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

Richard E. Edwards[1], Hao Zhang[1], Lynne E. Parker[1], Joshua R. New[2]

[1]Distributed Intelligence Lab
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville TN, USA

[2]Whole Building and Community Integration Group
Oak Ridge National Lab
Oak Ridge TN, United States

December 7, 2013

Funded by the United States
Department of Energy

Outline

Applying Kernel Methods to Large Datasets

- Direct Kernel application scales poorly
  - Requires $O(n^2)$ memory
  - Model solve time increases
  - Model selection time increases
- Scaling improvements
  - Faster model solvers
  - Problem decompositions
  - Low-rank Kernel approximations
- Most scaling improvements apply to standard SVMs

Applying Kernel Methods to Large Datasets

- Least Squares Support Vector Machine (LS-SVM)
  - Naive cross-validation model calibration complexity: $O(ln^3)$
  - Best exact leave-one-out (LOO) cross-validation complexity: $O(n^2)$
  - Best approximate cross-validation complexity: $O(m^2 n)$
- We can do better!
  - Approximate cross-validation complexity: $O(n \log n)$
  - Applies to LOO as well

Outline

Previous LS-SVM Model Selection

- T. Pahikkala et al (2006) and Cawley et al. (2004) obtained $O(n^2)$ LOO cross-validation
  - utilizes matrix inverse properties
- An et al. (2007) obtained $O(m^2 n)$ l-fold cross-validation
  - uses low-rank kernel approximation
  - removes redundancy from the validation process
  - introduces a new cross-validation algorithm
- L. Ding et al. (2011) obtained $O(ln \log n)$ l-fold cross-validation
  - $O(n^2 \log n)$ LOO cross-validation

Outline

Multi-Level Matrices

- Matrices indexed by factors
- Example 3-level matrix with factors: 2x2, 4x4, 2x2
  - $|M| = (2 \times 4 \times 2) \times (2 \times 4 \times 2)$
- Level 1:

$$M = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix}$$

- Level 2:

$$A_{00} = \begin{bmatrix} B_{00} & B_{01} & B_{02} & B_{03} \\ B_{10} & B_{11} & B_{12} & B_{13} \\ B_{20} & B_{21} & B_{22} & B_{23} \\ B_{30} & B_{31} & B_{32} & B_{33} \end{bmatrix}$$

- Level 3:

$$B_{00} = \begin{bmatrix} 5 & 6 \\ 1 & 2 \end{bmatrix}$$

Circulant Matrices

- A special Toeplitz matrix
- Its inverse is computed in $O(n \log n)$ via Fast Fourier Transform
- Example:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{bmatrix}$$

- General definition:

$$\begin{bmatrix} c_0 & c_1 & \cdots & c_n \\ c_n & c_0 & \cdots & c_{n-1} \\ \vdots & & \ddots & \vdots \\ c_1 & c_2 & \cdots & c_0 \end{bmatrix}$$

P-Level Circulant Matrices

- Combines Circulant Matrices and Multi-Level Circulant Matrices
  - Each level is a circulant matrix
  - All factors are now one diminsional
- Example 3-Level with factors 2, 4, 2:

$$M = \begin{bmatrix} A_0 & A_1 \\ A_1 & A_0 \end{bmatrix}$$

- Level 2:

$$A_0 = \begin{bmatrix} B_0 & B_1 & B_2 & B_3 \\ B_3 & B_0 & B_1 & B_2 \\ B_2 & B_3 & B_0 & B_1 \\ B_1 & B_3 & B_2 & B_0 \end{bmatrix}$$

- Level 3:

$$B_0 = \begin{bmatrix} 5 & 6 \\ 6 & 5 \end{bmatrix}$$

Outline

Overview

- We use same approximation method has L. Ding et al. (2011)
- We remove inefficienies from the cross-validation process
- Result: $n \log n$ LOO cross-validation
  - L. Ding et al.'s LOO cross-validation: $n^2 \log n$

Kernel Approximation via P-Level Circulant Matrices

- Song et al. (2010) introduced P-Level Circulant RBF Kernel approximation
  - allows $n \log n$ model solve time
  - allows fast model selection
- approximation converges as matrix level factors approach infinity
- result: $O(n + n2^p)$ complexity
  - However 2 to 3 factors work well
  - L. Ding et al. (2011) and our work
- One caveat: this approximation method only applies to RBF Kernels

Kernel Approximation via P-Level Circulant Matrices

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: $\quad T_j \leftarrow k(||jH_n||_2)$
6: **end for**
7: **for all** $j \in N$ **do**
8: $\quad D_j \leftarrow D_{j,0} \times D_{j,1} \times \cdots \times D_{j,p-1}$
9: $\quad U_j \leftarrow \sum_{l \in D_j} T_l$
10: **end for**
11: $\tilde{K} \leftarrow U$

Output: $\tilde{K}$

Efficient Cross-Validation

Theorem

Let $y^{(k)} = sign[g_k(x)]$ *denote the classifier formulated by leaving the* k*th group out and let* $\beta_{k,i} = y_{k,i} - g_k(x_{k,i})$. *Then* $\beta_{(k)} = C_{kk}^{-1}\alpha_{(k)}$.

- proven by An et al. (2007)
- Take aways:
  - Allows computing a single Kernel matrix inverse for all folds
  - Perform smaller inverses to compute the hold out result

Efficient Cross-Validation

Algorithm 2 Efficient Cross-Validation

Input: K (Kernel matrix), l (Number folds), y (response)

$K_\gamma^{-1} \leftarrow inv(K + \frac{1}{\gamma} I), \quad d \leftarrow 1_n^\mathsf{T} K_\gamma^{-1} 1_n$

$C \leftarrow K_\gamma^{-1} + \frac{1}{d} K_\gamma^{-1} 1_n 1_n^\mathsf{T} K_\gamma^{-1}$

$\alpha \leftarrow K_\gamma^{-1} y + \frac{1}{d} K_\gamma^{-1} 1_n 1_n^\mathsf{T} K_\gamma^{-1} y$

$n_k \leftarrow size(y)/l, \quad y^{(k)} \leftarrow zeros(l, n_k)$

for $k \leftarrow 1$, $k \leq l$ **do**

    Solve $C_{kk} \beta_{(k)} = \alpha_{(k)}$

    $y^{(k)} \leftarrow \text{sign}[y_{(k)} - \beta_{(k)}]$

    $k \leftarrow k + 1$

end for

$error \leftarrow \frac{1}{2} \sum_{k=1}^{l} \sum_{i=1}^{n_k} |y_i - y^{(k,i)}|$

Output: *error*

Approximate *l*-fold Cross-Validation

Theorem

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 An et al.'s Cross-Validation Algorithm is $O(n \log n)$

- Take aways:
  - This combination produces an $O(n \log n)$ runtime
  - Works for any *l*-fold, provided the factorizations allign

Extension to Kernel Ridge Regression

- An et al.'s changes to their algorithm:
  - Change C's value to K_γ^{-1}
  - Change α's value to $K_\gamma^{-1} y$
- Our theorm still holds under these settings

Outline

Experimental Setup

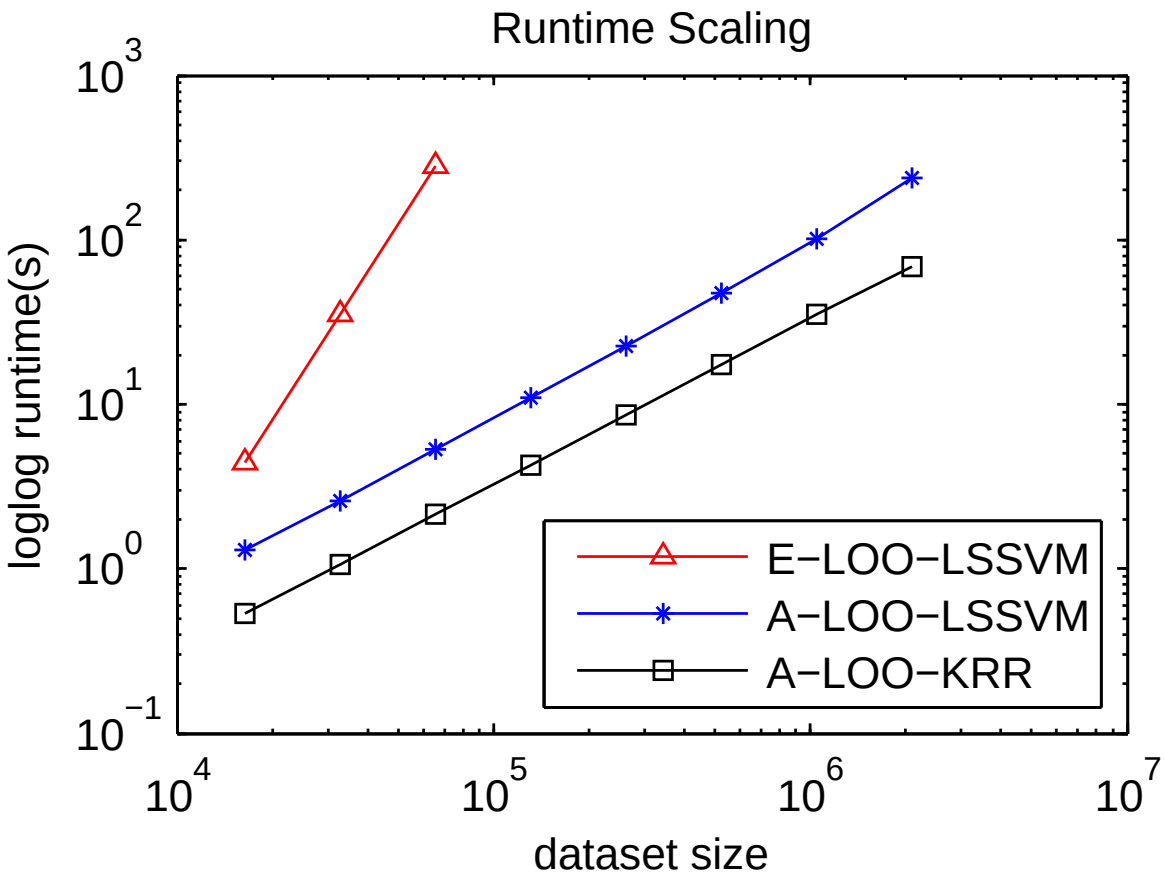
- Scaling
  - measured with randomly generated data
  - dataset sizes range from 2^{13} to 2^{20} samples
- Approximation Quality
  - measured on benchmark datasets
- Hyperparameter Selection Quality
  - Test exact models on real-world datasets

Single CPU Scaling Test

# Examples	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}
E-LOO	4.43s	35.25s	281.11s	–	–
A-LOO-LSSVM	1.3s	2.6s	5.32s	10.88s	22.45s
A-LOO-KRR	0.54s	1.06s	2.14s	4.3s	8.55s

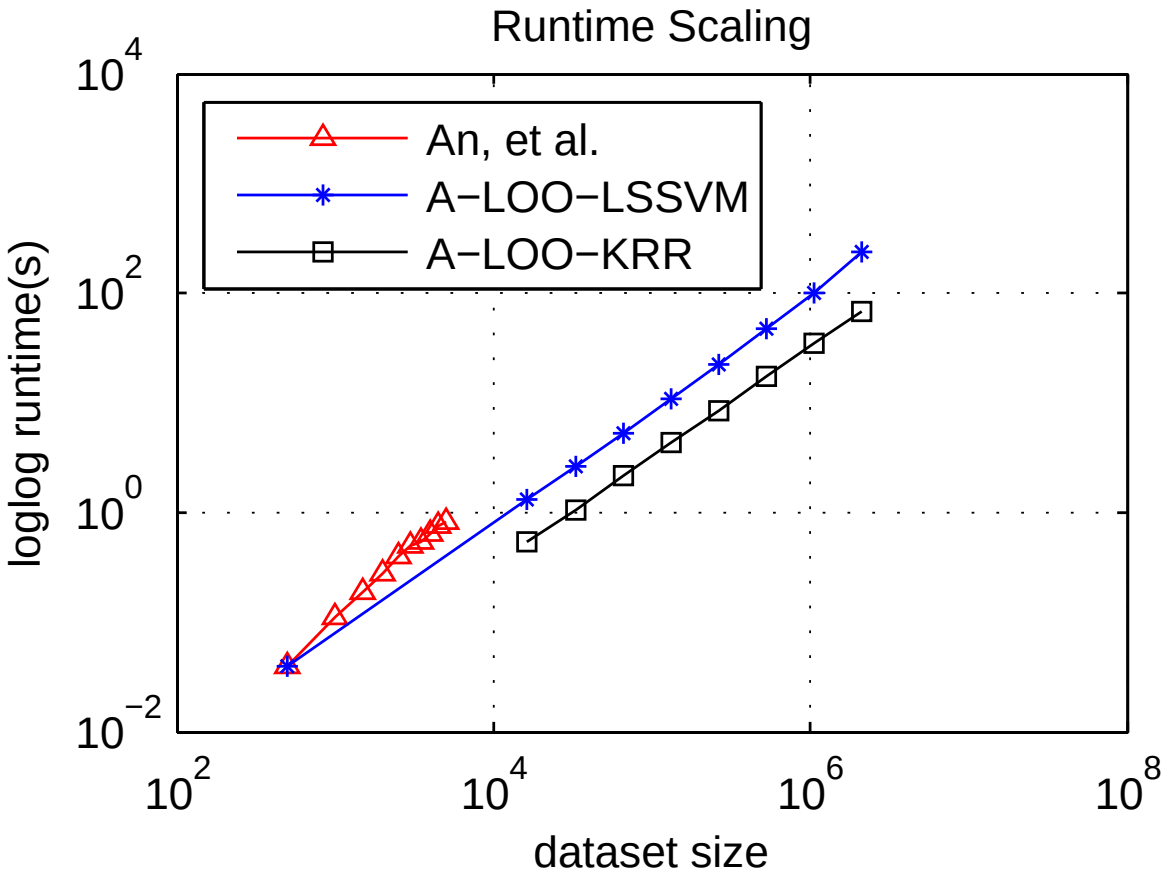
# Examples	2^{18}	2^{19}	2^{20}
E-LOO	–	–	–
A-LOO-LSSVM	47.41s	101.36s	235.83s
A-LOO-KRR	17.28s	35.39s	68.22s

Runtime Scaling Comparison



- A-LOO scales the same for LSSVM and KRR (same slopes)

Runtime Scaling Comparison



- We scale no worse than An et al's low-rank approximation
- We are assumption free, An et al. requires $m << n$

Benchmark Dataset Performance

Data set	#Train	#Test	A-Error(L. Ding, et al.)	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

- The real values selected effect approximation quality
- Hyperparameter selection is now $\mathbb{R}^{p+2}$, rather than $\mathbb{R}^2$

Real World Dataset

Data set	CoV(%)	MAPE(%)	CoV(%)	MAPE(%)
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

- Selected hyperparameters work well with exact models

Outline

Conclusion

- The approach provides an $O(n \log n)$ l-fold cross-validation method
- The approach scales well
- The approach selects hyperparameters that perform well with the exact model
- Hyperparameter selection is now $\mathbb{R}^{p+2}$, rather than $\mathbb{R}^2$