Approximate l-fold Cross-Validation with Least Squares SVM and Kernel Ridge Regression

Richard E. Edwards, Hao Zhang, Lynne E. Parker
Electrical Engineering and Computer Science
University of Tennessee
Knoxville, TN, United States
Email: {redwar15,haozhang,parker}@eecs.utk.edu

Joshua R. New
Whole Building and Community Integration Group
Oak Ridge National Lab
Oak Ridge TN, United States
Email: newjr@ornl.gov

***Abstract*—Kernel methods have difficulties scaling to large modern data sets. The scalability issues are based on computational and memory requirements for working with a large matrix. These requirements have been addressed over the years by using low-rank kernel approximations or by improving the solvers' scalability. However, Least Squares Support Vector Machines (LS-SVM), a popular SVM variant, and Kernel Ridge Regression still have several scalability issues. In particular, the $O(n^3)$ computational complexity for solving a single model, and the overall computational complexity associated with tuning hyperparameters are still major problems. We address these problems by introducing an $O(n \log n)$ approximate l-fold cross-validation method that uses a multi-level circulant matrix to approximate the kernel. In addition, we prove our algorithm's computational complexity and present empirical runtimes on data sets with approximately one million data points. We also validate our approximate method's effectiveness at selecting hyperparameters on real world and standard benchmark data sets. Lastly, we provide experimental results on using a multi-level circulant kernel approximation to solve LS-SVM problems with hyperparameters selected using our method.**

I. Introduction

The amount of digital data generated each year is increasing on an exponential scale. This data is generated through mobile devices, purchases, social networks, and much more. Understanding and extracting information from these large data sets is pivotal to advancing science and industry's decision making capabilities. However, existing statistical methods continue to struggle with handling problems that involve a large number of data points.

In particular, kernel methods have several difficulties. It is well known that a full-rank kernel matrix, representing n data points, requires $O(n^2)$ memory, and that the computational requirements greatly depend upon the learning technique. For example, traditional Support Vector Machine (SVM) [1] requires solving an optimization problem that scales linearly in n, but requires either storing the entire kernel matrix or recomputing it when needed [2]. SVM scalability problems have been greatly addressed over the years, either through problem decomposition [2] or by using faster methods to solve SVMs [3].

However, Least Squares SVM (LS-SVM) [4], a popular SVM variant, still suffers from several scalability issues. These issues include the computational complexity required for learning, $O(n^3)$, and tuning the learning method's hyperparameters, where n represents the total number of training examples. Solving an LS-SVM optimization problem involves solving a system of linear equations, which scales with the size of the kernel matrix. In addition, selecting hyperparameters that allow the learned model to generalize requires solving several LS-SVM problems and evaluating a model selection criteria function. Generally, l-fold cross-validation (CV) or a penalized criteria function[1] is used to estimate how well the resulting model will generalize. Using the exact criteria function in [5] requires $O(n^3)$ per model. While using l-fold CV requires $O(ln^3)$ per model with a naive implementation, better implementations require $O(n^3)$ and the exact Leave-one-out (LOO) CV method requires $O(n^2)$ [6], [7].

While SVM requires hyperparameter tuning as well, improved solvers reduce the computational cost per evaluation considerably. Reducing the overall computational cost for LS-SVM is primarily achieved by approximating the kernel matrix with a low-rank decomposition. However, a low-rank decomposition still produces an $O(m^3 + m^2 n)$ runtime per evaluation, where m is the rank of the approximation [8]. Low-rank approximation combined with an efficient l-fold CV implementation requires $O(m^2 n)$ computation [6]. Placing a restriction on m causes the computation to scale linearly in terms of n, if m is sufficiently smaller than $\frac{n}{2}$ [6]. However, m being sufficiently small is not practical for all data sets, which is demonstrated in [6].

Therefore, we are proposing a new approximation method for evaluating l-fold CV that scales solely in terms of n for all data sets. By combining a relatively new kernel approximation method from [9] with the efficient l-fold CV method in [6], we show that it is possible to evaluate each model in $O(n \log n)$ time. We provide proofs that establish the theoretical runtime and show validity, as well as experimental results with real world and benchmark data sets, and runtimes on large randomly generated data sets that establish scalability. Lastly, we show that our method is applicable to Kernel Ridge Regression (KRR), just like the original efficient l-fold CV method in [6].

The remainder of this paper is organized as follows. Section

[1]Penalized criteria functions combine a goodness of fit function with a penalty term.

II discusses previous work in hyperparameter tuning with LS-SVM. Section III presents required background information. Section IV presents proofs that establish algorithmic correctness and the $O(n \log n)$ runtime. Section V describes our experiments and discusses the experimental results. Section VI concludes the paper with a summary of our findings and future directions.

II. Related work

As mentioned in the introduction, selecting the optimal hyperparameters requires searching a continuous parameter space and evaluating an LS-SVM model per candidate hyperparameter setting. Two very common methods for evaluating model quality are penalized criteria functions and l-fold CV. L. Ding et al. [10] use the same approximation method as this work to reduce the computational complexity associated with solving the model and evaluating the criteria function from [5]. However, L. Ding et al.'s usage is very inefficient. Their overall runtime is $O(ln \log n)$, where l is the total number of folds. However, our method eliminates the redundancy and obtains an $O(n \log n)$ runtime. Section III-C discusses the approximation method in detail.

In addition, the criteria function in L. Ding et al.'s work is a biased estimator [10]. The authors argue, based on [11], that a model selection criteria is not required to be an unbiased estimator and that the model selection criteria only needs to provide a reliable estimation of the generalization error in hyperparameter space. While this argument is compelling, Cawley et al. [11] actually states that the model selection criteria needs to provide a reliable estimate of the true minimum test error, which can easily be obscured by bias when the true test error represents a complex surface in the hyperparameter space – especially since the criteria function in Song et al. [5] is convex (i.e. has a single minimum).

More work focuses on optimizing l-fold CV performance, because it tends to be a good indicator of generalization error [12]. Additionally, LOO CV provides an almost unbiased estimate of the true generalization error [13]. This is why [7] and [14] focus on using matrix properties to reduce the computational cost associated with computing a matrix inverse per fold. Both approaches achieve an O(n^2) runtime for LOO CV.

An et al.'s method [6] achieves an $O(n^3)$ runtime for exact l-fold CV and LOO CV. In addition, they present an approximate l-fold CV method, using the same algorithms, that uses a low-rank kernel approximation with an $O(nr^2)$ runtime, where r is the approximate kernel's rank. Their algorithm supports l-fold CV with LS-SVM classification, LS-SVM regression, and KRR [6]. While their exact method is slower than [7], [14], combining this method with the Song et al.'s kernel approximation [9] allows us to reduce the computational complexity to $O(n \log n)$, and compute a fast approximation of the almost unbiased LOO CV measure.

III. Preliminaries

This section provides necessary background information, including a short review of LS-SVM, KRR, approximating kernel matrices with multi-level circulant matrices, and the l-fold CV method by [6]. Our approximate l-fold CV approach is based on combining the last two items to create an even faster method.

A. Least squares support vector machine

Given a training set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^m$ and $y \in \{-1, 1\}$, LS-SVM can learn a classification function with the following form:

$$y(x) = sign[w^\mathsf{T} \varphi(\vec{x_i}) + b] \tag{1}$$

where $\varphi(*)$ represents a non-linear function that maps x to a higher dimensional feature space. This method can be used to learn regression functions by removing the sign decision component, and changing y's membership to $y \in \mathbb{R}$.

LS-SVM's learning method is based on two modifications to the well known SVM formulation [4]. Firstly, the loss function is defined as the squared error over all samples. Secondly, the inequality constraints are replaced with equality constraints. These two differences lead to the following formulation:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i^2 \tag{2}$$

with the following constraints:

$$w^\mathsf{T} \varphi(\vec{x_i}) + b + \xi_i = y_i \tag{3}$$

By applying Lagrange multipliers, the primal problem is transformed into the dual problem, which has the following closed form solution [15]:

$$\begin{bmatrix} 0 & 1_n^\mathsf{T} \\ 1_n & K + \frac{1}{\gamma} I_n \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix} \tag{4}$$

where $y = [y_1, y_2, \ldots, y_n]^\mathsf{T}$, $1_n = [1, 1, \ldots, 1]^\mathsf{T}$, $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_n]^\mathsf{T}$, and K represents the kernel matrix. Solving this system of equations produces the following classification function:

$$y(x) = sign[\sum_{i=1}^{n} \alpha_i K(\vec{x}, \vec{x_i}) + b] \tag{5}$$

where α and b are the solutions to equation 4.

B. Kernel ridge regression

KRR aims to find a non-linear function that represents the dependencies between the covariates $\{x_1, x_2, \ldots, x_n\}$ and the response variable $\{y_1, y_2, \ldots, y_n\}$ [16]. More precisely, KRR minimizes the following function:

$$\sum_{i=1}^{n} (y_i - w^\mathsf{T} \varphi(\vec{x_i})) + \lambda ||w||^2 \tag{6}$$

where λ represents a fixed penalty. Transforming this problem to the dual representation yields the following closed form solution:

$$(K + \lambda I)\alpha = y \tag{7}$$

where I is the identity matrix and K is the kernel matrix. Solving this system of equations yields the following non-linear function

$$y(x) = \sum_{i=1}^{n} \alpha_i K(\vec{x}, \vec{x_i}) \tag{8}$$

which is a special form of LS-SVM, $b = 0$.

C. Kernel approximation via multi-level circulant matrices

In order to provide a general definition for multi-level circulant matrices and to explain the approximation algorithm introduced by [9], we need to review standard multi-level matrices and multi-level indexing. Multi-level indexing is based on factorizing each dimension of an $m \times n$ matrix into regions such that $n = n_0 n_1 \ldots n_{p-1}$ and $m = m_0 m_1 \ldots m_{p-1}$ [17]. Each factor n_i and m_i specifies a level within the matrix and an index range for each dimension. This means a multi-level matrix is a matrix whose individual elements can be specified by two p dimensional index vectors and the p factors.

A circulant matrix is an $n \times n$ matrix that can be completely specified by its first row, because all remaining $n-1$ rows are permutations of the first row. A p-level circulant matrix is an extension that is defined as an $m \times m$ multi-level matrix where each level defines a $(p-i)$-level circulant matrix, where i represents the level. Each level i, other than level 1, can be viewed as an $n_i \times n_i$ block matrix where each sub matrix has dimension $\prod_{j=i+1}^{p-1} n_j \times \prod_{j=i+1}^{p-1} n_j$, and all rows in the block matrix are permutations of the first row. Level 1 is a standard circulant matrix. This means a p-level circulant matrix A can be completely represented by its first row $a_{0,l}$ where $0 = (0, 0, \ldots, 0)$ and l is a multi-level index. Representing the approximate kernel matrix as a p-level circulant matrix reduces the memory storage requirement from $O(n^2)$ to $O(n)$.

Algorithm 1 Kernel Approximation with P-level Circulant Matrix

```
Input: $M$ (Kernel's size), $F = \{n_0, n_1, \ldots, n_{p-1}\}$, $k$ (Kernel function)
 1: $N \leftarrow$ {All multi-level indices defined by F}
 2: $T \leftarrow zeros(M)$, $U \leftarrow zeros(M)$
 3: $H_n \leftarrow \{x_0, x_1, \ldots, x_{p-1}\} \in \mathbb{R}^p$ s.t. $\forall x_i \in H_n$, $x_i > 0$
 4: for all $j \in N$ do
 5:     $T_j \leftarrow k(||jH_n||_2)$
 6: end for
 7: for all $j \in N$ do
 8:     $D_j \leftarrow D_{j,0} \times D_{j,1} \times \cdots \times D_{j,p-1}$
 9:     $U_j \leftarrow \sum_{l \in D_j} T_l$
10: end for
11: $\tilde{K} \leftarrow U$
Output: $\tilde{K}$
```

Algorithm 1 presents Song et al.'s method for constructing the approximate kernel matrix [9]. The algorithm's runtime is $O(n + n2^p)$. The exponential component is from the fact that D_j grows exponentially in size as p increases. This is because $D_{j,s}$, where s represents the level, is defined as $D_{j,s} = \{0\}$ if $j_s = 0$ else $D_{j,s} = \{j_s, F(s) - j_s\}$, which means every additional factor increases the overall number of elements used in the set cross product. However, L. Ding et al. [10] showed that a small p (e.g., two or three) is sufficient for adequate approximation on classification problems. This means it is possible to fix p such that the runtime is $O(n)$.

More importantly, this approximation method does not use the original features within the data set, nor does it use transformations from the data. The approximation method constructs a kernel matrix that mimics the RBF kernel. Generically mimicking the RBF kernel is possible because all mappings to Hilbert space are many-to-one, and these mappings are controlled by the the kernel hyperparameters. This property reduces the technique's overall ability to support other kernel types, but provides large computational advantages by making the approximation a multi-level circulant matrix.

Approximation quality for this method was well established in Song et al. and L. Ding et al. [9], [10]. Song et al. proved that the approximate kernel matrix converges to the true kernel matrix as the number of factors, p, approaches infinity, where p is the number of factors used to partition the multi-level matrix and is equal to H_n's cardinality. Based on this proof and the established approximation runtime, $O(n + n2^p)$, a perfect approximation becomes exponentially difficult to compute. However based on L. Ding et al.'s research, a small number of factors produces adequate approximations.

While Song et al. and L. Ding et al. present excellent studies over the approximation's capabilities, our results show that the real numbers selected for H_n, a set of p positive real numbers selected by the user, greatly influence approximation quality (Section V). By studying how H_n influences performance, we illustrate that this method decreases computational complexity and increases the hyperparameter search space by p.

D. Cross-validation method

Algorithm 2 shows the process for calculating the l-fold CV error with LS-SVM classification. This method is centered around computing a single matrix inverse and solving l systems of equations, where l is the number of folds. The single inverse involves computing K_γ^{-1} and the l systems of equations are based on using the diagonal block matrices found in C (line 2) to estimate the predicted labels for the omitted folds. These diagonal block matrices are denoted C_{kk} and have dimension $n_v \times n_v$ where $n_v \simeq \frac{n}{l}$, k is the k-th fold, and n is the number of data points. An et al. [6], noticed that solving $C_{kk}\beta_k = \alpha_k$ provides the estimated error per omitted example in the k-th fold.

This algorithm can be easily modified to support KRR and LS-SVM regression. Removing the sign component and changing the error function are the only modifications required to change to LS-SVM regression. Changing to KRR regression requires setting C to K_γ^{-1}, setting α to $K_\gamma^{-1}y$, and making the LS-SVM regression modifications [6].

Algorithm 2 Efficient Cross-Validation

```
Input: $K$ (Kernel matrix), $l$ (Number folds), $y$ (response)
 1: $K_\gamma^{-1} \leftarrow inv(K + \frac{1}{\gamma}I), \quad d \leftarrow 1_n^\intercal K_\gamma^{-1} 1_n$
 2: $C \leftarrow K_\gamma^{-1} + \frac{1}{d} K_\gamma^{-1} 1_n 1_n^\intercal K_\gamma^{-1}$
 3: $\alpha \leftarrow K_\gamma^{-1} y + \frac{1}{d} K_\gamma^{-1} 1_n 1_n^\intercal K_\gamma^{-1} y$
 4: $n_k \leftarrow size(y)/l, \quad y^{(k)} \leftarrow zeros(l, n_k)$
 5: for $k \leftarrow 1$, $k \leq l$ do
 6:     Solve $C_{kk}\beta_{(k)} = \alpha_{(k)}$
 7:     $y^{(k)} \leftarrow \mathsf{sign}[y_{(k)} - \beta_{(k)}]$
 8:     $k \leftarrow k + 1$
 9: end for
10: $error \leftarrow \frac{1}{2}\sum_{k=1}^{l}\sum_{i=1}^{n_k} |y_i - y^{(k,i)}|$
Output: $error$
```

IV. $O(n \log n)$ Approximate l-fold cross-validation

Solving LS-SVM and KRR problems requires computing the penalized kernel matrix's inverse, K_γ^{-1}. However, computing K_γ^{-1} takes $O(n^3)$ operations. In addition, storing the full kernel matrix requires $O(n^2)$ memory. Using the exact kernel matrix is clearly not scalable to large problems. There are numerous published methods that find low-rank kernel approximations [8], as well as proofs that establish upper bounds on the approximation error [18]. The low-rank approximations produced by these methods are not guaranteed to be bounded in rank unless the bound is explicitly specified [6]. Specifying a bound that is too small will greatly degrade the approximation's representative power, while not specifying a bound can lead to a large number of operations to compute the approximate K_γ^{-1}.

On the other hand, the p-level circulant kernel matrix approximation provides superior computational performance over these other methods without a user defined bound. Its inverse can be computed in $O(n \log n)$, via the Fast Fourier Transform (FFT), and it only requires $O(n)$ memory, because the entire approximation can be represented as a single row vector. Despite these computational advantages, a naive l-fold CV approach using this approximation requires $O(ln \log n)$ time, which is $O(n^2 \log n)$ for LOO CV. This means evaluating the criteria in [10] on multiple folds is very inefficient, and is computationally inferior to the exact method as l increases.

By combining the p-level circulant kernel approximation (i.e., Algorithm 1) and the exact l-fold CV method (i.e., Algorithm 2), we overcome this limitation and fully utilize the approximation's computational capabilities. Replacing the kernel matrix used in Algorithm 2 with a p-level circulant kernel matrix reduces the overall algorithmic complexity to $O(n \log n)$ for all l. In addition, the algorithm is still a valid l-fold CV algorithm. To show both properties, we first need to prove Lemma 4.1 and Lemma 4.2. Using these Lemmas, we prove Theorem 4.3 and Theorem 4.4 that show our approximate l-fold CV algorithm is valid and has computational complexity $O(n \log n)$.

Lemma 4.1: If K is a p-level circulant matrix, then C (line 2) is also a p-level circulant matrix.

Proof: P-level circulant matrices are closed under inverse. In addition, p-level circulant matrices are closed under addition and multiplication if all matrices have the same level and factorization. This means that K_γ^{-1} is a circulant matrix based on these closure properties. In addition, $1_n 1_n^\intercal$ represents a matrix of all 1s with dimension $n \times n$, which means $1_n 1_n^\intercal$ is a p-level circulant matrix with the same factorization as K_γ^{-1}. Therefore, based on closure properties, C is a p-level circulant matrix with the same factorization as K. ■

Lemma 4.2: If K is a p-level circulant matrix with factorization $n = n_0 n_1 \dots n_{p-1}$ and $l = n_0 n_1 \dots n_s$ s.t. $s \leq p-1$, then C_{kk} is a $(p-s)$-level circulant matrix with dimension $\frac{n}{l} \times \frac{n}{l}$.

Proof: Based on Lemma 4.1, C is a p-level circulant matrix. Therefore, C can be partitioned into block matrices, based on its factorization, such that each block matrix is a $(p-s)$-level circulant matrix, where s is the level. Factorizing C according to $n_0 n_1 \dots n_s$ s.t $s \leq p-1$, produces a block matrix with exactly $n_0 n_1 \dots n_s \times n_0 n_1 \dots n_s$ blocks. This means there are exactly $l \times l$ blocks. In addition, each block must have dimension $\prod_{j=s+1}^{p-1} n_j \times \prod_{j=s+1}^{p-1} n_j$, which is equal to $\frac{n_0 n_1 \dots n_{p-1}}{n_0 n_1 \dots n_s} \times \frac{n_0 n_1 \dots n_{p-1}}{n_0 n_1 \dots n_s}$. Therefore, all of C's diagonal matrices are $(p-s)$-level circulant matrices with dimension $\frac{n}{l} \times \frac{n}{l}$. ■

Theorem 4.3: If K is a p-level circulant matrix with factorization $n = n_0 n_1 \dots n_{p-1}$ and $l = n_0 n_1 \dots n_s$ s.t. $s \leq p-1$, then the computational complexity for Algorithm 2 is O($n \log n$).

Proof: Based on Lemma 4.1 and Lemma 4.2, C_{kk} is a $(p-s)$-level circulant matrix with dimensions $\frac{n}{l} \times \frac{n}{l}$. This means the computational complexity associated with solving $C_{kk}\beta_k = \alpha$ is $O(\frac{n}{l} \log \frac{n}{l})$, because the FFT algorithm can be used to solve for β_k and has an $O(n \log n)$ computational complexity. Therefore, solving all l systems of equations requires $O(l(\frac{n}{l} \log \frac{n}{l}))$. When $l \neq n$ the computational complexity is $O(n \log n - n \log l)$, which is $O(n \log n)$, and if $l = n$, the computational complexity is $O(n)$. In addition, computing K_γ^{-1} requires $O(n \log n)$ time, via the same algorithm. This means if the computation required to compute C and α is bounded by $O(n \log n)$, then the overall computational complexity is $O(n \log n)$. Based on [9] and [10], multiplying two p-level circulant matrix requires $O(n)$ computational time after diagonalization, and diagonalizing each p-level circulant matrix requires $O(n \log n)$ time using the FFT algorithm. In addition, the final result can be recovered in $O(n \log n)$ time via the inverse FFT. The same process applies to addition as well. Therefore, C and α can be computed in $O(n \log n)$ time, and the overall runtime is $O(n \log n)$ for LS-SVM. Note, it is even simpler to show that the runtime is $O(n \log n)$ for KRR, because $C = K_\gamma^{-1}$ and $\alpha = K_\gamma^{-1}y$. ■

Theorem 4.4: If K is a p-level circulant matrix with factorization $n = n_0 n_1 \dots n_{p-1}$ and $l = n_0 n_1 \dots n_s$ s.t. $s \leq p-1$, then Algorithm 2 is valid (i.e., Algorithm 2 does not violate Theorem 1 in An et al. [6]).

Proof: Based on Lemma 4.2, all C_{kk} matrices have dimension $\frac{n}{l} \times \frac{n}{l}$. In addition, there are exactly l matrices along C's diagonal with those dimensions. This means combing this approximation method with Algorithm 2 does not violate the theorem presented in [6] that establishes algorithmic correctness. Therefore, this approximation is valid. ■

Based on Theorem 4.3, this approximation algorithm has computational complexity $O(n \log n)$, if l can be represented as $n_0 n_1 \dots n_s$ s.t. $s \leq p - 1$. This means all l are valid, if the approximation's factorization is controlled accordingly. This requirement is much lighter than restricting the kernel approximation's rank, because it generalizes to all data sets.

V. Experimental Results

To empirically verify our approximate l-fold approach, we designed three experiment types that test our method with $l = n$, named approximate leave-one-out (A-LOO). The first is a scalability experiment that tests our A-LOO method and the exact $O(n^3)$ LOO method, named E-LOO, on randomly generated data that increases exponentially in size. Our second experiment tests three different aspects about our approach: 1) it tests how well our method can select hyperparameters, 2) it illustrates how H_n influences the approximate kernel, and 3) it compares how well an approximate LS-SVM solution computed using the p-level circulant matrix compares against the exact LS-SVM solution. The third experiment tests how well hyperparameters selected via A-LOO perform when used to compute the exact LS-SVM solution. All experiments were run on a laptop with a 2.2 GHZ Intel i7 CPU and 8G of RAM using MATLAB 2011b.

Table I presents the results from testing the E-LOO, A-LOO with LS-SVM, and A-LOO with KRR on randomly generated data sets. A-LOO with KRR scales better with the data set's size. The scalability difference between A-LOO with KRR and A-LOO with LS-SVM is based on the overhead associated with computing C and α. As mentioned previously, the KRR method can directly compute C and α in two $O(n \log n)$ steps, while the LS-SVM approach is several $O(n \log n)$ and $O(n)$ steps. Despite the factor of 4 difference in runtime, plotting the times for both methods as a loglog plot reveals that both algorithms scale with the same slope (Figure 1). This means, empirically, both algorithms have the same Big-O complexity.

These scalability experiments were conducted using a single CPU core. This makes it easier to compare our times with the results in [6], which showed that their $O(nr^2)$ approximation method scaled approximately linearly for 500 to 5000 data points (Figure 2). Plotting their times and our runtimes on a loglog plot indicates that our A-LOO with LS-SVM scales no worse than their method. However, the runtimes from [6] were computed with an r that was guaranteed to never be greater than half the data points, and the experiments were run on a single-core 3.2 GHZ desktop. This implies that our algorithm has better scalability, and less assumptions that directly impact runtime, but can impact prediction performance.

Our second experiment set uses 10 classification benchmark data sets used by An et al. [6] and L. Ding et al. [10]. Table

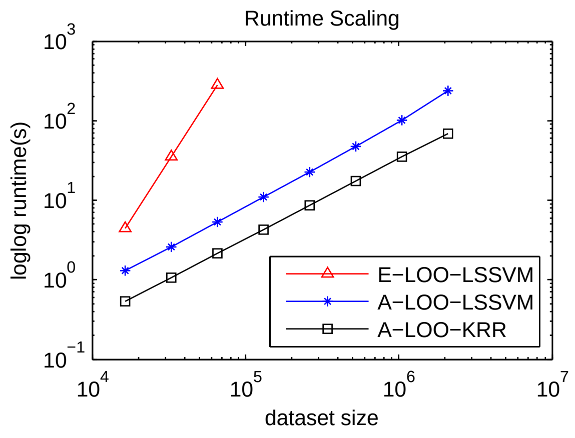

Fig. 1. A loglog runtime comparison across E-LOO-LSSVM, A-LOO-LSSVM, and A-LOO-KRR.

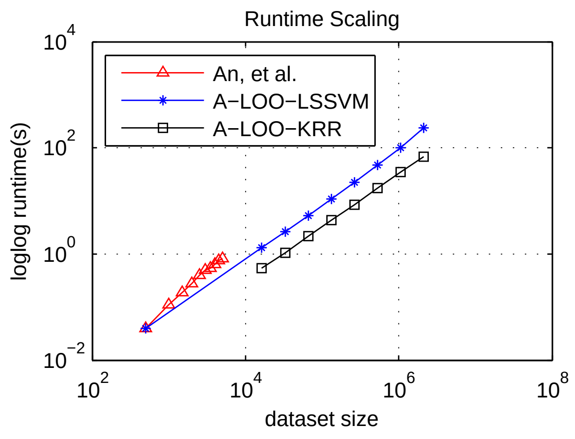

Fig. 2. A loglog runtime comparison across An, et al.'s A-LOO-LSSVM, A-LOO-LSSVM, and A-LOO-KRR.

II shows our results, and illustrates how classification error varies with respect to H_n (Algorithm 1), H_n's values were restricted to the sets $(1, 2)$ or $(10, 11)$. In addition, Table II shows how well an approximate LS-SVM solution fits the test data with hyperparameters selected using our A-LOO. The exact testing error in Table II is mostly the same as the exact error found in [6], but a few differ because we used a different grid-search resolution. However, the differences are not significantly large, implying we implemented the original l-fold algorithm correctly.

More importantly, the approximate LS-SVM error has less than a 5% difference from the exact error on 7 of the 10 data sets (1–7 in Table II), and presents moderate performance on 8–10 with less than a 15% difference. However, the performance on data sets 5, 7, and 10 differ greatly from the ones reported by L. Ding et al. [10]. Based H_n's impact on the approximation model's quality and L. Ding et al. not reporting how their H_n relates to the data specifically, we can only conclude that they may have manually optimized the approximation matrices' H_n components or felt that optimizing over H_n values was self evident and did not need to

TABLE I
COMPUTATION TIME ON A SINGLE CPU CORE

# Examples	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}
E-LOO	4.43s	35.25s	281.11	–	–	–	–	–
A-LOO-LSSVM	1.3s	2.6s	5.32s	10.88s	22.45s	47.41s	101.36s	235.83s
A-LOO-KRR	0.54s	1.06s	2.14s	4.3s	8.55s	17.28s	35.39s	68.22s

TABLE II
APPROXIMATE LS-SVM VERSUS EXACT LS-SVM

Data set	#Train	#Test	A-Error(L. Ding, et al. [10])	A-Error $H_n \in (1, 2)$	A-Error $H_n \in (10, 11)$	E-Error
1) Titanic	150	2051	22.897±1.427	23.82±1.44	22.80±0.68	22.92±0.43
2) B. Cancer	200	77	27.831±5.569	29.87±5.59	26.75±5.92	25.97±4.40
3) Diabetes	468	300	26.386±4.501	25.67±1.13	25.27±2.07	23.00±1.27
4) F. Solar	666	400	36.440±2.752	35.65±2.78	36.65±2.47	33.75±1.44
5) Banana	400	4900	11.283±0.992	14.10±1.74	18.98±1.76	10.97±0.57
6) Image	1300	1010	4.391±0.631	17.64±1.52	6.89±0.73	2.47±0.53
7) Twonorm	400	7000	2.791±0.566	15.64±25.71	6.85±8.86	2.35±0.07
8) German	700	300	25.080±2.375	29.93±1.61	27.40±1.79	21.87±1.77
9) Waveform	400	4600	Not Reported	19.85±3.87	17.57±1.93	9.77±0.31
10) Thyroid	140	75	4.773±2.291	29.33±4.07	17.33±3.89	4.17±3.23

TABLE III
EXACT LS-SVM PERFORMANCE USING HYPERPARAMETERS SELECTED VIA OUR A-LOO

Data set	CoV(%)	MAPE(%)	CoV(%)[19]	MAPE(%)[19]
House 1	19.6±1.69	15.3±0.47	20.1±0.81	16.1±0.85
Sensor A	1.3±0.05	1.0±0.05	–	–
Sensor B	17.2±4.89	10.8±0.25	–	–
Sensor C	12.0±2.31	7.8±0.68	–	–
Sensor D	1.4±0.09	0.9±0.03	–	–
S1	13.1±0.00	10.0±0.00	13.7±0.00	11.2±0.00
S2	3.1±0.00	4.7±0.00	6.4±0.00	4.5±0.00

be reported or studied. While we are only able to hypothesize about the result difference based on H_n's influence, L. Ding et al.'s results indicate that it is possible to obtain better results using this approximate LS-SVM mode on these three data sets. However, obtaining better results is dependent upon better exploring the hyperparameter space and selecting better values for H_n. In addition, during our experimentation process, we observed that it is not that difficult to manually set H_n through trial and error on these benchmark data sets, but real world applications most definitely require an automated approach. This means in exchange for our performance enhancements the hyperparameter selection space is expanded from $\mathbb{R}^2$ to $\mathbb{R}^{p+2}$, where p refers to the number of approximation factors.

In order to evaluate how well our A-LOO method selects hyperparameters, we tested the exact LS-SVM solution with hyperparameters selected by our method on several regression data sets. Our regression experiments utilized two real world data sets and restricted the H_n values to fall in the interval $(1, 2)$. The first data set is from the Great Energy Prediction Shootout [20], which contains environmental measurements and an unknown building's electrical consumption. The objective is to predict the unknown building's electrical consumption using the environmental measurements. The second data set contains approximately 140 sensor measurements collected every hour for an entire year from a residential home, and was provided by the authors of [19]. The objective with this data set is to predict future hourly sensor values, using previous sensor measurements. In addition, we evaluated the final model using the common metrics from the building spaces community, Coefficient of Variance (CoV) [20] and Mean Absolute Percentage of Error (MAPE) [21].

Table III shows the results from using our A-LOO algorithm to select hyperparameters on the regression data sets. House 1 refers to the overall home's electrical consumption, Sensors A through D refer to four randomly selected sensors from the residential home, and S1 and S2 refer to different instances of the Great Energy Prediction Shootout data found in [21]. Our LS-SVM results on S1 are better than the LS-SVM results presented in [19], which presents the only results for LS-SVM applied to this data set. In addition, the CoV measure for S2 is better as well.

Our regression results on the residential data set are very good as well. Our MAPE measure for House 1 (Table III) is statistically better than the MAPE results presented in [19] for order[2] 1, 2, and 3 LS-SVM models. In addition, our CoV measure is statistically better than order 1 and 3 LS-SVM models. This means our order 1 model is better than the House 1 models presented in [19], which implies that better hyperparameter tuning can greatly improve prediction accuracy. In addition, the hyperparameters selected using our A-LOO produced models that perform very well on the four randomly sampled sensors (Table III). Therefore, our A-LOO method is empirically effective at selecting reliable

[2]Order 1, 2, and 3 refer to the Markov order used for predicting the future residential electrical consumption. For example, order 2 uses sensor measurements at time t and $t - 1$ to predict a value at $t + 1$.

hyperparameters.

VI. Conclusion and future work

In summary, we presented an approximate l-fold CV method for LS-SVM and KRR, and proved that the method has an $O(n \log n)$ runtime. In addition, we showed that LS-SVM and KRR can effectively scale to large data sets (Table 1). We noted our method with LS-SVM method scales no worse than the $O(nr^2)$ method presented in [6], when all assumptions about r are satisfied. Additionally, our method with KRR scales extremely well. In addition, we showed that our l-fold CV method can select hyperparameters that can be used to solve LS-SVM problems with p-level circulant kernels and exact kernels. However, the approximate solution's quality is greatly impacted by selecting good values for H_n. This leads to hyperparameter tuning in $\mathbb{R}^{p+2}$, but yields a very large performance increase in terms of scalability.

We showed empirically that our method performs well, but a formal proof showing that our criteria is consistent is still required. It should be possible to prove consistency based on the proof presented in [10], because their criteria is a penalized squared error measure using the same approximation method. In addition, we need to further explore tuning H_n, because solving the exact model with the optimal hyperparameters still requires $O(n^3)$ time. However, this may not be an actual problem based on the general distributed optimization method presented in [22].

VII. Acknowledgements

This work was funded by field work proposal CEBT105 under the Department of Energy Building Technology Activity Number BT0305000. We would like to thank Amir Roth for his support and review of this project. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. Our work has been enabled and supported by data analysis and visualization experts at the NSF funded RDAV (Remote Data Analysis and Visualization) Center of the University of Tennessee, Knoxville (NSF grant no. ARRA-NSF-OCI-0906324 and NSF-OCI-1136246).

Oak Ridge National Laboratory is managed by UT-Battelle, LLC, for the U.S. Dept. of Energy under contract DE-AC05-00OR22725. This manuscript has been authored by UT-Battelle, LLC, under Contract Number DEAC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

References

[1] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag, 1995.

[2] T. Joachims, "Making large-Scale SVM Learning Practical," *Advances in Kernel Methods Support Vector Learning*, pp. 169–184, 1999.

[3] C. H. Teo, Q. Le, A. Smola, and S. V. N. Vishwanathan, "A scalable modular convex solver for regularized risk minimization," in *In KDD. ACM*, 2007.

[4] J. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[5] G. Song and Y. Xu, "Approximation of kernel matrices by circulant matrices and its application in kernel selection methods," *Frontiers of Mathematics in China*, vol. 5, pp. 123–160, 2010.

[6] S. An, W. Liu, and S. Venkatesh, "Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression," *Pattern Recognition*, vol. 40, no. 8, pp. 2154 – 2162, 2007.

[7] G. C. Cawley and N. L. Talbot, "Fast exact leave-one-out cross-validation of sparse least-squares support vector machines," *Neural Networks*, vol. 17, no. 10, pp. 1467 – 1475, 2004.

[8] F. R. Bach and M. I. Jordan, "Predictive low-rank decomposition for kernel methods," in *Proceedings of the 22nd International Conference on Machine Learning*. ACM, 2005, pp. 33–40.

[9] G. Song and Y. Xu, "Approximation of high-dimensional kernel matrices by multilevel circulant matrices," *Journal of Complexity*, vol. 26, no. 4, pp. 375 – 405, 2010.

[10] L. Ding and S. Liao, "Approximate Model Selection for Large Scale LSSVM," *Journal of Machine Learning Research – Proceedings Track*, vol. 20, pp. 165–180, 2011.

[11] G. Cawley and N. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *The Journal of Machine Learning Research*, vol. 11, pp. 2079–2107, 2010.

[12] K. Duan, S. Keerthi, and A. N. Poo, "Evaluation of simple performance measures for tuning SVM hyperparameters," *Neurocomputing*, vol. 51, no. 0, pp. 41 – 59, 2003.

[13] Luntz, "On estimation of characters obtained in statistical procedure of recognition," *Technicheskaya Kibernetica 3*, 1969.

[14] T. Pahikkala, J. Boberg, and T. Salakoski, "Fast n-fold cross-validation for regularized least-squares," in *Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence*, 2006.

[15] J. Suykens, T. V. Gestel, J. D. Brabanter, B. D. Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. World Scientific, 2002.

[16] C. Saunders, A. Gammerman, and V. Vovk, "Ridge Regression Learning Algorithm in Dual Variables," in *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 1998, pp. 515–521.

[17] E. E. Tyrtyshnikov, "A unifying approach to some old and new theorems on distribution and clustering," *Linear Algebra and its Applications*, vol. 232, pp. 1–43, 1996.

[18] C. Cortes, M. Mohri, and A. Talwalkar, "On the impact of kernel approximation on learning accuracy," in *Conference on Artificial Intelligence and Statistics*, 2010, pp. 113–120.

[19] R. E. Edwards, J. New, and L. E. Parker, "Predicting future hourly residential electrical consumption: A machine learning case study," *Energy and Buildings*, vol. 49, pp. 591–603, 2012.

[20] J. Kreider and J. Haberl, "Predicting hourly building energy use: the great energy predictor shootout- overview and discussion of results," *ASHRAE Transactions*, vol. 100, no. 2, pp. 1104–1118, 1994.

[21] K. Li, H. Su, and J. Chu, "Forecasting building energy consumption using Neural Networks and Hybrid Neuro-Fuzzy system: a Comparative Study," *Energy and Buildings*, 2011.

[22] N. Parikh and S. Boyd, "Block splitting for large-scale distributed learning," in *Neural Information Processing Systems (NIPS), Workshop on Big Learning*, 2011.