$

$$t_{i,j}(x,y) := x^{\delta_i} y^{\delta_j} T(x, b, \gamma_{i,j} + \mathbf{1}) - g_{i,j}(x,y), \text{ and}$$

$$h_{i,j}(x,y) := x^{\delta_i} y^{\delta_j} T(a, b, \gamma_i + \gamma_j + \mathbf{1}) - g_{i,j}(x,y) - t_{i,j}(x,y) - t_{i,j}(y,x),$$

where $\mathbf{1} \in \mathbb{N}^n$ is the vector of ones, $p^* : \mathbb{R}^{2n} \to \mathbb{R}^n$ is defined elementwise as $p^*(x,y)_i = \max\{x_i, y_i\}$, and $T : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{N}^n \to \mathbb{R}$ is defined as

$$T(x,y,\zeta) = \prod_{j=1}^{n} \left(\frac{y_j^{\zeta_j}}{\zeta_j} - \frac{x_j^{\zeta_j}}{\zeta_j} \right).$$

The proof of Corollary 7 can be found in (Colbert & Peet, 2020).

4.3. The TK Class is Dense

The density property differentiates the set of TK kernels from other sets of kernel functions (e.g. a linear combination of Gaussian kernels of fixed bandwidths).

From (Colbert & Peet, 2020) we have that the set of TK kernels satisfies the pointwise density property.

Theorem 8. *For any positive semidefinite kernel matrix K^* and any finite set $\{x_i\}_{i=1}^m$, there exists a $d \in \mathbb{N}$ and $k \in \mathcal{K}_T^d$ such that if $K_{i,j} = k(x_i, x_j)$, then $K = K^*$.*

4.4. TK Kernels are Universal

Finally we discuss the universality property of the class of TK kernels which ensures that every TK function can fit the training data well.

The following theorem from (Colbert & Peet, 2020) shows that any TK kernel with $P > 0$ is necessarily universal.

Theorem 9. *Suppose k is as defined in Eqn. (2) for some $P > 0$, $d \in \mathbb{N}$ and N as defined in Eqn. (5). Then k is universal.*

This theorem implies that even if we use the subset of TK kernels defined by $d = 0$, this subset is still universal.

5. An Efficient Algorithm for KL in Classification and Regression using TKs

In this section, we formulate the KL optimization problem for both classification and regression and represent this as a minimax saddle point problem. This formulation enables a decomposition into convex primal and dual sub-problems, $OPT\_A(P)$ and $OPT\_P(\alpha)$ with no duality gap. We then consider the Frank-Wolfe algorithm and show using Danskin's Theorem that the gradient step can be efficiently computed using the primal and dual sub-problems. Finally, we propose efficient algorithms for computing $OPT\_A(P)$ and $OPT\_P(\alpha)$: in the former case using an efficient SMO algorithm for convex QP and in the latter case, using an analytic solution based on the SVD.

5.1. Primal-Dual Decomposition

For convenience, we define the feasible sets for the sub-problems as

$$\begin{aligned}\mathcal{X} &:= \{P \in \mathbb{R}^{q\times q} \ : \ \text{trace}(P) = q, \ P > 0\}\\ \mathcal{Y}_c &:= \{\alpha \in \mathbb{R}^m \ : \ \sum_{i=1}^m \alpha_i y_i = 0, \ 0 \le \alpha_i \le C\},\\ \mathcal{Y}_r &:= \{\alpha \in \mathbb{R}^m \ : \ \sum_{i=1}^m \alpha_i = 0, \ \alpha_i \in [-C, C]\}.\end{aligned}$$

In this section, we typically use the generic form $\mathcal{Y}_*$ to refer to either $\mathcal{Y}_c$ or $\mathcal{Y}_r$ depending on whether the algorithm is being applied to the classification or regression problem. To define the objective function we use $\lambda(\alpha, P)$ to indicate

$$\lambda(\alpha, P) := -\frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m \alpha_i\alpha_j \int_a^b N_T^d(z, x_i)^T P N_T^d(z, y_j)dz, \tag{7}$$

where N_T^d are as defined in Eqn. (5). Additionally, we have $\kappa_c(\alpha) := \sum_{i=1}^m \alpha_i$ and

$$\kappa_r(\alpha) := -\epsilon\sum_{i=1}^m |\alpha_i| + \sum_{i=1}^m y_i\alpha_i.$$

where, again, we use $\kappa_* = \kappa_c$ for classification and $\kappa_* = \kappa_r$ for regression.

The KL optimization problem (OPT) for TK kernels is now defined as the following minimax saddle point optimization problem.

$$OPT_P := \min_{P\in\mathcal{X}} \max_{\alpha\in\mathcal{Y}_*} \quad \lambda(e_* \odot \alpha, P) + \kappa_*(\alpha), \tag{8}$$

where $\odot$ indicates elementwise multiplication, $e_c = y$ (vector of labels) for classification, and $e_r = \mathbf{1}_m$ (vector of ones) for regression.

Minimax Duality To find the dual of the KL optimization problem, we formulate two sub-problems:

$$OPT\_A(P) := \max_{\alpha\in\mathcal{Y}_*} \lambda(e_* \odot \alpha, P) + \kappa_*(\alpha) \tag{9}$$

and

$$OPT\_P(\alpha) := \min_{P\in\mathcal{X}} \ \lambda(e_* \odot \alpha, P) + \kappa_*(\alpha). \tag{10}$$

Now, we have that

$$OPT_P = \min_{P\in\mathcal{X}} \ OPT\_A(P)$$

and its dual is

$$\begin{aligned}OPT_D &= \max_{\alpha\in\mathcal{Y}_*} OPT\_P(\alpha) \\ &= \max_{\alpha\in\mathcal{Y}_*} \min_{P\in\mathcal{X}} \quad \lambda(e_* \odot \alpha, P) + \kappa_*(\alpha).\end{aligned} \tag{11}$$

The following lemma states that there is no duality gap between OPT_P and OPT_D - a property we will use in our termination criterion.

Lemma 10. *$OPT_P = OPT_D$. Furthermore, $\{\alpha^*, P^*\}$ solve OPT_P if and only if $OPT\_P(\alpha^*) = OPT\_A(P^*)$.*

Proof. See supplementary material for the proof. □

Finally, we note that $OPT\_A(P)$ is convex with respect to P - a property we will use in Thm. 14.

Lemma 11. *Let $OPT\_A(P)$ be as defined in 9. Then, the function $OPT\_A(P)$ is convex with respect to P.*

Proof. See supplementary material for the proof. □

5.2. Primal-Dual Frank-Wolfe Algorithm

For an optimization problem of the form

$$\min_{S\in\mathcal{X}} f(S),$$

where $\mathcal{X}$ is a convex subset of matrices and $\langle\cdot,\cdot\rangle$ is the Frobenius matrix inner product, the Frank-Wolfe (FW) algorithm is defined as in Algorithm 1.

Algorithm 1 The Frank-Wolfe Algorithm for Matrices.

`Initialize` P_0 as any point in $\mathcal{X}$.;
Step 1: $S_k = \arg\min_{S\in\mathcal{X}} \langle \nabla_Q f(Q)|_{Q=P_k}, S\rangle$
Step 2: $\gamma_k = \arg\min_{\gamma\in[0,1]} f(P_k + \gamma(S_k - P_k))$
Step 3: $P_{k+1} = P_k + \gamma_k (S_k - P_k)$, $k = k + 1$, return to step 1.

In our case, we have $f(Q) = OPT\_A(Q)$ so that

$$OPT_P = \min_{P\in\mathcal{X}} OPT\_A(P).$$

Unfortunately, implementation of the FW algorithm requires us to compute $\nabla_Q OPT\_A(Q)|_{Q=P_k}$ at each iteration. Fortunately, as shown in Subsections 5.3 and 5.4, we may efficiently compute the sub-problems OPT_A and OPT_P. Furthermore, in Theorem 13, we will show that these sub-problems can be used to efficiently compute the gradient $\nabla_Q OPT\_A(Q)|_{Q=P_k}$ - allowing for an efficient implementation of the FW algorithm. Theorem 13 uses Danskin's theorem as stated below. (Bertsekas et al., 1998).

Proposition 12 (Danskin's Theorem (Bertsekas et al., 1998)). *Let $\mathcal{Y} \subset \mathbb{R}^m$ be a compact set, and let $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be continuous such that $\phi(\cdot, \alpha) : \mathcal{X} \to \mathbb{R}$ is convex for each $\alpha \in \mathcal{Y}$. Then if,*

$$\mathcal{Y}_0(P) = \left\{ \bar{\alpha} \;\middle|\; \phi(P, \bar{\alpha}) = \max_{\alpha \in \mathcal{Y}} \phi(P, \alpha) \right\}.$$

consists of only one unique point, $\bar{\alpha}$, and $\phi(\cdot, \bar{\alpha})$ is differentiable at P then $f(P) = \max_{\alpha \in \mathcal{Y}} \phi(P, \alpha)$ is differentiable at P and

$$\nabla_P f(P) = \nabla_P \phi(P, \bar{\alpha}),$$

where $\nabla_P \phi(P, \bar{\alpha})$ is the vector with coordinates

$$\frac{\partial \phi(P, \bar{\alpha})}{\partial P_i}, \quad i = 1, ..., n.$$

Lemma 13. *If OPT_A and OPT_P are as defined in Eqns. (9) and (10), then for any $P_k \geq 0$, we have*

$$\arg\min_{S \in \mathcal{X}} \langle \nabla_Q OPT\_A(Q)|_{Q=P_k}, S \rangle = \arg OPT\_P(\arg OPT\_A(P_k)).$$

Proof. For simplicity, we define $D(\alpha)$ as in Eqn. (12) such that $\lambda(e_* \odot \alpha, P) := \langle D(\alpha), P \rangle$. Now, since $\lambda(\alpha, P)$ is strictly convex in α, for any $P_k > 0$, $OPT\_A(P_k)$ has a unique solution and hence we have by Danskin's Theorem that

$$\begin{aligned}
&\arg\min_{S \in \mathcal{X}} \left\langle \nabla_Q OPT_A(Q)|_{Q=P_k}, S \right\rangle \\
&= \arg\min_{S \in \mathcal{X}} \left\langle \nabla_Q \left[\max_{\alpha \in \mathcal{Y}_*} (\langle D(\alpha), Q \rangle + \kappa_*(\alpha)) \right]_{Q=P_k}, S \right\rangle \\
&= \arg\min_{S \in \mathcal{X}} \left\langle \nabla_Q \left[\langle D(\bar{\alpha}), Q \rangle + \kappa_*(\bar{\alpha}) \right]_{Q=P_k}, S \right\rangle
\end{aligned}$$

where $\bar{\alpha} = \arg OPT\_A(P_k)$. Hence,

$$\begin{aligned}
&\arg\min_{S \in \mathcal{X}} \left\langle \nabla_Q \left[\langle D(\bar{\alpha}), Q \rangle + \kappa_*(\bar{\alpha}) \right]_{Q=P_k}, S \right\rangle \\
&= \arg\min_{S \in \mathcal{X}} \left\langle \nabla_Q \left[\langle D(\bar{\alpha}), Q \rangle \right]_{Q=P_k}, S \right\rangle \\
&= \arg\min_{S \in \mathcal{X}} \left\langle D(\bar{\alpha}), S \right\rangle \\
&= \arg OPT_P(\bar{\alpha}) \\
&= \arg OPT_P(\arg OPT_A(P_k)).
\end{aligned}$$

□

We now propose the efficient implementation of the FW algorithm, as defined in Algorithm 2, based on efficient algorithms for computing OPT_A and OPT_P as will be defined in Subsections 5.3 and 5.4.

Algorithm 2 An Efficient FW Algorithm for TKL. Note that the stopping criterion is defined using the duality gap $OPT\_P(\alpha_k) - OPT\_A(P_k) > 0$, which is equivalent to the stopping criterion used in the standard FW algorithm.

`Initialize` $P_0 = I$, $k = 0$, $\alpha_0 = OPT\_A(P_0)$;
while $OPT\_P(\alpha_k) - OPT\_A(P_k) \geq \epsilon$ **do**
  Step 1a: $\alpha_k = \arg OPT\_A(P_k)$
  Step 1b: $S_k = \arg OPT\_P(\alpha_k)$
  Step 2: $\gamma_k = \arg\min_{\gamma \in [0,1]} OPT\_A(P_k + \gamma(S_k - P_k))$
  Step 3: $P_{k+1} = P_k + \gamma_k(S_k - P_k)$, $k = k + 1$
end while

In the following theorem, we use convergence properties of the FW algorithm to show that Algorithm 2 has worst-case linear convergence. Note that we use an primal-dual accelerator for quadratic convergence when higher accuracy is required, as defined in Subsection 5.5.

Theorem 14. *Algorithm 2 returns iterates P_k and α_k such that, $|\lambda(\alpha_k, P_k) + \kappa_*(\alpha_k) - OPT_P| < O(\frac{1}{k})$.*

Proof. If we define $f = OPT\_A$, then Theorem 13 shows that f is differentiable and, if the P_k satisfy Algorithm 2, that the P_k also satisfy Algorithm 1. In addition, Lemma 11 shows that $f(Q) = OPT\_A(Q)$ is convex in Q. It has been shown in, e.g. (Jaggi, 2013), that if $\mathcal{X}$ is convex and compact and $f(Q)$ is convex and differentiable on $Q \in \mathcal{X}$, then the FW Algorithm produces iterates P_k, such that, $f(P_k) - f(P^*) < O(\frac{1}{k})$ where

$$f(P^*) = \min_{P \in \mathcal{X}} f(P) = \min_{P \in \mathcal{X}} OPT\_A(P) = OPT_P.$$

Finally, we note that

$$\begin{aligned}
&\lambda(\alpha_k, P_k) + \kappa_*(\alpha_k) \\
&= \lambda(\arg OPT_A(P_k), P_k) + \kappa_*(\arg OPT_A(P_k)) \\
&= \max_{\alpha \in \mathcal{Y}_*} \lambda(\alpha, P_k) + \kappa_*(\alpha) = OPT_A(P_k) = f(P_k)
\end{aligned}$$

which completes the proof. □

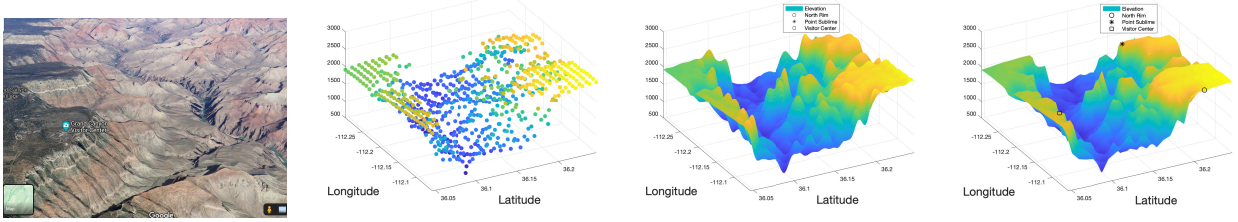
In the following subsections, we provide efficient algorithms for computing the sub-problems OPT_A and OPT_P.

5.3. Step 1, Part A: Solving $OPT\_A(P)$

For a given $P > 0$, $OPT\_A(P)$ is a convex Quadratic Program (QP). General purpose QP solvers have a worst-case complexity which scales as $O(m^3)$ (Ye & Tse, 1989) where, when applied to OPT_A, m becomes the number of samples. This computational complexity may be improved, however, by noting that OPT_A is compatible with the representation defined in (Chang & Lin, 2011) for QPs derived from SVM. In this case, the algorithm in LibSVM (Chang & Lin, 2011) can reduce the computational burden somewhat. This improved performance is illustrated in Figure 3 where we observe the achieved complexity scales as $O(m^{2.1})$. Note that for the 2-step algorithm proposed in this manuscript, solving the QP in $OPT\_A(P)$ is significantly slower that solving the Singular Value Decomposition (SVD) required for $OPT\_P(\alpha)$, which is defined in the following subsection. However, the achieved complexity of $O(m^{2.1})$ is also significantly faster than solving the large SDP, as described in (Lanckriet et al., 2004), (Qiu & Lane, 2005), and (Colbert & Peet, 2020). This complexity comparison will be further discussed in Section 6.

5.4. Step 1, Part B: Solving $OPT\_P(\alpha)$

For a given α, $OPT\_P(\alpha)$ is an SDP. Fortunately, however, this SDP is structured so as to admit an analytic solution using the SVD. To solve $OPT\_P(\alpha)$ we minimize $\lambda(e_* \odot \alpha, P)$ from Eq. (7) which, as per Corollary 7, is linear in P and can be formulated as



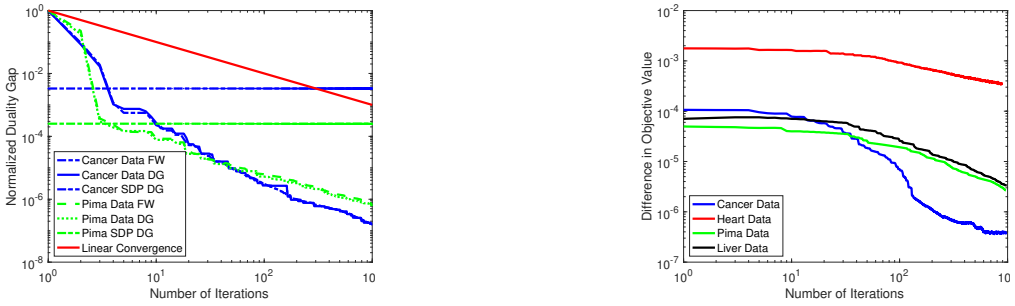
(a) An image from Google Maps of a section of the Grand Canyon corresponding to (36.04, -112.05) latitude and (36.25, -112.3) longitude.

(b) Elevation data ($m = 750$) from (Becker et al., 2009) for a section of the Grand Canyon between (36.04, -112.05) latitude and (36.25, -112.3) longitude.

(c) Predictor using a hand-tuned Gaussian kernel trained on the elevation data in (b). The Gaussian predictor poorly represents the sharp edge at the north and south rim.

(d) Predictor from Algorithm 2 trained on the elevation data in (b). The TK predictor accurately represents the north and south rims of the canyon.

Figure 1. Subfigure (a) shows an 3D representation of the section of the Grand Canyon to be fitted. In (b) we plot elevation data of this section of the Grand Canyon. In (c) we plot the predictor for a hand-tuned Gaussian kernel. In (d) we plot the predictor from Algorithm 2 for $d = 2$.



(a) The Frank-Wolfe error gap and duality gap for 1000 iterations of Algorithm 2, applied to two different classification data sets.

(b) The difference in objective value between Algorithm 2 with and without the 2nd stage Primal-Dual Booster after switching.

Figure 2. In (a) we plot the primal-dual gap from Algorithm 2 without the 2nd stage Primal-Dual Booster, and in (b) we plot the difference between the objective function when we switch to the 2nd stage Booster (after the threshold step length has been reached).

$$OPT\_P(\alpha) := \min_{\substack{P\in\mathbb{R}^{q\times q}\\ \text{trace}(P)=q\\ P\succ 0}} \lambda(e_* \odot \alpha, P) := \min_{\substack{P\in\mathbb{R}^{q\times q}\\ \text{trace}(P)=q\\ P\succ 0}} \langle D(\alpha), P\rangle$$

where,

$$D_{i,j}(\alpha) = \sum_{k,l=1}^{m} (\alpha_k y_k) G_{i,j}(x_k, x_l)(\alpha_l y_l) \qquad (12)$$

$$G_{i,j}(x,y) := \begin{cases} g_{i,j}(x,y) & \text{if } i \le \frac{q}{2}, j \le \frac{q}{2} \\ t_{i,j}(x,y) & \text{if } i \le \frac{q}{2}, j > \frac{q}{2} \\ t_{i,j}(y,x) & \text{if } i > \frac{q}{2}, j \le \frac{q}{2} \\ h_{i,j}(x,y) & \text{if } i > \frac{q}{2}, j > \frac{q}{2} \end{cases}$$

and g, t and h can be found in Corollary 7.

The following theorem gives an analytic solution for OPT_P using the SVD.

Theorem 15. *For a given α, denote symmetric $D_\alpha := D(\alpha) \in \mathbb{R}^{q\times q}$ as defined in Eqn. (12) and let $D_\alpha = V\Sigma V^T$ be its SVD. Let v be the right singular vector corresponding to the minimum singular value of D_α. Then $P^* = qvv^T$ solves $OPT\_P(\alpha)$.*

Proof. Recall $OPT\_P(\alpha)$ has the form

$$\min_{P\in\mathbb{R}^{q\times q}} \langle D_\alpha, P\rangle \quad \text{s.t. } P \succeq 0,\ \text{trace}(P) = q.$$

Denote the minimum singular value of D_α as $\sigma_{\min}(D_\alpha)$. Then for any feasible $P \in \mathcal{X}$, by (Fang et al., 1994) we have

$$\langle D_\alpha, P\rangle \ge \sigma_{\min}(D_\alpha)\text{trace}(P) = \sigma_{\min}(D_\alpha) q.$$

Now consider $P = qvv^T \in \mathbb{R}^{q\times q}$. P is feasible since $P \succeq 0$, and $\text{trace}(P) = q$. Furthermore,

$$\langle D_\alpha, P\rangle = q\,\text{trace}(V\Sigma V^T vv^T) = q\,\text{trace}(v^T V\Sigma V^T v)$$
$$= q\,\sigma_{\min}(D_\alpha)$$

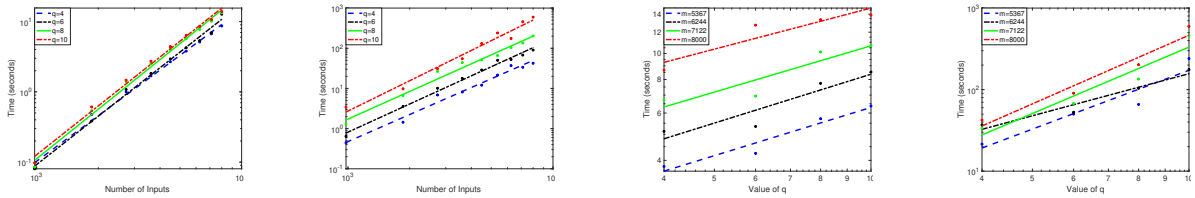
as desired. □

Note that the size, q, of D_α in $OPT\_P(\alpha)$ scales with the number of features, but not the number of samples (m). As a result, we observe that the OPT_P step of Algorithm 2 is significantly faster than the OPT_A step.

5.5. 2nd Stage Primal-Dual Booster

Implementation and numerical convergence analysis, included in Section 7, indicates that Algorithm 2 will often significantly exceeds linear convergence for the first several iterations. However, the convergence rate for 10+ iterations is consistently linear. While 10 iterations may be sufficient accuracy for most applications, occasionally we may require additional accuracy and for this case, we have implemented an Accelerated Primal-Dual (APD) algorithm based on the minimax momentum-style algorithms proposed in (Hamedani & Aybat, 2020), which are proven to have worst-case quadratic performance.

Because this APD algorithm is significantly slower for the first several iterations, it is only used if the step size in the Algorithm 2 falls below a predefined threshhold. Details of



(a) Numerical complexity analysis of TKL for classification versus m.

(b) Numerical complexity analysis of TKL for regression versus m.

(c) Numerical complexity analysis of TKL for classification versus q.

(d) Numerical complexity analysis of TKL for regression versus q.

Figure 3. In (a) and (b) we find log scale plots of the time taken to execute FW TKL for $P \in \mathbb{R}^{q \times q}$. The line of best linear fit is included for reference. In (c) and (d) we find log scale plots of the time taken to optimize TKL as a function of q for four different values of m.

this secondary algorithm are included in the supplementary material. While the transition to 2nd stage APD is clearly a heuristic, the numerical convergence studies in Section 6 show that this "booster" algorithm significantly reduces computation time when low error tolerances are used.

6. Numerical Convergence and Scalability

Here we consider the convergence properties and computational complexity of Algorithm 2.

6.1. Convergence Properties

To study the convergence properties of Algorithm 2, in Figure 2(a), we plot the duality gap between $OPT\_A(P_k)$ and $OPT\_P(\alpha_k)$ as a function of iteration number for the CANCER and PIMA datasets. Note that the typical FW error metric is based on a bound on the primal-dual gap and in practice we observe that these metrics are almost identical - as illustrated in Figure 2(a). Also included in Figure 2(a) is the duality gap in the SDP implementation of the TKL algorithm, as obtained from (Colbert & Peet). We do not include iterations of the SDP primal-dual algorithm as the complexity of these iterations is not comparable to the proposed algorithm. For reference, Fig. 2(a) also includes a plot of theoretical worst-case linear convergence. Finally, in Fig. 2(b), we study the benefits of the "boosted" FW-ADP algorithm for 4 datasets.

These figures show that in all cases, the FW TKL algorithm in practice achieves faster-than-linear convergence for several iterations and then linear convergence and that the second stage booster causes a significant decrease in the stated error metric. Finally, we note that after 100 iterations, the duality gap of the FW TKL algorithm is lower than that of the SDP-based TKL implementation.

6.2. Computational Complexity

In Figures 3, we plot the computation time of the FW TKL algorithm for both classification and regression on a desktop PC with an Intel i7-5960X CPU at 3.00 GHz and 128 Gb of RAM as a function of m and q, where m is the number of samples used to learn the TK kernel function and the size of P as $q \times q$ (so that q is a function of the number of features and the degree of the monomial basis Z_d). The data set for these plots is Combined Cycle Power Plant (CCPP) in (Tüfekci, 2014; Kaya et al., 2012), containing 4 features and $m = 9568$ samples. In the case of classification, labels with value greater than or equal to the median of the output were relabeled as 1, and those less than the median were relabeled as -1. To enable comparison with SimpleMKL, we use an identical stopping criterion of 10^{-2}. Figures 3(a-d) demonstrate that the complexity of Algorithm 2 scales as approximately $O(m^{2.28}q^{0.57})$ for classification and $O(m^{2.34}q^{2.40})$ for regression. These results are significantly lower with respect to m than the value of $O(m^{2.6}q^{1.9})$ reported in (Colbert & Peet, 2020) for binary classification using the SDP implementation. Aside from improved scalability, the overall time required for Algorithm 2 is significantly reduced when compared with the SDP algorithm in (Colbert & Peet, 2020), improving by two orders of magnitude in some cases. This is illustrated for classification using four data sets in Table 1. This improved complexity is likely due to the lower overhead associated with QP and the SVD.

7. Accuracy of the New TK Kernel Learning Algorithm for Regression

In this section, we compare the accuracy of the classification and regression solutions obtained from the FW TKL algorithm to the SimpleMKL, Neural Networks, and Random Forest algorithms. Specifically, we use the following implementations of these algorithms.

[TKL] Algorithm 2 with $d = 1$, $\epsilon = .1$ and we scale the data so that $x_i \in [0, 1]^n$, and then select $[a, b] = [0 - \delta, 1 + \delta]^n$, where $\delta \geq 0$ and C are chosen by 2-fold cross-validation;

[SMKL] SimpleMKL (Rakotomamonjy et al., 2008) with a standard selection of Gaussian and polynomial kernels with bandwidths arbitrarily chosen between .5 and 10 and polynomial degrees one through three - yielding approximately $13(n+1)$ kernels. We set $\epsilon = .1$ as in TKL and C is chosen by 2-fold cross-validation;

[NNet] A neural network with 3 hidden layers of size 50 using MATLABs (`patternnet` for classification and `feedforwardnet` for regression) implementation and stopped learning after the error in a validation set decreased sequentially 50 times.

[RF] The Random Forest algorithm (Breiman, 2004) as implemented on the scikit-learn python toolbox (Pedregosa et al., 2011) for classification and regression. We select

Table 1. We report the mean computation time (in seconds), along with standard deviation, for 30 trials comparing the SDP algorithm in (Colbert & Peet, 2020) and Algorithm 2. All tests are run on an Intel i7-5960X CPU at 3.00 GHz with 128 Gb of RAM.

Method	Liver	Cancer	Heart	Pima
SDP	95.75 ± 2.68	636.17 ± 25.43	221.67 ± 29.63	1211.66 ± 27.01
Algorithm 2	0.12 ± 0.03	0.41 ± 0.23	4.71 ± 1.15	0.80 ± 0.36

Table 2. Comparison of [TKL], [SMKL], [RF] and [NN] on 6 datasets. For each data set, the first column indicates: the number of features, n; the number of training samples, m; and the number of test samples, m_t, for each division. TSA is percentage of test samples correctly labeled and MSE is Mean Square Error in predicted output vs. true output in the test samples. All regression tests are run on a desktop with Intel i7-5960X CPU at 3.00 GHz and with 128 Gb of RAM. All classifications tests are run on a desktop with Intel i7-4960X CPU at 3.60 GHz and with 64 GB of RAM. N/A denotes that the indicated algorithm terminated unexpectedly due to memory (RAM) depletion.

Regression	Method	Error	Time (s)	Classification	Method	Accuracy (%)	Time (s)
Gas Turbine	TKL	0.23 ± 0.01	13580 ± 2060	Hill Valley	TKL	86.70 ± 5.49	86.78 ± 48.18
$n = 11$	SMKL	N/A	N/A	$n = 100$	SMKL	51.23 ± 3.55	2.81 ± 2.83
$m = 30000$	NNet	0.27 ± 0.03	1172 ± 100	$m = 1000$	NNet	70.00 ± 4.79	3.79 ± 1.75
$m_t = 6733$	RF	0.38 ± 0.02	16.44 ± 0.57	$m_t = 212$	RF	56.04 ± 3.27	0.75 ± 0.33
Airfoil	TKL	1.41 ± 0.44	49.87 ± 4.29	Shill Bid	TKL	99.76 ± 0.08	23.66 ± 2.63
$n = 5$	SMKL	4.33 ± 0.79	617.82 ± 161.63	$n = 9$	SMKL	97.71 ± 0.32	81.04 ± 13.11
$m = 1300$	NNet	6.06 ± 3.84	211.86 ± 41.04	$m = 5000$	NNet	98.64 ± 0.86	$3.56 \pm .60$
$m_t = 203$	RF	2.36 ± 0.42	0.91 ± 0.20	$m_t = 1321$	RF	99.35 ± 0.14	0.78 ± 0.36
CCPP	TKL	10.57 ± 0.82	626.76 ± 456.05	Abalone	TKL	84.61 ± 1.60	17.63 ± 3.77
$n = 4$	SMKL	13.93 ± 0.78	13732 ± 1490	$n = 8$	SMKL	83.13 ± 1.06	350.41 ± 175.15
$m = 8000$	NNet	15.20 ± 1.00	305.71 ± 9.25	$m = 4000$	NNet	84.70 ± 1.82	4.68 ± 0.64
$m_t = 1568$	RF	10.75 ± 0.70	1.65 ± 0.19	$m_t = 677$	RF	84.11 ± 1.33	0.98 ± 0.21

between 50 and 650 trees (in 50 tree intervals) using 2-fold cross-validation.

These algorithms were applied to 3 classification and 3 regression datasets. These datasets were chosen arbitrarily from (Dua & Graff, 2017) to contain a variety of number of features and number of samples. No other datasets were tested for relative performance and datasets were not "pre-screened". In both classification and regression, our accuracy metric uses 5 random divisions of the data into test sets (m_t samples $\cong 20\%$ of data) and training sets (m samples $\cong 80\%$ of data). For regression, the training data is used to learn the kernel and predictor. The predictor is then used to predict the test set outputs. The Mean Squared Error (MSE) of these predictions is listed in Table 2 along with standard deviation. Likewise for classification, the training data was used to obtain the kernel and classifier. The classifier was then used to predict the binary label. The percentage of correct labels is listed as Test Set Accuracy (TSA) in Table 2, along with standard deviation.

From Table 2, we see that the TKL algorithm significantly outperforms a carefully selected sample of state-of-the-art machine learning algorithms in average accuracy, with improvements in accuracy exceeding the standard deviation in 4 of 6 datasets. We note, however that average accuracy score of the NNET algorithm for classification improved on the TKL score for the Abalone dataset by $.09\%$, which is statistically insignificant, given the mean standard deviation of 1.5% for all algorithms on that dataset. The most significant increases in accuracy performance were on the Hill and Airfoil datasets, where TKL outperformed SimpleMKL at 1.41% vs 4.33% and at 86.70% vs. 51.23% respectively. These dramatic improvements may be due to some property of the data which makes it unsuitable for Gaussian kernels. For computation time, RF was uniformly fastest, as expected. SimpleMKL was consistently slowest (except for the Hill dataset, on which the accuracy was rather poor). Compared with NNET, the TKL algorithm was faster only on the Airfoil dataset, which is surprising, considering the significant accuracy performance improvement of TKL on that dataset.

To further illustrate the importance of density property and the TKL framework for practical regression problems, we used elevation data from (Becker et al., 2009) to learn a degree 2 TK kernel and associated SVM predictor representing the surface of the Grand Canyon in Arizona. This data set is particularly challenging due to the variety of geographical features. The result from the TKL algorithm can be seen in Figure 1(d) where we see that the regression surface visually resembles a photograph of this terrain, avoiding the artifacts present in Gaussian-based methods.

8. Conclusion

We have extended the TK kernel learning framework to regression problems and proposed an efficient algorithm for TK kernel learning based on a primal-dual decomposition combined with a FW type algorithm. The set of TK kernels is tractable, dense, and universal, implying that KL algorithms based on TK kernels are more robust than existing machine learning algorithms, an assertion supported by numerical testing on 6 relatively large and randomly selected datasets, testing which yielded uniform increases in accuracy of FW TKL over state-of-the-art alternatives.

Acknowledgements

References

Becker, J., Sandwell, D., Smith, W., Braud, J., Binder, B., Depner, J., Fabre, D., Factor, J., Ingalls, S., Kim, S., et al. Global bathymetry and elevation data at 30 arc seconds resolution: Srtm30_plus. Marine Geodesy, 32 (4):355–371, 2009.

Bertsekas, D., Hager, W., and Mangasarian, O. Nonlinear programming. 1998.

Boehmke, B. and Greenwell, B. Hands-On Machine Learning with R. CRC Press, 2019.

Breiman, L. Random forests. Machine Learning, 45:5–32, 2004.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Colbert, B. and Peet, M. TKL website. http://control.asu.edu/TKL. Accessed: 2021-01-01.

Colbert, B. and Peet, M. A convex parametrization of a new class of universal kernel functions. Journal of Machine Learning Research, 21(45):1–29, 2020. URL http://jmlr.org/papers/v21/19-594.html.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Fang, Y., Loparo, K., and Feng, X. Inequalities for the trace of matrix product. IEEE Transactions on Automatic Control, 39(12):2489–2490, 1994.

Gönen, M. and Alpaydın, E. Multiple kernel learning algorithms. Journal of Machine Learning Research, 2011.

Hamedani, E. Y. and Aybat, N. S. A primal-dual algorithm with line search for general convex-concave saddle point problems. arXiv: Optimization and Control, 2020.

Jaggi, M. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In Proceedings of the 30th international conference on machine learning, 2013.

Jain, A., Vishwanathan, S., and Varma, M. SPF-GMKL: generalized multiple kernel learning with a million kernels. In Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining, 2012.

Kaya, H., Tüfekci, P., and Gürgen, F. Local and global learning methods for predicting power of a combined gas & steam turbine. In Proceedings of the international conference on emerging trends in computer and electronics engineering, pp. 13–18, 2012.

Lanckriet, G., Cristianini, N., Bartlett, P., El Ghaoui, L., and Jordan, M. Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research, 2004.

Ni, K., Kumar, S., and Nguyen, T. Learning the kernel matrix for superresolution. In Proceedings of the IEEE Workshop on Multimedia Signal Processing, pp. 441–446, 2006.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

Qiu, S. and Lane, T. Multiple kernel learning for support vector regression. Computer Science Department, The University of New Mexico, Albuquerque, NM, USA, Tech. Rep, 2005.

Rakotomamonjy, A., Bach, F. R., Canu, S., and Grandvalet, Y. SimpleMKL. Journal of Machine Learning Research, 2008.

Recht, B. Convex Modeling with Priors. PhD thesis, Massachusetts Institute of Technology, 2006.

Schölkopf, B., Herbrich, R., and Smola, A. A generalized representer theorem. In International conference on computational learning theory, pp. 416–426, 2001.

Smola, A. and Schölkopf, B. A tutorial on support vector regression. Statistics and computing, 14(3):199–222, 2004.

Sonnenburg, S., Rätsch, G., Henschel, S., Widmer, C., Behr, J., Zien, A., De Bona, F., Binder, A., Gehl, C., and Franc, V. The SHOGUN machine learning toolbox. Journal of Machine Learning Research, 11(60):1799–1802, 2010.

Tüfekci, P. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. International Journal of Electrical Power & Energy Systems, 60:126–140, 2014.

Xu, Z., Jin, R., Yang, H., King, I., and Lyu, M. Simple and efficient multiple kernel learning by group lasso. In Proceedings of the 27th international conference on machine learning, pp. 1175–1182, 2010.

Yang, H., Xu, Z., Ye, J., King, I., and Lyu, M. Efficient sparse generalized multiple kernel learning. IEEE Transactions on neural networks, 22(3):433–446, 2011.

Ye, Y. and Tse, E. An extension of Karmarkar's projective algorithm for convex quadratic programming. Mathematical programming, 44(1-3):157–179, 1989.