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

Introduction

Related Work

Preliminaries

Approach

Experiments

Conclusion
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: $O(m^2n)$

- We can do better!
  - Approximate cross-validation complexity: $O(n \log n)$
  - Applies to LOO as well
Outline

Introduction

Related Work

Preliminaries

Approach

Experiments

Conclusion
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

Introduction

Related Work

Preliminaries

Approach

Experiments

Conclusion

Richard E. Edwards, Hao Zhang, Lynne E. Parker, Joshua R. New

The University of Tennessee

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

$$\begin{bmatrix}
c_0 & c_1 & \ldots & c_n \\
c_n & c_0 & \ldots & c_{n-1} \\
\vdots & \vdots & \ddots & \vdots \\
c_1 & c_2 & \ldots & c_0
\end{bmatrix}$$
P-Level Circulant Matrices

- Combines Circulant Matrices and Multi-Level Circulant Matrices
  - Each level is a circulant matrix
  - All factors are now one dimensional

- 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

Introduction

Related Work

Preliminaries

Approach

Experiments

Conclusion
Overview

- We use same approximation method has L. Ding et al. (2011)
- We remove inefficiencies 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. $N \leftarrow \{\text{All multi-level indices defined by } F\}$
2. $T \leftarrow \text{zeros}(M)$, $U \leftarrow \text{zeros}(M)$
3. $H_n \leftarrow \{x_0, x_1, \ldots, x_{p-1}\} \in \mathbb{R}^p \text{ s.t. } \forall x_i \in H_n, x_i > 0$
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: $\tilde{K}$
Efficient Cross-Validation

Theorem

Let $y^{(k)} = \text{sign}[g_k(x)]$ 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)

1. $K^{-1}_γ ← \text{inv}(K + \frac{1}{γ} I)$, \quad $d ← 1_n^T K^{-1}_γ 1_n$
2. $C ← K^{-1}_γ + \frac{1}{d} K^{-1}_γ 1_n 1_n^T K^{-1}_γ$
3. $α ← K^{-1}_γ y + \frac{1}{d} K^{-1}_γ 1_n 1_n^T K^{-1}_γ y$
4. $n_k ← \text{size}(y)/l$, \quad $y^{(k)} ← \text{zeros}(l, n_k)$
5. **for** $k ← 1, k ≤ l$ **do**
6. \quad **Solve** $C_{kk} β^{(k)} = α^{(k)}$
7. \quad $y^{(k)} ← \text{sign}[y^{(k)} - β^{(k)}]$
8. \quad $k ← k + 1$
9. **end for**
10. $error ← \frac{1}{2} \sum_{k=1}^l \sum_{i=1}^{n_k} |y_i - y^{(k,i)}|$

**Output:** $error$
Approximate \( l \)-fold Cross-Validation

**Theorem**

If \( K \) is a \( p \)-level circulant matrix with factorization \( n = n_0 n_1 \ldots n_{p-1} \) and \( l = n_0 n_1 \ldots n_s \) 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 align
Extension to Kernel Ridge Regression

- An et al.’s changes to their algorithm:
  - Change C’s value to $K_{\gamma}^{-1}$
  - Change $\alpha$’s value to $K_{\gamma}^{-1} y$

- Our theorem still holds under these settings
Outline

Introduction
Related Work
Preliminaries
Approach
Experiments
Conclusion
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

<table>
<thead>
<tr>
<th># Examples</th>
<th>$2^{13}$</th>
<th>$2^{14}$</th>
<th>$2^{15}$</th>
<th>$2^{16}$</th>
<th>$2^{17}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-LOO</td>
<td>4.43s</td>
<td>35.25s</td>
<td>281.11s</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>A-LOO-LSSVM</td>
<td>1.3s</td>
<td>2.6s</td>
<td>5.32s</td>
<td>10.88s</td>
<td>22.45s</td>
</tr>
<tr>
<td>A-LOO-KRR</td>
<td>0.54s</td>
<td>1.06s</td>
<td>2.14s</td>
<td>4.3s</td>
<td>8.55s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># Examples</th>
<th>$2^{18}$</th>
<th>$2^{19}$</th>
<th>$2^{20}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-LOO</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>A-LOO-LSSVM</td>
<td>47.41s</td>
<td>101.36s</td>
<td>235.83s</td>
</tr>
<tr>
<td>A-LOO-KRR</td>
<td>17.28s</td>
<td>35.39s</td>
<td>68.22s</td>
</tr>
</tbody>
</table>
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 \( m \ll n \)
Benchmark Dataset Performance

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

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

<table>
<thead>
<tr>
<th>Data set</th>
<th>CoV(%)</th>
<th>MAPE(%)</th>
<th>CoV(%)</th>
<th>MAPE(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>House 1</td>
<td>19.6±1.69</td>
<td>15.3±0.47</td>
<td>20.1±0.81</td>
<td>16.1±0.85</td>
</tr>
<tr>
<td>Sensor A</td>
<td>1.3±0.05</td>
<td>1.0±0.05</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sensor B</td>
<td>17.2±4.89</td>
<td>10.8±0.25</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sensor C</td>
<td>12.0±2.31</td>
<td>7.8±0.68</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sensor D</td>
<td>1.4±0.09</td>
<td>0.9±0.03</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>S1</td>
<td>13.1±0.00</td>
<td>10.0±0.00</td>
<td>13.7±0.00</td>
<td>11.2±0.00</td>
</tr>
<tr>
<td>S2</td>
<td>3.1±0.00</td>
<td>4.7±0.00</td>
<td>6.4±0.00</td>
<td>4.5±0.00</td>
</tr>
</tbody>
</table>

- Selected hyperparameters work well with exact models
Outline

Introduction

Related Work

Preliminaries

Approach

Experiments

Conclusion
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 $\mathbb{R}^2$