UCLouvain

École polytechnique de Louvain

Numerical investigation of a continuous relaxation for the Column Subset Selection Problem

Author: **Alexis Victor**
Supervisor: **Estelle Massart**
Readers: **Pierre-Antoine Absil, Estelle Massart, Bastien Massion**
Academic year 2023–2024
Master [120] in Mathematical Engineering

Abstract

This thesis presents a novel approach to approximating the Column Subset Selection Problem (CSSP) using a regularized continuous optimization method. The CSSP, which involves selecting a subset of columns from a given matrix to best approximate the original matrix, is a combinatorial problem that is NP-hard. This thesis introduces a method that transforms the combinatorial problem into a continuous optimization task by employing an L_1 orthogonal regularization term.

The proposed method is particularly well-suited for datasets where the number of features significantly exceeds the number of objects ($m \ll n$). The thesis includes a comprehensive analysis of the algorithm's performance against existing methods.

Results from experiments on both synthetic and real-world datasets demonstrate that the proposed method offers competitive precision and computational efficiency compared to existing methods. Its effectiveness in scenarios where the number of features far exceeds the number of objects ($m \ll n$) makes it a viable solution for large-scale applications.

Acknowledgment

First of all, I want to express my sincere thanks to my thesis supervisor Professor Estelle Massart for her constant support and guidance throughout this project. Her feedback and time were essential in helping me complete this thesis. The meetings and discussions we had were always valuable and helped me stay on track.

I also want to thank Bastien Massion, the PhD candidate who closely followed my work throughout the year. His advice and support were greatly appreciated and made a big difference in my progress.

Lastly, I want to thank my family and friends for their support and encouragement throughout this time. Their understanding and belief in me helped keep me going, especially during the tough moments.

Notation

Matrix Numbering

Consider the matrix X defined as:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

The i-th column of X, denoted by X_i, is:

$$X_{[:,i]} = X_i = \begin{bmatrix} x_{1i} \\ x_{2i} \\ \vdots \\ x_{mi} \end{bmatrix}$$

The i-th row of X, denoted by ${X^\top}_i$, is:

$$X_{[i,:]} = {X^\top}_i = \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{in} \end{bmatrix}$$

In this document, the notations X_i and $X_{[i,:]}$ will be preferred for referring to the i-th column and row of X, respectively.

Contents

Chapter 1

Introduction

Understanding the world is a universal pursuit, not just for mathematicians but for everyone. The first step in this process is being able to describe and differentiate various realities. In the context of data science, it involves organizing and analyzing datasets. A dataset consists of a group of objects, each described by a set of features. For instance, a dataset with m objects and n features can be stored in a matrix with m rows and n columns, where each row represents the features of a single object.

Scientists often apply methods such as classification, clustering, regression, etc, to extract more advanced insights from these datasets. However, to avoid overfitting and to reduce computational costs, it is often necessary to compress the dataset by reducing the number of features. One of the most commonly used techniques for this purpose is Principal Component Analysis (PCA).

The core idea of PCA is to generate a smaller number of s new features, called principal components, that capture the most important information from the original data and project the data in this new subspace. These components are designed to retain as much variance as possible from the original dataset. For example, as described in [6], consider a dataset that describes people using three features: age, height, and income. After applying PCA, the first principal component might be a combination such as $(\frac{1}{2}\text{age} - \frac{1}{\sqrt{2}}\text{height} + \frac{1}{2}\text{income})$. While PCA is powerful, it often falls short when it comes to providing meaningful interpretations. The principal components are mathematical abstractions that do not always have a direct physical or intuitive interpretation.

A non-mathematician might simply select two features that seem to capture the

most information about a person. Mathematically, this optimal selection can be described as minimizing the Frobenius norm of the difference between the original dataset matrix and its projection into the s-dimensional space spanned by the selected columns [2]. Finding such a subset of columns is a combinatorial problem, known as the Column Subset Selection Problem (CSSP).

This master thesis presents a new method to approximate the CSSP using regularized continuous optimization. It is focus on dataset that have more feature than 'objects'. In the matrix framework, describe above it translates as $n \geq m$. The method employs a regularization term that, when minimized, enforces the combinatorial selection of features [7]. This regularization transforms the combinatorial problem into a continuous optimization problem.

This master thesis presents a new method to approximate the CSSP using regularized continuous optimization. It focuses on datasets that have more features than objects. In the matrix framework described above, this translates to $n \geq m$. The method employs a regularization term that, when minimized, enforces the combinatorial selection of features [7]. This allows to then drop the combinatorial constraint. This allows the combinatorial constraint to be dropped, transforming the problem into a continuous optimization problem.

The thesis is organized as follows. Section 2 provides a brief overview of the mathematical tools used later. In Section 3, the CSSP is redefined using mathematical formalism, followed by a review of existing methods that solve, approximate or bound the CSSP. Section 4 offers a detailed description of the proposed method and analyzes its complexity. The results section provides an extensive empirical comparison of precision, computational cost, and efficiency between the proposed methods and state-of-the-art algorithms. The thesis concludes with a summary and suggests directions for future research.

Chapter 2

Theoretical background

If not specified, the definitions of the following theoretical concepts are derived from the book "Matrix Computations (Fourth Edition)" by Golub and Van Loan [4].

2.1 Matrix norms

$\|.\| : \mathbb{R}^{m\times n} \to \mathbb{R}$ is a matrix norm if the following three properties hold:

$$\begin{aligned} f(A) \geq 0, \quad A \in \mathbb{R}^{m\times n}, & \qquad (f(A) = 0 \text{ iff } A = 0) \\ f(A+B) \leq f(A) + f(B), & \qquad A, B \in \mathbb{R}^{m\times n}, \\ f(\alpha A) = |\alpha| f(A), & \qquad \alpha \in \mathbb{R}, A \in \mathbb{R}^{m\times n}. \end{aligned}$$

The two matrix norms used in this thesis are the Frobenius norm and the L_1 entrywise norm.

- **Frobenius Norm**: $\|A\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2\right)^{\frac{1}{2}}$
- **Entrywise L_1 Norm**: $\|A\|_1 = \sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|$

2.2 SVD

Theorem 1. ***Singular value decomposition [4]*** *If X is a real $m \times n$ matrix, then there exist orthogonal matrices*

$$U = \begin{bmatrix} u_1 & \cdots & u_m \end{bmatrix} \in \mathbb{R}^{m \times m} \quad \textit{and} \quad V = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix} \in \mathbb{R}^{n \times n}$$

such that

$$\begin{aligned} &U^\top X V = \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m \times n}, \quad p = \min\{m, n\}, \\ &X = U \Sigma V^\top \end{aligned}$$

Σ is an $m \times n$ rectangular diagonal matrix where the diagonal entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ are the singular values of X.

The time complexity for computing the SVD is $\mathcal{O}(\min(mn^2, m^2n))$.

2.2.1 Truncated SVD and low-rank approximation

The truncated singular value decomposition is a version of the SVD where only the top k singular values and corresponding singular vectors are kept. Let $X \in \mathbb{R}^{m \times n}$.

$$X = U \Sigma V^\top$$

For a given rank $s \in \{1, \ldots, n\}$, $p = \min(m, n)$, partition U, Σ, and V as follows:

$$U = \begin{bmatrix} U_s & U_{m-s} \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_s & 0 \\ 0 & \Sigma_{p-s} \end{bmatrix}, \quad V = \begin{bmatrix} V_s & V_{n-s} \end{bmatrix},$$

where $U_s \in \mathbb{R}^{m \times s}$, $\Sigma_s \in \mathbb{R}^{s \times s}$, and $V_s \in \mathbb{R}^{n \times s}$. Σ_p is an $(m-s) \times (n-s)$ rectangular diagonal matrix with diagonal entries $(\sigma_{n-s+1}, \ldots, \sigma_p)$.

The rank-s approximation of X, denoted by X_s, obtained from the truncated singular value decomposition, is given by:

$$X_s = U_s \Sigma_s V_s^\top$$

X_s minimizes the Frobenius norm of the difference between X and X_s:

$$\|X - X_s\|_F = \min_{\operatorname{rank}(\tilde{X}) \leq s} \|X - \tilde{X}\|_F = \sqrt{\sigma_{s+1}^2 + \cdots + \sigma_n^2}.$$

From here on, this approximation will be referred to as the best rank-s approximation.

2.2.2 PseudoInverse (Moore-Penrose inverse)

For $A \in \mathbb{R}^{m\times n}$, the Pseudoinverse $A^\dagger$ is the matrix satisfying the following properties:

$$\begin{aligned}&\text{(i)} \quad AXA = A,\\ &\text{(ii)} \quad XAX = X,\\ &\text{(iii)} \quad (AX)^\top = AX,\\ &\text{(iv)} \quad (XA)^\top = XA.\end{aligned}$$

The Pseudoinverse can also be defined using the SVD as $A^\dagger = V\Sigma^\dagger U^T$, where

$$\Sigma^\dagger = \operatorname{diag}\left(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_r}, 0, \ldots, 0\right) \in \mathbb{R}^{n\times m},$$

with $r = \operatorname{rank}(A)$.

2.3 Minimizers of orthogonal regularizers

The following theorem has been proved by E. Massart and B. Massion [7].

Theorem 2. *For $n > s$, the optimization problem:*

$$\min_{W\in\mathbb{R}^{n\times s}} \|WW^\top - I_n\|_1, \tag{2.1}$$

with $\|A\|_1 = \sum_{i,j} |A(i,j)|$ being the entrywise L_1 norm, has the solution set:

$$S = \left\{W \in \mathbb{R}^{n\times s} : W = P\begin{pmatrix} I_s \\ 0_{(n-s)\times s}\end{pmatrix} Q,\ P \in \Pi_n,\ Q \in \mathcal{O}_s\right\} \tag{2.2}$$

where Π_n is the set of all $n \times n$ permutation matrices, and $\mathcal{O}_s$ is the orthogonal group of size $s \times s$.

Chapter 3

Problem definition and existing methods

Definition 1. ***CSSP [2]:*** *Given a matrix $A \in \mathbb{R}^{m \times n}$ and a positive integer s, select s columns of A forming a matrix $C \in \mathbb{R}^{m \times s}$ such that the residual*

$$\|A - P_C A\|_F$$

is minimized over all possible $\binom{n}{s}$ choices for the matrix C, here $P_C = CC^{\dagger}$ denotes the projection onto the s-dimensional space spanned by the columns of C .

This problem is trivial when $\text{rank}(X) = r \leq s$, as a selection of s columns which r of which are linearly independent yields a zero-objective value. However, the problem has been proven NP-hard when $s < \text{rank}(X) = r \leq \min\{m, n\}$ [10].

3.1 Brute force

Since the problem is NP-complete, the only exact solution can be found using a brute-force approach, which involves evaluating the objective function for every possible combination of columns. For the selection of s columns from a matrix of size $m \times n$, the number of possible combinations is given by the binomial coefficient $\binom{n}{s}$. Using Stirling's approximation and considering the worst-case scenario where $s = \frac{n}{2}$, it becomes:

$$\binom{n}{s} = \frac{n!}{s!(n-s)!} \sim \sqrt{\frac{n}{2\pi s(n-s)}} \cdot \frac{n^n}{s^s(n-s)^{n-s}} \sim \frac{2^n}{\sqrt{n\pi}}$$

The theoretical computational complexity of the brute-force method is determined by this number of combinations multiplied by the computational cost of evaluating the objective function $F_C(C) = \|X - CC^\dagger X\|_F$, which is in $\mathcal{O}(mns)$. Specifically, the complexity of computing the pseudoinverse of C is in $\mathcal{O}(ms^2)$ which dominates $\mathcal{O}(mns)$. The matrix multiplication involved in computing F_C contributes an additional $\mathcal{O}(mns)$ from the operation $C(C^\dagger X)$. Therefore, the overall time complexity of the brute-force method is:

$$\text{BruteForce}(X) \in \mathcal{O}\left(\frac{2^n}{\sqrt{n\pi}} \cdot mns\right) = \mathcal{O}\left(2^n \cdot m\sqrt{n}s\right)$$

In practice, this method is only feasible for very small problem sizes.

3.2 Best low-rank approximation as a lower bound of the CSSP

This is exactly the same as the PCA. As seen in section 2.2.1, the low-rank approximation using the SVD truncated offers a lower bound for the Column Subset Selection Problem (CSSP).

Lemma 1. *For any matrix $X \in \mathbb{R}^{m\times n}$ and its truncated singular value decomposition $X_s = U_s\Sigma_s V_s^\top$, the following relation holds:*

$$\|X - (XV_s)(XV_s)^\dagger X\|_F = \|X - X_s\|_F \leq \|X - CC^\dagger X\|_F, \quad \forall C \in \mathbb{R}^{n\times s}. \quad (3.1)$$

Proof. The right inequality is known and proven by the Eckart-Young theorem [4]. For the left equality:

$$\begin{aligned}
\|X - (XV_s)(XV_s)^\dagger X\|_F &= \|X - U\Sigma(V^\top V_s)V_s^\top X^\dagger X\|_F \\
&= \|X - U\Sigma \begin{pmatrix} I_s \\ 0_{n-s\times s} \end{pmatrix} V_s^\top V \underbrace{\Sigma^{-1}U^T U\Sigma}_{I_m} V^T\|_F \\
&= \|X - U_s\Sigma_s \begin{pmatrix} I_s \\ 0_{n-s\times s} \end{pmatrix} V^\top\|_F \\
&= \|X - X_s\|_F
\end{aligned}$$

□

This lemma demonstrates that the objective function of the proposed method (see chapter 4) is equivalent to the loss function of the low-rank approximation using the truncated SVD. In other words, without the regularization term, the proposed method converges to the best low-rank s approximation of X.

3.3 Stochastic continuous landmark selection

Mathur et al. [8] propose a method that transforms the problem into a continuous one by incorporating a regularization term and applying the constraints through this regularization. Recall, The CSSP is defined by :

$$\begin{aligned} &\min \|X - P_s X\|_F \qquad P_s := CC^\dagger \\ =&\min -\operatorname{tr}[X^\top P_s X] \end{aligned}$$

The problem is reformulated to find the submatrix C of X by introducing a binary vector h.

$$\min_{h\in\{0,1\}^n} -\operatorname{tr}[X^\top P_s(h) X)] \qquad \text{subject to } \sum_{i=1}^{n} h_i \leq s \tag{3.2}$$

$P_{\mathbf{s}}(h)$ is the projection matrix onto $\operatorname{span}\{\mathbf{x}_i : h_i = 1, i = 1, \ldots, n\}$.

3.3.1 Regularization

The matrix $\widetilde{P}$ is introduced as a continuous generalization of P_s:

Definition 2. *For $t = (t_1, \ldots, t_n)^\top \in [0,1]^n$, let $T := Diag(t)$ be the diagonal matrix with diagonal elements $t_1, \ldots, t_n$, and define:*

$$\widetilde{P}(t) := XT\left[TX^\top XT + \delta(I - T^2)\right]^\dagger TX^\top,$$

where $\delta > 0$ is a fixed constant.

In this formulation, the binary vector h is replaced by the continuous vector t. The constraint is incorporated into the objective function through an associated regularization factor λ, leading to the following regularized problem for the CSSP:

$$\min_{t\in[0,1]^n} \underbrace{-\operatorname{tr}[X^\top \widetilde{P}(t) X] + \lambda \sum_{j=1}^{n} t_j}_{:=F_\lambda^X(t)} . \tag{3.3}$$

This problem is minimized using a stochastic gradient descent (SGD) approach. The domain of t is a hypercube $[0,1]^n$, and as t_i approaches 1, the corresponding column i is likely to be included in the submatrix C. The parameter λ influences the number of elements in t that converge to 1.

To obtain an approximate solution, SGD is applied to the regularized objective function. Starting from an interior point within the hypercube, the vector t evolves under the influence of SGD, gradually moving toward a corner point. During this process, some components of t_j are driven to zero effectively identifying which columns of X should not be selected.

To ensure an unbiased initial guess, the starting vector is set to $t = (1/2, \ldots, 1/2)$.

3.3.2 Stochastic gradient descent

The method solves the minimization problem 3.3 using a stochastic gradient descent. The stochastic gradient descent is defined by the following lemma:

Lemma 2. *Let $z \in \mathbb{R}^n$ be a random vector sampled from the Rademacher distribution*[1]*. For $T = Diag(t)$ where $t \in [0,1]^n$, define $K = X^\top X$, $Z = K - \delta I_n$, and $L_t = TZT + \delta I_n$. Then:*

$$
\begin{aligned}
a &= Kz, \\
b &= L_t^{-1}(t \odot a), \\
\phi &= b \odot Z(t \odot b) - a \odot b.
\end{aligned}
$$

The gradient $\nabla F_\lambda^X(t)$ is given by:

$$\nabla F_\lambda^X(t) = 2E[\phi] + \lambda \mathbf{1},$$

for $t \in (0,1)^n$.

Termination condition

The stopping criterion for the stochastic gradient descent method is not specified in the method description. To ensure consistent comparisons between methods,

[1] values are -1 or 1, each with probability $1/2$.

the chosen stopping criterion is aligned with the one used in the proposed method (section 4.2.4). Specifically, the method terminates if F_λ^X does not improve by more than ϵ over 10 consecutive iterations.

3.3.3 Box constraints

Applying the stochastic gradient descent to $F_\lambda^X(t)$ is not enough to solve the continuous extension of the CSSP. Indeed t needs to be enforce to $t \in [0,1]^n$, named as the box constraints; It is done using the following mapping $t_j(w_j) = 1 - \exp(-w_j^2), \quad j = 1, \ldots, n,$. In vector form the transformation is:

$$t(w) = 1 - \exp(-w \star w)$$

where $\star$ denotes the element-wise multiplication. Considering the optimization of the continuous CSSP,

$$w^* = \arg\min_{w \in \mathbb{R}^p} F_\lambda^X(t(w)),$$

the solution to (3.3) is attained by evaluating $t(w^*)$. Using this using the chain rule, the gradient develops as:

$$\frac{\partial F_\lambda^X(t(w))}{\partial w} = \frac{\partial F_\lambda^X(t(w))}{\partial t} \odot \left(2w \odot \exp(-w \odot w)\right).$$

3.3.4 Theoretical complexity analysis

When computing the stochastic gradient, instead of directly inverting L_t, the method solves the system $L_t b = t \odot a$. This also has a theoretical complexity of $\mathcal{O}(n^3)$, but it demonstrates better empirical computational time. Mathur suggests approximating the solution b using the conjugate gradient (CG) algorithm, an iterative method with a complexity of $\mathcal{O}(n^2\ell)$, where ℓ denotes the number of CG iterations [4]. In specific cases, this complexity can be reduced to $\mathcal{O}(n^2)$ under the assumption $\ell \ll n$, though this assumption generally does not hold. Thus, the general complexity for this step remains $\mathcal{O}(n^3)$, contrary to Mathur's conclusion.

At each iteration, M Monte Carlo samples are computed, leading to a per-iteration time complexity of $\mathcal{O}(n^3 M)$. Consequently, the overall time complexity is:

$$\mathcal{O}(NMn^3), \tag{3.4}$$

where N is the number of SGD iterations, and M represents the Monte Carlo sampling size.

3.3.5 Algorithm summary

The method is summarized in Algorithm 1:

Algorithm 1 Continuous Stochastic Landmark Selection

Input: $m \times n$ matrix A, integer s.
Output: $m \times s$ matrix C with s columns of A.

1: **Initial setup:**

- Define $t_0 = (1/2, \ldots, 1/2)$.
- Initialize $w^{(0)} \leftarrow \sqrt{-\ln(1 - t^{(0)})}$.

2: **Continuous Optimization Phase:**

- Perform SGD on the regularized objective function $F_{X,\lambda}(t(w)) = \|X - \widetilde{P}(t)X\|_F^2 + \lambda \sum_{j=1}^{n} t_j$. The iteration is:

$$w_{k+1} = w_k - \alpha \nabla_w F_{X,\lambda}(w_k) \odot (2w \odot \exp(-w \odot w)) .$$

- Termination condition: Stop when the objective function improves by less than tolerance ϵ over the last 10 iterations.

3: **Retrieving C from $\mathbf{t}^*$:**

- Compute $t^* = t(w^*) = 1 - \exp(-w^* \star w^*)$.
- Select the indices corresponding to the s largest elements of $\mathbf{t}^*$ as the columns for $\mathbf{C}$.

return $X_{[\mathbf{s}^*]}$

3.4 Randomized deterministic double phase

The method proposed by Boutsidis, Mahoney and Drineas [2] is a two-stage algorithm. The first stage is randomized, while the second stage is deterministic. In

the first stage, $c = \mathcal{O}(s \log s)$ columns are selected using the randomized leverage score method. This step provides a solid starting point for the deterministic second stage, where exactly s columns are selected.

The hyperparameter c plays a crucial role in the effectiveness of the method. It is important that c is larger than s to ensure that the deterministic phase has enough columns to choose from. However, if c is too large, the benefits of the first, randomized phase becomes null.

3.4.1 First phase: Randomized leverage scores

The randomized stage is based on leverage scores, using a specific matrix formalism to ease notation and combine with the second stage of the algorithm. The truncated singular value decomposition of X is computed to obtain the matrix of the top s right singular vectors, denoted $V_s \in \mathbb{R}^{s \times n}$. Using V_s, the n associated sampling probabilities are calculated as follows:

$$p_i = \|(V_s)_i\|_2^2 / s \tag{3.5}$$

The algorithm retains the i-th column of V_s with probability p_i. The number of columns chosen, denoted $\tilde{c}$, has an expected value of c. A sampling matrix S_1 of size $n \times \tilde{c}$ is then constructed so that $V_s^\top S_1$ is the subsampled matrix of V_s. A diagonal matrix D_1 is created with scaling factors $1/\sqrt{\min(1, cp_i)}$.

The randomized leverage score method would return AS_1.

3.4.2 Second phase: Deterministic column selection

In the deterministic stage, a square $(s \times s)$ submatrix is constructed by selecting s columns from the $(s \times \tilde{c})$ matrix $V_s^\top S_1 D_1$. This selection is performed using Algorithm 1 from Pan [9].

Pan's algorithm uses a rank-revealing LU factorization (RRLU) to identify the square submatrix with the largest local volume. The concepts of matrix volume and local maximum volume are defined as follows:

Definition 3. *[9] Let $A \in \mathbb{R}^{m \times n}$, and let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ ($p = \min(m, n)$)*

be the singular values of A. The volume of A is defined as:

$$vol(A) = \sigma_1(A)\sigma_2(A)\cdots\sigma_p(A) \tag{3.6}$$

Definition 4. *[9] Let $A \in \mathbb{R}^{m\times n}$ and B be a submatrix of A formed by any k columns (rows) of A. $vol(B) \neq 0$ is said to be a local maximum volume in A if:*

$$vol(B) \geq vol(B') \tag{3.7}$$

for any B' obtained by replacing one column (row) of B with a column (row) of A that is not in B.

Using the indices of the selected columns, the method constructs the $(\tilde{c}\times s)$ sampling matrix S_2, ensuring that the matrix with the maximum local volume at the end of the second phase is $V_s^\top S_1 D_1 S_2$.

The final matrix C, which is the solution to the CSSP problem, is given by XS_1S_2.

3.4.3 Complexity analysis

The algorithm runs in $\mathcal{O}(\min(mn^2, m^2n))$ time. Since this thesis focuses on problems where n is larger than m, the *Double-Phase* algorithm, when applied to our restricted CSSP problem, operates with a time complexity of $\mathcal{O}(m^2n)$.

3.4.4 Algorithm summary

The method is summarized in Algorithm 2

Algorithm 2 Randomized Deterministic Double Phase

Input: $m \times n$ matrix A, integer s.
Output: $m \times s$ matrix C with s columns of A.

1. **Initial setup:**
   - Compute the top s right singular vectors of A, denoted by V_s.
   - Compute the sampling probabilities p_i for $i \in [n]$, using 3.4.1
   - Let $c = \Theta(s \log s)$.
2. **Randomized Stage:**
   - For $i = 1, \ldots, n$, keep the i-th index with probability $\min\{1, cp_i\}$. If the i-th index is kept, keep the scaling factor $\sqrt{\min\{1, cp_i\}}$.
   - Form the sampling matrix S_1 and the rescaling matrix D_1
3. **Deterministic Stage:**
   - Run Algorithm 1 of Pan [9] on the matrix $V_s^\top S_1 D_1$ in order to select exactly s columns of $V_s^\top S_1 D_1$, thereby forming the sampling matrix S_2

return $X_{\mathbf{s}^*} = AS_1S_2$

Chapter 4

Proposed Method

4.1 Description of the method

The proposed method is called Column Subset Selection with L_1 *Orthogonal Regularization* (L_1OR). The key aspect of the method is the use of Theorem 2.3. This theorem allows to inject the combinatorial constraint into the objective function, enabling the use of continuous optimization techniques.

4.1.1 Regularization of the objective function

Recall, the CSSP is define as:

$$\min_{C \in \mathbb{R}^{m \times s}} \|X - CC^{\dagger}X\|_F^2$$

where C is a submatrix of X made of s columns of X. Note that we can write any submatrix C of X made of s columns of X as follows:

$$C = XP \begin{pmatrix} I_s \\ 0_{(n-s)\times s} \end{pmatrix} = XP_s \tag{4.1}$$

where $P \in \Pi_n$, the set of $n \times n$ permutation matrices and P_s the s first columns matrix of P.

The interpretation of this representation is first to permute the columns of the original matrix, and then to keep only the s first columns. Using this representation

for the submatrix, the column subset selection problem can be written as :

$$\begin{aligned} &\min_{C\in\mathbb{R}^{m\times s}} \|X - CC^{\dagger}X\|_F^2 \\ &\quad C = XP\begin{pmatrix} I_s \\ 0_{(n-s)\times s}\end{pmatrix} \\ &\quad P \in \Pi_n \end{aligned} \tag{4.2}$$

Therefore, problem 4.2 can be reformulated as a combinatorial problem such that the permutation matrix P is the only variable (the combinatorial aspect comes from the integer constraints on the entries of the matrix P). We make the following observation, the objective value in equation 4.2 is invariant to orthogonal rotations of C on the right. In other words any matrix:

$$XP\begin{pmatrix} I_s \\ 0_{(n-s)\times s}\end{pmatrix} Q = CQ =: XW$$

with $Q \in \mathcal{O}_s$ the set of orthogonal matrices, shares the same objective value as C. This implies the equivalence between problem 4.2 and:

$$\begin{aligned} &\min_{W\in\mathbb{R}^{n\times s}} \|X - XW(XW)^{\dagger}X\|_F^2 \\ &\quad W = P\begin{pmatrix} I_s \\ 0_{(n-s)\times s}\end{pmatrix} Q \\ &\quad P \in \Pi_n \\ &\quad Q \in \mathcal{O}_s \end{aligned} \tag{4.3}$$

Besides, it was shown by E. Massart and B. Massion [7],that for $n > s$, the optimization problem:

$$\min_{W\in\mathbb{R}^{n\times s}} \|WW^{\top} - I_n\|_1, \tag{4.4}$$

with $\|A\|_1 = \sum_{i,j} |A(i,j)|$ the entrywise l_1 norm, has for solution the set:

$$\mathcal{S} = \left\{ W \in \mathbb{R}^{n\times s} : W = P\begin{pmatrix} I_s \\ 0_{(n-s)\times s}\end{pmatrix} Q, P \in \Pi_n, Q \in \mathcal{O}_s \right\} \tag{4.5}$$

There follows that the feasible set of problem 4.3 coincides with the set $\mathcal{S}$ of solutions of problem 4.4. This naturally leads to a regularized approach for solving CSSP, replacing the constraints of problem 4.3 by the relaxed problem:

$$\min_{W\in\mathbb{R}^{n\times s}} \left\|X - XW(XW)^{\dagger}X\right\|_F^2 + \lambda \left\|WW^{\top} - I_n\right\|_1 \tag{4.6}$$

with λ a regularization parameter. This minimization is done using a gradient descent which is detailed later in the section 4.2. From W^*, it is possible to recover the permutation matrix P^* and then the submatrix C^*. Indeed, for any $W \in \mathcal{S}$ the following factorization of W is one possible singular value decomposition of W:

$$W = P \begin{pmatrix} I_s \\ 0_{(n-s)\times s} \end{pmatrix} Q \qquad P \in \Pi_n, Q \in \mathcal{O}_s \tag{4.7}$$

4.1.2 Retrieving C from W

Lemma 3. *Let $X \in \mathbb{R}^{p\times q}$ be a matrix with $p > q$. Suppose X has a singular value decomposition given by:*

$$X = U\Sigma V,$$

with all singular value equal to one, i.e., $\Sigma_i = 1 \quad \forall i \in \{1, \dots, q\}$. Then, for any matrix $B \in \mathcal{O}_p$, the matrix UB and $B_q^\top V$ are also a valid pair of left and right singular matrices for X. $B_q^\top$ denotes the submatrix made of the q first columns of $B^\top$.

Proof. Taking the specific singular value decomposition of X:

$$X = U \begin{pmatrix} I_m \\ 0_{(m-n)\times m} \end{pmatrix} V,$$

Injecting B,

$$\begin{aligned} X &= U(BB^\top) \begin{pmatrix} I_q \\ 0_{(p-q)\times q} \end{pmatrix} V \\ &= UB \begin{pmatrix} B_q^\top \\ 0_{p-q} \end{pmatrix} V \\ &= \underbrace{UB}_{U'} \begin{pmatrix} I_q \\ 0_{p-q} \end{pmatrix} \underbrace{B_q^\top V}_{V'} \\ X &= U'\Sigma V', \end{aligned}$$

U' and V' are well orthogonal as,

$$\begin{aligned} {U'}^\top U' = U'{U'}^\top = B^\top U^\top UB = UBB^\top U^\top = I_p \\ {V'}^\top V' = V'{V'}^\top = V^\top B_q B_q^\top V = B_q^\top UU^\top B_q = I_q \end{aligned}$$

□

From Lemma 3, after computing the SVD of W numerically using standard packages, the general form $W = P'IQ'$ is obtained, where P' is an orthogonal matrix, though not necessarily a permutation matrix. The first s left singular vectors of W form the matrix P'_s.

The submatrix P_s is a selection matrix with each column containing exactly one entry equal to 1, while all other entries are zero. Each row contains at most one non-zero entry. For example, if $X \in \mathbb{R}^{3\times 4}$ and $s = 2$, a possible form of P_s could be:

$$P_s = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

When multiplying X by P_s, resulting in $C = XP_s$, it becomes evident that the indices of the selected columns in C correspond to the indices of the non-zero rows in P_s.

Since $P'_s = P_sB$ (where B is an orthogonal matrix), the rows in P_s that are non-zero correspond to the same indices as the non-zero rows in P'_s. Consequently, the selected columns of X are determined by the indices of the non-zero rows in P'_s.

4.1.3 L_1 orthogonalization of W^*

However, W^* does not generally belong to the set $\mathcal{S}$. This is because W^* minimizes a penalized objective function that balances the trade-off between belonging to the set $\mathcal{S}$ and minimizing the objective function of the CSSP. This trade-off, along with the influence of the regularization factor λ and the step size α, is further discussed in Section 4.3.

It is hypothesized that with the appropriate choice of hyperparameters λ and α, W^* is near the set $\mathcal{S}$[1]. Assuming this proximity, the goal is to find a matrix $\bar{W} \in \mathcal{S}$ that represents the L_1 orthogonalization of W^*. The following method is proposed to find such a matrix $\bar{W}$:

Under the assumption that W^* is close to $\mathcal{S}$, its SVD can be expressed as:

$$W^* = \tilde{P}' \begin{pmatrix} \tilde{I}_s \\ 0_{(n-s)\times n} \end{pmatrix} Q',$$

[1] As $\|W^\top W - I_n\|_1 = \mathcal{O}(n-s)$

with $\tilde{I}_s$ approximately equal to I_s. By definition of the SVD, Q' is an orthogonal matrix.

To select the columns of X, the process proceeds as if W^* were in $\mathcal{S}$. However, instead of selecting the indices of the non-zero entries of P'_s, the indices corresponding to the s rows of $\tilde{P}'_s$ with the largest norms are chosen. The task of selecting the s columns of X is completed. The proposed method, named L_1 orthogonal regularization, is summarized in Algorithm 3 and the key points of the methods are discussed in the following section.

The matrix $\bar{W}$ can be constructed. This involves creating a matrix $\bar{P}$ as a permutation matrix, with the first s columns containing a 1 at the indices corresponding to the selected columns of X. The remaining $n - s$ columns are filled to complete the permutation matrix.

To illustrate this process, consider the example where $X \in \mathbb{R}^{3\times 4}$ and $s = 2$. A possible form of $\tilde{P}'$ could be:

$$\tilde{P}' = \begin{pmatrix} 0.01 & 0.03 & 0.003 & 0.02 \\ 0.8 & 0.4 & 0.5 & 0.3 \\ 0.04 & 0.01 & 0.03 & 0.02 \\ 0.2 & 0.9 & 0.6 & 0.9 \end{pmatrix}$$

Applying the procedure yields:

$$\bar{P}_s = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \qquad \text{and} \qquad \bar{P} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Finally, $\bar{W}$ is formulated as:

$$\bar{W} = \bar{P} \begin{pmatrix} I_s \\ 0_{(n-s)\times n} \end{pmatrix} Q'.$$

4.2 Gradient descent

4.2.1 Non convexity of the regularized objective function

The regularized objective function is not convex, which means that the method may not always reach the global minimum. However, this is not a significant issue

Algorithm 3 *L_1 Orthogonal Regularization*

Input: $m \times n$ matrix A, integer k, regularization parameter λ, step size α, tolerance ϵ.

Output: $m \times s$ matrix C with s columns of X.

1: **Initial setup:**

- Define W_0 as an $n \times s$ matrix with entries uniformly chosen between 0 and 1.

2: **Continuous Optimization Phase:**

- Gradient descent (GD) on the regularized objective function $F_{X,\lambda}(W) = \left\|X - XW(XW)^{\dagger}X\right\|_F^2 + \lambda \left\|WW^{\top} - I_n\right\|_1$. The iteration is

$$W_{k+1} = W_k - \alpha \nabla_W F_{X,\lambda}(W_k)$$

- Termination condition: stop when the objective function does not improve by more than the tolerance δ during the 10 last iteration.

3: **Retrieving C from W^* Phase:**

- $\tilde{P}' \tilde{I}' Q' = \mathrm{SVD}(W^*)$
- $P'_s = s$ first columns of $\tilde{P}'$
- $\mathbf{s}^* =$ indices of s largest norm of the lines of $\tilde{P}'_s$

return $X_{\mathbf{s}^*}$

in practice. In data science, particularly when dealing with matrices with a large number of columns, the Column Subset Selection Problem (CSSP) typically has many local minima with similar objective values. As a result, although the method may converge to a local minimum, the solution is often close in quality to the global minimum, making it effective for practical applications [3].

4.2.2 Non differentiability of the regularization term

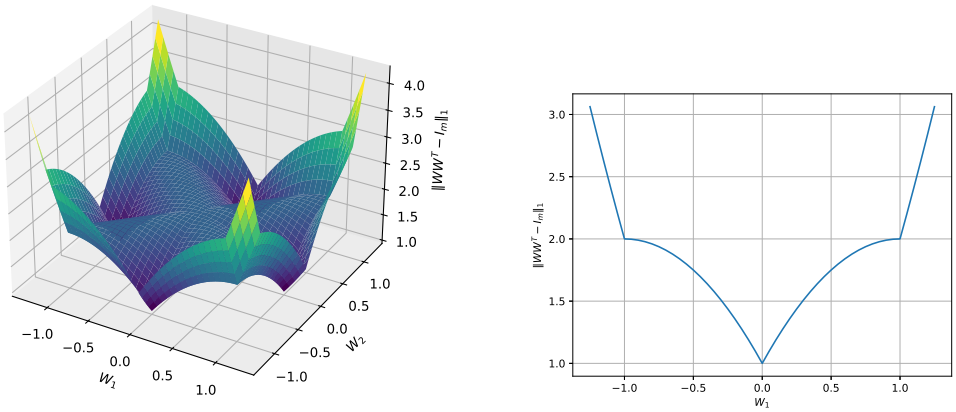
To find the optimal solution W^* for the problem defined in equation 4.6, gradient descent is employed due to its ability to handle large-scale problems. However, the function is not differentiable because of the regularization term $\left\|WW^{\top} - I_n\right\|_1$. This non-differentiability arises from the absolute value operator in the entrywise

norm 2.1. Specifically, non-differentiability occurs at points where an entry of $WW^\top - I_n$ equals zero. This typically happens when W is near its optimal values, such that $WW^\top$ is close to a diagonal matrix with s diagonal elements equal to 1.

Given that $W \in \mathbb{R}^{n \times s}$ with $\text{rank}(W) = s$ and $\text{rank}(W^\top) = s$, it follows that $\text{rank}(WW^\top) = s$. This conclusion arises by examining the null space of $W^\top$. Specifically, if x lies in the null space of $W^\top$, then:

$$WW^\top x = 0.$$

This implies that the null space of $WW^\top$ is identical to the null space of $W^\top$, confirming that $\text{rank}(WW^\top)$ must be s. Consequently, the matrices W that optimize the L_1 orthogonal regularization term are those for which $WW^\top$ is a diagonal matrix with all s diagonal entries equal to one, indicating orthonormal columns in W. Then the optimal value of the regularization term $\|WW^\top - I_n\|_1$ is $n - s$. This value represents the lower bound, as shown in Figure 4.3, and is used as a normalization factor in the results section.



(a) 3D visualization (b) $W_2 = 1$

Figure 4.1: $\left\|WW^\top - I_n\right\|_1$ with $W \in \mathbb{R}^{2\times 1}$

As observed in Figures 4.2 and 4.3, when using a sufficiently small step size, the algorithm tends to oscillate around a local minimum. The smaller the step size, the closer the algorithm gets to the optimal iterate. Since the gradient of $\left\|WW^\top - I_n\right\|_1$ does not converge to zero, a new termination condition, introduced in Section 4.2.4, is necessary. Using this revised termination condition, we plot the gradient descent of $\left\|WW^\top - I_n\right\|_1$ with $W \in \mathbb{R}^{10\times 5}$ in Figure 4.3.

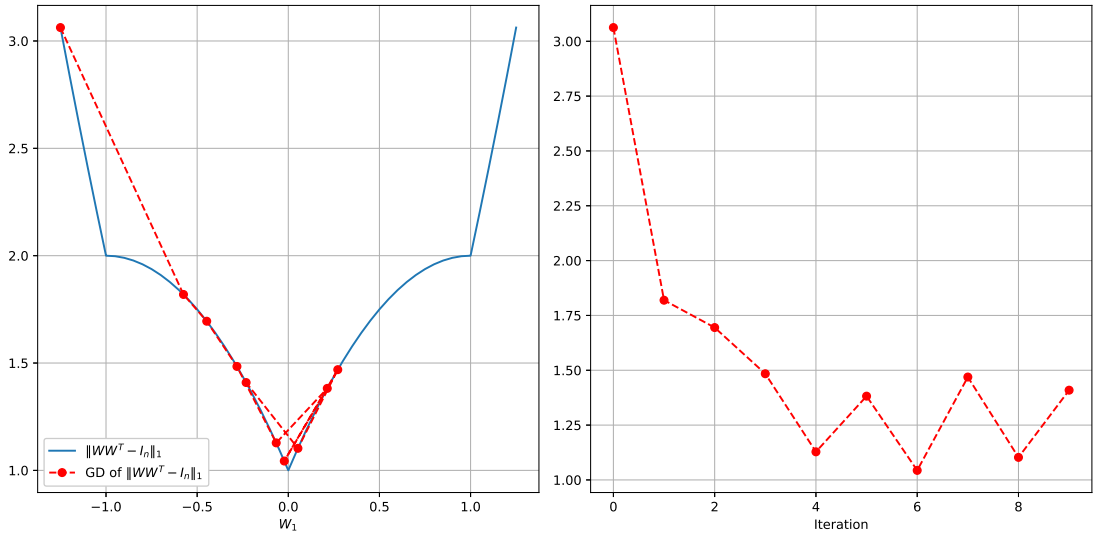


Figure 4.2: GD on $\left\|WW^\top - I_n\right\|_1$ with $\alpha = 0.15$, $W \in \mathbb{R}^{2\times 1}$

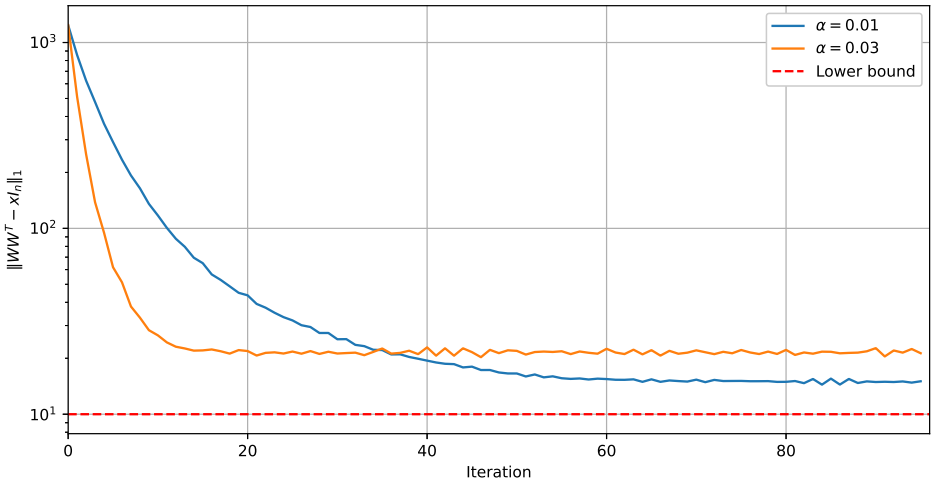


Figure 4.3: GD on $\left\|WW^\top - I_n\right\|_1$ with $\alpha = 0.01$, $W \in \mathbb{R}^{10\times 5}$

4.2.3 Analytical form of the gradient

Prior to deriving the gradient, the penalized objective function is expanded.

$$\begin{aligned} F_\lambda^X(W) =& \left\|X - XW(XW)^\dagger X\right\|_F^2 + \lambda \left\|WW^\top - I_n\right\|_1 \\ =& \sqrt{\operatorname{tr}\left(\left(X - XW\left(XW\right)^\dagger X\right)^\top \left(X - XW\left(XW\right)^\dagger X\right)\right)}^2 + \lambda \left\|WW^\top - I_n\right\|_1 \end{aligned} \tag{4.8}$$

By definition of the pseudo inverse, $XW\left(XW\right)^\dagger$ is symmetric. Then $\left(XW\left(XW\right)^\dagger\right)^\top = XW\left(XW\right)^\dagger$ and $XW\left(XW\right)^\dagger XW = XW$.

$$\begin{aligned} F_\lambda^X(W) =& \operatorname{tr}\left(\left(X^\top - X^\top XW\left(XW\right)^\dagger\right)\left(X - XW\left(XW\right)^\dagger X\right)\right) + \lambda \left\|WW^\top - I_n\right\|_1 \\ =& \operatorname{tr}\Big(X^\top X - X^\top XW(XW)^\dagger X - X^\top XW(XW)^\dagger X \\ &+ X^\top \underbrace{XW\left(XW\right)^\dagger XW}_{=XW}\left(XW\right)^\dagger X\Big) + \lambda \left\|WW^\top - I_n\right\|_1 \\ =& \operatorname{tr}\left(X^\top X\right) - \operatorname{tr}\left(X^\top XW(XW)^\dagger X\right) - \operatorname{tr}\left(X^\top XW(XW)^\dagger X\right) \\ &+ \operatorname{tr}\left(X^\top XW\left(XW\right)^\dagger X\right) + \lambda \left\|WW^\top - I_n\right\|_1 \\ =& -\operatorname{tr}\left(X^\top XW(XW)^\dagger X\right) + \lambda \left\|WW^\top - I_n\right\|_1 \end{aligned}$$

For clarity, the objective function and the regularization term are detailed separately.

Gradient of the objective function

$$\begin{aligned} &\left(\nabla_W - \operatorname{tr}\left(X^\top XW(XW)^\dagger X\right)\right)_{i,j} \\ &= \frac{\partial}{\partial W_{i,j}}\left(-\operatorname{tr}\left(X^\top XW(XW)^\dagger X\right)\right) \\ &= -\operatorname{tr}\left(X^\top \frac{\partial}{\partial W_{i,j}}\left((XW)(XW)^\dagger\right)X\right) \quad \text{By linearity of the trace} \end{aligned} \tag{4.9}$$

d_{ij} is defined as the differential operator $\frac{\partial}{\partial W_{i,j}}$ with $i = 1, \ldots, n \quad j = 1, \ldots, s$ and $A = XW$. For clarity of the reader $d_{ij}AA = d_{ij}(A)A, \quad d_{ij}A^\dagger = d_{ij}\left(A^\dagger\right)$ and

$d_{ij}A^\top = d_{ij}\left(A^\top\right)$. The gradient is expressed as:

$$-\operatorname{tr}\left(X^\top d_{ij}\left(AA^\dagger\right)X\right) \tag{4.10}$$

Expanding $d_{ij}(AA^\dagger)$:

$$\begin{aligned} d_{ij}(AA^\dagger) &= d_{ij}(AA^\dagger AA^\dagger) && \text{since } A = AA^\dagger A \\ &= \underbrace{d_{ij}(AA^\dagger)AA^\dagger}_{(1)} + \underbrace{AA^\dagger d_{ij}(AA^\dagger)}_{(2)} \end{aligned}$$

Finding an expression for (1) by expending $d_{ij}(AA^\dagger A)$:

$$\begin{aligned} & d_{ij}(AA^\dagger A) = d_{ij}(AA^\dagger)A + AA^\dagger d_{ij}A \\ \iff \quad & d_{ij}A = d_{ij}(AA^\dagger)A + AA^\dagger d_{ij}A \\ \iff \quad & d_{ij}(AA^\dagger)A = (I - AA^\dagger)d_{ij}A \\ \iff \quad & d_{ij}(AA^\dagger)AA^\dagger = (I - AA^\dagger)d_{ij}AA^\dagger \end{aligned}$$

Finding an expression for (2) by expending $\left(d_{ij}(AA^\dagger)AA^\dagger\right)^\top$:

$$\begin{aligned} \left(d_{ij}(AA^\dagger)AA^\dagger\right)^\top &= \left(AA^\dagger\right)^\top \left(d_{ij}(AA^\dagger)\right)^\top \\ &= AA^\dagger d_{ij}\left(\left(AA^\dagger\right)^\top\right) \\ \iff \quad AA^\dagger d_{ij}\left(AA^\dagger\right) &= \left(d\left(AA^\dagger\right)AA^\dagger\right)^\top && \text{which is the transpose of (1)} \\ \iff \quad AA^\dagger d_{ij}\left(AA^\dagger\right) &= \left((I - AA^\dagger)d_{ij}AA^\dagger\right)^\top \end{aligned}$$

Injecting (2) and (3) in (1):

$$d_{ij}(AA^\dagger) = \left(I - AA^\dagger\right)d_{ij}AA^\dagger + \left(\left(I - AA^\dagger\right)d_{ij}AA^\dagger\right)^\top$$

Injecting $d_{ij}(AA^\dagger)$ in the gradient 4.10:

$$-\operatorname{tr}\left[X^\top\left(\left(I - AA^\dagger\right)d_{ij}AA^\dagger + \left(\left(I - AA^\dagger\right)d_{ij}AA^\dagger\right)^\top\right)X\right] \tag{4.11}$$

$$
\begin{aligned}
&- \operatorname{tr}\left[\left(\left(I - AA^\dagger\right) d_{ij}AA^\dagger + \left(\left(I - AA^\dagger\right) d_{ij}AA^\dagger\right)^\top\right) XX^\top\right] && (4.12)\\
&= - \operatorname{tr}\left[\left(I - AA^\dagger\right) d_{ij}AA^\dagger XX^\top\right] - \operatorname{tr}\left[\left(\left(I - AA^\dagger\right) d_{ij}AA^\dagger\right)^\top XX^\top\right] \\
&= - 2\operatorname{tr}\left[\left(I - AA^\dagger\right) d_{ij}AA^\dagger XX^\top\right] && (4.13)\\
&= - 2\operatorname{tr}\left[d_{ij}A \underbrace{A^\dagger XX^\top \left(I - AA^\dagger\right)}_{:=K}\right] && (4.14)\\
&= - 2\operatorname{tr}\left[d_{ij}AK\right] && (4.15)
\end{aligned}
$$

The equations 4.12 and 4.14 are obtained using the cyclicity property of the trace operator, i.e. $\operatorname{tr}(ABCD) = \operatorname{tr}(BCDA) = \operatorname{tr}(CDAB) = \operatorname{tr}(DABC)$. The equation 4.13 comes from the fact that $\operatorname{tr}(X^\top) = \operatorname{tr}(X)$. The implementation of the current form of the gradient requires two nested loops, as the trace of a new expression must be computed for each term in the gradient.

Let's now develop $d_{ij}A$ in order to simplify the trace operator. Recall the matrix $\frac{\partial W}{\partial W_{i,j}}$ is null everywhere except its component i, j that is equal to 1. We define $E_{i,j} := \frac{\partial W}{\partial W_{i,j}}$.

$$
\begin{aligned}
d_{ij}A &= \frac{\partial A(W)}{\partial W_{i,j}} \\
&= X\frac{\partial W}{\partial W_{i,j}} \\
&= XE_{i,j} \\
&= \begin{pmatrix} 0_{n\times j-1} & X_{:,i} & 0_{n\times s-j}\end{pmatrix}
\end{aligned}
$$

With this explicitation of $dA_{i,j}$ the gradient can be developed further to allow for vectorized computation. It can be observed that $K \in \mathbb{R}^{s\times m}$ and $d_{ij}A \in \mathbb{R}^{m\times s}$. This yields:

$$
\begin{aligned}
\operatorname{tr}\left[d_{ij}AK\right] &= \operatorname{tr}\left[\begin{pmatrix} 0_{n\times j-1} & X_i & 0_{n\times s-j}\end{pmatrix} E\right] \\
&= \operatorname{tr}\left[X_i (K^\top)_j\right] \\
&= \sum_{k=1}^{m} X_{k,i}K_{j,k} \\
\implies \qquad \operatorname{tr}\left(d_{ij}AK\right) &= \left(X^\top K^\top\right)_{i,j} = \left((EX)^\top\right)_{i,j} \qquad i = 1,\dots,,n \quad j = 1,\dots,s
\end{aligned} \tag{4.16}
$$

The gradient of the objective function can be expressed as follows:

$$
\begin{aligned}
&= -2(KX)^\top \\
&= -2\left(A^\dagger X X^\top \left(I - AA^\dagger\right) X\right)^\top \\
&= -2\left(X^\top X X^\top (XW)^{\dagger^\top}\right) + 2\left(X^\top (XW)(XW)^\dagger X X^\top (XW)^{\dagger^\top}\right)
\end{aligned} \tag{4.17}
$$

Gradient of the regularization term

$$
\begin{aligned}
&\left(\nabla_W \lambda \left\|WW^\top - I_n\right\|_1\right)_{i,j} = \\
&= \lambda \frac{\partial}{\partial W_{i,j}} \sum_{k=1}^{n}\sum_{l=1}^{n} \left|\left(WW^\top - I_n\right)_{k,l}\right| \\
&= \lambda \sum_{k=1}^{n}\sum_{l=1}^{n} \operatorname{sign}\left(WW^\top - I_n\right)_{k,l} \left(E_{i,j}W + WE_{i,j}\right) \\
&= \lambda \sum_{k=1}^{n}\sum_{l=1}^{n} \operatorname{sign}\left(WW^\top - I_n\right)_{k,l} \left(\begin{pmatrix} 0_{i-1\times n} \\ W_j^\top \\ 0_{n-i\times n} \end{pmatrix} + \begin{pmatrix} 0_{n\times j-1} & W_i & 0_{n\times n-j} \end{pmatrix}\right) \\
&= \lambda \left(\operatorname{sign}\left(WW^\top - I_n\right)_{:,i} W_j + \operatorname{sign}\left(WW^\top - I_n\right)_j W_i\right)
\end{aligned}
$$

In matrix form, this yields:

$$
\nabla_W \left(\lambda \left\|WW^\top - I_n\right\|_1\right) = \lambda \left(\operatorname{sign}\left(WW^\top - I_n\right) W + \operatorname{sign}\left(WW^\top - I_n\right)^\top W\right) \tag{4.18}
$$

Then the final form of the gradient of the objective function 4.8 is:

$$
\begin{aligned}
\nabla F_\lambda^X(W) = &-2\left(X^\top X X^\top (XW)^{\dagger^\top}\right) + 2\left(X^\top (XW)(XW)^\dagger X X^\top (XW)^{\dagger^\top}\right) \\
&+ \lambda \left(\operatorname{sign}\left(WW^\top - I_n\right) W + \operatorname{sign}\left(WW^\top - I_n\right)^\top W\right)
\end{aligned} \tag{4.19}
$$

4.2.4 Termination condition

Due to the non-differentiability of the function, the gradient may not converge to zero, and the value of the objective function may not decrease consistently at each iteration. As illustrated in Figures 4.2 and 4.3, the iterates W_i tend to oscillate

around a local minimum. Therefore, a termination condition that does not rely on the value of the gradient is necessary. The stopping criterion is defined as follows: if over the last 10 iterations, the best result has not improved by more than a specified delta, the algorithm terminates and returns the best iteration. This stopping criterion is detailed in the pseudo-code below:

$F_{\text{best}} = \infty$
$n_{\text{best}} = 0$
$n = 0$
$W_i =$ starting point
while True **do**
    $W_i = W_i - \alpha \nabla_W F_\lambda^X(W_i)$
    $n = n + 1$
    if $F_\lambda^X(W_i) + \epsilon < F_{\text{best}}$ **then**
        $F_{\text{best}} = F(X, W_i, \lambda)$
        $W_{\text{best}} = W_i$
        $n_{\text{best}} = n$
    end if
    if $(n - n_{\text{best}} > 10)$ **then**
        $W_i = W_{\text{best}}$
        break
    end if
end while

4.3 Role and determination of the hyperparameters λ and α

The hyperparameters λ (penalization term) and α (step size) in gradient descent iterations significantly influence how close W^* is to the set $\mathcal{S}$.[2] The solution obtained for W^* through gradient descent may not exactly belong to the set $\mathcal{S}$ defined in (2.2). Although numerical approximations may prevent W^* from lying precisely within $\mathcal{S}$, the impact is minimal due to the following reasons:

1. The optimum represents a balance between the CSSP objective function and the penalization term.

[2]This "closeness" can be interpreted as the distance (e.g., Euclidean) between W^* and $\bar{W}$, the projection of W^* onto $\mathcal{S}$. However, finding such a matrix $\bar{W}$ is generally not feasible.

2. The non-differentiability of the penalized objective function causes oscillations around the minimum, which prevents gradient descent from reaching the exact minimum.

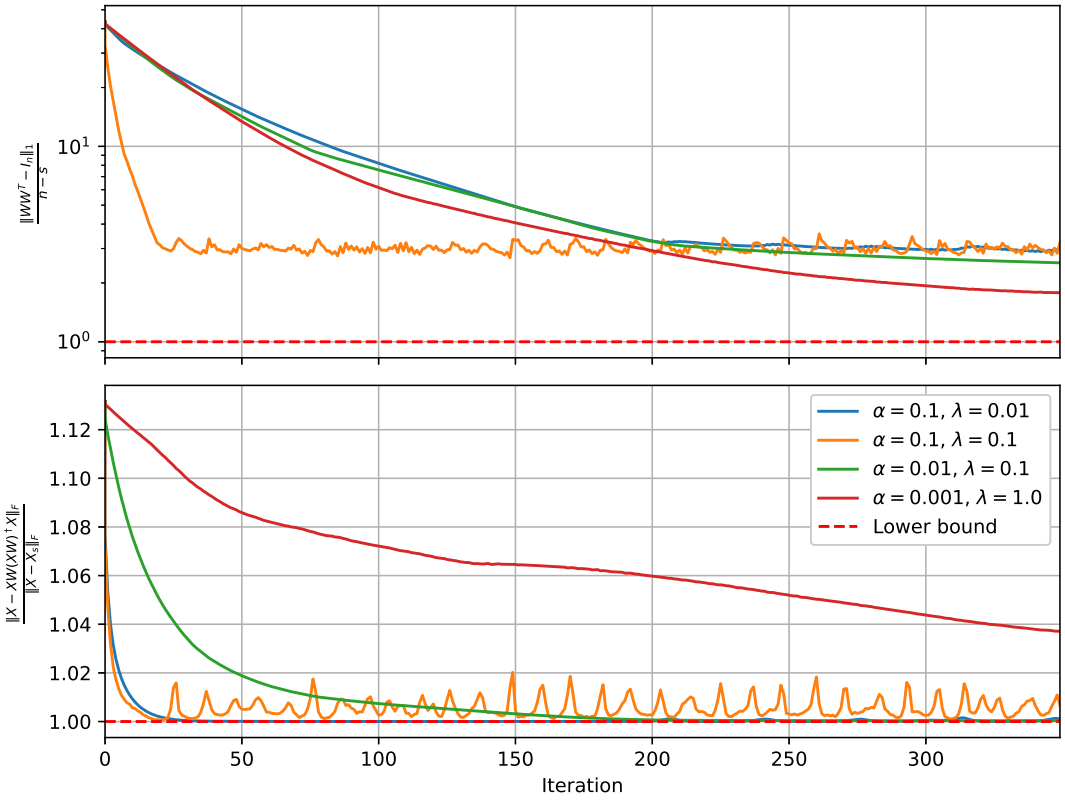


Figure 4.4: Gradient Descent on $\left\|X - XW(XW)^\dagger X\right\|_F^2 + \lambda \left\|WW^\top - I_n\right\|_1$ with 350 fixed iterations, $X \in \mathcal{U}[0,1]^{10\times 20}$, $s = 5$.

As λ increases, the influence of the penalization gradient becomes more significant compared to the gradient of the primary objective function. This increase in λ also intensifies the oscillations in the penalized objective function. To ensure that W remains close to the set $\mathcal{S}$, the step size α must be reduced as λ increases, which helps to minimize and control these oscillations.

This behavior is illustrated in Figure 4.4. In this figure, the y-axis of the upper graph represents the penalized term normalized by $n - s$, which, as explained in Section 4.1.3, is its minimum value. The y-axis of the lower graph shows the objective function normalized by the best low-rank approximation described in section 3.2.

The impact of Step 3 in the algorithm (1.3) on the precision of the proposed method remains somewhat uncertain. This step involves approximating the numerically optimal W^* within the set $\mathcal{S}$. The hyperparameters play a crucial role in determining how close W^* is to $\mathcal{S}$, but it is challenging to assess how close W^* needs to be for the approximation $\bar{W}$ to yield a practically acceptable objective value, specifically $\|X - X\bar{W}(X\bar{W})^\dagger X\|_F$.

4.3.1 Theoretical complexity analysis

The overall theoretical time complexity of the proposed algorithm is primarily driven by the gradient descent step. The complexity of the gradient descent depends on the number of iterations multiplied by the time complexity of computing the gradient. The gradient, as expressed in Equation (4.19), consists of three terms.

First term

$$-2\left(X^\top X X^\top (XW)^\dagger\right)$$

In this term, computing $X^\top X X^\top$ requires $2m^2n$ operations, but since it is constant across iterations, its cost can be neglected. The computation of the pseudoinverse $(XW)^\dagger$ has a complexity of $\mathcal{O}(mns + \min(m^2s, ms^2))$. The multiplication $(X^\top X X^\top)$ and $(XW)^\dagger$ require mns operations. Therefore, the time complexity of this term is $\mathcal{O}(2mns + ms^2)$.

Second term

$$2\left(X^\top (XW)(XW)^\dagger X X^\top (XW)^{\dagger\top}\right)$$

For this term, the computation of $XX^\top$ requires m^2n operations, but like the first term, it remains constant across iterations. The matrices XW and $(XW)^\dagger$ are already computed in the first term. By organizing the matrix multiplications as follows:

$$2\underbrace{\underbrace{X^\top (XW)}_{(a)}(XW)^\dagger}_{(b)}\underbrace{(XX^\top)(XW)^{\dagger\top}}_{(c)}$$

the computation of terms (a), (b), and (c) each requires mns operations. The final multiplication of terms (b) and (c) also takes mns operations. Given that some terms are reused from the first term, the time complexity of the second term is $\mathcal{O}(4mns)$.

Third term

$$\lambda \left(\text{sign}(WW^\top - I_n)W + \text{sign}(WW^\top - I_n)^\top W \right)$$

In this term, the multiplication $WW^\top - I_n$ requires $n^2 s$ operations. The 'sign' function is applied element-wise with a complexity of $\mathcal{O}(n^2)$. Therefore, the time complexity of the third term is $\mathcal{O}(n^2 s)$.

Overall complexity

Thus, the overall complexity of the method can be summarized as:

$$\mathcal{O}(N(6mns + ms^2 + n^2 s)) = \mathcal{O}(N \max(mns, n^2 s))$$

where N is the number of iterations of the gradient descent.

Chapter 5

Results

This section provides a comprehensive analysis of the precision and speed of the proposed algorithm for the Column Subset Selection Problem (CSSP). The performance of the proposed method is evaluated against two established algorithms: the *Double-Phase* algorithm by Boutsidis et al. [2] and the *Stochastic Continuous Landmark Selection* algorithm by Mathur et al. (2024) [8]. The goal is to highlight the advantages and limitations of the proposed approach.

For clarity, the proposed method (L_1 *Orthogonal Regularization*), the *Double-Phase*, and the *Stochastic Continuous Landmark Selection* algorithms are henceforth respectively referred to as L_1O, DP and SLS.

5.1 Datasets

The numerical experiments were conducted using three distinct datasets, comprising both real and synthetic data:

- **Random**: This dataset consists of a matrix with 20 rows and 50 columns, where each entry is generated from a uniform distribution between 0 and 1.
- **Arrhythmia**: This dataset contains measurements from 484 electrocardiograms (ECG), with each ECG recording 280 measurements [1].
- **MNIST1K** : This dataset is derived from the MNIST database, consisting of $60,000$ images of handwritten digits. Each image is a 28×28 pixel grid,

represented as a vector of 784 pixels [5].

For the numerical experiments, the Arrhythmia and MNIST1K datasets were standardized. A random sample of 100 entries was drawn from each dataset for each experiment. The dimensions of the datasets used in the experiments are as follows: Random (20×50), Arrhythmia (100×280), and MNIST1K (100×784). It is important to note that for each trial of every experiment conducted with different methods, the same samples from the MNIST1K and Arrhythmia datasets were used.

	Random	**Arrhythmia**	**MNIST1K**
m	20	100	100
n	50	280	784

Table 5.1: Summary of m and n for each dataset

5.2 Hyperparameter selection

Regularization term λ and step size α

Sections 3.3 and 4.3 provide detailed explanations of the roles of the hyperparameters λ and α in the *Stochastic Continuous Landmark Selection* method and L_1 *Orthogonal Regularization*. The impact of the two hyperparameters λ and α are not independent. Consequently, determining the optimal pair λ, α requires a grid search approach to obtain the optimal pair.

Figures 5.1 and 5.2 shows the approximation factor for both the SLS and L_1O for different sets of hyperparameters λ and α for each dataset. The approximation factor is defined as the ratio of the objective function's value to the optimal error of the low-rank s approximation X_s.

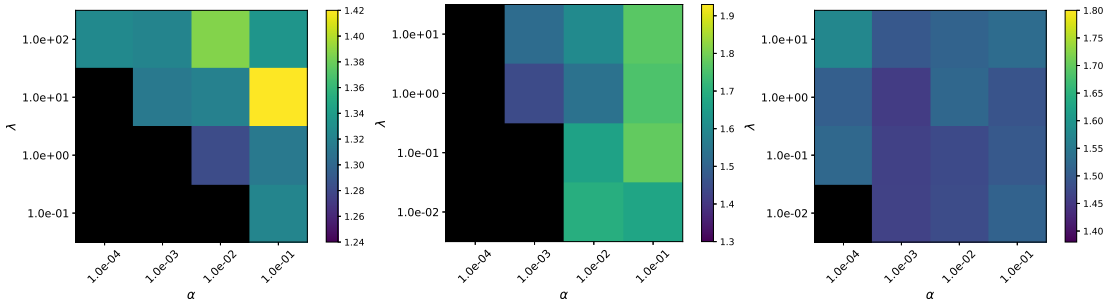
$$\text{Approximation factor} = \frac{\|X - CC^{\dagger}X\|_F}{\|X - X_s\|_F}$$

The stopping criterion for the experiments has a tolerance of $\epsilon = 10^{-3}$. Each figure presents the mean results from 10 trials of the experiment. For specific hyperparameter pairs, if the method fails to find an optimal value within 2000 iterations for the Random dataset, or within 400 iterations for the other two

datasets in at least one of the trials, the corresponding grid cell is marked with a black square. For a pair of hyperparameters (λ, α) associated with a black square, it is possible that the last iterate performed better than other pairs (λ, α), but the method had not yet fully converged.

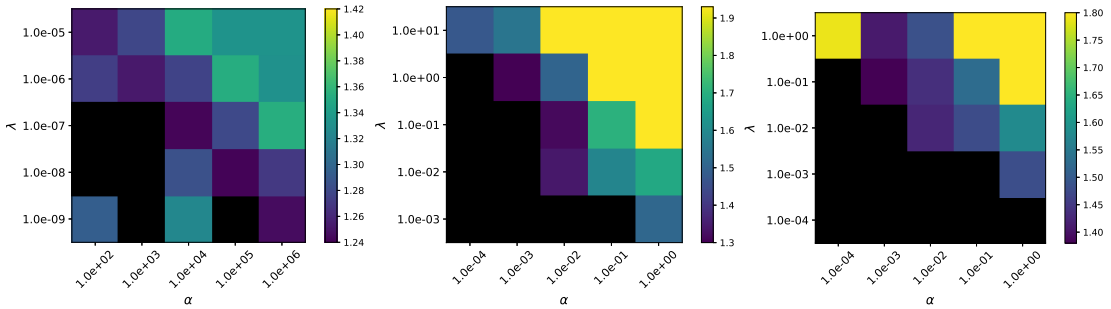
The hyperparameters are determined for a fixed number of columns to be selected, this approach is not problematic. Indeed it can be observed in Figures 5.5, 5.6, 5.5 that when the selection factor $\mathbf{s}$ changes, the SLS and L_1O's approximation factor trends show consistent and similar behavior to that of the *Uniform Sampling* and *Double-Phase* algorithms.

The pairs of optimal hyperparameters for each configuration is resumed in Table 5.2.



(a) Random (b) Arrithmia (c) MNIST1K

Figure 5.1: Grid Search of the approximation factor for SLS's hyperparameters (λ and α for the three datasets.



(a) Random (b) Arrithmia (c) MNIST1K

Figure 5.2: Grid Search of the approximation factor for L_1O's hyperparameters (λ and α for the three datasets.)

For SLS, the value of the optimal hyperparameters are similar for the three datasets. For L_1O, the optimal hyperparameter values are similar only for both the MNIST1K

Method	Random ($s = 50\%m$)	Arrhythmia ($s = 30\%m$)	MNIST1K ($s = 30\%m$)
SLS	$\lambda^* = 1$, $\alpha^* = 10^{-2}$	$\lambda^* = 1$, $\alpha^* = 10^{-3}$	$\lambda^* = 1$, $\alpha^* = 10^{-3}$
L_1O	$\lambda^* = 10^{-8}$, $\alpha^* = 10^{5}$	$\lambda^* = 1$, $\alpha^* = 10^{-3}$	$\lambda^* = 10^{-1}$, $\alpha^* = 10^{-3}$

Table 5.2: Optimal Hyperparameters λ^* and α^* for SLS and L_1O across the different datasets.

and Arrhythmia datasets.

In contrast, for the Random dataset, it appears that $\lambda^* \ll \alpha^*$. The unusually large α^* may be explained by the possibility that the gradient of the objective function has very small values, requiring a much larger scaling factor compared to the regularization term. Interestingly, the product $\alpha^* \cdot \lambda^*$ equals 10^{-3}, which is consistent with the values observed for the other datasets. This consistency is logical since the regularization term does not directly depend on the dataset.

Figure 5.3 supports the hypothesis. The upper part of the figure illustrates the evolution of the objective function and the regularization term (multiplied by λ) for the Random and MNIST1K datasets. The objective function is normalized by $\|X - X_s\|_F$, the Frobenius norm of the residual from the low-rank s approximation of X (as detailed in Section 3.2), allowing for comparison across the two datasets. Meanwhile, the regularization term is normalized by $n - s$, its lower bound.

The lower part of the figure displays the average absolute value of the gradient entries, multiplied by the relevant hyperparameters in the gradient descent process, specifically α for the gradient of the objective function and $\alpha \cdot \lambda$ for the regularization term gradient. This was calculated by multiplying these hyperparameters by the entrywise norm of the gradient and dividing by its size ($n \times s$). The have the following expressions:

$$\alpha \cdot \text{mean of } \nabla_W F^X = \frac{\alpha \sum_{i=1}^{n} \sum_{j=1}^{s} \left| \left(\nabla_W \|X - XW(XW)^\dagger X\|_F \right)_{i,j} \right|}{n \cdot s}$$

$$\alpha \cdot \lambda \cdot \text{mean of } \nabla_W R_\lambda = \frac{\alpha \cdot \lambda \sum_{i=1}^{n} \sum_{j=1}^{s} \left| \left(\nabla_W \|WW^\top - I_n\|_F \right)_{i,j} \right|}{n \cdot s}$$

This upper part Figure 5.3 demonstrates also that the L_1O algorithm effectively minimizes the objective function. The proposed method reduces the regularization term's value until it reaches the tradeoff determined by the optimal hyperparameters. This trade-off is illustrated in the lower part of Figure 5.3. Analyzing the figure, it can be observed that when the stopping criterion is fulfilled (at the last iteration)

the ratio between these two terms is similar, $\sim (10)$, across both datasets.

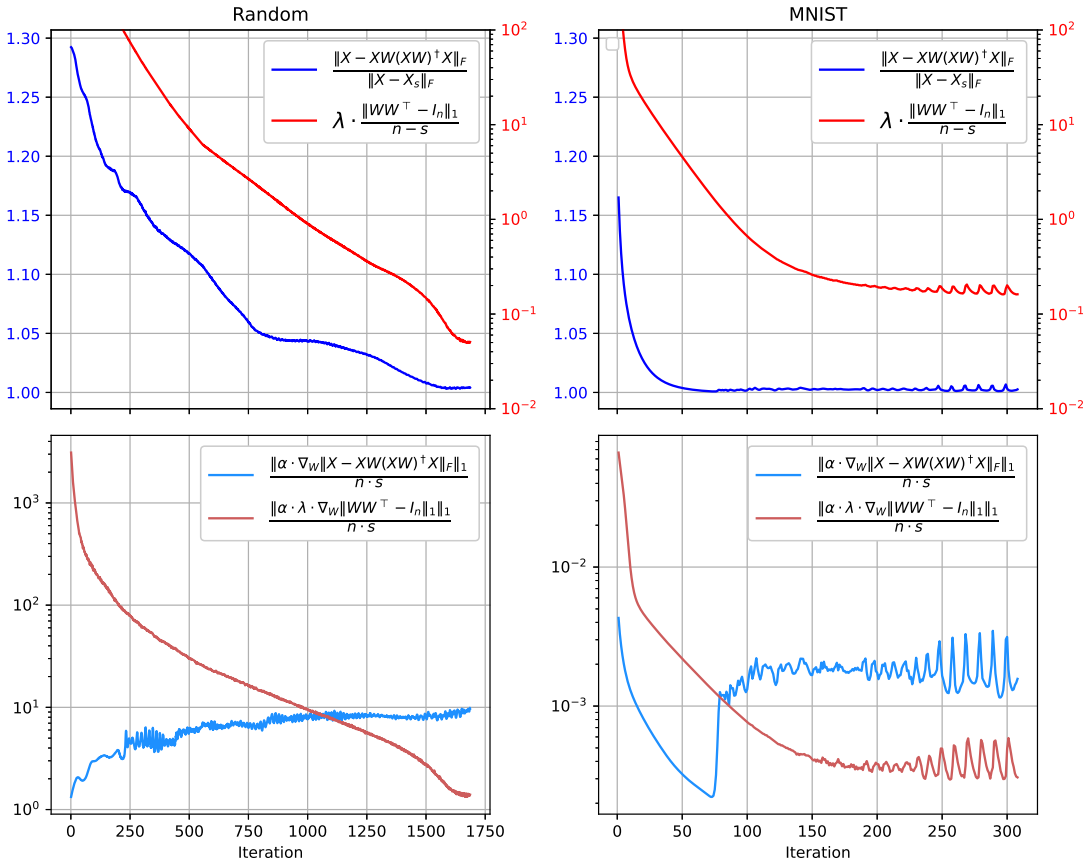


Figure 5.3: Behavior of the Objective Function, Regularization Term, and Gradients with Optimal Hyperparameters in the L_1OR Algorithm.

Stochastic Continuous Landmark Selection's Monte Carlo sampling size M

Figure 5.4 shows the influence of the Monte Carlo sample size M on the approximation factor for the SLS method. As M increases the precision, mesured by the approximation factor, of the method improves. However, the improvement becomes small after $M = 5$. Since the size M is directly proportional to the time complexity of the method, $M = 5$ is a good choice. Indeed, it provides a good trade-off between precision and time complexity.

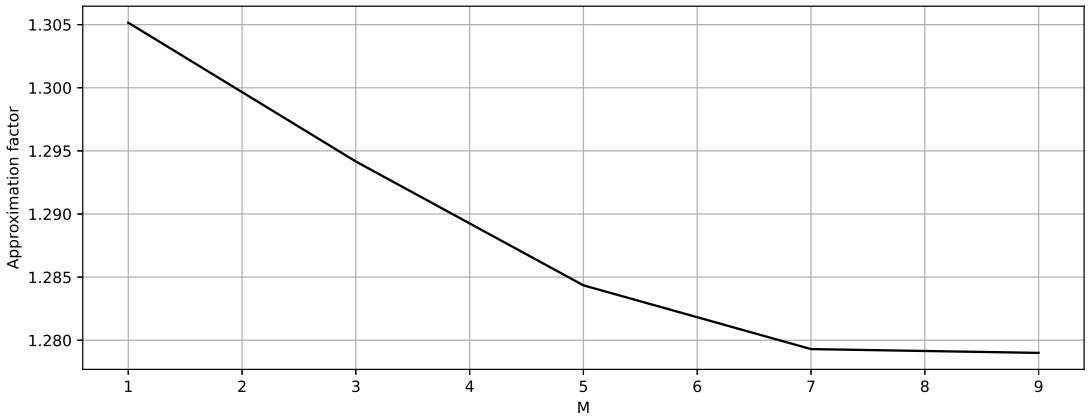


Figure 5.4: influence of M

5.3 Performance analysis

Figures 5.5, 5.6 and 5.7 display the mean approximation factor over 20 trials for the proposed method and existing methods (Section 3) across the Random, Arrhythmia, and MNIST1K datasets. Due to the random nature of Uniform Sampling, its results are averaged over 200 trials to ensure consistency.

The primary objective in this section is to compare the performance of the proposed method relative to the *Stochastic Continuous Landmark Selection* (SLS) method, which also relies on continuous optimization using a regularization. Since the SLS method is recent (published in 2024), the Double-Phase Algorithm serves to validate the consistency of the results, as it is an older method based on a well-established algorithm [9]. Uniform Sampling is used as a baseline method.

Achieving an approximation factor of 1 is theoretically impossible unless the matrix composed of the first s left singular vectors of the dataset matrix lies within the set $\mathcal{S}$ (as defined in 2.2). Determining the infimum of the approximation factor requires solving the Column Subset Selection Problem (CSSP), which is NP-complete. An exact solution would necessitate a brute-force approach, which is computationally infeasible for datasets of this size. Consequently, it remains impossible to determine how close each method is to the optimal subset column selection.

It is important to note that as s approaches m, the value of the objective function for the low rank-s approximation nears zero, naturally leading to an increase in the approximation factor. Conversely, when s is close to zero, the error in the low rank-s approximation becomes larger, making the approximation factor smaller.

The results indicate that for meaningful values of s, all three methods significantly outperform a random selection of s columns. The meaningful values of s are those far from 0, where brute-force combinatorial testing ($\binom{n}{s}$) is impractical. The proposed method consistently outperforms the SLS method across all three datasets and nearly the entire range of $s \in [0, m]$. However, it is also observed that, except for the Random dataset, the Double-Phase Algorithm achieves higher accuracy compared to the proposed method.

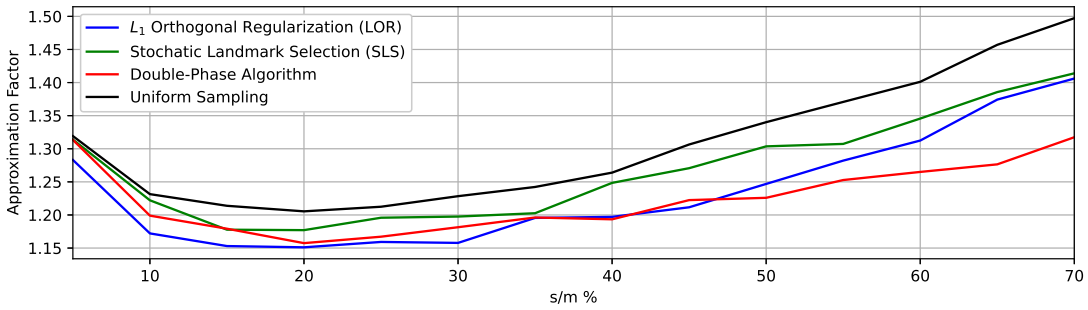


Figure 5.5: mean approximation factor over 20 trials (Random)

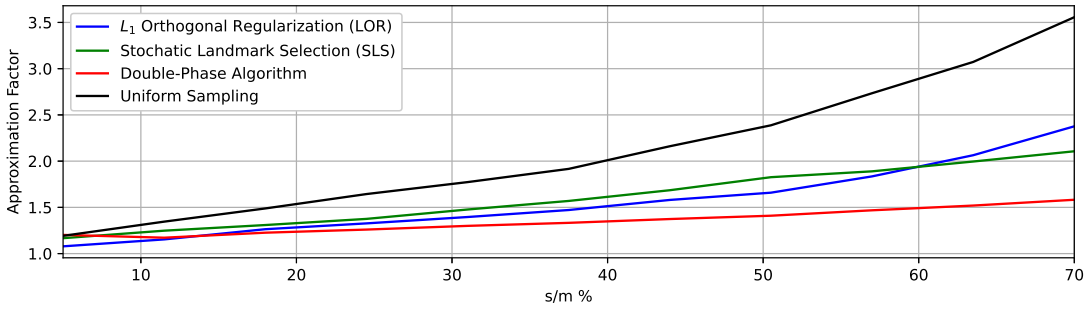


Figure 5.6: mean approximation factor over 20 trials (Arrithmia)

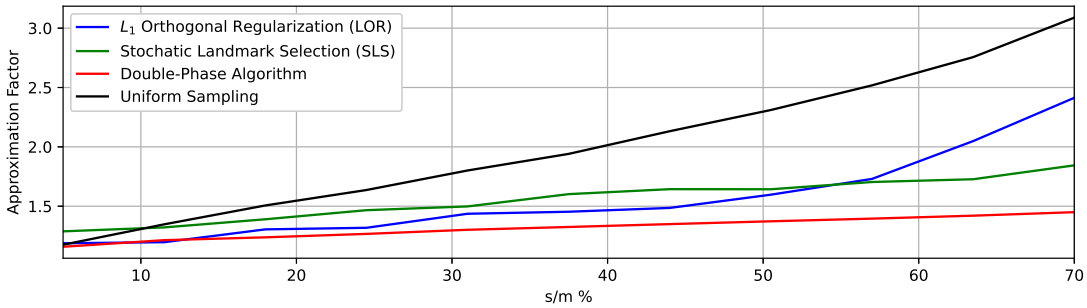


Figure 5.7: mean approximation factor over 20 trials (MNIST1K)

5.4 Computational efficiency

5.4.1 Computational time analysis

This subsection focuses on comparing the empirical computational complexity of the proposed method with the SLS method.

To perform a fair and reliable comparison, both methods were implemented in Python. The code is available on a public GitHub repository [11]. Computationally intensive steps, such as matrix multiplication, pseudo-inverse, and inverse calculations, were executed using ‘numpy.linalg‘ functions ‘dot‘, ‘pinv‘, and ‘inv‘. To ensure a fair comparison, the number of operations required was minimized.

The deterministic sub-algorithm of the Double-Phase method involves an unconventional column pivoting and the triangularization of a Hessenberg matrix using a specified strategy, for which no efficient built-in method exists in ‘numpy‘.

When considering computational cost, the comparison of the proposed method with SLS is the most appropriate, as both are iterative methods. Their similar implementation in Python ensures the consistency of the results.

Time comparison with evolution of the problem size

In this analysis, the problem size is kept constant with respect to n. The matrix X is defined as $X \in \mathbb{R}^{m \times n}$, with $s = \frac{m}{2} = \frac{n}{4}$, and its entries are uniformly distributed between 0 and 1. The computational complexities per iteration for the L_1O and the SLS methods are $\mathcal{O}(mns)$ and $\mathcal{O}(n^3)$, respectively. Given this, the computational

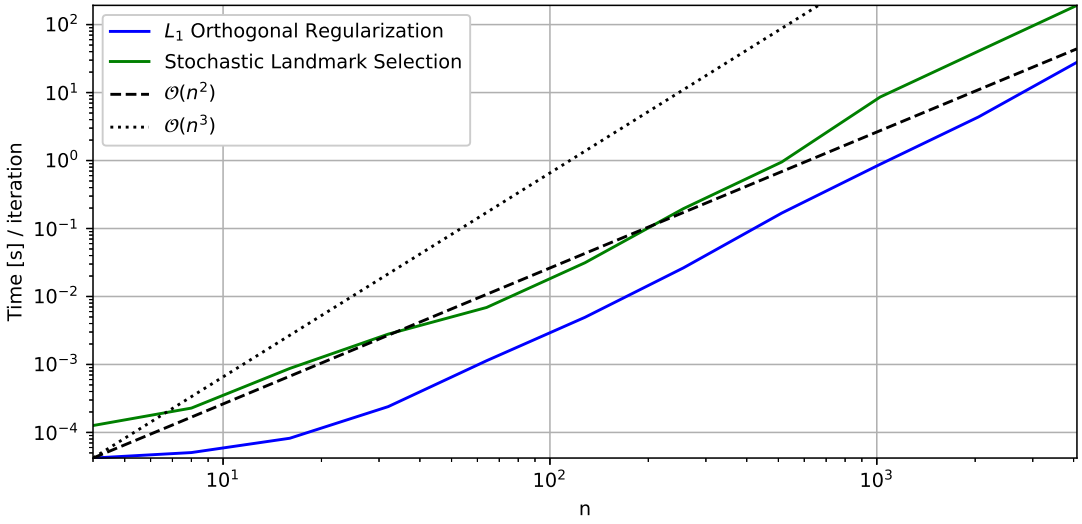


Figure 5.8: Empiric computational time

cost of both methods is expected to follow a cubic growth pattern. However, as seen in Figure 5.8, the actual time complexity growth for both methods appears closer to quadratic. This deviation from cubic growth may be due to large hidden constants that reduce the observed complexity.

The L_1O method outperforms the SLS method in terms of computational time by roughly a factor of 10. Given that the SLS used a Monte Carlo sample size of $M = 5$, computing its computational time for one sample per iteration, the difference with the time complexity per iteration of the L_1O method is modest.

Despite this, the theoretical time complexity per iteration $mns < n^3$ suggests that there are scenarios where the proposed L_1O method could be significantly faster, especially in applications involving large datasets with a relatively small number of selected columns s. The difference in computational cost between L_1O and SLS arises from the fact that the proposed method computes the pseudo-inverse of $XW \in \mathbb{R}^{m\times s}$, which has a time complexity of $\mathcal{O}(ms^2)$, while solving $L_t a = b$ in SLS has a time complexity of $\mathcal{O}(n^3)$ despite what Mathur says with its $O(n^2)$ conjugate gradient argument.

Figure 5.9 further supports these findings. In this experiment, 700 items were sampled from MNIST1K, with fixed dimensions $m = 700$ and $n = 784$, and the number of selected columns varied from 5% to 95% of m. The results show that while the computational time for the L_1O method increases with s, the computational time for the SLS method remains stable, consistent with theoretical expectations.

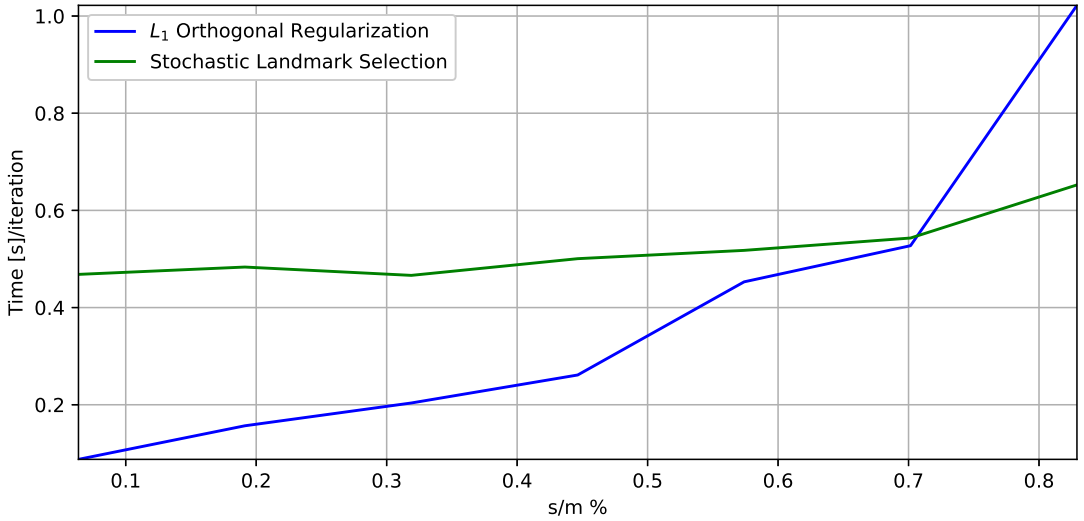


Figure 5.9: Empiric computational time

5.4.2 Efficiency

Analyzing the empirical computational time and the precision of the SLS and the L_1O method separatly present some limitations. Indeed the comparisons were only focused on the computational time per iteration and the precision test were focusing on the approximation factor of the columns choice given by the algorithm.

Test procedure

Recall that in the L_1O method, W is the $n \times s$ matrix that is updated at each iteration. In the SLS method, T is a diagonal matrix with the vector t on its diagonal also updated at each iteration. Each entry of t ranges between 0 and 1, indicating whether the column corresponding with the index of the entry, should be selected.

Figures 5.10, 5.12, and 5.11 display the approximation factor values if the L_1O's gradient descent and the SLS's Stochastic Gradient Descent had been stopped after a certain amount of time (right) or a specific number of iterations (left). At the end of continuous optimization, neither method has yet finalized the column selection process. The L_1OR method requires a truncated SVD on W^*, while the SLS method needs to identify the s largest elements of the vector t. This explains why the stopping criterion is based not on the error from an actual column selection,

but on the values of the regularized objective function.

To generate the figures, the values of W and T were saved at each iteration (and every 3 iterations for the Random dataset), along with the corresponding time and iterations. After the algorithm reached its stopping point, the actual column selection s was computed for each saved instance of W and T. The figures then illustrate the resulting approximation factors. It is important to highlight that the two methods are displayed on different x-axes, corresponding their respective progressions. For each dataset the number of columns s to select and the value of the hyperparameters are listed in Table 5.2.

Results

Figures 5.10, 5.12, and 5.11 confirm that the L_1O method is significantly faster than the SLS method. Depending on the specific configuration, the total computational time of the L_1O method is approximately 5 to 40 times faster than that of the SLS method. The figures also illustrate that as the size of n increases relative to the other problem dimensions s and m, the L_1O method's speed advantage over the SLS method becomes more pronounced.

The downward-upward-downward pattern observed in the L_1O method's CSSP loss curve is not observed in the regularized objective function as illustrated in the upper part of Figure 5.3 where the regularized objective function consistently decreases until the stopping criterion is satisfied. Considering the definition of the stopping criterion, it is evident that any upward movement in the regularized objective function would trigger the criterion, causing the optimization process to halt.

A similar behavior is observed in the SLS method, particularly in Figure 5.11, where the curve initially rises before descending. In Figures 5.10 and 5.12, the SLS curve also increases after a rapid decrease. These patterns highlight a fundamental limitation of both methods: they are optimizing a regularized objective function, i.e. a loss function that is not directly aligned with the objective function of the CSSP.

It can be oberved that the SLS method takes an extended time to stop. One possible explanation, as displayed in the figures 5.10, 5.12, 5.11, lies in the nature of how the method approaches the problem. The figures only shows the approximation factor for selecting s columns. However, the method actually constructs a vector T with values that tend toward the bounds of the interval $(0, 1)$, which is then used

to create a sampling matrix $X_T = XT$ that minimize the regularized objective function. By selecting the s largest values of T, the method chooses a submatrix of this already sampled matrix X_T. The key point is that the method focuses on minimizing the matrix X_T, rather than directly finding an optimal submatrix with exactly s columns. This process might explain why the method takes longer to stop. The termination condition is based on X_T and not on the s largest values of T. This hypothesis might also explains why the SLS method is more accurate than the proposed method for values of s close to m.

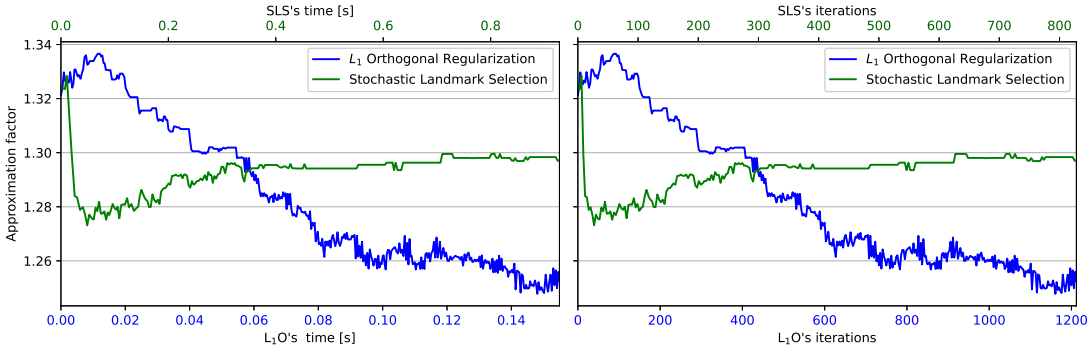


Figure 5.10: Approximation factor Hyperparameters determination (Random, 20 trials, $s = 0.5m$)

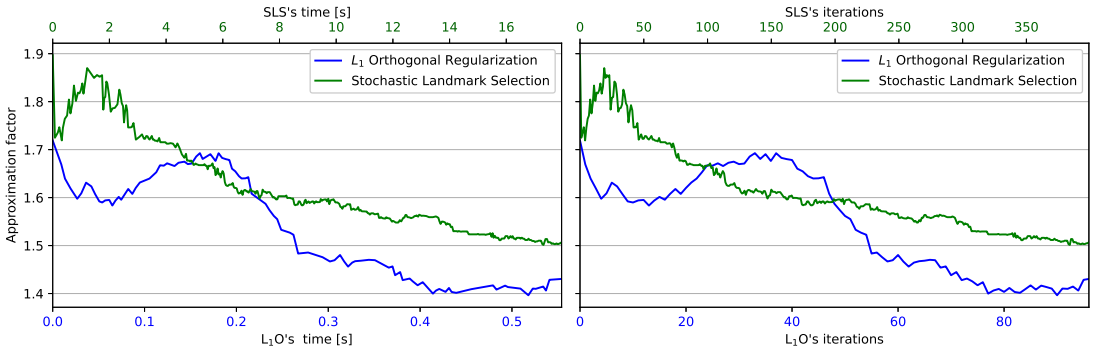


Figure 5.11: Approximation factor Hyperparameters determination (Arrithmia, 5 trials, $s = 0.3m$

When the stopping criterion is reached, the L_1O method demonstrates better accuracy, indicated by a smaller approximation factor, compared to the SLS method, as shown in Figures 5.5, 5.6, and 5.7. It can be conclude that the L_1O method outperform in term of precision and empirical computational speed the SLS method.

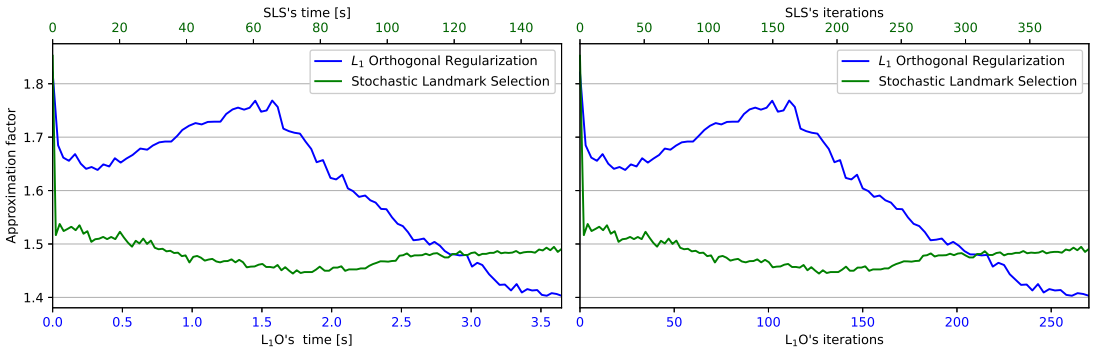


Figure 5.12: Approximation factor Hyperparameters determination (MNIST1K, 5 trials, $s = 0.3m$)

Chapter 6

Conclusion

The goal of this master thesis was to develop a new method for selecting columns for the CSSP problem using Theorem 2.3. This theorem enables the removal of the combinatorial constraint, allowing the use of linear optimization tools. The primary objective was to demonstrate the potential of this method by comparing it to another method that also relaxes the combinatorial constraint and uses gradient descent.

Chapter 1 introduced the importance and role of dimension reduction in data science. It explained, through a basic example, the limitations of Principal Component Analysis (PCA) and the need to reduce the dimensionality of a dataset by projecting it onto existing features. This naturally led to the presentation of the Column Subset Selection Problem (CSSP) and its main challenge: the computational difficulty of solving it exactly. The chapter concluded with a brief introduction to the proposed method and the potential benefits it could bring.

Chapter 2 reviewed some of the mathematical tools required for the remainder of the thesis. It primarily detailed the Singular Value Decomposition (SVD) and its properties, as well as the theorem on which the proposed method is based.

Chapter 3 formally defines the CSSP and reviews existing methods that attempt to solve, bound, or approximate the CSSP. Special emphasis is placed on the Stochastic Landmark Selection method, as it is also a method using a relaxation of the combinatorial constraint using a regularization term. Thus the SLS method provides the most significant and consistent results when compared to our method. Each method is also accompanied by a brief discussion of its theoretical complexity.

Chapter 4 details the proposed method, called L_1 Orthogonal Regularization. This chapter explains how Theorem 2.3 is used to relax the combinatorial constraint, transforming it into an unconstrained regularized continuous optimization problem. The chapter also develops the analytical form of the gradient of the regularized objective function. It then presents the gradient descent approach used to solve the problem and discusses the method's hyperparameters. The chapter concludes with an in-depth discussion of the method's theoretical time complexity.

Chapter 5 first determines the optimal parameters for the different methods to be tested. It then examines the precision and empirical time complexity of the presented methods across three different datasets. The first dataset is artificial, with matrix entries uniformly distributed, used to test the methods on data without any particular shape or meaning. The second dataset consists of images of digits (MNIST), and the third dataset contains Electroencephalograms (ECGs), known as the Arrhythmia dataset, used to detect abnormal heartbeats, a condition known as arrhythmia. Chapter 5 shows that the proposed method is faster than the Stochastic Landmark Selection (SLS) method. It is demonstrated that the empirical computational cost per iteration aligns with the theoretical complexity, $\mathcal{O}(Nn^3)$ for the SLS method and $\mathcal{O}(\max(mns, n^2s))$ for the proposed method. One of the main advantages of the proposed method stems from this difference in complexity. Additionally, the proposed method achieves better precision despite using a less advanced optimization technique.

As a result, this master thesis successfully achieved its objectives of developing the proposed method and exploring its potential. The performance comparisons in Chapter 5 clearly indicate that this initial version of the method already offers valuable performance in terms of precision and complexity. This thesis lays the groundwork for more advanced methods.

The performance of a version using a stochastic gradient descent or the investigation of methods with non-constant step sizes seems promising. Furthermore, adjusting the regularization factor λ should be explored, as it could potentially minimize the scale of the approximation made in Step 3 of the proposed method 4.1.3.

Declaration of Generative AI and AI-Assisted Technologies in the Writing Process

During the preparation of this work, the author used *ChatGPT 4.0* from OpenAI to verify grammar and spelling, enhance sentence structure, improve clarity, fluency and ensure coherence and readability throughout the manuscript. It is important to note that *ChatGPT 4.0* did not contribute any information, data, or original content to this work: its role was solely to assist in refining the existing text. After each use of this tool, the author thoroughly reviewed and edited the content as needed and take full responsibility for the final content of the publication.

Bibliography

[1] A. Asuncion and D. J. Newman. Uci machine learning repository. `https://archive.ics.uci.edu/ml`, 2007. University of California, Irvine, School of Information and Computer Sciences.

[2] Christos Boutsidis, Michael W Mahoney, and Petros Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 968–977. SIAM, 2009.

[3] Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, 2014.

[4] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013.

[5] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The mnist database of handwritten digits, 1998.

[6] Michael W. Mahoney and Petros Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, January 20 2009.

[7] Massart E. Massion B. Minimizers of orthognal regularizers. work in progress.

[8] Anant Mathur, Sarat Moka, and Zdravko Botev. Column subset selection and nyström approximation via continuous optimization. In *Proceedings of the Winter Simulation Conference*, WSC '23, page 3601–3612. IEEE Press, 2024.

[9] C.-T. Pan. On the existence and computation of rank-revealing lu factorizations. *Linear Algebra and its Applications*, 316:199–222, 2000.

[10] Yaroslav Shitov. Column subset selection is np-complete. *Linear Algebra and its Applications*, 610:52–58, 2021.

[11] Alexis Victor. Numerical investigation cssp. https://github.com/AlexisVictor/CSSP, 2024.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**