Randomized SVD, CUR Decomposition, and SPSD Matrix Approximation

Shusen Wang
Outline

• CX Decomposition & Approximate SVD
• CUR Decomposition
• SPSD Matrix Approximation
CX Decomposition

• Given any matrix $A \in \mathbb{R}^{m \times n}$

• The CX decomposition of $A$
  1. Sketching: $C = AP \in \mathbb{R}^{m \times c}$
  2. Find $X$ such that $A \approx CX$
     • E.g. $X^* = \text{argmin}_X ||A - CX||_F^2 = C^+A$
     • It costs $O(mnc)$
Let the sketching matrix $\mathbf{P} \in \mathbb{R}^{n \times c}$ be defined in the table.

$$\min_{\text{rank}(\mathbf{X}) \leq k} \| \mathbf{A} - \mathbf{CX} \|^2_F \leq (1 + \epsilon) \| \mathbf{A} - \mathbf{A}_k \|^2_F$$

<table>
<thead>
<tr>
<th></th>
<th>Uniform sampling</th>
<th>Leverage score sampling</th>
<th>Gaussian projection</th>
<th>SRHT</th>
<th>Count sketch</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbf{C} \geq$</td>
<td>$0 \left( \nu k \left( \log k + \frac{1}{\epsilon} \right) \right)$</td>
<td>$0 \left( k \left( \log k + \frac{1}{\epsilon} \right) \right)$</td>
<td>$0 \left( \frac{k}{\epsilon} \right)$</td>
<td>$0 \left( (k + \log n) \left( \log k + \frac{1}{\epsilon} \right) \right)$</td>
<td>$0 \left( k^2 + \frac{k}{\epsilon} \right)$</td>
</tr>
</tbody>
</table>

$\nu$ is the column coherence of $\mathbf{A}_k$
CX Decomposition ⇔ Approximate SVD

- CX decomposition ⇔ approximate SVD

\[ A \approx CX \]
**CX Decomposition ⇔ Approximate SVD**

- CX decomposition ⇔ approximate SVD

\[ A \approx CX = U_C \Sigma_C V_C^T X \]

**SVD:**
\[ C = U_C \Sigma_C V_C^T \in \mathbb{R}^{m \times c} \]

**Time cost:** \( O(mc^2) \)
CX Decomposition $\Leftrightarrow$ Approximate SVD

- CX decomposition $\Leftrightarrow$ approximate SVD

$$A \approx CX = U_C \Sigma_C V_C^T X = U_C Z$$

Let $\Sigma_C V_C^T X = Z \in \mathbb{R}^{c \times n}$

SVD: $C = U_C \Sigma_C V_C^T \in \mathbb{R}^{m \times c}$

Time cost: $O(mc^2 + nc^2)$
CX Decomposition ⇔ Approximate SVD

• CX decomposition ⇔ approximate SVD

\[ A \approx CX = U_C \Sigma_C V_C^T X = U_C Z = U_C U_Z \Sigma_Z V_Z^T \]

SVD: \( C = U_C \Sigma_C V_C^T \in \mathbb{R}^{m \times c} \)

Let \( \Sigma_C V_C^T X = Z \in \mathbb{R}^{c \times n} \)

SVD: \( Z = U_Z \Sigma_Z V_Z^T \in \mathbb{R}^{c \times n} \)

Time cost: \( O(mc^2 + nc^2 + nc^2) \)
CX Decomposition $\Leftrightarrow$ Approximate SVD

- CX decomposition $\Leftrightarrow$ approximate SVD

$$A \approx CX = UC \Sigma_C V_C^T X = UCZ = UCU_Z \Sigma_Z V_Z^T$$

- SVD: $C = U_C \Sigma_C V_C^T \in \mathbb{R}^{m \times c}$
- SVD: $Z = U_Z \Sigma_Z V_Z^T \in \mathbb{R}^{c \times n}$

$m \times s$ matrix with orthonormal columns

$s \times n$ matrix with orthonormal rows

diagonal matrix

Time cost: $O(mc^2 + nc^2 + nc^2 + mc^2)$
**CX Decomposition ⇔ Approximate SVD**

- CX decomposition ⇔ approximate SVD
- Done! Approximate rank c SVD: \( A \approx (U_G U_L) \Sigma_L V_L^T \)

\[
A \approx CX = U_C \Sigma_C V_C^T X = U_C Z = U_C U_Z \Sigma_Z V_Z^T
\]

- **Time cost:** \( O(mc^2 + nc^2 + nc^2 + mc^2) = O(mc^2 + nc^2) \)
CX Decomposition $\iff$ Approximate SVD

- CX decomposition $\iff$ approximate SVD

- Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{C} \in \mathbb{R}^{m \times c}$, the approximate SVD costs
  - $O(mnc)$ time
  - $O(mc + nc)$ memory
CX Decomposition

• The CX decomposition of $A \in \mathbb{R}^{m \times n}$
  • Optimal solution: $X^* = \text{argmin}_X \|A - CX\|_F^2 = C^+A$
  • How to make it more efficient?
CX Decomposition

• The CX decomposition of $\mathbf{A} \in \mathbb{R}^{m \times n}$
  
  - Optimal solution: $\mathbf{X}^* = \arg\min_{\mathbf{X}} \| \mathbf{A} - \mathbf{CX} \|_F^2 = \mathbf{C}^\dagger \mathbf{A}$
  
  - How to make it more efficient?

A regression problem!
Fast CX Decomposition

- Fast CX [Drineas, Mahoney, Muthukrishnan, 2008][Clarkson & Woodruff, 2013]
  - Draw another sketching matrix $S \in \mathbb{R}^{m \times s}$
  - Compute $\tilde{X} = \arg\min_X \|S^T(A - CX)\|_F^2 = (S^T C)^\dagger (S^T A)$
  - Time cost: $O(ncs) + \text{TimeOfSketch}$
  - When $s = \tilde{O}(c/\epsilon)$,
    $$\|A - C\tilde{X}\|_F^2 \leq (1 + \epsilon) \cdot \min_X \|A - CX\|_F^2$$
Outline

• CX Decomposition & Approximate SVD
• CUR Decomposition
• SPSD Matrix Approximation
CUR Decomposition

- Sketching
  - $\mathbf{C} = \mathbf{AP}_\mathbf{C} \in \mathbb{R}^{m \times c}$
  - $\mathbf{R} = \mathbf{P}_\mathbf{R}^T \mathbf{A} \in \mathbb{R}^{r \times n}$
- Find $\mathbf{U}$ such that $\mathbf{CUR} \approx \mathbf{A}$
- CUR $\iff$ Approximate SVD
  - In the same way as "$\mathbf{CX} \iff \text{Approximate SVD}$"
CUR Decomposition

• Sketching
  • $C = AP_C \in \mathbb{R}^{m \times c}$
  • $R = P^T_RA \in \mathbb{R}^{r \times n}$

• Find $U$ such that $CUR \approx A$

• CUR $\iff$ Approximate SVD
  • In the same way as “$CX \iff$ Approximate SVD”

• 3 types of $U$
CUR Decomposition

- Type 1 [Drineas, Mahoney, Muthukrishnan, 2008]:

\[ U = \left( P_R^T A P_C \right)^\dagger \]
CUR Decomposition

• Type 1 [Drineas, Mahoney, Muthukrishnan, 2008] :
  \[ U = \left( P_R^T A P_C \right)^+ \]

• Recall the fast CX decomposition
  \[ A \approx C \tilde{X} = C \left( P_R^T C \right)^+ \left( P_R^T A \right) \]
CUR Decomposition

• Type 1 [Drineas, Mahoney, Muthukrishnan, 2008]:
  \[ \mathbf{U} = (\mathbf{P}_R^T \mathbf{A} \mathbf{P}_C)^\dagger \]

• Recall the fast CX decomposition
  \[ \mathbf{A} \approx \mathbf{C} \tilde{\mathbf{X}} = \mathbf{C}(\mathbf{P}_R^T \mathbf{C})^\dagger (\mathbf{P}_R^T \mathbf{A}) = \mathbf{CUR} \]
CUR Decomposition

• Type 1 [Drineas, Mahoney, Muthukrishnan, 2008]:
  \[ U = (P_R^T A P_C)^\dagger \]

• Recall the fast CX decomposition
  \[ A \approx C \tilde{X} = C (P_R^T C)^\dagger (P_R^T A) = CUR \]

• They’re equivalent: \( C \tilde{X} = C U R \)
CUR Decomposition

• Type 1 [Drineas, Mahoney, Muthukrishnan, 2008]:
  \[ U = \left( P^T_R A P_C \right)^\dagger \]

• Recall the fast CX decomposition
  \[ A \approx C\tilde{X} = C(P^T_R C)^\dagger (P^T_R A) = \text{CUR} \]

• They’re equivalent: \( C\tilde{X} = C U R \)

• Require \( c = \tilde{O}\left(\frac{k}{\epsilon}\right) \) and \( r = \tilde{O}\left(\frac{c}{\epsilon}\right) \) such that
  \[ \|A - \text{CUR}\|_F^2 \leq (1 + \epsilon) \|A - A_k\|_F^2 \]
CUR Decomposition

• Type 1 [Drineas, Mahoney, Muthukrishnan, 2008]:
  \[ U = (P_R^T A P_C)^\dagger \]

• Efficient
  • \(O(rc^2) + \text{TimeOfSketch}\)

• Loose bound
  • Sketch size \(\propto \epsilon^{-2}\)

• Bad empirical performance
CUR Decomposition

• Type 2: Optimal CUR

\[ U^* = \min_U \| A - \text{CUR} \|_F^2 = C^\dagger A R^\dagger \]
CUR Decomposition

• Type 2: Optimal CUR

\[ U^* = \min_U \| A - CUR \|_F^2 = C^\dagger AR^\dagger \]

• Theory [W & Zhang, 2013], [Boutsidis & Woodruff, 2014]:

  • \( C \) and \( R \) are selected by the adaptive sampling algorithm
  • \( c = O \left( \frac{k}{\epsilon} \right) \) and \( r = O \left( \frac{k}{\epsilon} \right) \)
  • \( \| A - CUR \|_F^2 \leq (1 + \epsilon) \| A - A_k \|_F^2 \)
CUR Decomposition

• Type 2: Optimal CUR

$$U^* = \min_U \| A - CUR \|_F^2 = C^\dagger AR^\dagger$$

• Inefficient
  • $O(mnc) + \text{TimeOfSketch}$
CUR Decomposition

• Type 3: Fast CUR [W, Zhang, Zhang, 2015]
  • Draw 2 sketching matrices $S_C$ and $S_R$
  • Solve the problem
    \[ \tilde{U} = \min_\mathbf{U} \left\| S_C^T (A - \text{CUR}) S_R \right\|_F^2 = (S_C^T S_C)^\dagger (S_C^T A S_R) (R S_R)^\dagger \]
• Intuition?
CUR Decomposition

- The optimal $\mathbf{U}$ matrix is obtained by the optimization problem

$$\mathbf{U}^* = \min_{\mathbf{U}} \| \mathbf{CUR} - \mathbf{A} \|_F^2$$
CUR Decomposition

- Approximately solve the optimization problem, e.g. by column selection

\[
\min_{\mathbf{U}} \quad \text{subject to} \quad \mathbf{U} \mathbf{U}^\top = \mathbf{U}^\top \mathbf{U} = \mathbf{I}
\]

\[
\mathbf{F} = \begin{bmatrix}
\mathbf{U} & \mathbf{E} & \mathbf{F}
\end{bmatrix}
\]
CUR Decomposition

- Solve the small scale problem

\[
\min_{U} \left\| \begin{array}{c}
\text{red}
\end{array} \right\| - \left\| \begin{array}{c}
\text{blue}
\end{array} \right\| \right\|_{F}^{2}
\]
CUR Decomposition

• Type 3: Fast CUR [W, Zhang, Zhang, 2015]
  - Draw 2 sketching matrices $S_C \in \mathbb{R}^{m \times s_c}$ and $S_R \in \mathbb{R}^{n \times s_r}$
  - Solve the problem
    $$\tilde{U} = \min_U \left\| S_C^T (A - CUR) S_R \right\|_F^2 = (S_C^T C)^\dagger (S_C^T A S_R)(R S_R)^\dagger$$

• Theory
  - $s_c = O \left( \frac{c}{\epsilon} \right)$ and $s_r = O \left( \frac{r}{\epsilon} \right)$
  - $\left\| A - C\tilde{U}R \right\|_F^2 \leq (1 + \epsilon) \cdot \min_U \left\| A - CUR \right\|_F^2$
CUR Decomposition

• Type 3: Fast CUR [W, Zhang, Zhang, 2015]
  • Draw 2 sketching matrices $S_C \in \mathbb{R}^{m \times s_c}$ and $S_R \in \mathbb{R}^{n \times s_r}$
  • Solve the problem
    \[
    \tilde{U} = \min_U \left\| S_C^T (A - CUR) S_R \right\|_F^2 = \left( S_C^T C \right)^\dagger \left( S_C^T A S_R \right) \left( R S_R \right)^\dagger
    \]

• Efficient
  • $O(s_c s_r (c + r)) + \text{TimeOfSketch}$

• Good empirical performance
A:
\[ m = 1920 \]
\[ n = 1168 \]

C and R:
- \( c = r = 100 \)
- uniform sampling

\( \mathbf{A} \):
- \( m = 1920 \)
- \( n = 1168 \)

\( \mathbf{C} \) and \( \mathbf{R} \):
- \( c = r = 100 \)
- uniform sampling

Type 1: Fast CX
- \( s_c = 2c, \quad s_r = 2r \)

Type 2: Optimal CUR
- \( s_c = 4c, \quad s_r = 4r \)

Type 3: Fast CUR
- \( s_c = 4c, \quad s_r = 4r \)
Conclusions

• Approximate truncated SVD
  • CX decomposition
  • CUR decomposition (3 types)

• Fast CUR is the best
Outline

• CX Decomposition & Approximate SVD
• CUR Decomposition
• SPSD Matrix Approximation
Motivation 1: Kernel Matrix

• Given $n$ samples $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and kernel function $\kappa(\cdot, \cdot)$.
• E.g. Gaussian RBF kernel

$$\kappa(x_i, x_j) = \exp \left(-\frac{||x_i - x_j||^2}{\sigma^2}\right).$$
Motivation 1: Kernel Matrix

- Given \( n \) samples \( \mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d \) and kernel function \( \kappa(\cdot, \cdot) \).
- E.g. Gaussian RBF kernel
  \[ \kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp \left( - \frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{\sigma^2} \right). \]
- Computing the kernel matrix \( \mathbf{K} \in \mathbb{R}^{n \times n} \)
  - where \( k_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j) \)
  - costs \( O(n^2d) \) time
Motivation 2: Matrix Inversion

• Solve the linear system

$$(K + \alpha I_n)w = y$$

to find $w \in \mathbb{R}^n$.

- $K \in \mathbb{R}^{n \times n}$ is the kernel matrix
- $y = [y_1, \ldots, y_n] \in \mathbb{R}^n$ contains the labels
Motivation 2: Matrix Inversion

• Solve the linear system

\[(K + \alpha I_n)w = y\]

to find \(w \in \mathbb{R}^n\).

• Solution: \(w^* = (K + \alpha I_n)^{-1}y\)
Motivation 2: Matrix Inversion

• Solve the linear system

\[(K + \alpha I_n)w = y\]

\[\text{to find } w \in \mathbb{R}^n.\]

• Solution: \( w^* = (K + \alpha I_n)^{-1}y \)

• It costs
  • \( O(n^3) \) time
  • \( O(n^2) \) memory.
Motivation 2: Matrix Inversion

• Solve the linear system

\[(K + \alpha I_n)w = y\]

to find \(w \in \mathbb{R}^n\).

• Solution: \(w^* = (K + \alpha I_n)^{-1}y\)

• It costs
  • \(O(n^3)\) time
  • \(O(n^2)\) memory.

• Performed by
  • Kernel ridge regression
  • Least squares kernel SVM
Motivation 3: Eigenvalue Decomposition

• Find the top $k \ll n$ eigenvectors of $K$.
• It costs
  • $\tilde{O}(n^2k)$ time
  • $O(n^2)$ memory.
Motivation 3: Eigenvalue Decomposition

- Find the top $k$ ($\ll n$) eigenvectors of $\mathbf{K}$.
- It costs
  - $\tilde{O}(n^2k)$ time
  - $O(n^2)$ memory.
- Performed by
  - Kernel PCA ($k$ is the target rank)
  - Manifold learning ($k$ is the target rank)
Computational Challenges

• Time costs
  • Computing kernel matrix: $O(n^2d)$
  • Matrix inversion: $O(n^3)$
  • Rank $k$ eigenvalue decomposition: $O(n^2k)$
Computational Challenges

• Time costs
  • Computing kernel matrix: $O(n^2d)$
  • Matrix inversion: $O(n^3)$
  • Rank $k$ eigenvalue decomposition: $O(n^2k)$

At least quadratic time!
Computational Challenges

• Time costs
  • Computing kernel matrix: $O(n^2d)$
  • Matrix inversion: $O(n^3)$
  • Rank $k$ eigenvalue decomposition: $O(n^2k)$

• Memory costs
  • Inversion and eigenvalue decomposition: $O(n^2)$
Computational Challenges

• Time costs
  • Computing kernel matrix: $O(n^2d)$
  • Matrix inversion: $O(n^3)$
  • Rank $k$ eigenvalue decomposition: $O(n^2k)$

• Memory costs
  • Inversion and eigenvalue decomposition: $O(n^2)$
  • Because
    • the numerical algorithms are pass-inefficient
    • form $K$ and keep it in memory
Computational Challenges

• Time costs
  • Computing kernel matrix: $O(n^2d)$
  • Matrix inversion: $O(n^3)$
  • Rank $k$ eigenvalue decomposition: $O(n^2k)$

• Memory costs
  • Inversion and eigenvalue decomposition: $O(n^2)$
  • Because
    • the numerical algorithms are pass-inefficient
    • $\Rightarrow$ form $K$ and keep it in memory

When $n = 10^5$, the $n \times n$ matrix costs 80GB memory!
How to Speedup?

• Efficiently form the low-rank approximation

\[ K \approx C \ U \ C^T \]
How to Speedup?

- Efficiently form the low-rank approximation
  \[ \mathbf{K} \approx \mathbf{C} \mathbf{U} \mathbf{C}^\top \]

- Equivalent \( \mathbf{K} \approx \mathbf{L} \mathbf{L}^\top \)
Efficient Matrix Inversion

• Solve the linear system \((K + \alpha I_n)w = y:\)

\[ w^* = (K + \alpha I_n)^{-1}y \]
Efficient Matrix Inversion

• Approximately solve the linear system \((K + \alpha I_n)w = y\):
  
  • Replace \(K\) by \(LL^T\): \(w^* = (K + \alpha I_n)^{-1}y \approx (LL^T + \alpha I_n)^{-1}y\)
Efficient Matrix Inversion

• Approximately solve the linear system \( (K + \alpha I_n)w = y \)
  
  • Replace \( K \) by \( LL^T \): \( w^* = (K + \alpha I_n)^{-1}y \approx (LL^T + \alpha I_n)^{-1}y \)
  
  • Expand the inversion by the Woodbury identity
Efficient Matrix Inversion

• Approximately solve the linear system \((K + \alpha I_n)w = y\)
  
  • Replace \(K\) by \(LL^T\): \(w^* = (K + \alpha I_n)^{-1}y \approx (LL^T + \alpha I_n)^{-1}y\)
  
  • Expand the inversion by the Woodbury identity

\[
(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}
\]
Efficient Matrix Inversion

• Approximately solve the linear system $(K + \alpha I_n)w = y$
  
  • Replace $K$ by $LL^T$: \[ w^* = (K + \alpha I_n)^{-1}y \approx (LL^T + \alpha I_n)^{-1}y \]
  
  • Expand the inversion by the Woodbury identity
  \[ w^* \approx \alpha^{-1}y + \alpha^{-1}L(\alpha I + L^T L)^{-1}L^Ty \]
Efficient Matrix Inversion

• Approximately solve the linear system \((K + \alpha I_n)w = y\)
  
  • Replace \(K\) by \(LL^T\): \(w^* = (K + \alpha I_n)^{-1}y \approx (LL^T + \alpha I_n)^{-1}y\)
  
  • Expand the inversion by the Woodbury identity
    \[ w^* \approx \alpha^{-1}y + \alpha^{-1}L(\alpha I + L^TL)^{-1}L^Ty \]

• Time cost: \(O(n^c^2)\)

Linear in \(n\), much better than \(O(n^3)\)
Efficient Eigenvalue Decomposition

• Approximately compute the $k$-eigenvalue decomposition of $K$
  • SVD: $L = U_L \Sigma_L V_L$
  • $K \approx LL^T = U_L \Sigma_L^2 U_L^T$
Efficient Eigenvalue Decomposition

• Approximately compute the $k$-eigenvalue decomposition of $K$
  • $\text{SVD: } L = U_L \Sigma_L V_L$
  • $K \approx LL^T = U_L \Sigma^2_L U_L^T$
  • Approximate $k$-eigenvalue decomposition of $K$
    • eigenvectors: the first $k$ vectors in $U_L$

• Time cost: $O(nc^2)$
Efficient Eigenvalue Decomposition

- Approximately compute the \( k \)-eigenvalue decomposition of \( K \)
  - SVD: \( L = U_L \Sigma_L V_L \)
  - \( K \approx LL^T = U_L \Sigma^2_L U_L^T \)
  - Approximate \( k \)-eigenvalue decomposition of \( K \)
    - eigenvectors: the first \( k \) vectors in \( U_L \)

- Time cost: \( O(nc^2) \)
  - Much lower than \( \tilde{O}(n^2k) \)
Sketching Based Models

• How to find such an approximation?

\[ K \approx C U C^T \]
Sketching Based Models

• How to find such an approximation?

\[ \mathbf{K} \approx \mathbf{C} \mathbf{U} \mathbf{C}^T \]

• Sketching based Methods: \( \mathbf{C} = \mathbf{K} \mathbf{S} \in \mathbb{R}^{n \times c} \) is a sketch of \( \mathbf{K} \).
  • \( \mathbf{S} \in \mathbb{R}^{n \times c} \) can be column selection or random projection matrix.
Sketching Based Models

• How to find such an approximation?

\[ K \approx C U C^T \]

• Sketching based Methods: \( C = KS \in \mathbb{R}^{n \times c} \) is a sketch of \( K \).
  • \( S \in \mathbb{R}^{n \times c} \) can be column selection or random projection matrix

• Three methods:
  • The prototype model [HMT11, WZ13, WLZ16]
  • The fast model [WZZ15]
  • The Nyström method [WS15, GM13]
The Prototype Model

• Objective: $K \approx CUC^T$

• Minimize the approximation error by

$$U^* = \arg\min_U \left\| K - CUC^T \right\|_F^2 = C^+K(C^+)^T.$$
The Prototype Model

- Objective: \( K \approx CUC^T \)
- Minimize the approximation error by
  \[ U^* = \arg\min_U \|K - CUC^T\|_F^2 = C^\dagger K(C^\dagger)^T. \]

Extension of the random SVD to SPSD matrix [HMT11]
The Prototype Model

• Objective: $K \approx CUC^T$

• Minimize the approximation error by

$$U^* = \arg\min_U \left\| K - CUC^T \right\|_F^2 = C^\dagger K (C^\dagger)^T.$$ 

• Time: $O(n^2c)$

• The time complexity is nearly the same to the $k$-eigenvalue decomposition.
• It is much faster than the $k$-eigenvalue decomposition in practice.
The Prototype Model

- Objective: $K \approx CUC^T$
- Minimize the approximation error by
  \[
  U^* = \arg\min\limits_U \|K - CUC^T\|^2_F = C^+K(C^+)^T.
  \]
- Time: $O(n^2c)$
- #Passes: one
The Prototype Model

• Objective: $K \approx \text{CUC}^T$

• Minimize the approximation error by

$$U^* = \arg\min_U \|K - \text{CUC}^T\|_F^2 = C^\dagger K(C^\dagger)^T.$$  

• Time: $O(n^2c)$

• #Passes: one

• Memory: $O(nc)$
  
  • Put $k_{ij}$ in memory only when it is visited

  • Keep $C^\dagger$ in memory
The Prototype Model

• Error Bound
  • $k \ll n$ is arbitrary integer
  • $P$ samples $c = O\left(\frac{k}{\epsilon}\right)$ columns by adaptive sampling
  • $\mathbb{E} \left\| K - CU^*C^T \right\|_F^2 \leq (1 + \epsilon) \left\| K - K_k \right\|_F^2$
The Prototype Model

• Limitations
  • $\mathbf{u}^* = \mathbf{c}^\dagger \mathbf{K} (\mathbf{c}^\dagger)^T$
  • Time cost is $O(n^2 c)$
  • Requires observing the whole of $\mathbf{K}$
The Prototype Model

- Prototype model: $K \approx C U^* C^T$, where
  
  $$U^* = \underset{U}{\text{argmin}} \|K - CUC^T\|_F^2.$$
The Fast Model

• Column/row selection
  • Form $P^TKP$ and $P^TC$
The Fast Model

- Column/row selection
  - Form $P^T_K P$ and $P^T_C$

![Diagram showing matrix operations and dimensions]
The Fast Model

• $K \approx C \tilde{U} C^T$, where

$$\tilde{U} = \arg\min_U \left\| P^T(K - CUC^T)P \right\|_F^2.$$
The Fast Model

- Prototype model: $\mathbf{U}^* = \arg\min_{\mathbf{U}} \left\| \mathbf{K} - \mathbf{CUC}^T \right\|_F^2 = \mathbf{C}^\dagger \mathbf{K} (\mathbf{C}^\dagger)^T$

- Fast model: $\tilde{\mathbf{U}} = \arg\min_{\mathbf{U}} \left\| \mathbf{P}^T (\mathbf{K} - \mathbf{CUC}^T) \mathbf{P} \right\|_F^2 = (\mathbf{P}^T \mathbf{C})^\dagger (\mathbf{P}^T \mathbf{K} \mathbf{P}) (\mathbf{C}^T \mathbf{P})^\dagger$. 
The Fast Model

- Prototype model: $U^* = \arg\min_U \|K - CU_C^T\|^2_F = C^T K (C^T)^T$

- Fast model: $\bar{U} = \arg\min_U \|P^T (K - CU_C^T) P\|^2_F = (P^T C)^\dagger (P^T K P) (C^T P)^\dagger$.

- Theory
  - $p = O \left( \sqrt{\frac{nc}{\epsilon}} \right)$
  - $P$ is column selection matrix (according to the row leverage scores of $C$)
  - Then $\|K - C\bar{U}C^T\|^2_F \leq (1 + \epsilon) \|K - U^*C^T\|^2_F$

The faster model is nearly as good as the prototype model!
The Fast Model

• Prototype model: $U^* = \arg\min_U \left\| K - CUC^T \right\|_F^2 = C^\dagger K(C^\dagger)^T$

• Fast model: $\tilde{U} = \arg\min_U \left\| P^T(K - CUC^T)P \right\|_F^2 = (P^TC)^\dagger (P^TKP)(C^TP)^\dagger$.

• Theory
  • $p = O\left(\sqrt{\frac{nc}{\epsilon}}\right)$
  • $P$ is column selection matrix (according to the row leverage scores of $C$)
  • Then $\left\| K - C\tilde{U}C^T \right\|_F^2 \leq (1 + \epsilon) \left\| K - CU^*C^T \right\|_F^2$

• Overall time cost: $O(p^2c + nc^2) = O(nc^3/\epsilon)$
  linear in $n$
The Nyström Method

\[ S(n \times c) : \text{column selection matrix} \]

\[ C = K S(n \times c), \quad W = S J K S = S J C(c \times c) \]

The Nyström method:

\[ K \approx C W^7 C J K \]
The Nyström Method

- $S (n \times c)$: column selection matrix
- $C = KS (n \times c)$
The Nyström Method

- $S (n \times c)$: column selection matrix
- $C = KS (n \times c)$, $W = S^T KS = S^T C (c \times c)$
The Nyström Method

- $\mathbf{S} (n \times c)$: column selection matrix
- $\mathbf{C} = \mathbf{KS} (n \times c)$, $\mathbf{W} = \mathbf{S}^T \mathbf{KS} = \mathbf{S}^T \mathbf{C} (c \times c)$
- The Nyström method: $\mathbf{K} \approx \mathbf{CW}^\dagger \mathbf{C}^T$
The Nyström Method

• $S (n \times c)$: column selection matrix
• $C = KS (n \times c)$, $W = S^T KS = S^T C (c \times c)$
• The Nyström method: $K \approx C W^+ C^T$
• New explanation:
  • Recall the fast model: $\tilde{X} = \arg\min_X \left\| P^T (K - XC^T) P \right\|_F^2$
The Nyström Method

- \( S (n \times c) \): column selection matrix
- \( C = KS (n \times c) \), \( W = S^T KS = S^T C (c \times c) \)
- The Nyström method: \( K \approx C W^\dagger C^T \)
- New explanation:
  - Recall the fast model: \( \tilde{X} = \arg\min_X \|P^T(K - CXC^T)P\|_F^2 \)
  - Setting \( P = S \), then
    \[
    \tilde{X} = \arg\min_X \|S^T(K - CXC^T)S\|_F^2
    \]
The Nyström Method

- \( S (n \times c) \): column selection matrix
- \( C = KS (n \times c) \), \( W = S^T KS = S^T C (c \times c) \)
- The Nyström method: \( K \approx C W^\dagger C^T \)

New explanation:
- Recall the fast model: \( \tilde{X} = \arg\min_X \|P^T(K - CXC^T)P\|_F^2 \)
- Setting \( P = S \), then

\[
\tilde{X} = \arg\min_X \|S^T(K - CXC^T)S\|_F^2 \\
= (S^T C)^\dagger (S^T KS)(C^T S)^\dagger
\]
The Nyström Method

- \( S (n \times c) \): column selection matrix
- \( C = KS (n \times c), W = S^T KS = S^T C (c \times c) \)
- The Nyström method: \( K \approx CW^+ C^T \)
- New explanation:
  - Recall the fast model: \( \tilde{X} = \text{argmin}_X \| P^T (K - CXC^T) P \|_F^2 \)
  - Setting \( P = S \), then
    \[
    \tilde{X} = \text{argmin}_X \| S^T (K - CXC^T) S \|_F^2
    = (S^T C)^+ (S^T KS) (C^T S)^+
    = W^+ W W^+.
    \]
The Nyström Method

• \( S (n \times c) \): column selection matrix

• \( C = KS (n \times c) \), \( W = S^T KS = S^T C (c \times c) \)

• The Nyström method: \( K \approx CW^\dagger C^T \)

• New explanation:
  
  • Recall the fast model: \( \tilde{X} = \arg\min_X \left\| P^T (K - XC^T)P \right\|_F^2 \)
  
  • Setting \( P = S \), then

\[
\tilde{X} = \arg\min_X \left\| S^T (K - XC^T)S \right\|_F^2
\]

\[
= (S^T C)^\dagger (S^T KS)(C^T S)^\dagger
\]

\[
= W^\dagger WW^\dagger = W^\dagger
\]
The Nyström Method

• \( S (n \times c) \): column selection matrix
• \( C = KS (n \times c), W = S^T KS = S^T C (c \times c) \)
• The Nyström method: \( K \approx C W^\dagger C^T \)

New explanation:

• Recall the fast model: \( \tilde{X} = \arg\min_X \| P^T (K - CXC^T) P \|^2_F \)
• Setting \( P = S \), then

\[
\tilde{X} = \arg\min_X \| S^T (K - CXC^T) S \|^2_F = (S^T C)^\dagger (S^T KS) (C^T S)^\dagger = W^\dagger WW^\dagger = W^\dagger
\]

• The Nyström method is special instance of the fast model.
The Nyström Method

• $S (n \times c)$: column selection matrix

• $C = KS (n \times c)$, $W = S^T KS = S^T C (c \times c)$

• The Nyström method: $K \approx CW^\dagger CT$

• New explanation:
  
  • Recall the fast model: $\tilde{X} = \arg\min_X \left\|P^T(K - CXC^T)P\right\|_F^2$
  
  • Setting $P = S$, then
    
    $\tilde{X} = \arg\min_X \left\|S^T(K - CXC^T)S\right\|_F^2$
    
    $= (S^TC)^\dagger (S^T KS)(C^T S)^\dagger$
    
    $= WW^\dagger = W^\dagger$

• The Nyström method is a special instance of the fast model.
• It is an approximate solution to the prototype model.
The Nyström Method

• Cost
  • Time: $O(nc^2)$
  • Memory: $O(nc)$
The Nyström Method

• Cost
  • Time: $O(nc^2)$
  • Memory: $O(nc)$

Very efficient!
The Nyström Method

• Cost
  • Time: $O(nc^2)$
  • Memory: $O(nc)$

• Error bound: weak

Very efficient!
Comparisons

\[ C = KS \in \mathbb{R}^{n \times c}, \quad W = S^T KS = S^T C \in \mathbb{R}^{c \times c} \]

- SPSD matrix approximation: \( K \approx CUC^T \)
  - The prototype model: \( U = C^\dagger K (C^\dagger)^T \)
  - The fast model: \( U = (P^T C)^\dagger (P^T KP)(C^T P)^\dagger \)
  - The Nyström method: \( U = W^\dagger \)
Comparisons

- \( \mathbf{C} = \mathbf{KS} \in \mathbb{R}^{n \times c}, \mathbf{W} = \mathbf{S}^T \mathbf{KS} = \mathbf{S}^T \mathbf{C} \in \mathbb{R}^{c \times c} \)
- SPSD matrix approximation: \( \mathbf{K} \approx \mathbf{CUC}^T \)
  - The prototype model: \( \mathbf{U} = \mathbf{C}^\dagger \mathbf{K} (\mathbf{C}^\dagger)^T \)
  - The fast model: \( \mathbf{U} = (\mathbf{P}^T \mathbf{C})^\dagger (\mathbf{P}^T \mathbf{KP})(\mathbf{C}^T \mathbf{P})^\dagger \)
  - The Nyström method: \( \mathbf{U} = \mathbf{W}^\dagger \)

When \( \mathbf{P} = \mathbf{I}_n \), the prototype model \( \Leftrightarrow \) the fast model
Comparisons

• $C = KS \in \mathbb{R}^{n \times c}$, $W = S^T KS = S^T C \in \mathbb{R}^{c \times c}$

• SPSD matrix approximation: $K \approx CUC^T$
  
  • The prototype model: $U = C^T K (C^T)^T$
  • The fast model: $U = (P^T C)^T (P^T KP) (C^T P)^T$
  • The Nyström method: $U = W^T$

When $P = S$, the Nyström method $\Leftrightarrow$ the fast model
Comparisons

- $c = 150$, $n = 100c$, vary $p$ from $2c$ to $40c$

\[
\frac{\left\| K - \text{UCU}^T \right\|^2_F}{\left\| K \right\|^2_F}
\]

The Nyström Method
$O(nc^2)$ time

The Fast Model
$O(nc^2 + p^2c)$ time

The Prototype Model
$O(n^2c)$ time
Conclusions

• Motivations
  • Avoid forming the kernel matrix
  • Avoid inversion/decomposition

• Prototype model, fast model, Nystrom
  • They have connections
  • The fast model and Nystrom are practical