Approximate multiple kernel learning with least-angle regression

Martin Stražar, Tomáž Curk*

University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, Ljubljana SI 1000, Slovenia

A R T I C L E   I N F O

Article history:
Received 16 August 2018
Revised 12 November 2018
Accepted 10 February 2019
Available online 18 February 2019

Keywords:
Kernel methods
Kernel approximation
Multiple kernel learning
Least-angle regression

A B S T R A C T

Kernel methods provide a principled way for general data representations. Multiple kernel learning and kernel approximation are often treated as separate tasks, with considerable savings in time and memory expected if the two are performed simultaneously.

Our proposed Mklaren algorithm selectively approximates multiple kernel matrices in regression. It uses Incomplete Cholesky Decomposition and Least-angle regression (LAR) to select basis functions, achieving linear complexity both in the number of data points and kernels. Since it approximates kernel matrices rather than functions, it allows to combine an arbitrary set of kernels. Compared to single kernel-based approximations, it selectively approximates different kernels in different regions of the input spaces.

The LAR criterion provides a robust selection of inducing points in noisy settings, and an accurate modelling of regression functions in continuous and discrete input spaces. Among general kernel matrix decompositions, Mklaren achieves minimal approximation rank required for performance comparable to using the exact kernel matrix, at a cost lower than 1% of required operations. Finally, we demonstrate the scalability and interpretability in settings with millions of data points and thousands of kernels.

© 2019 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY license. (http://creativecommons.org/licenses/by/4.0/)

1. Introduction

Kernel functions provide general data representations and are used to numerically represent many object types, such as vectors, strings or trees. Multiple kernel learning (MKL) determines relevance of predefined kernels for a particular task. Kernel-based algorithms often involve evaluating the kernel for all pairs of data points – the kernel matrix – implying at least quadratic time and storage complexity [30]. In this work, we present a principled way to approximate a relevant subset of kernels in regression.

Kernel approximation methods can be split into two groups. First, approximations of the kernel function are data independent, space- and time- efficient, but constrained to kernels of particular functional forms, characterized by smoothness, differentiability, stationarity or input domain-based properties. Functional decompositions rely on approximating the implicit feature mapping or through spectral densities [25,26,34,36]. The second major group are the data-dependent, kernel matrix approximations. A subset of the input data points – termed the active set or inducing points – is selected. The kernel matrix is then approximated by projections to the basis functions centered on the active set. The active set selection can be based on expected error reduction [12,17,38], statistical leverage [120,29,41], or k-means++ [24]. Side information such as regression targets can be exploited in supervised learning [4]. Approximation of a general kernel matrix is bounded by the $O(n^2)$ complexity of solving linear systems determined by a matrix with $n$ data points and rank $r$. Highly optimized methods can further reduce complexity, but their applications are limited to specific kernel families [19,31,42].

Optimizing kernel hyperparameters is straightforward for differentiable kernels [5,23]. Alternatively, kernel functions can be learned from spectral densities [13,40]. The multiple kernel learning (MKL) problems are quadratic or semidefinite programs, and assume access to the exact kernel matrix [14]. Sparse Pseudo Input Gaussian Processes jointly optimize both the active set and the hyperparameters, but are limited to real number input spaces and differentiable kernels [7,32,39].

We propose a joint low-rank kernel approximation and MKL for regression allowing for arbitrary kernels and input spaces. The Mklaren algorithm is based on supervised Incomplete Cholesky Decomposition to simultaneously learn low-rank multiple kernel approximations and a regression model. The proposed active set selection is based on least-angle regression (LAR), an iterative implementation of the Lasso [11]. It addresses the following drawbacks of the existing approaches:

* Corresponding author.

E-mail addresses: martin.strazar@fri.uni-lj.si (M. Stražar), tomaz.curk@fri.uni-lj.si (T. Curk).

https://doi.org/10.1016/j.neucom.2019.02.030

0925–2312/© 2019 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license. (http://creativecommons.org/licenses/by/4.0/)
1. Existing kernel matrix approximation methods rarely address a multiple kernel scenario. Indeed, the linear span of the kernel matrix sum is equivalent to the concatenation of the implicit feature spaces. However, we show that this property is lost with an approximation of the kernel matrix sum (see A.5).

2. Standard feature selection methods in linear regression do not scale efficiently to kernel basis functions selection. We provide robust approximation and scoring of the basis functions.

3. The most efficient kernel approximations pose significant limitations on input data and types of kernels. Approximating the kernel matrix suffers no loss of generality and is useful when different families of kernels need to be used. We show several such examples.

The Mklaren algorithm is derived in Section 2. The related methods are described in Section 3 and compared experimentally to evaluate the differences in output function spaces, inducing point selection, multiple kernel learning, compactness and time complexity (Section 4). Auxiliary experiments and derivations are presented in the Appendix. The implementations of all evaluated algorithms and experiments is provided as a Python-based MKL library – mstrazar/mklaren on Github.

Notation. Lower-case bold letters denote vectors (e.g. \(\mathbf{g}\)), uppercase bold letters are matrices (e.g. \(\mathbf{G}\)), and parentheses index vectors and matrices (e.g. \(G(i,j)\)) is a scalar at row \(i\) and column \(j\). Subscripts link vectors to data points or kernels, i.e. \(\mathbf{g}_{ij}\) is a vector related to kernel \(q\) and data point \(i\). Kernel functions are denoted as \(k\). Notation \(k(\mathbf{x}_i, \cdot)\) is a basis function of \(k\) centered at data point \(\mathbf{x}_i\). Kernel matrices \(\mathbf{K}\) or functions \(k(\cdot, \cdot)\) are used interchangeably as appropriate, keeping in mind that the kernel matrix need never be computed entirely.

2. The Mklaren algorithm

We start by introducing the basic terminology of multiple kernel regression and present ideas underpinning Mklaren, Incomplete Cholesky Decomposition and Least-angle regression. Derivation of particular steps and complexity analysis conclude this section.

2.1. Multiple kernel regression

Let \{\(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\)\} be a set of input data points associated with target output values \(\mathbf{y} \in \mathbb{R}^n\). Let \(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_p\) be inner product spaces, each containing \{\(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\)\} and endowed with respective kernel functions \(k_1, k_2, \ldots, k_p\). The \(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_p\) can all be equal (e.g. real-valued vector spaces) or may contain different representations of the same objects (e.g. a text document can be seen both as a vector of word counts or a long unstructured string).

The kernels \(k_q\) are positive definite mappings from \(\mathcal{X}_i \times \mathcal{X}_j\) to \(\mathbb{R}\). In practice, they enable learning non-linear output functions \(f\), that map from \(\mathcal{X}_1 \times \mathcal{X}_2 \ldots \times \mathcal{X}_p\) to \(\mathbb{R}\) and thus approximate \(\mathbf{y}\). Each input point \(\mathbf{x}_i\) generates a basis function \(k_q(\mathbf{x}_i, \cdot)\) in terms of kernel \(k_q\), generating a representation. Examples of kernels and their underlying prior assumptions are listed in Section 3.2.

The output functions \(f\) assume the following parametric form:

\[
f = \sum_{q=1}^{p} \sum_{i=1}^{n} \mu_q \alpha_q k_q(\mathbf{x}_i, \cdot).
\] (1)

Here, the vector \(\mu \in \mathbb{R}^p\) represents the kernel weights, proportional to the relevance of each kernel, while \(\alpha\) are standard regression parameters.

Common MKL algorithms solve for \(\mu\), \(\alpha\) or both, and differ in computational complexity and constraints on \(\mu\), \(\alpha\). For instance, \(\mu\) can be learned independently, be limited to unconstrained, convex or linear combination [9], or be subject to norm constraints [27]. The parameters \(\mu\), \(\alpha\) can also be constrained jointly using matrix norms [15]. Alternatively, the kernel matrix can be optimized directly with convex optimization techniques [18].

2.2. Approximate multiple kernel learning with Mklaren

Computational complexity can be reduced by selecting a subset of all basis functions in Eq. (1). Evaluating \(k_q(\mathbf{x}_i, \mathbf{x}_j)\) for each pair \(\mathbf{x}_i, \mathbf{x}_j\) in the training set determines kernel matrices \(\mathbf{K}_q \in \mathbb{R}^{n \times n}\). These can be approximated with matrices \(\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_p\), where \(\mathbf{G}_q \in \mathbb{R}^{n \times r_q}\) and the total rank \(r = \sum_q r_q < n\).

Mklaren solves the above problem in a regression setting. Briefly, Incomplete Cholesky decomposition (ICD) is used to construct each \(\mathbf{G}_q\). The selection of basis functions is guided by least-angle regression (LAR). A principal challenge in adapting LAR to kernel learning is avoiding the evaluations of all possible \(k_q(\mathbf{x}_i, \mathbf{x}_j)\). This adaptation is our principal contribution and hinges on approximation of the correlations between \(k_q(\mathbf{x}_i, \cdot)\) and \(\mathbf{y}\). An example scenario and the steps of Mklaren are illustrated in Fig. 1. The complete pseudo code is listed in Algorithm 1, below.

Algorithm 1: The Mklaren algorithm.

Input:
\[
\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \text{ set of points in } \mathcal{X},
\quad k_1, k_2, \ldots, k_p \text{ kernel functions on } \mathcal{X} \times \mathcal{X},
\quad \mathbf{y} \in \mathbb{R}^n \text{ regression targets, with } \mathbf{1}^\top \mathbf{y} = 0,
\quad r \text{ total rank, } \delta \text{ look-ahead columns, } \lambda \text{ regularization parameter}.
\]

Result:
\[
\mathbf{G}_1 \in \mathbb{R}^{n \times r_1}, \mathbf{G}_2 \in \mathbb{R}^{n \times r_2} \ldots, \mathbf{G}_p \in \mathbb{R}^{n \times r_p} \text{ Cholesky factors},
\quad \mathbf{H} \in \mathbb{R}^{n \times r} \text{ combined feature space},
\quad \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p \text{ active sets of pivot indices},
\quad \mathbf{f} \in \mathbb{R}^n \text{ regression estimate on the training set},
\quad \mathbf{w} \in \mathbb{R}^p \text{ regression coefficients determining function estimator } \hat{f}.
\]

1. Initialize:

2. \(\mathbf{H} = \mathbf{0}\), residual \(\mathbf{e} = \mathbf{y}\), bisector \(\mathbf{u} = \mathbf{0}\), regression estimate \(\mathbf{f} = \mathbf{0}\), active sets \(\mathcal{A}_q = \emptyset\) and counters \(r_q = 0\) for \(q \in [1, \ldots, p]\).

3. Compute initial ICDs with \(\delta\) columns for \(\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_p\).

4. while \(\sum_q r_q < r\) do

5. Compute \(\bar{\mathbf{a}}_{q,i}\) and \(\bar{\mathbf{c}}_{q,i}\) for each kernel \(q\) and pivot \(i \notin \mathcal{A}_q\) (Eq. 8).

6. Select \(q, i\) based on the minimum in Eq. 5.

7. Compute \(\mathbf{g}_{q,i}\) (Eq. 3) and \(\mathbf{h}_{q,i}\) (Eq. 6):

8. \(\mathbf{g}_{q,i}(\cdot, r_q) \leftarrow \mathbf{g}_{q,i}^r\),

9. \(\mathbf{H}(\cdot, \sum_q r_q) \leftarrow \mathbf{h}_{q,i}\),

10. \(r_q \leftarrow r_q + 1\), \(\mathcal{A}_q \leftarrow \mathcal{A}_q \cup \{i\}\).

11. Recompute \(\mathbf{G}_q(\cdot, r_q + 1; r_q + \delta)\) using standard ICD.

12. Compute true \(\bar{\mathbf{a}}_{q,i}\) and \(\bar{\mathbf{c}}_{q,i}\) (Eq. 5).

13. Compute the bisector \(\mathbf{u}\) of columns in \(\mathbf{H}\) except \(\mathbf{h}_{q,i}\) (Eq. 1).

14. Compute step \(\alpha\) for \(\mathbf{h}_{q,i}\) (Eq. 5) and update

15. \(\mathbf{f} \leftarrow \mathbf{f} + \alpha \mathbf{u}\),

16. \(\mathbf{e} \leftarrow \mathbf{e} - \alpha \mathbf{u}\).

17. Solve linear system \(\mathbf{Hw} = \mathbf{f}\) for \(\mathbf{w}\) using Eq. 2.

\[\text{Available at https://github.com/mstrazar/mklaren.}\]
2.3. Incomplete Cholesky decomposition

Incomplete Cholesky decomposition (ICD, ref. [12]) is a numerical method to approximate symmetric positive definite matrices. Comparing to simpler approximations, e.g. random sampling of basis functions, it implements a cheap heuristic to search in unexplored input regions. For simplicity, we introduce the method assuming a single kernel matrix.

A positive semidefinite symmetric matrix $K \in \mathbb{R}^{n \times n}$ is approximated by Cholesky factors $G \in \mathbb{R}^{n \times r}$, computed in time $O(n^2 r) \prec O(n^2)$, such that $K \approx G G^T$. Candidate pivots are listed in the remaining set, $\mathcal{J} = \{1, 2, \ldots, n\} - r$ of them will be added to the active set, initialized as $A = \emptyset$. Initially, $G \in \mathbb{R}^{n \times r}$ is set to $0$. For each pivot $i$, a lower bound on reduction in absolute error $\| K - G G^T \|_1$ is kept in a vector $\mathbf{d}$, initialized with diagonal entries $d(i) = \| K(i, i) \|$.

At iteration $j = 1, 2, \ldots, r$, a pivot $i$ is selected from the remaining set $\mathcal{J}$ as

$$
i = \arg\max_{i \in \mathcal{J}} \mathbf{d}(i), \quad \mathcal{J} = \{1, 2, \ldots, n\} - A, \tag{2}
$$

and its pivot column $g_i = G(:, i)$ is computed as

$$
g_i(i) = G(i, j) = \sqrt{d(i)},

\quad g_i(j) = G(\mathcal{J}, j) = \frac{1}{G(i, j)} \left( K(\mathcal{J}, i) - \sum_{t=1}^{j-1} G(\mathcal{J}, t) G(i, t) \right). \tag{3}
$$

The selected pivot is added to $A$ and auxiliary variables are updated

$$
d \leftarrow d - g_i^2; \quad A \leftarrow A \cup \{i\}, \quad j \leftarrow j + 1.
$$

Since $K$ is determined by a kernel $k$, the pivot columns $g_i$ can also be seen through basis functions $k(x_i, \cdot)$ with fixed inducing points $x_i$. The pivot selection in Eq. (2) can be tailored to a specific task, e.g. kernel regression.

2.4. Basis function selection based on least-angle regression

Least-angle regression (LAR) is a feature selection method and an approximation to the Lasso solution path [11]. With multiple kernels, each $K_q$ corresponds to its approximation $G_q \in \mathbb{R}^{n \times q}$. Concatenating the Cholesky factors $G_q$ would form a combined kernel approximation $H$. Note that $H$ is not equivalent to a Cholesky decomposition of the uniform sum of base kernels $K_{\text{unif}} = \sum_{q=1}^{p} K_q$.

As the matrix $H$ is constructed iteratively, it requires a non-trivial adaptation of the LAR-based basis function selection. Let $h_{q,j}$ denote a scaled and centered pivot column $g_{q,j}$ (Eq. 6), corresponding to kernel $K_q$ and data point $x_i$, s.t. $\|h_{q,j}\|_2 = 1$ and $\mathbf{1}^T h_{q,j} = 0$. We describe how $h_{q,j}$ are selected to form $H$. The exact computation of all $h_{q,j}$ is indeed unfeasible, but for simplicity, suppose that $h_{q,j}$ are known. In Section 2.6, we then describe how to approximate $h_{q,j}$.

The auxiliary variables for computing the Cholesky factors of each $K_q$ are initialized as in Section 2.3, above. The feature matrix $H$, regression estimate $\mathbf{f}$, and the residual $\mathbf{e}$ are initialized as

$$
\mathbf{H} = 0, \quad \mathbf{f} = 0, \quad \text{and} \quad \mathbf{e} = y.
$$

A unique pair (kernel, pivot) $q, i$ is selected for each column $j$. At $j = 1$, the first vector is chosen to maximize correlation $c_{q,i} = \mathbf{e}^T h_{q,i} = \rho(\mathbf{e}, h_{q,i})$.

$$
q, i = \arg\max_{q \in \{1, 2, \ldots, p\}, x_i \in A_q} \rho(\mathbf{e}, h_{q,i}).
$$

and thus $H(:, j) - h_{q,i} \rightarrow A_q \leftarrow A_q \cup \{i\}$.

After iteration $j$, $H$ contains $j$ vectors, through which we can draw a bisector $\mathbf{u}$, having $\|\mathbf{u}\| = 1$ and making equal, less than 90 degree angles with $\mathbf{e}$ (A.1). Updating the regression estimate $\mathbf{f}$ along direction $\mathbf{u}$ and the residual $\mathbf{e}$,

$$
\mathbf{f}^{\text{new}} \leftarrow \mathbf{f} + \alpha \mathbf{u}, \quad \mathbf{e}^{\text{new}} \leftarrow \mathbf{e} - \alpha \mathbf{u}, \tag{4}
$$

causes the correlations $c_j = \rho(\mathbf{e}, H(:, j))$ to change equally for all $H(:, j)$ and an arbitrary step size $\alpha \in \mathbb{R}$. The value $\alpha$ is set con-
servatitively such that some new vector $h_{q,i}$ not in $H$ will have the same correlation with $e^{\text{new}}$ as all the vectors already in $H$:
$$
\rho(e^{\text{new}}, H(:, 1)) = \ldots = \rho(e^{\text{new}}, H(:, j)) = \rho(e^{\text{new}}, h_{q,i}).
$$
The step size $\alpha$ and $h_{q,i}$ are selected as follows. Define the following quantities
$$
C = c_1, \quad A = (1^T H 1)^{-1/2}.
$$
Then
$$
\alpha = \min_{q,i} \left\{ \frac{C - c_{q,i}}{A - a_{q,i}}, \frac{C + c_{q,i}}{A + a_{q,i}} \right\},
$$
where $c_{q,i} = \rho(e, h_{q,i})$, $a_{q,i} = \rho(u, h_{q,i})$, and $\min^+$ is the minimum over positive arguments for each $q, i$. The minimizer of Eq. (5) determines $H(:, j + 1) = h_{q,i}$. The variables $j, c_j, f, e, A_q$ are updated as above, according to $\alpha$. In the last iteration $j = r$, the step simplifies to $\alpha = C/A$, yielding the full least-squares solution for $H$ and $y$.

2.5. Scaling and centering the basis functions

Since LAR assumes centered and scaled basis functions, for each pivot column $g_{q,i}$ we define
$$
h_{q,i} = s_q G_{q,i}/\|P g_{q,i}\|,
$$
with the centering operator $P = (I - \frac{1}{n} 1 1^T)$ and the sign of the correlation with the residual $s_q = (P g_{q,i})^T e$. Importantly, the transformed basis functions and their correlations with the residual $e$ are directly comparable among kernels.

2.6. Look-ahead approximation

Explicit calculation of all $g_{q,i}$ and corresponding $h_{q,i}$ would yield a complexity of order $O(n^2)$. The issue is addressed by using approximations $\hat{g}_{q,i}$ and $\hat{h}_{q,i}$ that are less expensive to compute. The approach is based on selecting $\delta$ look-ahead columns, spanning a look-ahead approximation to the kernel matrix [4,7].

By definition of ICD in Eq. (3), the values of a candidate pivot column $g_{q,i}$ at step $j = r_q$ and pivot $i \notin A_q$ are
$$
\hat{g}_{q,i} = \frac{(K_q - \sum_{t=1}^{r_q-1} G_q(:, t) G_q(:, t)^T)(:, i)}{\sqrt{d(t)}}.
$$
In place of a complete kernel matrix $K_q$, a look-ahead approximation with $\delta$ additional columns is used
$$
\hat{g}_{q,i} = \frac{G_q(:, r_q+1:r_q+\delta)}{\sqrt{d(t)}}
$$
and the additional columns are computed as $\delta$ standard Cholesky steps (Section 2.3). This in turn defines approximate values $\hat{g}_{q,i}$
$$
\hat{g}_{q,i} = \frac{(L_q - \sum_{t=1}^{r_q-1} G_q(:, t) G_q(:, t)^T)(:, i)}{\sqrt{d(t)}}
$$
and
$$
= G_q(:, r_q+1:r_q+\delta) G_q(:, r_q+1:r_q+\delta)^T.
$$
Given $\hat{g}_{q,i}$ and consequently $\hat{h}_{q,i}$, the computation of approximate $\tilde{\xi}_{q,i}$ equals
$$
\hat{\xi}_{q,i} = e^T \hat{h}_{q,i} = \frac{(\hat{P} g_{q,i})^T e}{\|P g_{q,i}\|}.
$$
Cancelling out $\frac{1}{\|P g_{q,i}\|}$, the correlations are computed using
$$
\|\hat{P} g_{q,i}\|^2 = \|P G_q(:, r_q+1:r_q+\delta) G_q(:, r_q+1:r_q+\delta)\|_2^2
$$
and
$$
A_1 = \left( G_q^T G_q - \frac{1}{n} G_q^T \mathbf{1} \mathbf{1}^T G_q \right) (r_q+1:r_q+\delta, r_q+1:r_q+\delta),
$$
(9)
Computation of $\tilde{\xi}_{q,i}$ is analogous. Correctly ordering the computations yields a computational complexity of $O(\delta^2)$ per column. Note that matrices $G_q^T G_q$, $G_q^T \mathbf{1} \mathbf{1}^T G_q$, $e^T G_q$, $e^T \mathbf{1} \mathbf{1}^T G_q$ are the same for all columns (independent of $i$) and need to be computed once only per iteration.

The pair $q, i$ is selected according to Eq. (5), but with $\tilde{\xi}_{q,i}$ and $\tilde{\xi}_{q,i}$ in place of exact values. A Cholesky step (Eq. (3)) is then performed to compute the exact $g_{q,i}$ and corresponding $h_{q,i}$. The vector $h_{q,i}$ is then treated as the only remaining candidate for computing the correct step size $\alpha$ in Eq. (5) with exact values for $a_{q,i}$ and $c_{q,i}$. Crucially, as the newly discovered $h_{q,i}$ might have a higher correlation with the residual than the previously selected vectors — a situation that cannot happen in canonical LAR — the value of $\alpha$ is allowed to be negative for this update.

Finally, a new set of $\delta$ look-ahead columns is to be recomputed using the standard Cholesky steps. Note that increasing $\delta$ monotonically increases the approximation accuracy and is limited only by the available computational resources. See Section 2.8 for more discussion.

2.7. The Mklaren algorithm

The steps described in the previous sections complete the Mklaren algorithm (Algorithm 1). Given a sample of $n$ data points with targets $y$ and $p$ kernel functions, the model hyperparameters are: the maximum rank $r$ of the combined feature matrix, the number of look-ahead columns $\delta$ and $l_2$ norm regularization parameter $\lambda$ (constraining $f$, discussed in A.4).

The variables related to regression estimate ($f$, residual $e$ and bisector $u$) and individual decompositions $G_q$ (active sets $A_q$, column counters $r_q$) are initialized in lines 1-2. Each $G_q$ is initialized using standard ICD with $\delta$ look-ahead columns, as described in Section 2.6 (line 3).

The main loop is executed for $r$ iterations, until the sum of selected pivot columns equals $\sum r_q = r$, where at each iteration a kernel $k_q$ and a pivot column $i \notin A_q$ are selected and added to $G_q$ and consequently the combined feature matrix $H$. For each kernel $k_q$ and each pivot $i \notin A_q$, $\tilde{\xi}_{q,i}$ and $\tilde{\xi}_{q,i}$ are computed. Based on these approximate values, the kernel $k_q$ and pivot $i$ are selected. Given the optimal $k_q$ and pivot $i$, the pivot column $g_{q,i}$ and $h_{q,i}$ are computed. The new pivot column $g_{q,i}$ is added to $G_q(:, r_q)$, $r_q$ is incremented and the $\delta$ columns at $G_q(:, r_q+1:r_q+\delta)$ are recomputed using standard ICD (lines 5-11). Having computed the exact $g_{q,i}$, the true values $a_{q,i}$ and $c_{q,i}$ can be computed and the regression estimate $f$ and the residual $e$ are updated (lines 12-16).

The regression coefficients $w$ solving for $H w = f$ and required for out-of-sample prediction are obtained by constructing $H$ and solving the linear system discussed in A.2 (line 17).

Instead of stopping after consuming the predefined time/memory budget determined by $r$, another possibility is to estimate the expected generalization error. This type of an approach is sketched in A.6. Other types of high probability bounds related to kernel approximation can also be readily applied to Mklaren [13].

2.8. Computational complexity

The Mklaren algorithm scales linearly both in the number of data points $n$ and kernels $p$. Assuming $r < n$, the computational
complexity is
\[ O(n^3 + r^2 (r^2 + npb_3 + n\delta^2) + nr^2 + r^3) = O(n^2 + npb_3^2). \] (10)

The look-ahead Cholesky decompositions are standard Cholesky decompositions with \( \delta \) pivots and complexity \( O(n^2 \delta) \). The main loop is executed \( r \) times. The selection of kernel and pivot pairs includes inverting \( \mathbf{H}^T \mathbf{H} \) of size \( r \times r \) subject to complexity \( O(r^3) \). However, as each step is a rank-one modification to \( \mathbf{H}^T \mathbf{H} \), the Morrison-Sherman-Woodbury lemma can be used to achieve complexity \( O(r^2) \) per update \[21\]. The computation of correlations in Eq. (8) are computed for \( p \) kernels in \( O(npb_3^2) \). Recomputation of \( \delta \) Cholesky factors requires standard Cholesky steps of complexity \( O(n^2 \delta^2) \). The computation of the gradient step and updating the regression estimate is \( O(n) \). The QR decomposition in Eq. (2) takes \( O(n^2) \), and the computation of linear transform \( \mathbf{T} \) in Eq. (3) is of complexity \( O(r^3 + nr^2) \). The computational complexity bound in Eq. (10) also provides natural parameter bounds — for total complexity lower than \( O(n^2 \delta) \) which equal \( r < \sqrt{\pi} \) and \( \delta < \sqrt{\pi} / \sqrt{r} \). The complexity of solving \( n \times n \) linear systems (including linear regression) of \( O(n^3) \) can be improved accordingly by setting \( r < n \) and \( \delta < \sqrt{\pi} / \sqrt{r} \).

3. Related work

The experiments elucidate the differences between methods implementing approximate multiple-kernel Ridge regression, which are reviewed here.

3.1. A note on compared methods

All the experiments assume a data set with \( \{x_1, x_2, \ldots, x_n\} \) points in \( p \) inner product spaces \( \lambda_{\mathbf{X}} \) with corresponding kernel functions \( k_{\mathbf{X}} \) and real number targets \( y = \{y_1, y_2, \ldots, y_n\} \). The dimension of the reduced feature space equals \( r < n \) for all methods, hereby listed according to the increasing number of assumptions about the data or kernels.

Cholesky with Side Information (CSI), Incomplete Cholesky Decomposition (ICD) and the Nyström method all approximate the kernel matrix using \( r \) inducing points defining the active set \( \mathcal{A} \subset \{1, 2, \ldots, n\} \), with \( |\mathcal{A}| = r \) and the basis functions \( k_{\mathbf{X}}(x_i, \cdot) \) for \( i \in \mathcal{A} \), for a kernel \( k_{\mathbf{X}} \). Importantly, no further restrictions on the kernels or input spaces are assumed. The methods can be seen as discrete optimization and differ in sampling of \( \mathcal{A} \); ICD approximates the reduction in error (Section 2.3); the Nyström method uses approximate leverage scores which determine the sampling distribution; CSI uses a matching pursuit-based heuristic.2 In contrast to Mklaren, only a single kernel is assumed; to implement MKL, we approximate an unweighted sum of all \( p \) kernels up to rank \( r \) (\( K_{\text{unif}} \)); see Section 2.5.

Random Fourier Features (RFF) approximate the frequency components of the basis functions [26]. The matching pursuit optimization is used; \( \delta \) basis functions are generated for each \( k_{\mathbf{X}} \), and selected greedily up to rank \( r \) [37]. Non-stationary RFF (RFF-NS) are obtained by sampling from the spectral densities proposed by Tön et al. [35].

Sparse Pseudo Input Gaussian Processes (SPGPs) explore a space of models with the largest capacity by continuously optimizing both the locations of the inducing points and kernel hyperparameters [32]. This limits its applications to real vector spaces \( \lambda_{\mathbf{X}} = \mathbb{R}^d \) of finite (and not too large) dimension \( d \) with continuous and differentiable kernels. Although the locations of pseudo inputs are optimized, the zero-mean prior over function values on the pseudo inputs tends to an uniform, well-spread placement.

2 With matching pursuit, we refer to the approach which chooses the basis functions in a greedy manner and solves for the full least-squares estimate at each step.

As an empirical lower bound on prediction error we used the \( L_2 \)-regularized Kernel learning (\( L_2 \)-KRR), optimizing both the kernel weights and the regression coefficients given the exact kernel matrix \[8\]. To evaluate multiple kernel learning regardless of approximation, we used the state-of-the-art methods Align (independently learned weights for each kernel), Linear (linear combination) and Convex (convex combination) from Cortes et al. [9].

3.2. Kernels used in this study

We briefly review kernels used in this section, with emphasis on the interpretation of respective hyperparameters.

Linear kernel. The linear kernel corresponds to the canonical inner product
\[ k_{\text{lin}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'. \] (11)
This kernel is included in order to perform feature selection experiments.

Exponentiated-quadratic kernel. The infinite differentiability of the exponential-quadratic generates smooth output functions:
\[ k_{\exp}(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2). \] (12)
with the frequency hyperparameter \( \gamma \) or length scale \( \ell = \sqrt{\gamma} \) which determines the distance \( \|\mathbf{x} - \mathbf{x}'\| \) at which the output functions covary.

The Matérn kernel. Both the smoothness \( \nu \) and the length scale \( \ell \) of the output functions are controlled [28]. With \( \nu \) of form \( t + 1/2 \), for integer \( t \), the kernel is a product of a \( t \)-degree polynomial and an exponential of the distance \( \|\mathbf{x} - \mathbf{x}'\| \). We use the Matérn 3/2 kernel as an example
\[ k_{\text{Matern}}(\mathbf{x}, \mathbf{x}') = \left(1 + \sqrt{3}\|\mathbf{x} - \mathbf{x}'\|/\ell\right) \exp\left(-\sqrt{3}\|\mathbf{x} - \mathbf{x}'\|/\ell\right). \] (13)

Periodic-exponential kernel. The periodic-exponential is useful for modeling periodic signals, such as time series. It is defined as multiple exponentiated-quadratic kernels with length scale \( \ell \), repeated at frequency \( \omega \).
\[ k_{\text{per}}(\mathbf{x}, \mathbf{x}') = \exp\left(-2 \sin(\omega \|\mathbf{x} - \mathbf{x}'\|) \right)^2 \ell^2. \] (14)

String kernels. The data points are strings up to length \( L \) sampled from a finite alphabet \( \mathcal{B} \). The spectrum kernel between strings \( s, s' \in \mathcal{B}^L \) is a product of substrings counts:
\[ k_{\text{SPEC}}(s, s') = \sum_{i=|s'|-|s|}^{\min(|s|, |s'|)} \text{count}(s_i, s_{i+|s'|}) \times \text{count}(s_i, s_{i+|s'|}), \] (15)
with the substring length \( \ell < L \), and \( \text{count}(s, s') \) equals the number of occurrences of \( u \) in \( s \). Explicitly constructing the feature space corresponding to \( k_{\text{SPEC}} \) is unpractical as its dimension grows as \( |\mathcal{B}|^{2 \ell} \). The substring length \( \ell \) is analogous to the bandwidth of the exponentiated-quadratic kernel, as increasing \( \ell \) implies a more stringent similarity threshold.

4. Experimental results and discussion

This section presents extensive empirical comparison of Mklaren against approximate kernel methods, multiple kernel learning methods and hybrid methods. The methods are first compared on a simple 1D real line; Section 4.1 discusses the differences in output function shapes and their effect on performance in interpolation and extrapolation. The applicability of kernel matrix approximations to non-numerical, string data is discussed in Section 4.2. Systematic comparison with related kernel approximations follow in Section 4.3, multiple kernel learning methods and interpretability in Section 4.4, and inducing points selection approaches in B.2. Finally, the empirical execution times are assessed in Section 4.5.
4.1. Modeling one-dimensional functions

One of the simplest input spaces is arguably a regularly-sampled interval on the real line $\mathbb{R}$. It helps to illustrate the differences in the output functions and choice of inducing points. We perform two sets of experiments: interpolation (test points are intermixed with training points) and extrapolation (test points lie away from the training points). Time series are natural scenarios where these tasks are relevant.

The Appliances energy prediction data set\(^3\) contains time series measurements of temperature in nine building compartments, denoted $T_1 - T_9$. The measurements are recorded at regular 10 minute intervals over 18 weeks. Each week is thus temperature time series of length 1008.

4.1.1. Interpolation

Signals were split into training, validation and test sets by (round-robin) assigning each consecutive time point in a different set. The training set was used to fit the data on exponentiated-quadratic kernels (Eq. (12)) with the hyperparameter $\gamma$ in $[10^{-4}, 10^{-2}, 1, 10^2, 10^4]$ and the Matérn 3/2 kernel (Eq. (13)) with the length scale hyperparameter assuming equivalent values. The validation set was used to tune the regularization hyperparameter $\lambda$ within $[0, 10^{-1}, 10^{-0.5}, 1, 10^{0.5}, 10^1]$. The look-ahead parameter for Mklaren, CSI, RFF and RFF-NS was set to $\delta = 10$, while rank was set to the number of days in a week ($r = 7$). The performance evaluation metric was the root mean-squared error (RMSE) on the test sets, averaged over all 18 weeks.

The supervised methods selecting components from different length scales (i.e. Mklaren, SPGP, RFF and RFF-NS) performed best (Table 1). The difference between Mklaren and other kernel matrix approximations ICD, CSI and Nyström are statistically significant at $P < 0.05$ (Friedman rank test, Fig. 2). With the exponentiated-quadratic kernel, Mklaren performed best in 4 out of 9 data sets, and 8 out of 9 cases with the Matérn kernel.

Strikingly, Mklaren slightly outperformed the SPGP model with continuous optimization of inducing points. Since Mklaren imposes no prior on inducing point locations, it can sometimes appear more flexible by placing the inducing points non-uniformly, while SPGP will favour uniform placement. This is confirmed on Fig. 3, highlighting the differences in method-specific output function spaces.

\(^3\) https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction.
Most similar to Mklaren, RFF samples functions with different frequencies, approximated with periodic components. However, input region-specific dependencies on different length scales are not entailed by the stationary RFF. The improved performance by Mklaren is likely attributable to direct exploitation of the target signal, rather than the two-step approach of initially generating random features and subsequently using matching pursuit. The non-stationary RFF-NS, somewhat unexpectedly, performed slightly worse than stationary RFF. We speculate that this is due to the specific way the optimization is implemented; as both approaches sample periodic basis functions and the signal variance appears constant across the input region, stationary features are not at a notable disadvantage. Since both methods are allowed the same number of look-ahead draws, and as RFF samples from a smaller space, the probability of finding useful basis functions appears slightly larger.

4.1.2. Extrapolation

The true potential of time series prediction lies in extrapolation to future times. Here, we compared the approximate multiple kernel learning methods to an autoregressive integrated moving average (Arima) baseline [22]. We used the same data as above, but employed a different splitting strategy. The first half ($n_{tr}$ = 504) points was used for training, the following quarter ($n_{val}$ = 252) for selecting regularization parameter $\lambda$, and the last quarter ($n_{te}$ = 252) for measuring the final performance.

To identify long-term trends, we use a combination of linear $k_{\text{lin}}$ (Eq. (11)) and periodic $k_{\text{per}}$ kernels (Eq. (14)). The hyperparameters of $k_{\text{per}}$ can assume 13 different values each $\omega \in \{10^{-3}, 10^{-2.5}, 10^{-2}, \ldots, 10^{1}\}$ and $\epsilon \in \{10^{-3}, 10^{-2.5}, 10^{-2}, \ldots, 10^{3}\}$. Together with $k_{\text{lin}}$, this results in 170 kernels. We used Mklaren, CSI, ICD and the Nyström method with rank $r = 7$, and look-ahead $\delta = 70$. With Arima, we fit multiple linear models $f_1, f_2, \ldots, f_{504}$ one for each possible delay $r \in 1 \ldots 504$ in the training set. To predict the value at $y_{t+r}$, we used the feature representation of last 7 time points, $x_\delta = (y_{t-6}, y_{t-5}, \ldots, y_{t})$ with model $f_r$. This resulted in fitting a total of 504 $\times 7 = 3528$ coefficients with a $L_2$ (Ridge) regression parameterized by $\lambda$.

An example fit for each method is shown on Fig. 4. As the Arima model performs perfectly within the training set, the accuracy suffers a heavy drop with longer delays (test region in cyan). Similarly, the single kernel approximations CSI, ICD and the Nyström method are not able to pursue a clear future trend due to a uniform mixture of all frequencies and length scales. Mklaren identified a small number of frequencies and a slight linear trend, an advantage that is confirmed also numerically. As the mean-squared
error varies substantially, we select the method with lowest RMSE and count the number of wins within all combinations of compartments and weeks (Table 2). Here, Mklaren is the best performing method in more than half of the cases, with the other methods sharing the wins nearly equally. The low performance of Arima can be explained by the test data being very far in the future, with a low number of corresponding delays in the training set.

In summary, Mklaren delivers competitive performance even when compared to flexible inducing point and hyperparameter optimization methods. The performance benefits from different kernels in different input regions and adaptable placement of inducing points.

### 4.2. Non-numerical input data: regression with strings

String data are common in text mining and bioinformatics [33]. The data points \( s_1, s_2, \ldots, s_n \) are strings of length \( L \), sampled from a finite alphabet, e.g. \( B = \{A, C, G, T\} \) representing DNA bases. Real number targets \( y \in \mathbb{R}^n \) are associated to each \( s_i \). We focus on the spectrum kernel (Eq. (15)), the inner product of common substring occurrences. The exponential complexity of constructing the explicit feature space renders SPGP and RFF unfeasible, so general kernel matrix approximations need to be used. By means of synthetic data, we first examine whether correct substring lengths can be identified. Second, the utility of our approach is shown with a real biological data set of RNA sequences.

**Experiments on synthetic data.** Random strings of length \( L = 30 \) are generated by uniformly sampling from \( B^L \). The sizes of training, validation and test sets were set to \( n_{tr} = 500, n_{val} = n_{te} = 5000 \). Ten spectrum kernels with \( \ell \in \{1, 2, \ldots, 10\} \) were used. A ground truth active set \( A \) of \( r = 7 \) inducing points was sampled from the training set. The underlying true kernel matrix was \( L = \mathbf{K}_{\ell=4}(\cdot, A)\mathbf{K}_{\ell=4}(A, A)^{-1}\mathbf{K}_{\ell=4}(\cdot, A)^T \) with \( \mathbf{K}_{\ell=4} \) the spectrum kernel matrix with a fixed \( \ell = 4 \). Targets were sampled from a degenerate Gaussian Process, \( y \sim \mathcal{N}(0, I) \).

Out of 30 cross-validation comparisons, Mklaren achieved the lowest RMSE in 23 cases (76.6 %), while CSI in the remaining 7 cases (23.3 %) \( (P < 0.0017, \text{Wilcoxon paired signed rank test}) \). The critical distance plots showing the differences in ranks at \( P < 0.05 \) (Fig. 5a). Differences in the results can be explained by the effect of approximating the kernel term as discussed in the Section 4.1, but this time with discrete rather than continuous input space. Recall that different substring lengths are analogous to signal frequency components (Section 3.2); we investigate how different \( \ell \) affect the shape of the output functions. Due to the \( d = \sum_{i=1}^{10} 4^i = 1,398,100 \) dimensional feature space, model visualization is challenging. We devise a visualization approach in B.1.

**Fig. 5b** shows a typical example where Mklaren outperforms the kernel matrix sum \( \mathbf{K}_{\text{skernel}} \) approximations and most closely matches the profile of the true output. The methods approximating the kernel sum are prone to the short length substrings (i.e. \( \ell = 1, 2, 3 \)), which are analogous to high frequency continuous signals in Section 4.1. These confirms Mklaren is suitable for modelling discrete string data and identifying suitable substring lengths.

**Predicting RNA-binding protein binding affinities.** A biological case study contains RNA sequences of \( L = 30 \) and \( B = \{A, C, G, U\} \), along with affinities of nine RNA-binding proteins [16]. There are 18 data

---

**Table 2**

| Method       | win % | CSI | ICD | Arima | Nyström |
|--------------|-------|-----|-----|-------|---------|
| Mklaren      | 51.7  | 17.3| 10.5| 10.5  | 8.0     |

---

**Fig. 4.** Selected example of a time series extrapolation with periodic and extrapolation kernels \( (T_i, \text{last week}, \lambda = 0.1) \), and a baseline Arima model. Validation and test regions are shown in gray and cyan, respectively.
Table 3
Root-mean squared error (RMSE) on the RBP binding affinity prediction data sets using low-rank matrix reconstruction on string kernels.

|     | n | Mklnren | CSI | Nyström | ICD |
|-----|---|---------|-----|---------|-----|
| U1A (set B) | 15,192 | 0.836 ± 0.010 | 0.836 ± 0.012 | 0.839 ± 0.013 | 0.847 ± 0.012 |
| U1A (set A) | 15,294 | 0.818 ± 0.009 | 0.818 ± 0.012 | 0.825 ± 0.010 | 0.830 ± 0.012 |
| VTS1 (set B) | 28,115 | 0.965 ± 0.008 | 0.965 ± 0.007 | 0.968 ± 0.006 | 0.970 ± 0.006 |
| VTS1 (set A) | 28,542 | 0.965 ± 0.007 | 0.966 ± 0.007 | 0.967 ± 0.006 | 0.969 ± 0.006 |
| Fusip (set B) | 31,148 | 0.927 ± 0.011 | 0.929 ± 0.008 | 0.939 ± 0.007 | 0.943 ± 0.010 |
| Fusip (set A) | 31,300 | 0.927 ± 0.009 | 0.932 ± 0.009 | 0.940 ± 0.007 | 0.941 ± 0.010 |
| RBM4 (set B) | 34,288 | 0.790 ± 0.024 | 0.811 ± 0.038 | 0.834 ± 0.021 | 0.878 ± 0.032 |
| RBM4 (set A) | 34,628 | 0.794 ± 0.022 | 0.812 ± 0.024 | 0.834 ± 0.016 | 0.877 ± 0.022 |
| SLM2 (set B) | 37,588 | 0.871 ± 0.010 | 0.871 ± 0.011 | 0.881 ± 0.008 | 0.887 ± 0.011 |
| SLM2 (set A) | 38,002 | 0.875 ± 0.011 | 0.884 ± 0.012 | 0.891 ± 0.010 | 0.899 ± 0.012 |
| PTB (set B) | 39,536 | 0.859 ± 0.004 | 0.861 ± 0.006 | 0.864 ± 0.005 | 0.866 ± 0.006 |
| PTB (set A) | 39,822 | 0.862 ± 0.004 | 0.862 ± 0.004 | 0.866 ± 0.004 | 0.869 ± 0.006 |
| SF2 (set B) | 40,774 | 0.812 ± 0.021 | 0.818 ± 0.017 | 0.839 ± 0.011 | 0.854 ± 0.024 |
| SF2 (set A) | 41,197 | 0.808 ± 0.014 | 0.823 ± 0.019 | 0.829 ± 0.011 | 0.851 ± 0.011 |
| HuR (set B) | 41,625 | 0.758 ± 0.007 | 0.761 ± 0.006 | 0.779 ± 0.008 | 0.775 ± 0.012 |
| HuR (set A) | 41,864 | 0.756 ± 0.009 | 0.753 ± 0.007 | 0.772 ± 0.010 | 0.769 ± 0.012 |
| YB1 (set B) | 45,397 | 0.957 ± 0.006 | 0.958 ± 0.004 | 0.960 ± 0.004 | 0.964 ± 0.004 |
| YB1 (set A) | 45,857 | 0.964 ± 0.005 | 0.962 ± 0.005 | 0.965 ± 0.005 | 0.967 ± 0.005 |

sets, 2 replicates for each protein, with the number of sequences n in tens of thousands. We used ntr = ntv = 3000 randomly selected sequences for training and validation. All the remaining sequences were used for testing. The training-validation-testing process was repeated 30 times. For all methods, the rank was set to r = 10 and the look-ahead to δ = 10 (for Mklnren and CSI).

Mklnren and CSI achieve lowest average RMSE on all data sets (Table 3). The differences are slight, but statistically significant — Mklnren achieves the minimal average RMSE in 16 out of 18 data sets (P = 0.0028. Wilcoxon paired signed-rank test versus CSI). Low-rank approximations are suitable with data sets where the objects are accessed only through the kernel, where Mklnren achieves competitive performance.

4.3. Savings due to low-rank approximation

The principal difference of Mklnren compared to other approximations is simultaneous approximation of multiple kernels and the guarantee that all the basis functions are equally correlated with the residual at each step. We evaluate the minimal necessary rank to achieve equivalent error to full-rank MKL method L2-KRR, considered an empirical lower bound. The experimental design is inspired by Bach and Jordan [4], and uses 28 regression data sets.

We have used p = 10 exponentiated-quadratic kernels (Eq. (12)) with varying bandwidth γ ∈ {10−3, 10−2, ..., 105, 106}. The look-ahead δ was set to 10 for Mklnren, CSI, RFF and RFF-NS. All features were standardized and the targets y were centered. Up to 3000 data points were per data set, ensuring sensible time to compute the exact L2-KRR. The performance was assessed using 5-fold cross-validation; a training set (60% of the size) and validation sets (20%) were used to infer the approximation and select the regularization parameter λ from [0, 10−5, 10−4, ..., 101], respectively. The performance was scored by RMSE on the remaining test data. Cases where no method achieved the target were removed from comparison (mv, pole, stock, concrete).

Table 4 shows the minimal setting of r where the performance is at most one standard deviation away from the performance obtained by L2-KRR. The differences in ranks among all evaluated methods are shown in Table 4. The differences are statistically significant (P < 2.2 × 10−16). Friedman rank-sum test) and are summarized on Fig. 6. The best two methods are SPGP and Mklnren,
Table 4
Minimal rank $r$ at which the error of each method is at most one std. dev. away from L2-KRR. In bold: lowest rank among kernel matrix approximations (Mklaren, CSI, ICD, Nyström). Ratio denotes approximate savings in operations for solving linear systems, i.e., $\frac{m^2}{n}$, i.e., $\frac{m^2}{n}$. (column ratio in Table 4). The highest ratio (worst case) was $6.4 \times 10^{-3}$ for Friedman, indicating that the solution can be found with a mere 0.64% operations, and orders of magnitude lower for most other data sets.

![Critical distance (CD) diagram](image)

whose rankings are statistically indistinguishable at $P < 0.05$ (Friedman test), although SPGP consistently achieves the minimal rank. Expectedly, Mklaren, RFF and RFF-NS perform comparatively, due to selection of basis functions from a given set of kernels centered on the training set. Only Mklaren and SPGP achieve the goal within the given rank constraints in all cases. There is a larger gap between Mklaren and other kernel matrix approximation methods, attributable to supervised selection of inducing points from multiple kernels, and the difference in function spaces, discussed in A.5 and Section 4.1. The difference between Mklaren and CSI is statistically significant when examined independently (Wilcoxon paired signed-rank test, $P = 0.009$). Expectedly, Mklaren outperforms general kernel matrix approximations CSI, ICD, and Nyström when they approximate the kernel matrix sum up to rank $r$. This holds even if the three methods approximate each of the $p$ kernels individually, up to rank $\left[ \frac{p}{2} \right]$ (data not shown). Note that this way, the minimal achievable rank is 10, which is greater than the minimum in half of the cases in Table 4.

In all cases, the goal can be achieved with $r$ much lower than $n$, ensuring a low-complexity model in terms of degrees of freedom (Section A.6). The fraction of required operations for solving the corresponding linear systems with Mklaren are estimated as $\frac{m^2}{n}$ (column ratio in Table 4). The highest ratio (worst case) was $6.4 \times 10^{-3}$ for Friedman, indicating that the solution can be found with a mere 0.64% operations, and orders of magnitude lower for most other data sets.

Due to practical difficulties in ensuring consistent implementations (programming language, etc.) we opt for rank $r$ as a measure of effective complexity. The absolute timings are nevertheless provided in Section 4.5. In summary, Mklaren provides most compact approximations comparing to general kernel matrix approximations, solving the linear systems for a small fraction of cost. The performance is also competitive to highly optimized methods for exponentiated-quadratic kernels. The kernels that are not included in the model can be discarded, which is corroborated in the following section.

4.4. Multiple kernel learning and interpretability

The comparison of Mklaren to full-rank MKL methods is challenging, as it is unrealistic to expect improved predictive performance with approximations. We focus on the ability of Mklaren to select from a large set of kernels in linear time, without sacrificing the consideration of between-kernel correlations. Here, we evaluated both the predictive performance and interpretability.

Inspired by Cortes et al. [9], we use four sentiment analysis data sets compiled by Blitzer et al. [6]. The data are user reviews of books ($n = 5501$), electronics ($n = 5901$), kitchens ($n = 5149$) and dvds ($n = 5118$), with target ratings in the integer interval $[1, 5]$. The features are counts of the 4000 most frequent unigrams and bigrams, represented by rank-one linear kernels. The splits into training and test set were provided by the authors of the data set.

We compared Mklaren with MKL methods, maximizing centered kernel alignment by Cortes et al. [9]: Align Linear and Convex, described in Section 3.1 and adapted for kernel Ridge regression. Prior to optimization, the features were initially filtered according to the descending centered alignment metric for Align,
Linear, Convex, which ensured the same dimension of the feature space. When using Mklaren, the r pivot columns were selected from the complete set of p = 4000 kernels. The parameter δ was set to 1. Note that in this scenario, Mklaren is equivalent to the original LAR method, thus excluding the effect of low-rank approximation.

The performance was measured via 5-fold cross-validation. At each step, 80% of the training set was used for kernel matrix approximation (Mklaren) or determining kernel weights (Align, Linear, Convex). The remaining 20% of the training set was used for selecting regularization parameter λ from the set \(0, 10^{-3}, 10^{-2}, \ldots, 10^{3}\), and the final performance was reported on the test set as RMSE. The results for different settings of rank r are shown in Table 5, with the uniform kernel sum included as a baseline. Mklaren outperformed all other MKL methods that assume complete kernel matrices, and its RMSE is more than one standard deviation away from that of any other methods. On all four data sets, the performance appears to saturate at \(r = 160\).

The interpretability of kernel selection is assessed on the \(kitchen\) data set. Each of the methods infers an ordering of kernels; with Mklaren, it is defined by the pivot columns; with alignment-based methods, by the kernel weight vector. In Fig. 7, we display the performance of up to 40 features. The full least-squares coefficients \(w_{\text{ols}}\) is used for each feature subset up to step \(j\). The arrows indicate the sign of the coefficient \(w_{\text{ols}}\). The gradient in explained variance is higher for features corresponding to words associated to strong sentiments: \textit{great}, \textit{good}, \textit{love}, etc. Not surprisingly, the order in which features are added to the model critically influences the explained variance. Due to the equivalence between LAR and Lasso, the strong sentiment features are identified early, irrespective of their magnitude. On the other hand, the regularized centered alignment methods appear to select words with a high prior frequency, e.g. propositions. These results show that the Mklaren model can identify a sensible combinations of kernels with a small number of iterations.

4.5. Empirical execution times

Implementations of all discussed methods were compared in terms of training time. Data was generated as described in Section B.2, assuming \(k_{\text{exp}}\) kernels (Eq. (12)), rank \(r = 30\), input dimension \(d = 100\). regularization \(\lambda = 0.1\) and look-ahead \(\delta = 10\) for Mklaren, CSI and RFF. All methods were presented with an \(X \in \mathbb{R}^{n \times d}\) input matrix, a \(y \in \mathbb{R}^{n}\) target vector and \(k_1, k_2, \ldots, k_p\) kernel functions. The time to compute the complete kernel matrix was included where relevant. The processes were allowed memory resources of 512 GB RAM and a time limit of one hour.

Timings of different scenarios for varying \(n \in \{10^2, 10^{2.5}, 10^3, 10^{3.5}, 10^4\}\) and \(p = 10\) are shown in Table 6. For up to 316 examples, the full-rank MKL methods are comparable to low-rank approximations. SPGP is least efficient due to optimization of inducing points. For large data sets, only low-rank matrix approximations met the time constraints, with RFF being most efficient due to skipping computation of the kernel entirely. The linear complexity is confirmed for Mklaren; it takes 3.9-5.1 times longer than Nyström due to look-ahead steps for 10 kernels.

The performance with varying the number of kernels \(p \in \{10^1, 10^2, 10^3 \ldots, 10^{5}\}\) and \(n = 1000\) is assessed in Table 7. Note that only MKL-based methods are compared. The methods with quadratic complexity in \(p\) (Linear, Convex, L2-KRR, SPGP) exceeded the constraints when \(p\) exceeded 100. Mklaren is second most efficient up to \(p = 31\), after which the overhead in selectively evaluating the kernel outweighs pre-computation of the complete kernel matrix.

With these results, we empirically demonstrate the scalability of Mklaren to large data sets in both \(n\) and \(p\), achieving the best performance among supervised kernel matrix approximations.
Fig. 7. Increase in explained variance upon incrementally including features to an ordinary least-squares model (the kitchen data set). The order of features is determined by the magnitude of kernel weights for Align, Linear and Convex or the order of selection by Mkla伦. Arrows indicate positive (black) or negative (gray) sign of the feature in the model weight vector upon inclusion. Under-scored are words “great” and “not”, which significantly alter the explained variance when discovered by the Align, Linear and Convex models.
Table 6
Timings (in seconds) for increasing number of data points n. A dash (-) denotes cases where processes exceeded the given resources. The rows are sorted by: 1) number of cases not exceeding one hour and 2) average rank.

| method | n    | 100  | 316  | 1000 | 3162 | 10,000 | 31,622 | 100,000 | 316,227 | 1,000,000 |
|--------|------|------|------|------|------|--------|--------|---------|---------|-----------|
| RFF    | 0.03 | 0.08 | 0.12 | 0.26 | 0.73 | 2.45   | 8.86   | 32.7    | 105.22  |
| ICD    | 0.08 | 0.17 | 0.39 | 1.08 | 3.53 | 10.63  | 36.4   | 117.23  | 370.54  |
| Nyström| 0.09 | 0.18 | 0.41 | 1.17 | 3.59 | 11.53  | 39.73  | 136.57  | 433.78  |
| Mkla ren| 0.35 | 0.73 | 2.11 | 5.87 | 17.05| 53.69  | 183.74 | 637.17  | 2070.78 |
| SGPB   | 3.47 | 4.06 | 9.13 | 14.76| 21.32| 90.28  | 387.06 | 1011.26 | 3202.28 |
| CSI    | 0.58 | 1.31 | 3.79 | 17.19| 101.27| 246.11 |       |         |         |           |
| Uniform| 0.07 | 0.44 | 4.73 | 44.31| 593.77|        |        |         |         |           |
| L2-KRR | 0.11 | 0.55 | 8.2  | 115.83|        |        |        |         |         |           |
| Align  | 0.09 | 0.59 | 9.66 | 158.17|        |        |        |         |         |           |
| Linear | 0.29 | 2.78 | 52   | 1461.83|        |        |        |         |         |           |
| Convex | 0.31 | 2.83 | 52.46| 1463.16|        |        |        |         |         |           |

Table 7
Timings in seconds for increasing number of kernels (p) for n = 1000. Sorting and notation is defined as in Table 6.

| method | p   | 10  | 17  | 31  | 56  | 100 | 177 | 316 |
|--------|-----|-----|-----|-----|-----|-----|-----|-----|
| RFF    | 0.11| 0.16| 0.25| 0.35| 0.85| 1.29| 2.27|     |
| Uniform| 4.59| 6.09| 5.57| 7.36| 10.54|16.34|27.07|     |
| Mkla ren| 1.97| 2.86| 4.93| 8.42| 14.27|23.98|41.4 |     |
| L2-KRR | 7.58| 7.99| 8.95|10.61|13.68|19.75|32.84|     |
| Align  | 9.3 | 13.08|18.91|30.06|46.66|79.9 |142.59|     |
| SGPB   | 8.92| 13.91|29.48|69.56|     |     |     |     |
| Linear | 51.96| 128.55|384.29|1194.68|     |     |     |     |
| Convex | 52.4| 129.33|384.72|1190.79|     |     |     |     |

5. Conclusion

Subquadratic complexity in the number of training examples is essential in large-scale application of kernel methods. Learning the kernel matrix efficiently from the data and the selection of relevant portions on the data early reduces the time and storage requirements. Using a greedy low-rank approximation to multiple kernels, we achieve linear complexity in the number of kernels and data points without sacrificing the consideration of in-between kernel correlations.

When used with low-rank approximations, the unweighted sum of kernel matrices is not equal to concatenation of the feature spaces induced by the kernels. Sampling the inducing point-centered basis functions only from the relevant kernels enables using different hyperparameters in different regions of the input space. This property proved important both in matching the output function shapes in one-dimensional, continuous signals, string data sets, as well as in multidimensional data sets.

The Mklaren algorithm is a flexible approach for multiple kernel regression. It is kernel function independent, with data assumed to be accessed only through kernels. Nevertheless, its performance is comparable with methods that are optimized for specific kernels or types of input spaces. Natural extensions of the approach would include different target tasks, such as classification, ranking or general linear models. If the used kernels are differentiable with respect to hyperparameters, the latter could also be optimized at the time of approximation by appropriate modification of the inducing point selection heuristic. With the abundance of different data representations, we expect kernel methods to remain ubiquitous in machine learning applications.

Funding

This work was supported by grants P2-0209, J7-5460, J1–8150, and J3–9263 from the Slovenian Research Agency.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi: 10.1016/j.neucom.2019.02.030.

References

[1] A.E. Alaoui, M.W. Mahoney, Fast randomized kernel methods with statistical guarantees, Statistics 1050 (2014) 1–17.
[2] J. Alcalá-Fdez, L. Sanchez, S. Garcia, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, Others: KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput-A Fusion Found, Methodol, Appl. 13 (3) (2009) 307–318.
[3] F. Bach, Sharp analysis of low-rank kernel matrix approximations, in: Conference Learning Theory, 2013, pp. 185–209.
[4] F.R. Bach, M.I. Jordan, Predictive low-rank decomposition for kernel methods, in: Proceedings of the International Conference on Machine Learning (ICML), ACM Press, New York, New York, USA, 2005, pp. 33–40.
[5] C.M. Bishop, Pattern Recognition and Machine Learning, Information Science and Statistics, Springer, 2006.
[6] J. Blitzer, M. Dredze, F. Pereira, Biographies, Bollywood, boom-boxes and blenders: domain adaptation for sentiment classification, in: Proceedings of the Association for Computational Linguistics (ACL), 7 (2007) pp. 440–447.
[7] Y. Cao, M. Brubaker, D. Fleet, A. Hertzmann, Efficient optimization for sparse Gaussian process regression, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2015) 1–1.
[8] C. Cortes, M. Mohri, A. Rostamizadeh, L-2 regularization for learning Kernels, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2009, pp. 109–116.
[9] C. Cortes, M. Mohri, A. Rostamizadeh, Algorithms for learning Kernels based on centered alignment, J. Mach. Learn. Res. 13 (2012) 795–828.
[10] J. Demmler, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[11] B. Efron, T. Hastie, Least angle regression, Ann. Stat. 32 (2) (2004) 407–499.
[12] S. Fine, K. Scheinberg, Efficient SVM training using low-rank kernel representations, J. Mach. Learn. Res. 2 (2001) 243–264.
[13] Y. Gal, R. Turner, Improving the Gaussian process sparse spectrum approxima-
tion by representing uncertainty in frequency inputs, in: Proceedings of the International Conference on Machine Learning, (ICML), 2015.

Declarations of interest

None.


Z. proteus., divergences, ear (ICML), classifiers, J. programming, A. processes spasta.2018.02.002

B. Kulis, M. Sustik, L.S. Dhillon, Low-rank Kernel learning with Bregman matrix divergences, J. Mach. Learn. Res. 10 (2009) 341–376.

G. Lanckriet, N. Cristianini, Learning the kernel matrix with semidefinite programming, J. Mach. Learn. Res. 5 (2004) 27–72.

Q. Le, T. Särös, A. Smola, Fastfood—approximating kernel expansions in logarithmic time, in: Proceedings of the International Conference on Machine learning (ICML), 28, 2013.

M. Li, W. Bi, J.T. Kwok, B.-L. Lu, Large-scale Nyström kernel matrix approximation using randomized SVD, IEEE Trans. Neural Netw. Learn. Syst. 26 (1) (2015) 152–164.

C.D. Meyer, Matrix Analysis and Applied Linear Algebra, SIAM, 2000.

T.C. Mills, T.C. Mills, Time Series Techniques for Economists, Cambridge University Press, 1991.

Y. Mohsenzadeh, H. Sheikhzadeh, S. Member, Gaussian kernel width optimization for sparse Bayesian learning, IEEE Trans. Neural Netw. Learn. Syst. 26 (4) (2015) 709–719.

D. Ongie, T. Gärtner, Nyström method with Kernel K-means++ samples as landmarks, in: Proceedings of the 34th International Conference on Machine Learning, 70, 2017, pp. 2652–2660.

J. Pennington, F.X. Yu, Spherical random features for polynomial kernels, in: Advances in Neural Information Processing Systems (NIPS), 2015, pp. 1837–1845.

A. Rahimi, B. Recht, Random features for large-scale kernel machines, in: Advances in Neural Information Processing Systems (NIPS), 2007, pp. 1177–1184.

A. Rakotomamonjy, S. Chanda, -norm multiple kernel learning with low-rank kernels, Neurocomputing 143 (2014) 68–79.

C.E. Rasmussen, in: Gaussian Processes for Machine Learning, MIT Press, 2006.

A. Rudi, R. Camoriano, L. Rosasco, Less is more: Nyström computational regularization, Adv. Neur. In. (2015) 1657–1665.

B. Schölkopf, A.J. Smola, Learning with kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, 2002.

S. Si, C.-J. Hsieh, I. Dhillon, Memory efficient kernel approximation, in: Proceedings of the International Conference on Machine learning (ICML), 32, 2014.

E. Snoek, Z. Ghahramani, Sparse Gaussian processes using pseudo-inputs, in: Proceedings of the International Conference on Machine learning (ICML), 2006.

S. Sonnenburg, G. Rätsch, B. Schölkopf, Large scale genomic sequence SVM classifiers, in: Proceedings of the International Conference on Machine learning (ICML), ACM, 2005, pp. 848–855.

Z. Szabo, Optimal rates for random Fourier features, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 2015, pp. 1144–1152.

J.-F. Tao, S. Flaxman, D. Sedjrović, S. Bhart, Spatial mapping with Gaussian processes and nonstationary fourier features, Spat. Stat. (2017), doi: 10.1016/j.spasta.2018.02.002.

A. Vedaldi, A. Zisserman, Efficient additive kernels via explicit feature maps, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 480–492, doi:10.1109/TPAMI.2011.153.

P. Vincent, Y. Bengio, Kernel matching pursuit, Mach. Learn. 48 (1–3) (2002) 165–187, doi:10.1023/A:1019355821559.

C. Williams, M. Seeger, Using the Nyström method to speed up kernel ma- chines, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 2001, pp. 682–688.

A. Wilson, H. Nickisch, Kernel interpolation for scalable structured Gaussian processes (KISS-GP), Int. Conf. Mach. Learn. (2015) 1775–1784.

A. Wilson, R. Adams, Gaussian process kernels for pattern discovery and extrapolation, Int. Conf. Mach. Learn. (2013) 1067–1075.

Z. Xu, R. Jin, B. Shen, S. Zhu, Nyström approximation for sparse kernel methods: theoretical analysis and empirical evaluation, in: Proceedings of the Conference on Artificial Intelligence (AAAI), 2015, pp. 3115–3121.

Z. Yang, A. Wilson, A. Smola, L. Song, A la carte-learning fast kernels, Artificial Intelligence and Statistics, 2015, pp. 1098–1106.

---

Martin Stršlar received a masters and a doctoral degree in computer science from the University of Ljubljana, Faculty of Computer and Information Science in 2013 and 2018, respectively.

His research interest span scalable machine learning, data integration, kernel methods and Bayesian statistics. The main applications of the developed models are in bioinformatics: modelling with next-generation sequencing (NGS) data sets, protein-RNA iterations and single-cell RNA sequencing.

Dr. Stršlar was the recipient of the 2012 iGEM Best Health and Medicine prize, and 2012 iGEM Best Modeling prize. He is one of the core contributors of data mining software platforms Orange (https://orange.biolab.si) and Single cell Orange (https://singlecell.biolab.sl).

Tomáš Curk received a doctoral degree in computer science from the University of Ljubljana, Faculty of Computer and Information Science in 2007.

His research interest include bioinformatics, machine learning and data integration, with applications in modeling RNA-seq data on gene expression and iCLIP and RB-Dmap data on protein-RNA interaction.

Dr. Curk serves as Vice-Dean for research at the Faculty of Computer and Information Science from 2016. He is one of the initial contributors to the data mining software Orange (http://orange.biolab.si), gene expression analysis software dictyExpress (http://dictyexpress.org), gene interaction analysis software SNPsyn (http://snpSyn.biolab.com/tomaz/icount).