Data-dependent Performance Modeling of Linear Solvers for Sparse Matrices

Jae-Seung Yeom†
Greg Bronevetsky‡
Jayaraman Thiagarajan†
Abhinav Bhatle†
Tzanio Kolev†

†{yeom2,jayaraman1,bhatle1,kolev1}@llnl.gov
‡bronevet@google.com
Overview of the problem

- Many computational science and engineering (CSE) code rely on solving sparse linear systems.
- Performance of a linear solver depends on:
  - input problem
  - data representation
  - algorithmic
  - Implementation
  - platform
- Need to choose an optimal solver for a given problem.
Problem setup

- Given an input matrix and a set of PC-LS choices, predict the fastest choice to solve the linear problem for novice users.

- Two sources of matrix data
  - MFEM data: representative of PDE-based simulations
    - E.g., heat transfer, structural mechanics, accelerator design, radiation diffusion flow,
  - University of Florida (UFL) data: collected from various domains

- PC-LS choices from *Trilinos* (ver. 11.12.1): Aztec + Ifpack

<table>
<thead>
<tr>
<th>Matrix data $N_M$</th>
<th>PC-LS choices $N_C = N_P \times N_S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFEM</td>
<td>UFL</td>
</tr>
<tr>
<td>879 matrices</td>
<td>361 chosen matrices</td>
</tr>
<tr>
<td>generated for</td>
<td></td>
</tr>
<tr>
<td>PDE-based problems</td>
<td></td>
</tr>
</tbody>
</table>

**Preconditioners**
- NONE, JACOBI, NEUMANN, LS, DOM_DECOMP, ILU, ILUT, IC, ICT, CHEBYSHEV, POINT_RELAXATION, BLOCK_RELAXATION, AMESOS, SORA, IHSS

**Linear solvers**
- GMRES, GMRES_CONDNUM, GMRESR, CG, CG_CONDNUM, CGS, BICGSTAB, TFQMR, FIXED_PT
## Candidate model features to understand prediction capability

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{row}$</td>
<td>number of rows</td>
</tr>
<tr>
<td>$NNZ$</td>
<td>number of non-zeros, $nnz(A)$</td>
</tr>
<tr>
<td>$NNZ_L$</td>
<td>number of non-zeros in lower triangle</td>
</tr>
<tr>
<td>$NNZ_U$</td>
<td>number of non-zeros in upper triangle</td>
</tr>
<tr>
<td>$NNZ_{max}$</td>
<td>maximum number of non-zeros per row</td>
</tr>
<tr>
<td>$NNZ_{avg}$</td>
<td>average number of non-zeros per row</td>
</tr>
<tr>
<td>$DE_{max}$</td>
<td>largest element in magnitude along diagonal</td>
</tr>
<tr>
<td>$DE_{min}$</td>
<td>smallest non-zero element in magnitude along diagonal</td>
</tr>
<tr>
<td>$bw_L$</td>
<td>bandwidth of lower triangle</td>
</tr>
<tr>
<td>$bw_U$</td>
<td>bandwidth of upper triangle</td>
</tr>
<tr>
<td>$NO$</td>
<td>number of ones</td>
</tr>
<tr>
<td>$NDD_{row}$</td>
<td>number of rows that are strictly diagonally dominant, i.e., satisfying $2</td>
</tr>
<tr>
<td>$spread_L$</td>
<td>normalized sum of distances from diagonal of non-zero elements in lower triangle $\sum (i - j)/S$ where $i &gt; j$, $a_{i,j} \neq 0$, and $S = \sum_{i=1}^{N_{row} - 1} i * (N_{row} - i)$</td>
</tr>
<tr>
<td>$spread_U$</td>
<td>normalized sum of distances from diagonal of non-zero elements in upper triangle $\sum (j - i)/S$ where $j &gt; i$, $a_{i,j} \neq 0$, and $S = \sum_{i=1}^{N_{row} - 1} i * (N_{row} - i)$</td>
</tr>
<tr>
<td>$symm$</td>
<td>property showing how symmetric the matrix is as $1 - \frac{nnz(A - A^T)(nnz(A) - nnz_d(A))}{(nnz(A) - nnz_d(A))}$ where $A^T$ is a non-conjugate transpose of $A$ and $nnz_d(A)$ is number of non-zeros on diagonal of $A$</td>
</tr>
<tr>
<td>$symm_s$</td>
<td>property showing how skew-symmetric the matrix is as $1 - \frac{(nnz(A + A^T) - nnz_d(A))}{(nnz(A) - nnz_d(A))}$</td>
</tr>
<tr>
<td>$symm_p$</td>
<td>property showing how symmetric the non-zero pattern of matrix is as $\frac{1 - \frac{nnz(P - P^T)(nnz(P) - nnz_d(P))}{(nnz(P) - nnz_d(P))}}{\frac{1}{nnz_d(P)}}$ where $p_{i,j} = 1$ if $a_{i,j} \neq 0$, 0 otherwise</td>
</tr>
<tr>
<td>$</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td></td>
</tr>
</tbody>
</table>

### Basic: readily computable

#### $\rho(A)$
- spectral radius

#### $\lambda_2$
- second largest absolute eigenvalue

#### $\lambda_{min}$
- non-zero smallest absolute eigenvalue

#### $\sqrt{\lambda_1/\lambda_2}$
- where $\lambda_1 = \rho(A)$

#### $\sqrt{\text{peak}}$
- peak $= \lambda_1 / \left( \left( \sum_i \lambda_i \right) - \lambda_1 \right)$

#### $\sqrt{\kappa}$
- where $\kappa$ is the condition number computed by Matlab: $\kappa = \text{cond} \left( A, \text{true} \right)$

#### $\sigma_{90}$
- portion of $\sqrt{\sigma} > 0.9 \times (\sqrt{\sigma_{\max}} - \sqrt{\sigma_{\min}}) + \sqrt{\sigma_{\min}}$ where $\sigma$ is a singular value

#### $\sigma_{10}$
- portion of $\sqrt{\sigma} \leq 0.1 \times (\sqrt{\sigma_{\max}} - \sqrt{\sigma_{\min}}) + \sqrt{\sigma_{\min}}$

### Advanced: expensive to compute

#### $N_{dim}$
- number of spatial dimensions (from the mesh)

#### $N_{row}'$
- $N_{row} / N_{dim}$ if P2, $N_{row}$ otherwise

#### $Mesh_{do}$
- finite element discretization order

#### $Mesh_{rt}$
- mesh refinement level

#### $\sqrt{\kappa'}$
- approximation to condition number term $\sqrt{\kappa}$, where $\kappa' = \left( N_{row}' \right)^2 / N_{dim}$

#### $\sqrt{\kappa'_{h_{\min}}}$
- approximation to condition number term $\sqrt{\kappa}$, where $\kappa'_{h_{\min}} = (1/\min(h))^2$

#### $\sqrt{\kappa'_{h_{\max}}}$
- approximation to condition number term $\sqrt{\kappa}$, where $\kappa'_{h_{\max}} = (1/\max(h))^2$

### Domain-specific: expert knowledge

- Basic: readily computable
- Advanced: expensive to compute
- Domain-specific: expert knowledge
Challenges in data and initial attempt with linear regression to predict execution time

- Identifying/designing important features
- Too many branching on categorical features → not scalable/not general
- Highly skewed distribution
- System noise

\[
N_{\text{runs}} = N_M \times N_C \times 11 \times 10
\]
Selecting features based on relative importance identified by Gradient Boosted Machine (GBM)

4 sets to understand the impact of feature choices

F1 Basic
- $R^2 = 0.71$
- Select if greater

F2 Basic + Advanced
- $R^2 = 0.79$

F3 Basic + Domain-specific
- $R^2 = 0.86$
- GBM fit score
- Can we do better?

F4 Basic + Advanced + Domain-specific
- $R^2 = 0.88$
Neural word embedding

- Based on neural network
- Low-dimensional representation
  - Vector space $W$
- More similar words closer in space
- Preserves many linguistic regularities and patterns
- Word2Vec (Mikolov et al., ’13)
  - Train 100 billion words in a day

\[ W^T x \rightarrow h \quad f(W'^T h) \rightarrow y \]

\[ n = N_M + N_C \]

vec("Madrid") - vec("Spain") + vec("France") \approx vec("Paris")
Modeling approach
performance vector space by word2vec

- **Performance vector space (PVS)**
  - Weight matrix ($W$) learned between input layer and hidden layer
  - To represent matrix relation in a lower dimension space

- **Map from PVS to label**
  - Weight matrix ($W'$) learned between hidden layer and output layer
  - To determine a set of PC-LSs that are effective for the given matrix
Overall workflow

**Learning**

- Build Regression Model
- Total Runtime
- Feature Importance
- Select Important Features

**Training Data**

- Feature Set
- Performance Data
- Build Matrix-Label Association
- Compute Vector Embedding

**Novel Test Sample**

- Compute Sparse Representation
- Compute Vector Space Projection

**Predicting**

- Select similar training examples based on matrix features
- Predict fastest PC-LS and associated T for a new matrix

Predicting

- K-NN Classifier
- Majority Voting
Predicting execution time for MFEM matrices predicted vs. observed

- **F1 Basic**
  - SVM
  - K-NN
  - Proposed

- **F2 Basic + Advanced**
  - SVM
  - K-NN
  - Proposed

- **F3 Basic + Domain-specific**
  - SVM
  - K-NN
  - Proposed

- **F4 Basic + Domain-specific**
  - SVM
  - K-NN
  - Proposed
Prediction error (ARE)

\[ E = \frac{|y_p - y_o|}{y_o} \]

\( y_p \): predicted time

\( y_o \): observed time

Impact of ineffective outlier handling in classification

<table>
<thead>
<tr>
<th>Feature set</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>Blind</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>1.58</td>
<td>2.08</td>
<td>3.42</td>
<td>2.06</td>
<td></td>
</tr>
<tr>
<td>K-NN</td>
<td>3.66</td>
<td>1.68</td>
<td>0.39</td>
<td>1.76</td>
<td>0.14</td>
</tr>
<tr>
<td>Ours</td>
<td>0.18</td>
<td>0.025</td>
<td>0.03</td>
<td>0.025</td>
<td></td>
</tr>
</tbody>
</table>
Predicting execution time for UFL matrices
predicted vs. observed

**F1 Basic**

**F2 Basic + Advanced**

<table>
<thead>
<tr>
<th>Feature set</th>
<th>F1</th>
<th>F2</th>
<th>Blind</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>3.52</td>
<td>2.93</td>
<td></td>
</tr>
<tr>
<td>K-NN</td>
<td>3.41</td>
<td>2.69</td>
<td>2.19</td>
</tr>
<tr>
<td>Ours</td>
<td>2.13</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>
- Our method outperforms in terms of absolute relative error (ARE)
  - in the presence of outlying data
  - Even when classification accuracy is lower than SVM and k-NN
- This helps users to identify optimal or close to optimal PC-LS choices

<table>
<thead>
<tr>
<th>Feature set</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>Blind</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>1.58</td>
<td>2.08</td>
<td>3.42</td>
<td>2.06</td>
<td></td>
</tr>
<tr>
<td>K-NN</td>
<td>3.66</td>
<td>1.68</td>
<td>0.39</td>
<td>1.76</td>
<td>0.14</td>
</tr>
<tr>
<td>Ours</td>
<td>0.18</td>
<td>0.025</td>
<td>0.03</td>
<td>0.025</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature set</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>Blind</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>73.4</td>
<td>75.8</td>
<td>76.0</td>
<td>80.1</td>
<td></td>
</tr>
<tr>
<td>K-NN</td>
<td>72.8</td>
<td>75.1</td>
<td>74.6</td>
<td>78.6</td>
<td>69</td>
</tr>
<tr>
<td>Ours</td>
<td>70.1</td>
<td>74.2</td>
<td>75.2</td>
<td>77.9</td>
<td></td>
</tr>
</tbody>
</table>
## Conclusion

- Data-dependent performance modeling

- Highly challenging dataset
  - Large number of classes, categorical features, limited amount, skewed label distribution, outlying data

- Formulated the problem as a classification problem of predicting the fastest PC-LS choice for a given matrix
  - Neural word embedding to take advantage of patterns in relationship
    - Between matrices, and between matrices and PC-LS choices
  - Consistent feature space and performance vector space (PVS)

- User-centered metric
  - Outperforms in the presence of outlying data in terms of absolute relative error (ARE)