RANDOMLY PIVOTED PARTIAL CHOLESKY STRUCTURED DATA

ELLEN BURNS, LUKE GREEN, JOSHUA KIMBER

Abstract. We investigate the effectiveness of RP-Cholesky and its variants for computing low-rank approximations of radial basis function (RBF) kernel matrices. Previous work has focused primarily on real-world datasets [CETW24]. We aim to evaluate whether the observed characteristics for real-world data hold for both synthetic and randomized data where we can control the presence of noise. Our study provides insights into the performance and robustness of RP-Cholesky methods across different types of data matrices.

1. Research question and overview of the existing literature/methods

Kernel methods play a major role in machine learning algorithms and distill information about the pairwise similarities of N data points into a dense positive-semidefinite (psd) kernel matrix with dimensions $N \times N$ [CETW24]. When applied to real-world data, kernel matrices often have a low-rank structure. Thus, low-rank approximations are often considered as they allow the reduction of both temporal and spatial complexity [Bac13].

Cholesky factorization was first introduced in 1910 as an algorithm that breaks down a positive definite matrix A into a lower triangular matrix L multiplied by its transpose $A = LL^T$. Several decades after it first appeared, mathematicians began playing with the idea of swapping rows or columns of the matrix A to be able to find a lower rank approximation to A. This can be done by stopping the process of the algorithm early and just using the selected diagonal values of the matrix for the approximation. In machine learning applications, the most common selection strategy is greedy pivoting, which at each iteration chooses the column corresponding to the largest diagonal entry of A.

Mathematicians later recognized that this strategy can heavily emphasize outliers, leading to poor approximation [CETW24]. Randomly Pivoted Partial Cholesky (RP-Cholesky) was introduced as a way to balance exploration and exploitation by randomly sampling pivots proportional to the diagonal entries of the current residual. [CETW24] recently advocated for the adoption of RP-Cholesky into machine learning pipelines and provided

Date: December 23, 2025.

a thorough study demonstrating its superiority to Greedy Pivoting when applied to kernel matrices over real world datasets.

The purpose of this project is to provide further analysis of the sensitivities and tradeoff between exploration and exploitation of the selection criterion for RP-Cholesky applied to kernel matrices over controlled synthetic datasets. We hope to reveal the correlation in performance between the amount of randomness in the selection criterion and noise in the data.

2. Implementation of the algorithm/methodology

In our implementation, we modify Algorithm 1 from [CETW24] to also include the parameter β, which acts as an exponent on the diagonal entries of the residual matrix (see line 8). It is this parameter that allows us to analyze the tradeoff between exploration and exploitation. It is important to note that $\beta = 1$ is the standard RP-Cholesky algorithm.

Algorithm 1 Randomly Pivoted Cholesky (RP-Cholesky)

```
Input: PSD matrix $A \in \mathbb{R}^{N \times N}$, rank $k$, exponent $\beta$
Output: Pivot set $S = \{s_1, \dots, s_k\}$, factor $F$, approximation $\widehat{A}$
 1: Initialize $F \leftarrow \mathbf{0}_{m \times k}$ and $d \leftarrow \operatorname{diag}(A)$ and $S = \{\}$
 2: for $i = 1$ to $k$ do
 3:     if $\beta = \infty$ then                                   ▷ Greedy pivoting
 4:         $s_i \leftarrow \arg\max_j d_j$
 5:     else if $\beta = 0$ then                                   ▷ Uniform pivoting
 6:         Sample $s_i \sim 1/(m - \texttt{len(S)})$
 7:     else                                                      ▷ $\beta$-randomized pivoting
 8:         Sample $s_i \sim d_j^\beta / \sum_{j=1}^{m} d_j^\beta$
 9:     end if
10:     $S \leftarrow S \cup \{s_i\}$                        ▷ Add the pivot index to the list $S$
11:     $g \leftarrow A(:, s_i)$                             ▷ retrieve corresponding column
12:     $g \leftarrow g - F(:, 1{:}i-1)F(s_i, 1{:}i-1)$      ▷ No overlap with prev. columns
13:     $F(:, i) \leftarrow g / \sqrt{g_{s_i}}$                    ▷ Update approximation
14:     $d \leftarrow d - g^2 / g_{s_i}$                          ▷ Track residual diagonal
15:     $d \leftarrow \max(d, 0)$                    ▷ Ensure diagonal remains non-negative
16: end for
17: $\widehat{A} \leftarrow FF^\top$
18: return $S, F, \widehat{A}$
```

In our tests we chose to analyze the performance of $\beta \in \{0, 0.5, 1, 2, \infty\}$ on RBF kernel matrices size $N = 1000$. Each entry of the kernel matrix is defined as

$$k_{ij} = \exp\left(\frac{||x_i - x_j||^2}{2\ell^2}\right)$$

with $||\cdot|| = ||\cdot||_2$ and $\ell = 0.5$. Similar to [CETW24] we use the relative trace error to measure the performance of the algorithm.

This is defined as $\operatorname{tr}(A - \widehat{A})/\operatorname{tr}(A)$ where $\widehat{A}$ is a rank k approximation of A. All data is generated using `make_moons` and `make_circles` from the Sci-Kit Learn package.

3. Visualization/analysis of algorithm convergence or usefulness

Using synthetically generated "circle" data with minimal noise in Figure 1 it appears that $\beta = 1, 2$ typically outperform the other variants of the algorithm. It appears that the uniform pivoting (blue) is the least efficient in this case, with the greedy (purple) algorithm performs moderately well. However, real-world data sets are typically noisy, motivating an investigation of the effects of increased noise.

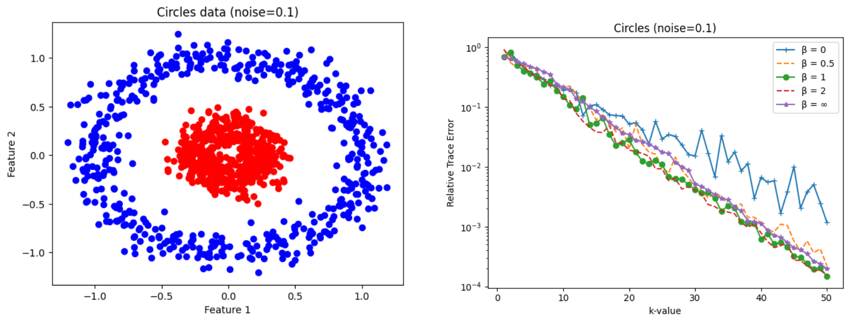

Figure 1. The plot on the left shows the synthetic data generated with minimal noise, and the plot on the right shows low-rank approximations and their corresponding errors with ranks $k = 1, \ldots, 50$.

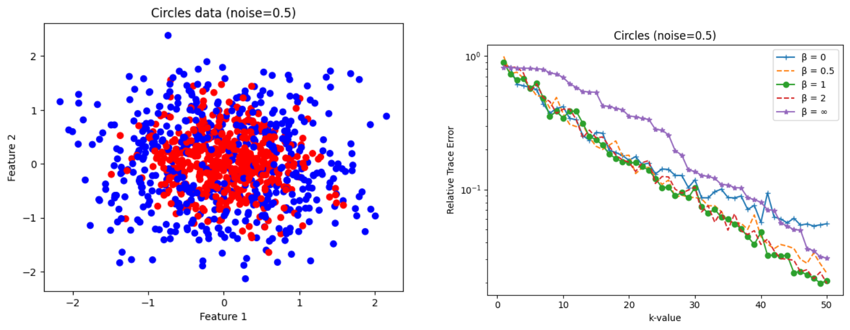

Figure 2. The plot on the left displays the synthetic data with significant noise, and the plot on the right shows low-rank approximations and their corresponding errors of the with ranks $k = 1, \ldots, 50$.

When noise is increased, see Figure 2, there is a major decline in the performance of greedy algorithm, while standard RP-Cholesky maintains its efficiency. This behavior mirrors what is observed in the real-world data experiments reported in [CETW24]. Perhaps the most interesting result is that uniform pivoting performs better than the greedy algorithm until about $k = 45$.

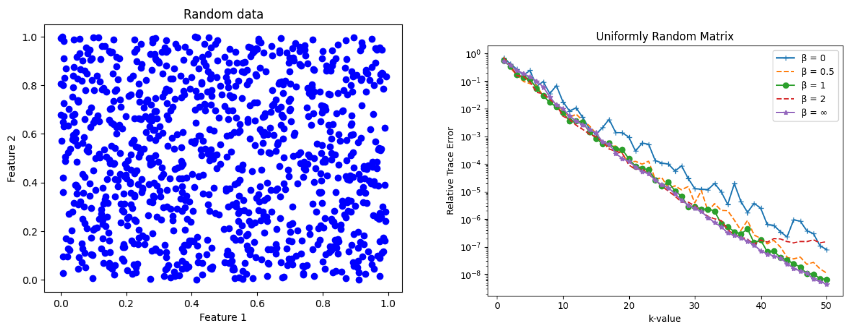

FIGURE 3. The plot on the left displays the randomized data, and the plot on the right shows low-rank approximations and their corresponding errors with ranks $k = 1, \dots, 50$.

In the case of randomized data in Figure 3, it appears that the greedy algorithm performs just as well as the standard RP-Cholesky as k increases. This result shows something different than what was discussed in [CETW24]. If data are unstructured then the greedy algorithm appears to be just as efficient as standard RP-Cholesky. These results may suggest there is a noise threshold, where if the data are "unstructured enough" it may not matter whether the greedy algorithm or standard RP-Cholesky is applied.

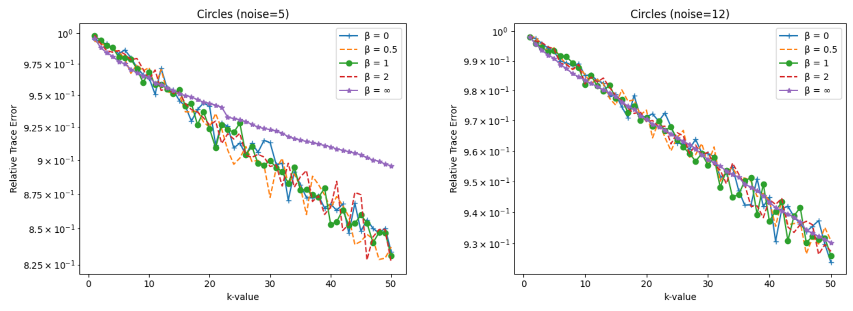

FIGURE 4. The plot on the left shows the errors for an approximated RBF matrix with data having a noise level of 5, and the plot on the right displays the errors for another approximation with a noise level of 12.

As shown in Figure 4, the noise level must increase substantially before the greedy method becomes comparable to the other variants. The plot on the left corresponds to a noise level ten times larger than that used in Figure 2, yet the greedy algorithm still performs poorly. Only when the noise level exceeds twenty times the previous value of 0.5 does noticeable improvement emerge. Although these two plots do not establish a definitive threshold, they raise important questions about how to navigate the transition between structured and randomized data and how to choose the most appropriate RP-Cholesky variant.

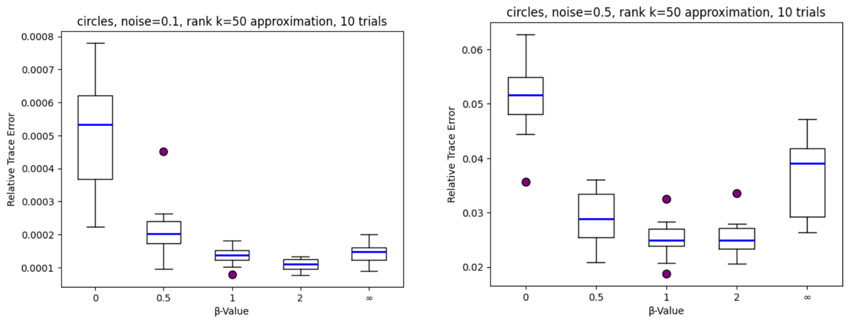

Figure 5. These box plots display the distribution of error across 10 trials for each β-value. Each trial is computing a rank $k = 50$ approximation of the matrix. The left plot uses circle data and noise level 0.1, and the right plot uses circle data noise level 0.5.

To analyze the error distributions, ten independent trials were run for each β-value, with each trial computing a rank-$k = 50$ approximation of an RBF kernel matrix. The resulting distributions are shown in Figure 5. As expected, the distributions indicate that RP-Cholesky outperforms both the uniform and greedy variants.

Several avenues for further investigation remain. The present analysis considers only RBF kernel matrices, and it may be informative to extend these experiments to other kernel types. More broadly, the question of how randomness and structure interact in determining algorithmic performance remains an open and intriguing direction for future work.

4. Societal Impact

As a simple example of the speedup attained by low-rank approximations, consider the following computations:

$$(xy^T)z \quad \text{and} \quad x(y^Tz).$$

The first computation has both temporal and spatial complexity $\mathcal{O}(n^2)$, while the second $\mathcal{O}(n)$. Modern day deep neural networks can take several weeks/months to train, and inference is expensive. Low-rank approximations are foundational to neural network compression techniques being developed today.

References

[Bac13] Francis Bach. Sharp analysis of low-rank kernel matrix approximations, 2013.

[CETW24] Yifan Chen, Ethan N. Epperly, Joel A. Tropp, and Robert J. Webber. Randomly pivoted cholesky: Practical approximation of a kernel matrix with few entry evaluations, 2024.