UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingeniería de Sistemas
Informáticos

**Column subset selection in practice:
efficient heuristics and regularization**

TESIS DOCTORAL

Bruno Ordozgoiti Rubio
Máster Universitario en Ciencias y Tecnologías de la Computación
por la Universidad Politécnica de Madrid

2018

Departamento de Sistemas Informáticos
Escuela Técnica Superior de Ingeniería de Sistemas
Informáticos

**Column subset selection in practice:
efficient heuristics and regularization**

Bruno Ordozgoiti Rubio

Máster Universitario en Ciencias y Tecnologías de la Computación
por la Universidad Politécnica de Madrid

Dirigida por

Alberto Mozo Velasco

Jesús García López de Lacalle

Doctor en Informática

Doctor en Matemáticas

2018

Acknowledgements

I would first like to thank my supervisors, Alberto Mozo and Jesús García López de Lacalle, for their support. Many others have contributed, whether knowingly or unknowingly, to the completion of this thesis. One person I feel particularly compelled to mention is Professor Antonio Hernando, who has always lent an ear to my questions and has never hesitated to generously share his nuanced understanding of a wide range of topics in machine learning and mathematics. Other academics who I have held constructive discussions with are José Ignacio Tello, Luis Miguel Pozo and José Ramón Sánchez Couso. I am also grateful to Sandra Gómez for helping me navigate the labyrinth of academic bureaucracy.

I must also mention the colleagues I have had the privilege to work with during the last few years, in particular Bo Zhu, Stanislav Vakaruk, Alexandru Mara, Enrique Fernández and Álex Martínez. Our exchanges during this time were a constant source of enlightenment and entertainment.

I would also like to express my gratitude to Aris Gionis and the team he supervises at Aalto University, in Finland, with whom I spent six months as a visitor. I cannot overstate how much I learned during that period, in our numerous conversations about science, academia and Finnish weather.

I am of course enormously grateful to my parents, who despite the mercurial nature of my ambitions have never withdrawn their support. And to Mariela, whose company makes life arbitrarily better.

Abstract

Today, data are available at an unprecedented scale. An overwhelming quantity of Internet-connected devices generate a constant trickle of pieces of information all over the world, much of which are processed in real time or stored for later use.

Making sense of these enormous data sets is often an challenging endeavour. Their size demands the use of massive computational resources, which motivates the design of efficient algorithms. Additionally, these data usually contain measurements of a large number of variables, which poses a wide variety of problems. To address the latter, a family of techniques commonly referred to as dimensionality reduction is studied.

In this thesis we address the problem of feature selection, a subset of dimensionality reduction methods that preserve the semantic meaning of the original data variables. To do so, we analyze a problem formulation known as column subset selection. A significant advantage of column subset selection is that the models it produces are simple and in some cases easy to interpret. In an age where notable advances in applied computer science are met with growing concerns about ethics and transparency, model simplicity can become a key requirement in many scenarios.

The column subset selection problem has received significant attention in the computer science literature over the last few years, mainly from a theoretical perspective. Here we analyze the problem from a more practical standpoint. Our contributions can be summarized as follows.

First, we propose the use of a local search heuristic. We show empirically that it outperforms existing algorithms and derive elementary approximation guarantees.

Furthermore, we take advantage of the nature of the problem formulation to derive an efficient implementation suitable for practical use.

Second, we introduce regularized formulations of the problem. We derive a greedy algorithm for these new objectives and demonstrate empirically that it produces improved subsets with respect to multiple criteria.

Resumen

Hoy en día, los datos están disponibles en un volumen sin precedentes. Una cantidad abrumadora de dispositivos conectados a Internet genera un flujo constante de información por todo el mundo, mucha de la cual se procesa en tiempo real o se almacena para usos posteriores.

Extraer conocimiento de estos ingentes conjuntos de datos suele ser una tarea difícil. Su tamaño exige el uso de recursos computacionales masivos, lo que motiva el diseño de algoritmos eficientes. Además, estos datos suelen contener mediciones de una gran cantidad de variables, y esto acarrea una amplia serie de problemas. Para atajarlos, se estudia una familia de técnicas conocida como "reducción de la dimensionalidad".

En esta tesis abordamos el tema de la selección de variables, un subconjunto de las técnicas de reducción de la dimensionalidad con la particularidad de que preservan el significado semántico de las variables originales. Para ello, analizamos un problema conocido "selección de un subconjunto de columnas". Una ventaja significativa de este problema es que da lugar a modelos simples y en ocasiones fáciles de interpretar. En la actualidad, los avances en ciencias de la computación aplicada van a menudo acompañados de preocupaciones sobre ética y transparencia, por lo que la simplicidad de los modelos utilizados puede ser clave en muchos casos.

El problema de la "selección de un subconjunto de columnas" ha recibido bastante atención durante los últimos años, principalmente desde un punto de vista teórico. En este texto analizamos el problema desde una perspectiva más práctica. Las contribuciones aquí recogidas se resumen a continuación.

En primer lugar proponemos el uso de una heurística de búsqueda local. Mostramos empíricamente que ofrece un rendimiento superior al de otros algoritmos y demostramos garantías de aproximación elementales. Además, aprovechamos la naturaleza del problema para formular una implementación eficiente adecuada para el uso práctico.

En segundo lugar, introducimos formulaciones regularizadas del problema. Proponemos un algoritmo voraz para optimizarlas y demostramos empíricamente que los subconjuntos de columnas resultantes son superiores con respecto a múltiples criterios.

Contents

1	Introduction	1
1.1	Contributions	6
2	Preliminaries	8
2.1	Notation	8
2.2	The singular value decomposition	10
2.3	The QR decomposition	11
2.4	The Moore-Penrose pseudoinverse	11
3	The column subset selection problem	13
3.1	Variants	14
3.2	Approximation	15
3.2.1	Lower bounds	16
3.2.2	Existing guarantees	17
3.2.3	Open problems	18
4	Background	20
4.1	Rank-revealing QR factorizations	20
4.2	The CUR decomposition	22

4.3	Randomized algorithms	23
4.3.1	Adaptive and volume sampling	26
4.3.2	Leverage score sampling	26
4.4	Heuristics	27
4.4.1	Greedy algorithm	28
4.4.2	A* search	29
5	Efficient local search	31
5.1	Implementation	33
5.2	Analysis	41
6	Regularization	44
6.1	Regularization in column subset selection	48
6.2	Regularized greedy column subset selection	50
6.2.1	Algorithms for regularized greedy column subset selection . .	52
7	Numerical experiments	65
7.1	Local search	65
7.1.1	Data	67
7.1.2	Objective function	68
7.1.3	Running time	69
7.1.4	Reconstruction capabilities	71
7.1.5	Benefits of randomized initialization	72
7.1.6	Iterations	73
7.1.7	Approximation error	74
7.1.8	Kahan matrix	75
7.2	Regularized column subset selection	77

7.2.1	Generalization ability	80
7.2.2	Stability	81
7.2.3	Conditioning	83
7.2.4	Clustering	84
7.2.5	Image reconstruction	87
7.2.6	Running time	89
8	Conclusions & future work	100
	Appendix A Proofs	114
	Appendix B Detailed derivations	125

Chapter 1

Introduction

The central challenges in modern applications of computer science almost invariably involve some form of data processing. This is always a daunting task: data collected by measuring natural phenomena are often noisy, unreliable, redundant, either frustratingly scarce or unfathomably numerous, and sometimes covertly changing over time. However, data usually hold valuable information, and our goal in this thesis is to study methods to help us uncover it.

Of course, the array of data processing techniques available to the applied computer scientist is too vast to cover in a single monograph. We therefore choose to concern ourselves with the problem of dimensionality reduction, and within that scope, we focus on variable or feature selection.

High dimensionality, that is, a large number of variables in data, can at first seem advantageous. The more variables we can measure, the larger the amount of information that will become available to us about the phenomenon we are studying. Nevertheless, things are unfortunately not that simple. In fact, high dimensionality can be so problematic that it has led to the notion of “*curse of dimensionality*”.

To quote Richard E. Bellman, who first coined the term, the curse of dimensionality is "*a malediction hat has plagued the scientist from earliest days*" [3].

The example that Bellman proposes to illustrate this notion, given in the context of calculus of variations, is illuminating. Consider a function $f : \mathbb{R} \rightarrow \mathbb{R}$. If we want a discrete account of its behavior in the interval $[0, 1]$ to a fidelity of 0.1, we need 10 points. How about a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$? If we want the sampled points to be separated by a distance of at most 0.1, we will need 10^2 such points. In $\mathbb{R}^3$ we will need 10^3 points. More generally, the number of points required to obtain a representative sample of the support increases exponentially in the dimensionality.

Another problem of high dimensionality comes from the excess of degrees of freedom. Consider a set of n observations $\{(x_i, y_i) : 1, \dots, n\} \subset \mathbb{R}^d \times \mathbb{R}$. We are tasked with building a model to predict the value of y_i given x_i . We constrain ourselves to functions of the form $w^T x_i$. The problem of finding the best estimate of w so as to minimize the sum of squared errors

$$E(w) = \sum_{i=1}^n (w^T x_i - y_i)^2$$

amounts to approximately solving the following linear system:

$$Xw = y$$

where X is a matrix whose i -th row is x_i^T and $y = (y_1, \dots, y_n)^T$. If d grows approaching n , the system will become increasingly close to compatibility and the error of approximately solving it will most likely decrease significantly. Whenever $d \geq n$, the system can be solved exactly. Now, consider that the entries of both X and y consist entirely of values sampled uniformly at random from $[0, 1]$. The error of the obtained

solution suggests we can perfectly estimate y_i given new observations of x_i , but this could not be further from the truth!

Another concern is simply the impact that dimensionality can have on the required computational resources. For instance, the approximate solution of a linear system as the one mentioned above requires roughly $O(nd^2)$ computations. If we are dealing with tens of thousands of variables—which is not rare in fields like computer vision or bioinformatics—the quadratic dependence on dimensionality can be problematic.

These are just a few of the reasons why high dimensionality can be troublesome, although additional illustrating examples can be found in the literature [30]. At this point we hope that the pursuit of methods to deal with the potential hurdles of processing high-dimensional data is well motivated. As mentioned above, the family of techniques developed for this purpose is commonly known as *dimensionality reduction*.

One way to approach dimensionality reduction is to work under the hypothesis that despite the high dimensionality of a given data set, the behaviour of the observations can be explained by a small number of driving factors. Perhaps the most well-known method for this purpose is principal component analysis (PCA) [48]. If we interpret the set of observations for each of our variables as a vector, PCA reveals an orthogonal basis for the subspace they span. What is interesting about this basis is that the vectors it is composed of can be sorted by the amount of data variance they capture. That is, it provides us with low-dimensional subspaces where we can project our data with minimal loss of information in some sense.

In close relation to PCA we have the singular value decomposition (SVD) [70, 37]. The SVD is a factorization that reveals the closest possible low-rank approximation—in a family of matrix norms—we can obtain. The SVD is in fact equivalent to

PCA, save for a translation required by the latter so that the loss is measured in sample variance rather than a plain entrywise matrix norm. The SVD has proved useful as a means to uncover explanatory factors in several fields, from recommender systems [17] to geophysics [69].

Other approaches to dimensionality reduction involving the uncovering of small sets of explanatory factors have recently found success in the machine learning literature [43, 51, 49].

Sometimes, however, uncovering explanatory factors might not be entirely convenient for the task at hand. Imagine a scenario where a team of doctors in a hospital want to develop a predictive model for estimating the risk a patient has of contracting a disease. Their variable set might consist of blood sugar levels, weight, age, and additional values obtained through blood analyses and other tests.

If they were to use one of the methods mentioned above, such as PCA, to uncover a few explanatory factors, they would probably obtain more robust predictions than if they used the whole variable set. However, the model, and thus the predictions, would be structured in terms of the resulting abstract factors. The doctors might prefer a model that makes predictions based directly on the original variables. For instance, it is useful for them to know if, say, blood sugar levels and weight are especially influential on the likelihood of developing the disease.

The task of reducing the dimensionality by choosing a subset of the available variables, rather than uncovering a set of abstract explanatory factors, is known as feature selection. In addition to producing models expressed in terms of the original variables, which can be meaningful to domain experts, it is advantageous when collecting these variables is costly. In the scenario described above, for instance, some of the involved tests might be expensive or even inflict physical pain, so by reducing the number of required variables we could be decreasing the strain put on

patients and saving the hospital money.

Feature selection has received a considerable level of attention in the machine learning literature [41, 78, 46, 59], and is arguably one of the central problems in data analysis. Unlike some other forms of dimensionality reduction, feature selection, once we establish formally what makes a good feature subset, is in general a form of combinatorial optimization, and thus the problems that arise in this domain are often hard to solve.

The medical scenario described above is an example of *supervised* feature selection; given a target value we are tasked with finding the variables that carry the most predictive power. There exist numerous proposals attempting to find good feature subsets for this purpose in different contexts [72, 32, 8, 74].

The counterpart to this is *unsupervised* feature selection. In this case there is no target variable, so the choice of features concerns finding subsets which are in some sense especially meaningful with respect to the structure of the data. As in the supervised case, there exists a wealth of proposals for unsupervised feature selection in the literature, employing a variety of criteria for locating good subsets [58, 24, 42, 79, 11, 77, 80, 54, 45, 65, 44, 76, 23, 75].

One possible approach for this purpose is commonly referred to as column subset selection, and can be informally described as follows: given a matrix A , find the subset of k of its columns with which we can build a linear model that best approximates the rest. This problem is the main focus of this dissertation. Column subset selection has received considerable attention, especially in the numerical linear algebra and the theoretical computer science communities.

A significant advantage of column subset selection is its linear nature. Even though this limits the approximation capabilities of the chosen subsets, the resulting models are simple and in some cases easy to interpret. Nowadays, the adoption of

computational methods to help in decision-making processes is undoubtedly becoming prevalent across many different domains. However, there are increasing concerns over their accountability and the need to establish causal relationships between the inputs these mechanisms receive and the outputs they produce. Therefore, we argue that techniques like column subset selection are to become an essential element of the computer science practitioner’s toolbox.

The purpose of this thesis is to study column subset selection in practice. Our contributions are summarized next.

1.1 Contributions

Efficient local search We first set out to address the following question: can we improve the performance of existing methods for column subset selection in practice? To this end, we analyze some well-known methods and analyze their possible pitfalls. This leads to the choice of a local-search heuristic. Our results can be summarized as follows:

- We show that one-swap local search for column subset selection can be implemented in $O(mnk)$ operations per iteration, involving an initial computation of cost $O(mn^2)$, for an input matrix of size $m \times n$ and the choice of k columns.
- We empirically demonstrate its superiority with respect to other algorithms in a variety of real-world data sets.
- We derive elementary approximation guarantees.

Regularization In the second part of this thesis we address the problem of regularization. Column subset selection, as is usually formulated in the literature, is

unregularized and hence inadequate for practical purposes. We address this problem, producing the following results:

- We propose a regularized formulation of the column subset selection problem.
- We produce an algorithm to greedily maximize this objective and show that each step can be solved in essentially $O(\min\{mn, n^2\})$ time complexity.
- We propose alternative objectives for feature selection in practice.
- We offer a lower bound for the error of the proposed problems.

Chapter 2

Preliminaries

2.1 Notation

Throughout this dissertation we use capital letters (A) to denote matrices and lowercase letters (v) to denote vectors. Occasionally, we use capital letters (S) to denote sets. The distinction between sets and matrices should be clear from the context. $A_{i\cdot}$ is the i -th row and $A_{\cdot i}$ the i -th column of the matrix A . A_{ij} is the entry in the i -th row and j -th column of matrix A . In the case of a vector v , its i -th entry is denoted by v_i , unless different notation is temporarily introduced (this will be made clear in the text). $\mathbb{R}^{m \times n}$ denotes the set of real-valued matrices of size $m \times n$.

$\|v\|_2 = \sqrt{\sum_i v_i^2}$ denotes the ℓ -2 norm of the vector v . $\|A\|_F = \sqrt{\sum_i \sum_j A_{ij}^2}$ denotes the Frobenius norm of the matrix A , and $\|A\|_2 = \max_x \frac{\|Ax\|_2}{\|x\|_2}$ is its spectral norm.

We also employ the following additional notation:

- C^+ denotes the Moore-Penrose generalized inverse of C [38].
- H_k^i denotes a diagonal $k \times k$ matrix whose entries are all 1, except for element

H_{ii} which is zero.

- Given a matrix $A \in \mathbb{R}^{m \times n}$ and a set $S \subset \{1, \dots, n\}$ with k elements, A_S is the $m \times k$ matrix comprised of the columns of A whose indices are in the set S .
- Given $A \in \mathbb{R}^{m \times n}$, $A \setminus i$ is a $m \times n - 1$ submatrix of A resulting from the removal of column i .
- In our pseudocode we employ the function `uniSampleWithoutReplacement( $S, k$ )`, which returns a sample of $k \in \mathbb{N}$ elements drawn uniformly at random without replacement from the set S .
- In our pseudocode, for a set R and some $i \in \mathbb{N}$, if we employ the notation $R[i]$ we consider the set to be ordered and $R[i]$ to be its i -th element.
- e_i is the i -th vector of the canonical basis of the indicated dimensionality.
- Given two matrices A and B , $(A|B)$ is the matrix resulting from appending the columns of B to A . E.g., if $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{m \times k}$, then $(A|B) \in \mathbb{R}^{m \times n+k}$ and consists of the columns of both matrices.
- $\sigma_i(A)$ denotes the i -th largest singular value of A .
- $\mathcal{P}(S)$ denotes the power set S
- $|S|$ is the cardinality of set S
- If x_i is a vector, then x_{ii} is its i -th element.
- $[n]$ denotes the subset of $\mathbb{N}$ defined as $\mathbb{N} \cap [1, n]$,
- Given vectors x, y of the same dimension, $x \circ y$ denotes the element-wise product of x and y .

2.2 The singular value decomposition

Given any matrix $A \in \mathbb{R}^{m \times n}$, we can write

$$A = U\Sigma V^T$$

where

- $U \in \mathbb{R}^{m \times m}$ is orthogonal, and its columns are known as left singular vectors of A .
- $V \in \mathbb{R}^{n \times n}$ is orthogonal, and its columns are known as right singular vectors of A .
- $\Sigma \in \mathbb{R}^{m \times n}$ is upper-diagonal with nonnegative entries $\sigma_1 \geq \dots \geq \sigma_n$. These are the singular values of A . The number of positive singular values equals the rank of A .

This is known as the singular value decomposition of A [37], and it is unique up to permutation of singular vectors corresponding to equal singular values.

It is well known that

$$\|A\|_F^2 = \sum_{i=1}^n \sigma_i^2$$

An important property of the singular value decomposition is that it provides the solution to the following problem:

$$\min_{X, \text{rank}(X)=k} \|A - X\|_F$$

In particular, the minimizer is given by

$$A_k = \sum_{i=1}^k \sigma_i u_i v_i^T$$

where $u_i = U_{:,i}$, $v_i = V_{:,i}$. In addition,

$$\|A - A_k\|_F^2 = \sum_{i=k+1}^n \sigma_i^2$$

2.3 The QR decomposition

Given any matrix $A \in \mathbb{R}^{m \times n}$, we can write

$$A = QR$$

where

- $Q \in \mathbb{R}^{m \times m}$ is orthogonal
- $R \in \mathbb{R}^{m \times n}$ is upper-trapezoidal

It follows that the i -th column of A can be written as a linear combination of the first i columns of Q .

2.4 The Moore-Penrose pseudoinverse

Given any matrix $A \in \mathbb{R}^{m \times n}$, we say A^+ is its pseudoinverse if it satisfies the following properties:

- $AA^+A = A$

- $A^+AA^+ = A^+$
- $(AA^+)^T = AA^+$
- $(A^+A)^T = A^+A$

If $A = U\Sigma V^T$ is the singular value decomposition of A , then $A^+ = V\Sigma^-U^T$, where

$$\Sigma_{ij}^- = \begin{cases} 1/\Sigma_{ij} & \text{if } \Sigma_{ij} > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (2.1)$$

Chapter 3

The column subset selection problem

Column subset selection is a problem in combinatorial optimization consisting in choosing a representative subset of columns of a matrix. The quality of the representation is measured as the error incurred in approximating the whole matrix using the subspace spanned by the chosen columns. Formally, it can be defined as follows.

Problem 1. *Column Subset Selection Problem (CSSP).* Given a matrix $A \in \mathbb{R}^{m \times n}$ and a positive integer k smaller than the rank of A , let $\mathcal{A}_k$ denote the set of $m \times k$ matrices comprised of k columns of A . Find $C \in \mathcal{A}_k$ so as to minimize

$$\|A - CC^+A\|_F^2 \tag{3.1}$$

where C^+ is the Moore-Penrose pseudoinverse of C .

The CSSP is known to be UG-Hard [15], and recent —though as of yet unverified— work proves its NP-hardness [68] by a reduction from graph three-coloring to a de-

cision version of the problem.

3.1 Variants

The problem of approximating a matrix using a subset of its columns has been studied in a variety of forms. First, we need not limit ourselves to the minimization of the Frobenius norm, but can use any matrix norm. The choices that are by far the most common in the literature are, however, Frobenius and spectral. Throughout this text, ξ can be replaced freely with either of $F, 2$.

A simple modification of the problem is to relax the limitation on the number of chosen columns in order to improve the approximation guarantees, letting us pick any number $r \geq k$. The question that arises naturally is the following: what is the minimum value of r we need to achieve a certain error? This can be formalized as follows.

Problem 2. *r-CSSP (r-Column Subset Selection Problem)* Given a matrix $A \in \mathbb{R}^{m \times n}$ and a real number $\delta > 0$, define $g(C, r) = \min_{C \in \mathcal{A}_r} \|A - CC^+A\|_\xi^2$. Find

$$\min\{r \in \mathbb{N} : g(C, r) \leq \delta\} \quad (3.2)$$

Another variation considers the choice of $r \geq k$ columns, but limits the rank of the approximation. Given $C \in \mathbb{R}^{m \times r}$, define $\Pi_{C,k}^\xi(A)$ to be the best rank- k approximation of A in norm ξ within the subspace spanned by the columns of C .

Problem 3. *ESSP (Enclosing Subspace Selection Problem)* Given a matrix $A \in \mathbb{R}^{m \times n}$ and a real number $\delta > 0$, define $h(C, r) = \min_{C \in \mathcal{A}_r} \|A - \Pi_{C,k}^\xi(A)\|_\xi^2$. Find

$$\min\{r \in \mathbb{N} : h(C, r) \leq \delta\} \quad (3.3)$$

Finally, we can consider a more general statement of CSSP, where the set of columns we can choose from is not necessarily a subset of the columns of the target matrix.

Problem 4. GCSSP (*Generalized Column Subset Selection Problem*). *Given matrices $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{m \times p}$ and a positive integer k , let $\mathcal{B}_k$ denote the set of $m \times k$ matrices comprised of k columns of B . Find $C \in \mathcal{B}_k$ so as to minimize*

$$\|A - CC^+A\|_F^2 \quad (3.4)$$

Of course, GCSSP can be extended as above to allow the choice of $r > k$ columns.

3.2 Approximation

In combinatorial optimization, the goal is usually to find efficient algorithms which provide provable performance guarantees with respect to optimal solutions [73]. The subset selection problems described above, however, are peculiar in that regard. A candidate solution $C \in \mathbb{R}^{m \times k}$ provides a rank- k approximation of the target matrix, CC^+A . Therefore, it is natural to measure the obtained error with respect to the best possible rank- k approximation [5], given by the singular value decomposition [70]. In the case of CSSP, r -CSSP and ESSP, the resulting guarantees obviously apply to optimal solutions.

In the case of GCSSP, though, seeking approximation bounds with respect to the best rank- k approximation is pointless. Simply consider an instance such that $A^TB = 0$. A reasonable alternative could be to derive approximations in terms of $\Pi_{B,k}^\xi(A)$, although we know of no such work. Existing proposals for GCSSP target optimal solutions instead [1].

3.2.1 Lower bounds

Bounds are known for the best possible achievable value of the ratio

$$\alpha = \frac{\|A - \Pi_{C,k}^\xi(A)\|_F^2}{\|A - A_k\|_F^2}$$

where A_k is the best rank- k approximation. Note that if $r = k$, CSSP and ESSP are equivalent. The lower bounds presented here are on the value of α that any algorithm *can hope to guarantee*.

For the case $r = k, \xi = F$, Deshpande et al. provide an example—and a matching algorithm—where $\alpha \geq k + 1$ for any subset of k columns of the matrix [19]. The tight example is simple and stems from geometric intuition, so we discuss it briefly. Consider a square matrix A of order $k + 1$ whose columns are the vertices of a regular simplex lying in the affine hyperplane $\{x_{k+1} = \alpha\} \subset \mathbb{R}^{k+1}$. Assume $p = (0, 0, \dots, 0, \alpha)$ is the centroid of said vertices. If α is small enough, the best rank- k approximation is given by a matrix A_k spanning the hyperplane $\{x_{k+1} = 0\}$. Since we can express the $k + 1$ points in the simplex perfectly except for their last coordinate,

$$\|A - A_k\|_F^2 = \sum_i \alpha^2 = (k + 1)\alpha^2$$

The loss in approximating A using any subset of k of its columns equals the loss in approximating the remaining column. The crucial observation made by the authors is that by definition of the centroid

$$p = \frac{1}{k + 1} \sum_i A_{\cdot i}$$

the ratio between said loss (by picking, say, the first k columns) and the loss in

approximating p is

$$\frac{d(C, A_{:k+1})^2}{d(C, p)^2} = (k+1)^2$$

Since α is arbitrary, we can choose it so that $d(C, p)^2 \geq (1 - \epsilon)\alpha^2$ for any $\epsilon > 0$. It follows that

$$d(C, A_{:k+1})^2 \geq (1 - \epsilon)(k+1)^2\alpha^2 = (1 - \epsilon)(k+1)\|A - A_k\|_F^2$$

For $r > k, \xi = F, \alpha \geq 1 + \frac{2k}{r}$ [20]. In spectral norm, in the case $r = k$, a lower bound for α is $\frac{n}{k}$ [18], whereas in the case $r > k$, a lower bound of $\frac{n}{r}$ was given by Boutsidis et al. [5].

The question of finding lower bounds for the ratios

$$\frac{\|A - \Pi_{C,k}^\xi(A)\|_F^2}{\|A - \Pi_{C_{opt},k}^\xi(A)\|_F^2} \qquad \frac{\|A - CC^+A\|_F^2}{\|A - C_{opt}C_{opt}^+A\|_F^2}$$

where C_{opt} is the optimal column choice have not, to the best of our knowledge, been addressed in the literature. It is however interesting, as it would provide insight on the approximability of this problem.

3.2.2 Existing guarantees

Again, we consider the ratio

$$\alpha = \frac{\|A - \Pi_{C,k}^\xi(A)\|_\xi^2}{\|A - A_k\|_F^2}$$

For the case $r = k, \xi = 2$, the best known result comes from the numerical linear algebra literature [39]. The approximation factor is of $\alpha = f^2k(n - k) + 1$ for any

$f > 1$. The algorithm requires $O((m + n \log_f n)n^2)$ operations.

For the case $r > k, \xi = 2$, a near-optimal factor of $O(\frac{\rho}{r})$, where ρ is the rank of A , was obtained by Boutsidis et al. [5].

When $r = k, \xi = F$ the best known approximation factor is $k + 1$ [18].

When $r > k, \xi = F$ a near-optimal factor of $1 + O(\frac{k}{r})$ can be achieved [5].

Since $\|A - \Pi_{C,k}^\xi(A)\|_\xi^2 \geq \|A - CC^+A\|_\xi^2$, any algorithm providing an approximation guarantee for ESSP with $r > k$ translates directly to r -CSSP. For the case $r > k, \xi = F$, the ratio for r -CSSP can be improved up to $\frac{r+1}{r-k+1}$, which is in fact optimal [40].

Recently, it was shown that an approximation of factor $(1 - \epsilon)$ with respect to the optimum can be obtained for the maximization equivalent of the GCSSP picking $O(\frac{k}{\sigma_k^2(S)\epsilon})$ columns with a greedy algorithm, where $\sigma_k^2(S)$ is the square of the smallest singular value of an optimal matrix [1]. In a similar result, the greedy choice of $\tilde{O}(\frac{k \log k}{\sigma_n^2(A)\epsilon^2})$ columns leads to a $(1 + \epsilon)$ approximation of $\|A - A_k\|_F^2$ [16].

3.2.3 Open problems

As shown in this section, the literature is now rich in results for the different variants of column subset selection. Near-optimal polynomial-time algorithms have been shown to exist for ESSP and CSSP. However, there remains a variety of questions left to answer surrounding this topic. We enumerate some of these below:

- Can $\Pi_{C,k}^2$ be computed exactly in polynomial time? This problem was highlighted by Boutsidis et al. [5].
- Can the optimum of GCSSP be approximated to a factor of $1 + \epsilon$ with a number of columns independent of $\sigma_k(S)$?
- More generally,

- what is the best approximation factor we can hope to achieve for GCSSP in polynomial time?
- what is the best additive error approximation we can hope to achieve for GCSSP in polynomial time?

Chapter 4

Background

In this section we offer an overview of some of the most relevant concepts that arise in the literature related to column subset selection problems.

4.1 Rank-revealing QR factorizations

The CSSP can be traced back to early works in numerical linear algebra describing rank-revealing matrix factorizations. These methods were motivated by the need to solve ill-conditioned linear systems numerically [35, 36], leading to the so-called QR decomposition with column pivoting (QRP) [9]. The issue at hand can be illustrated as follows. Given a full column-rank matrix $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, consider a linear system:

$$Ax = b$$

The approximate solution is given by $x = (A^T A)^{-1} A^T b$. If A is nearly numerically rank-deficient —i.e. it has a large condition number— then the inverse $(A^T A)^{-1}$

is prone to large numerical errors. Thus, we are interested in finding a subset of columns of A that are well-conditioned, but without significantly undermining the ability of the full matrix to approximate b .

This notion is formalized through the concept of rank-revealing factorizations. We limit our exposition to QR decompositions, although rank-revealing factorizations can be of other types [64].

Definition 1. *Rank-revealing QR factorization*

Given a matrix $A \in \mathbb{R}^{m \times n}$, we consider a factorization of the form

$$A\Pi = QR = Q \begin{pmatrix} R_{11} & R_{12} \\ & R_{22} \end{pmatrix} \quad (4.1)$$

where Q is orthogonal, Π is a permutation matrix, $R_{11} \in \mathbb{R}^{k \times k}$. This factorization is said to be rank-revealing if

$$\begin{aligned} \sigma_{\min}(R_{11}) &\geq \frac{\sigma_k(A)}{p_1(k, n)} \\ \sigma_{\max}(R_{22}) &\leq \sigma_{k+1}(A)p_2(k, n) \end{aligned}$$

where $p_i(k, n)$ are functions bounded by low-degree polynomials in k and n .

If the matrix A has numerical rank k , the high-order gap between $\sigma_k(A)$ and $\sigma_{k+1}(A)$ will be revealed by an adequate choice of Π . The reader wondering why this is the case should note that by the nature of the QR decomposition, if the block R_{22} is “small”, then most of the content of matrix A can be explained by a subspace of dimension k . These factorizations enjoy varied applications [13].

The relevance of these factorizations in this thesis arises from the following fact.

If $C = A\Pi_{[k]}$ then

$$\|A - CC^+A\|_\xi^2 = \|R_{22}\|_\xi^2$$

Therefore, any bound on the block R_{22} translates immediately to the CSSP, both in Frobenius and in spectral norm.

In the numerical linear algebra literature, bounds are usually found for $\|R_{22}\|_2$. These can easily be applied to the Frobenius norm by observing the following:

$$\|R_{22}\|_F^2 \leq \text{rank}(R_{22})\|R_{22}\|_2^2 = \text{rank}(R_{22})\sigma_1(R_{22})^2$$

One of the first methods for finding such a factorization is due to Chan [12], and guarantees $p_2(k, n) = \sqrt{n(n-k)}2^{n-k}$ in $O(mn^2)$ running time. A subsequent result by Gu & Eisenstat [39] yields the best known guarantee for the choice of exactly k columns in a deterministic fashion. Their algorithm runs in $O((m + n \log_f n)n^2)$ time and ensures $p_2(k, n) = \sqrt{f^2 k(n-k) + 1}$ for any $f > 1$, which is the guarantee mentioned in section 3.2.2.

4.2 The CUR decomposition

The *CUR* decomposition [22] factorizes $A \in \mathbb{R}^{m \times n}$ as

$$A = CUR$$

where C is a subset of k columns of A , R is a subset of r rows and $U = C^+AR^+$ is chosen to minimize the reconstruction error in Frobenius norm. Column subset selection is a special case where $R = A$, which by the definition of the pseudoinverse

results in

$$CUR = CUA = CC^+AA^+A = CC^+A$$

The CUR decomposition is motivated by the same reasons as column subset selection, namely the difficulty to interpret matrix factorizations like the singular value decomposition. In addition, the reduced matrix size results in more efficient computations. An algorithm achieving $\epsilon\|A\|_F$ additive error was described by Drineas et al. [22]. A near-optimal algorithm was given recently by Boutsidis and Woodruff, achieving $(1 + \epsilon)$ multiplicative error with $O(\frac{k}{\epsilon})$ rows and columns.

4.3 Randomized algorithms

Randomization has brought significant progress to the column subset selection landscape. In addition to their importance, the techniques underlying most randomized approaches motivate the choice of the heuristic we study in this thesis. We therefore devote part of our discussion to some of the most relevant results.

Most work in this area stems from the seminal paper by Frieze and Vempala [31], who showed that given a matrix A , in time polynomial in $k, \frac{1}{\epsilon}$, we can construct a description of a matrix D^* that approximates the best rank- k approximation of A to additive error, i.e.

$$\|A - D^*\|_F^2 \leq \|A - A_k\|_F^2 + \epsilon\|A\|_F^2$$

The description is a matrix of orthonormal columns Y such that $AD^* = AYY^T$. That is, the algorithm produces a set of orthonormal vectors.

In order to obtain the approximation guarantee, certain sampling assumptions are made. Namely, that we can sample rows from A such that row i is chosen with

probability

$$P_i \geq c \frac{\|A_{i:}\|_2^2}{\|A\|_F^2}$$

for some constant $c \leq 1$, and from a row i we can sample entry j with probability

$$Q_{j|i} \geq c \frac{P_{ij}}{P_i}, \quad P_{ij} = \frac{A_{ij}^2}{\|A\|_F^2}$$

The importance of these assumptions becomes apparent in the algorithm, which can be described as follows:

1. Sample p rows from A independently, where row i is sampled with probability $P_i \geq c \frac{\|A_{i:}\|_2^2}{\|A\|_F^2}$.
2. Form matrix $S \in \mathbb{R}^{p \times n}$. $A_{i:}/\sqrt{pP_i}$ is a row of S if $A_{i:}$ was sampled.
3. Sample p columns from S independently, where row j is sampled with probability $P'_j \geq \frac{c\|S_{:j}\|_2^2}{2\|S\|_F^2}$.
4. Form matrix $W \in \mathbb{R}^{p \times p}$. $S_{:j}/\sqrt{pP'_j}$ is a column of S if $S_{:j}$ was sampled.
5. Compute the top k singular vectors of W , $u^{(1)}, \dots, u^{(k)}$.
6. Let $T = \{t : \|Wu^{(t)}\|_2^2 \geq \frac{c\epsilon}{8k}\|W\|_F^2\}$. Output $v^{(t)} = \frac{S^T u^{(t)}}{\|W^T u^{(t)}\|_2}$ for $t \in T$.

The output vectors $v^{(t)}$ are the description of the matrix D^* . An important detail for performance is that for the second sampling step, we do not need to compute $\|S\|_F^2$, as we can take advantage of the fact $\sum_{i \in I} \frac{Q_{j|i}}{p} \geq \frac{c\|S_{:j}\|_2^2}{2\|S\|_F^2}$, where I is the set of row indices that were sampled and added to S , to suitably define P'_j .

The main insight behind this algorithm is that a random sample of a matrix must provide an approximation of said matrix. The crucial question is then which properties should the sampling process satisfy such that said approximation is good.

A key ingredient for this purpose is the following fact. If we take a sample S of s rows of A and define

$$w^{(t)} = \frac{1}{s} \sum_{i \in S} \frac{u_i^{(t)}}{P_i} A_{i:}.$$

then

$$\mathbb{E} [\|w^{(t)} - \sigma_t v^{(t)}\|_2^2] \leq \frac{1}{sc} \|A\|_F^2$$

What these statements mean is the following. Observe that $U^T A = \Sigma V^T$, that is, $\sum_i u_i^{(t)} A_{i:} = \sigma_t v^{(t)}$. This implies that using a scaled sample of s rows we can get a bounded-error approximation (in expectation) of $\sigma_t v^{(t)}$.

Another key ingredient is the fact that a sample of p scaled rows taken according to the scheme described above provides a good approximation of $A^T A$. Specifically,

$$Pr(\|S^T S - A^T A\|_F \geq \theta \|A\|_F^2) \leq \frac{1}{\theta^2 cp}$$

for any $\theta > 0$. The same result can of course be used for column sampling.

Using these tools, it is possible to prove the additive error approximation of the resulting low-rank approximation. The intuition behind this approach might seem obvious; a random sample of a matrix approximates said matrix, of course. However, these results formalized the extent to which this is true, and how to exploit the structure of the matrix to sample judiciously. In addition, they laid the groundwork for many of the subsequent advances in randomized linear algebra¹ [21] .

---

¹Note that even though the work of Frieze, Kannan & Vempala was published in 2004, a preliminary version of the paper appeared in the Proceedings of the 39th Symposium on Foundations of Computer Science in 1998

4.3.1 Adaptive and volume sampling

The sampling approach introduced by Frieze et al. [31] was improved using a technique called *adaptive sampling* [19]. This method arises from the simple observation that sampling independently can lead to the choice of redundant columns. Consider a sample of t sets of s columns of matrix A . Define $E_j = A - C_j C_j^+ A$, where $C_j = A_{\bigcup_{i=1}^{j-1} S_i}$. Column i is picked with probability

$$p_i^{(j)} = \frac{\|(E_j)_{:i}\|_2^2}{\|E_j\|_F^2}$$

That is, we build each column sample by considering the residual of A not explained by the previously sampled columns. This process decreases the additive error exponentially.

A technique that generalizes this notion is known as *volume sampling* [19], and consists in sampling subsets of columns with probability proportional to the volume of the simplex they form. Note that the addition of a redundant column will not result in an expansion of the volume spanned by the simplex. *Volume sampling* leads to multiplicative error of factor $k + 1$. Deshpande et al. gave an existence result showing that any matrix contains $k + \frac{k(k+1)}{\epsilon}$ columns achieving a $(1 + \epsilon)$ relative error approximation, and based on this fact they described a polynomial-time approximation scheme (PTAS) for projective clustering.

4.3.2 Leverage score sampling

Boutsidis et al. achieved notable approximation improvements for picking exactly k columns using a method known as leverage-score sampling. The i -th leverage score

of a matrix A can be defined as

$$p_i = \frac{1}{z} \|(V_k)_{i:}\|_2^2$$

where V_k is a matrix whose columns are the first k right singular vectors of A and z is a normalizing constant. By sampling columns independently with probabilities proportional to these values and then reducing the sample to exactly k columns using a factorization with pivoting, the authors achieve an approximation factor of $O(k^2 \log k)$ in Frobenius norm. A slight modification of the leverage scores permits guarantees in spectral norm as well.

4.4 Heuristics

The theoretical computer science community has yielded a large wealth of algorithmic results, providing near-optimal approximation guarantees for some of the varied forms in which column subset selection might present itself. These methods generally measure the quality of the obtained low-rank approximation and the related analyses rely heavily on their linear-algebraic properties. However, there is still a significant interest in studying the performance of purely combinatorial approaches. On one hand, not much is known about the hardness of approximation of column subset selection problems in general, as tight examples exist only for best rank- k subspaces. On the other, traditional combinatorial algorithms are generally much more convenient for practical implementation and exhibit good empirical performance [28, 27, 62]. Furthermore, recent works provide approximation guarantees for methods in this category [1, 16, 63].

Here we discuss some of the most relevant heuristics and the associated results.

4.4.1 Greedy algorithm

One of the most popular heuristics for column subset selection is the greedy algorithm. In addition to providing remarkable empirical performance, it can be implemented very efficiently [28, 27].

The greedy optimization of the CSSP objective is based on the following observation. Let $A \in \mathbb{R}^{m \times n}$ and let C be a matrix composed by a proper subset of the columns of A . Let $\hat{C}$ denote the matrix that results from adding a column of A to C , that is $\hat{C} = (C|A_{:w})$ for some column index w . Then

$$A - \hat{C}\hat{C}^+A = A - CC^+A - E_{:w}E_{:w}^+E \quad (4.2)$$

where $E = A - CC^+A$. This is easily seen considering that CC^+ is a projection onto the space spanned by the columns of C , and therefore all the columns of E are orthogonal to those of CC^+A .

Equation 4.2 implies that we can easily perform a greedy selection of columns, by updating A at each step to obtain the corresponding residual E and choosing a column w of E to minimize $\|E - E_{:w}E_{:w}^+E\|_F^2$. Furthermore, the minimizing column can be found efficiently. In [28] it is shown that

$$\operatorname{argmin}_i \|E - E_{:i}E_{:i}^+E\|_F^2 = \operatorname{argmax}_i \frac{\|G_i\|_2^2}{G_{ii}} \quad (4.3)$$

where $G = E^TE$. In addition, the values of $\|G_i\|_2^2$ and G_{ii} for all i can be updated efficiently every time we incorporate a new column to our subset [28]. Finally, the algorithm can be effectively run in a distributed manner taking advantage of the celebrated result known as Johnson-Lindenstrauss Lemma [47, 27]

Civril & Magdon-Ismail showed that if the greedy algorithm chooses $r = \tilde{\Omega}(\frac{k \log k}{\epsilon^2 \sigma_n^2(A)})$,

then

$$\|A - CC^+A\|_F^2 \leq (1 + \epsilon)^2 \|A - A_k\|_F^2$$

In subsequent work, an approximation scheme was given for the optimal solution [1]. If the greedy algorithm chooses $r = \Omega(\frac{k}{\epsilon\sigma_k^2(S)})$ columns, where S is an optimal k -column submatrix, then

$$\|CC^+A\|_F^2 \geq (1 - \epsilon) \|SS^+A\|_F^2$$

This work targets the maximization of $\|CC^+A\|_F^2$, which is equivalent to the minimization of $\|A - CC^+A\|_F^2$. The required number of columns is moreover shown to be optimal in the sense that there exists an instance of the problem where the greedy algorithm will need $\Omega(\frac{k}{\epsilon\sigma_k^2(S)})$ to approximate the optimum accordingly.

These results are strong. However, it would be interesting to determine whether other heuristics can achieve similar approximations without a dependence on $\sigma_k(S)$ or $\sigma_n(A)$. The value of $\sigma_n(A)$ can be arbitrarily small. In the case of $\sigma_k(S)$, the authors assume the columns of S to be normalized to unit norm, so $\sigma_k(S)$ is guaranteed to be at most 1, and can also be arbitrarily small. In the case of data coming from real-world measurements, the optimal subset is likely to be well-conditioned. However, a certain amount of correlation is to be expected. This can result on a significant increase in the number of columns required by the greedy algorithm to guarantee the desired relative approximation.

4.4.2 A* search

Since column subset selection is most likely NP-hard, practical algorithms often focus on finding approximate solutions. The optimal subset can be found by naively

enumerating all $\binom{n}{k}$ candidates and computing their approximation error. This, however, is infeasible for even moderate values of n, k . In cases where finding the optimal subset is of interest, it is compelling to attempt to design algorithms that can do this more efficiently.

A proposal along these lines is an adaptation of A* search [66] to column subset selection [2]. Column subsets are nodes in the search graph, which is pruned taking advantage of the following bound

$$\|A - C_{OPT}C_{OPT}^+A\|_F^2 \leq (k+1)\|A - A_k\|_F^2$$

where C_{OPT} is an optimal k -column subset. The computation of the eigenvalues of the residuals at each step are significantly optimized to produce a relatively efficient algorithm.

Even if the algorithm can find an optimal subset faster than a naive method, the problem remains intractable for moderately large values of n and k . Nevertheless, it can be useful for some purposes, such as evaluating the performance of algorithms with respect to the optimum or testing theoretical claims.

Chapter 5

Efficient local search

As discussed in chapter 4, column subset selection has received considerable attention from the theoretical point of view, and near-optimal approximation algorithms are known for some variants of the problem. However, even though there exist asymptotically optimal algorithms, these are often impractical, involving large or even unknown constants —see, for instance, the complete version¹ of [7]. Therefore, it is interesting to assess how well we can do in practice with easy-to-implement heuristics. Here we examine a well-known heuristic for combinatorial optimization, namely local search.

Our main focus is on the empirical performance and the efficiency of the algorithm. However, we also pay attention to the theoretical properties of the algorithm, carrying out analyses to provide insight on its behaviour and deriving elementary approximation guarantees.

The motivation of using local search for column subset selection is based on the following key idea: some of the methods we have discussed in chapter 4 attempt

---

¹<https://arxiv.org/pdf/0812.4293>

to find good subsets by examining columns individually. In the case of randomized methods this is mainly due to the convenience of sampling independently, which simplifies the analysis significantly. Consider, for instance, leverage-score sampling (see section 4.3.2). The leverage score is large for a matrix column if it is similar to a left singular vector. Since V_k is a submatrix of an orthogonal matrix, high-scoring columns tend to be isolated in that regard. However, we have observed that in real datasets this isolation is rare, and fairly correlated columns can have similar moderately high scores. This can lead to columns with a considerable amount of redundancy being sampled at the expense of candidates that might contribute more to the final subset despite their relatively low leverage scores. This is alleviated in part by adaptive sampling (see section 4.3.1), but one still has to sample an initial subset of k/ϵ columns.

A similar problem affects the greedy algorithm (see section 4.4.1), which is in fact a drawback of greedy algorithms in any hard combinatorial setting. Columns are picked one by one, choosing the one that minimizes the objective function accounting for the ones chosen before. This scheme, however, does not account for the fact that more columns will be chosen afterwards. The example below illustrates how a greedy approach might fail. Let us consider the following matrix.

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 + \epsilon & 0 \\ 1 & 0 & 0 & 1 + \epsilon \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (5.1)$$

for some small ϵ . Let us consider that we want to pick k columns. The best choice for

$k = 1$ is the first column. However, that column does not belong to the best subset for $k = 2$, which contains the second and fourth. A greedy approach would choose the first column at the first iteration and never discard it, resulting in a sub-optimal final subset even in this very simple example.

The local search algorithm, on the contrary, starts from a random subset and updates feature choices taking the entirety of the subset into account. An interesting empirical observation is that even a single iteration will often yield better results than other algorithms, despite the fact that the initial subset is picked uniformly at random.

A drawback of local search is that a straightforward implementation would be very inefficient. We therefore derive a series of non-trivial optimizations that make it possible to draw subsets of tens or hundreds of features in a few seconds or milliseconds on a consumer-grade computer.

5.1 Implementation

Algorithm 1 is an adaptation of local search for CSSP. For an input matrix $A \in \mathbb{R}^{m \times n}$ let us consider that we want to pick k columns. First, an initial subset R of k columns is chosen uniformly at random without replacement (line 2), forming a matrix $C = A_R \in \mathbb{R}^{m \times k}$. Then, the algorithm iterates until convergence as follows. For $i = 1, \dots, k$, column i is removed from C , forming matrix $\tilde{C} \in \mathbb{R}^{m \times k-1}$, and is replaced by another column such that the objective function (3.1) is minimized over all possible $n - k + 1$ replacements (we don't rule out the column we removed). The algorithm converges when no single column replacement yields an improvement in the objective function anymore.

Algorithm 1 Local search for column subset selection

```
1: procedure LocalSearch( $A, k$ )
2:    $R \leftarrow \text{uniSampleWithoutReplacement}(\{1 \dots n\}, k)$ 
3:    $C \leftarrow A_R$ 
4:   Compute  $C^+$ 
5:   while not converged do
6:     for  $i = 1, \dots, k$  do
7:        $\tilde{C} \leftarrow CH_k^i$   $\triangleright$  Zero out column  $i$ 
8:        $w \leftarrow \text{argmin}_j \|A - (\tilde{C}|A_{:j})(\tilde{C}|A_{:j})^+ A\|_F$ 
9:        $C \leftarrow (\tilde{C}|A_{:w}) \setminus i$ 
10:       $R[i] \leftarrow w$ 
11:    end for
12:  end while
13:  output  $R$ 
14: end procedure
```

`LocalSearch` can find very good approximations to the CSSP objective function. However, it can be very slow when the values of k and n grow slightly. We now present a series of derivations that enable the design of an efficient, equivalent algorithm.

In [28], it is shown that when the column subset is constructed in a greedy fashion, the next minimizing column can be found fast. In our case we cannot rely on this fact, since our algorithm does not construct the subset greedily, but rather it runs by iteratively removing one column and replacing it with a new one. We nevertheless use some results proved in [28]. If we have a column subset of A , forming matrix C , the following theorem provides us with a simple expression to identify the best single column to append to matrix C , based on the matrix $E = A - CC^+A$.

Theorem 1. *Let $A \in \mathbb{R}^{m \times n}$. For some $k \in \mathbb{N}, k < \text{rank}(A)$ let $C \in \mathbb{R}^{m \times k}$ be a*

matrix comprised of a subset of the columns of A . Let $E = A - CC^+A$. Then

$$\operatorname{argmin}_i \|A - (C|A_{:i})(C|A_{:i})^+A\|_F = \operatorname{argmax}_i \frac{\|E^T E_{:i}\|_2^2}{\|E_{:i}\|_2^2}$$

This means that if we have computed matrix E , we can easily find the best column to add. If we define $F = E^T E$, we can express this objective as

$$\operatorname{argmax}_i \frac{\|F_{:i}\|_2^2}{F_{ii}} \quad (5.2)$$

In [28], efficient formulae are given for recomputing $\|F_{:i}\|^2$ and F_{ii} once a column has been appended to matrix C . Our algorithm, however, does not build the column subset incrementally, but it iteratively replaces each column by another. Therefore, not only does it require to update these values when a column is added to C , but also when it is removed (or equivalently zeroed out to be replaced by a different one). We now present a series of derivations that allow us to do this efficiently.

The key derivations are the efficient update of the Moore-Penrose pseudoinverse of C , the subsequent efficient update of the residual matrix $E = A - CC^+A$ and the fast update of the numerator and denominator of (5.2) to determine the winning column at each step of the algorithm.

We first point out how to update the Moore-Penrose generalized inverse of C . In [57], efficient formulae for rank-1 updates of this type of matrix are provided. Six different cases are considered, depending on the nature of the vectors in which the update can be decomposed. We employ this result to derive efficient updates in our context. Note that proofs for all original results presented here can be found in appendix A.

Proposition 1. *Let $A \in \mathbb{R}^{m \times n}$. For some $k \in \mathbb{N}, k < \operatorname{rank}(A)$ let $C \in \mathbb{R}^{m \times k}$ be*

a matrix comprised of a subset of the columns of A such that $\text{rank}(C) = k$. Let $\tilde{C} \in \mathbb{R}^{m \times k}$ be the matrix resulting from zeroing-out column i in C (i.e., column i of $\tilde{C}$ is comprised uniquely of zeros). Let $\rho = ((C^+)_{i,:})^T$ (the i -th row of C^+ as a column vector). Then

$$\tilde{C}^+ = C^+ - \|\rho\|_2^{-2} C^+ \rho \rho^T$$

Second, we indicate how to update the residual matrix E when a column is removed.

Proposition 2. Let $\tilde{C} \in \mathbb{R}^{m \times k} = CH_k^i$, $E = A - CC^+A$, $\tilde{E} = A - \tilde{C}\tilde{C}^+A$, $\rho = ((C^+)_{i,:})^T$. Then

$$\tilde{E} = E + C_{:,i} \rho^T A + \|\rho\|_2^{-2} \tilde{C} C^+ \rho \rho^T A$$

We now provide efficient formulae to compute (5.2). We define $F = E^T E$, $\tilde{F} = \tilde{E}^T \tilde{E}$ and the vectors

$$f = (\|F_{:,1}\|_2^2, \dots, \|F_{:,n}\|_2^2)$$

$$g = (F_{11}, \dots, F_{nn})$$

$$\tilde{f} = (\|\tilde{F}_{:,1}\|_2^2, \dots, \|\tilde{F}_{:,n}\|_2^2)$$

$$\tilde{g} = (\tilde{F}_{11}, \dots, \tilde{F}_{nn})$$

Proposition 3. Let $\delta = (\tilde{E}_{:,j})^T \tilde{E}$ and $\gamma = E^T E \delta$. Then

$$\tilde{f} = f + \|\delta\|_2^2 (\delta \circ \delta) \delta_j^{-2} + 2(\gamma \circ \delta) \delta_j^{-1}$$

$$\tilde{g} = g + (\delta \circ \delta) \delta_j^{-1}$$

The objective to find the current best column is

$$\operatorname{argmax}_i \frac{\tilde{f}_i}{\tilde{g}_i}$$

Equivalent derivations yield the update formulae to use when a column is chosen and added to the subset:

$$f = \tilde{f} + \|\delta\|_2^2 (\delta \circ \delta) \delta_j^{-2} - 2(\gamma \circ \delta) \delta_j^{-1}$$

$$g = \tilde{g} - (\delta \circ \delta) \delta_j^{-1}$$

We now give an efficient formula to update C^+ once the winning column of the current iteration is added.

Proposition 4. *Let $C \in \mathbb{R}^{m \times k}$ be the matrix resulting from adding column w of A to $\tilde{C}$ at position i . Let $\omega = \tilde{E}_{:,w}$. Then*

$$C^+ = \tilde{C}^+ - \|\omega\|_2^{-2} (\tilde{C}^+ A_{:,w} \omega^T - e_i \omega^T)$$

Finally, since in this case we have added a column to the subset, we can employ the result proved in lemma 2 of [28] to update E .

$$E = \tilde{E} - \omega \omega^T \tilde{E} \|\omega\|_2^{-2} \tag{5.3}$$

where $\omega = \tilde{E}_{:,w}$.

These derivations enable the design of `EfficientLocalSearch` (Algorithm 2), equivalent to `LocalSearch` but much more efficient.

Despite the efficiency of `EfficientLocalSearch`, two problems can still arise.

First, if m is sufficiently large the running time can be prohibitive and the matrix might not fit in main memory. Second, the initial computation of f and g (for (5.2)) requires the computation of the product $E^T E \in \mathbb{R}^{n \times n}$. If n is too large, this imposes significant memory and computational requirements. We now show how to address these two potential problems.

Dealing with huge matrices

We now prove two results that allow us to deal with matrices comprised of a large number of rows or a large number of columns. The next theorem implies that in order to find an approximation to the solution of the CSSP on a $m \times n$ matrix, we can operate on an $n \times n$ matrix and obtain the same result. An example of the benefits brought by this result is the following. We ran our algorithm on a commodity four-core PC with 8GB of RAM on a dataset of 11,000 rows (the USPS dataset described in section 7.1). Without using the result from Theorem 2 the algorithm took over 6 seconds. Using it, the running time was reduced to 0.28 seconds.

This result can also be beneficial when dealing with matrices with a huge number of rows that do not fit in main memory. These can be preprocessed fast taking advantage of distributed computing platforms such as Apache Spark [25], which provide efficient parallelized implementations of the singular value decomposition. The resulting matrix is compact and can therefore be processed by our algorithm in the main memory of a modest machine.

Theorem 2. *Let $A \in \mathbb{R}^{m \times n}$, $m > n$, $\text{rank}(A) = n$ and let $U^T A V = \Sigma$ be its singular value decomposition. Let $\mathcal{S}^{n \times k}$ be the set of $n \times k$ column sampling matrices (permuted*

and column-truncated identity matrices). Then

$$\begin{aligned}
& \operatorname{argmin}_{S \in \mathcal{S}^{n \times k}} \|A - AS(AS)^+A\|_F^2 \\
&= \operatorname{argmin}_{S \in \mathcal{S}^{n \times k}} \|\Sigma V^T - \Sigma V^T S(\Sigma V^T S)^+ \Sigma V^T\|_F^2
\end{aligned} \tag{5.4}$$

Applying Theorem 2 In order to apply this theorem, if the input matrix A is such that $m \gg n$, then A should be replaced with ΣV^T on entry.

The following theorem implies that we do not need to compute the product $E^T E$ explicitly, which could impose prohibitive memory requirements for large values of n , i.e. large numbers of columns.

Theorem 3. *For some matrix A and a column subset C , let $E = A - CC^+A$ and let $E = U\Sigma V^T$ be its singular value decomposition. Let $f_i = \|E^T E_{:i}\|_2^2$, $g_i = \|E_{:i}\|_2^2$. Then*

$$\begin{aligned}
f_i &= \|(\Sigma^2 V^T)_{:i}\|_2^2 \\
g_i &= \|(\Sigma V^T)_{:i}\|_2^2
\end{aligned}$$

This result is especially advantageous when the rank of our data matrix is much smaller than the number of columns, a circumstance that arises frequently in certain domains such as bioinformatics or image processing.

Applying Theorem 3 This theorem can be applied to our algorithm as follows. If the input matrix A is such that $m \ll n$, $E = A - CC^+A$ and $E = U\Sigma V^T$ is the singular value decomposition of E , then lines 3 and 4 of `EfficientLocalSearch` should be replaced by

$$F \leftarrow \Sigma^2 V^T$$

$$f_i \leftarrow \|(\Sigma^2 V^T)_{:i}\|_2^2 \text{ for } i = 1 \dots n$$

$$g_i \leftarrow \|(\Sigma V^T)_{:i}\|_2^2 \text{ for } i = 1 \dots n$$

Complexity analysis

The complexity of **EfficientLocalSearch** is dominated by the initial requirement to compute either $A^T A$ or the singular value decomposition of A . Given efficient and parallelized implementations of these operations, however, the running time in practice is generally more heavily impacted by the complexity of the loop. The main operations to be performed are the updates of C^+ and E (when removing and when adding a column) and the computation of δ and γ . The computation of the pseudoinverse generally takes $O(mk^2)$ time. However, taking advantage of propositions 1 and 4 this can be done in $O(mk)$ time, while E is updated twice in $O(mn)$ time. On the other hand, δ and γ can be computed in $O(mn)$ time, and f and g in $O(n)$. We therefore have that the necessary operations in the loop take $O(2mk + 3mn + 4n) = O(mn)$ time (note that k is always smaller than the rank of the input matrix). Since these operations are run for each column of the candidate subset, each iteration requires $O(mnk)$ operations. If $m \gg n$, using Theorem 2 the complexity of the loop can be reduced to $O(n^2k)$ at the cost of an $O(mn^2)$ operation at the beginning of the algorithm. Since existing implementations of the SVD are very efficient and often run in parallel, this can yield significant benefits in practice. The product $E^T E$ at the beginning takes $O(mn^2)$, but if $n > m$ this can be reduced to $O(m^2n)$ using Theorem 3.

5.2 Analysis

In this section we present a formal analysis of the behaviour of the proposed algorithm.

The following lemma, which provides a lower bound for the increment at each iteration of the greedy algorithm, is crucial in the derivation of our results. Note that this lemma is stated in terms of $\|CC^+A\|_F^2$, whose maximization is equivalent to minimizing $\|A - CC^+A\|_F^2$.

Lemma 1. *[1] Let B be the solution to the CSSP in Frobenius norm and let $\tilde{C}$ be a column subset of the target matrix A such that $\|\tilde{C}\tilde{C}^+A\|_F \leq \|BB^+A\|_F$. Let us assume that the columns of B are normalized to unit norm. Then there exists a column v in B such that if $C = (\tilde{C}|v)$*

$$\|CC^+A\|_F^2 - \|\tilde{C}\tilde{C}^+A\|_F^2 \geq \sigma_k^2(B) \frac{(\|BB^+A\|_F^2 - \|\tilde{C}\tilde{C}^+A\|_F^2)^2}{4k\|BB^+A\|_F^2}$$

We can now derive a first lower bound for the norm recovered by local search. Here we consider that the columns of B are normalized so that we can use lemma 1, but note that scaling this matrix does not affect the solution.

Lemma 2. *Let C be the matrix output by Algorithm 1, and let B be the solution to the CSSP for some input matrix A and some integer $k \geq 2$. Let the columns of B be normalized to unit norm. Then*

$$\|CC^+A\|_F^2 \geq \sigma_k^2(B) \frac{\sigma_k^4(BB^+A)}{4\|BB^+A\|_F^2} + \sum_{i=n-k+2}^n \sigma_i^2(A)$$

We now show that a locally optimal solution *with any number of columns greater than 1* enjoys multiplicative approximation guarantees with respect to an optimal

solution.

Lemma 3. *Let $C \in \mathbb{R}^{m \times k'}$ be the matrix output by Algorithm 1 for $k' \geq 2$, and let B be the solution to the CSSP for some input matrix A and some integer $k \geq 2$. Let the columns of B be normalized to unit norm. Then*

$$\|CC^+A\|_F^2 \geq O\left(\frac{\sigma_k^2(B)}{k}\right) \|BB^+A\|_F^2$$

Algorithm 2 Efficient implementation of local search for column subset selection

---

```

1: procedure EfficientLocalSearch( $A, k$ )
2:    $R \leftarrow \text{uniSampleWithoutReplacement}(1 \dots n, k)$ 
                                      $\triangleright$  Optionally apply theorem 2 or 3

3:    $F \leftarrow E^T E$ 
4:    $f_i \leftarrow \|F_{:i}\|_2^2$ ;  $g_i \leftarrow F_{ii}$  for  $i = 1 \dots n$ 
5:    $C \leftarrow A_R$ 
6:   while not converged do
7:     for  $i = 1, \dots, k$  do
8:        $j \leftarrow R[i]$ 
9:        $\tilde{C} \leftarrow C H_k^i$   $\triangleright$  Zero out column  $i$ 
10:       $\tilde{C}^+ \leftarrow C^+ - \|\rho\|_2^{-2} C^+ \rho \rho^T$   $\triangleright$  Prop. 1
11:       $S_1 \leftarrow C_{:i} \rho^T A$ 
12:       $S_2 \leftarrow \|\rho\|_2^{-2} \tilde{C} C^+ \rho \rho^T A$ 
13:       $\tilde{E} \leftarrow E + S_1 + S_2$   $\triangleright$  Prop. 2
14:       $\delta \leftarrow \tilde{E}_{:j}^T \tilde{E}$ ;  $\gamma \leftarrow E^T E \delta$ 
15:       $\tilde{f} \leftarrow f + \|\delta\|_2^2 (\delta \circ \delta) \delta_j^{-2} + 2(\gamma \circ \delta) \delta_j^{-1}$ 
16:       $\tilde{g} \leftarrow g + (\delta \circ \delta) \delta_j^{-1}$   $\triangleright$  Prop. 3
17:       $w \leftarrow \text{argmax}_h \tilde{f}_h / \tilde{g}_h$ 
18:       $\delta \leftarrow \tilde{E}_{:w}^T \tilde{E}$ ;  $\gamma \leftarrow \tilde{E}^T \tilde{E} \delta$ 
19:       $f \leftarrow \tilde{f} + \|\delta\|_2^2 (\delta \circ \delta) \delta_w^{-2} - 2(\gamma \circ \delta) \delta_w^{-1}$ 
20:       $g \leftarrow \tilde{g} - (\delta \circ \delta) \delta_w^{-1}$ 
21:       $C^+ \leftarrow \tilde{C}^+ - \|\omega\|_2^{-2} (\tilde{C}^+ A_{:w} \omega^T - e_i \omega^T)$ 
                                      $\triangleright$  Prop. 4

22:       $E \leftarrow \tilde{E} - \omega \omega^T \tilde{E} \|\omega\|_2^{-2}$   $\triangleright$  (5.3)
23:      checkConvergence()
24:       $R[i] \leftarrow w$ 
25:       $C \leftarrow A_R$ 
26:     end for
27:   end while
28:   Output  $R$ 
29: end procedure

```

Chapter 6

Regularization

So far we have examined column subset selection from a purely algorithmic perspective, that is, focusing on the complexity of the involved algorithms and the quality of the obtained approximations. However, additional considerations are required when using these tools in practice. To illustrate this we discuss the concept of regularization.

In all computational practices involving data analysis, regularization is essentially akin to the notion of Ockham’s razor. This principle is often attributed to 14th century logician William of Ockham¹, and since it was first written it has taken many forms. We shall state it here in a way that is most frequently employed in scientific endeavours.

In the face of two competing theories that make the exact same predictions, one should favor the simpler one

---

¹Ockham is however not the first to have stated the principle. For a discussion on the matter, see <https://www.britannica.com/topic/Occams-razor>

Figure 6.1: Some observations

The way this translates to the matters we currently concern ourselves with is best illustrated by an example. Imagine we are to build a model to make predictions about a certain phenomenon. We are given m measurements of the phenomenon in question. Each these measurements consists of a pair $(x_i, y_i) \in \mathbb{R}^2$, where x_i is the value of an independent variable and y_i is the observed value of the phenomenon. We want to be able to predict the value of y_i for future observations of x_i . Figure 6.1 provides a possible realization of this setting. The data have been generated by drawing each x_i from a uniform distribution between 0 and 1, and then setting $y_i = 1.2x_i + 0.8x_i^2 + \epsilon$, where ϵ is normally distributed with 0 mean and a variance of 0.1.

A sensible first approach is to seek a linear function that relates the two variables, and then measure the approximation error as the squared difference between the prediction and the observation. This is known as least-squares estimation, or linear regression, and can be formulated as follows. If $X = (x_1, \dots, x_m)^T$ and $y = (y_1, \dots, y_m)$, we seek $w \in \mathbb{R}$ so as to minimize

$$\|Xw - y\|_2^2 = \sum_{i=1}^m (wx_i - y_i)^2$$

Figure 6.2: Linear fit

The minimizer $\hat{w} = X^+y$ will result in the linear function depicted in figure 6.2. It might be the case that the resulting fit is unsatisfactory. We can nevertheless obtain a better fit if we enrich our matrix X with more variables. A standard way to do this is to replace each observation x_i with a d -degree polynomial in x_i :

$$p(x) = \sum_{j=1}^d \alpha_j x_i^j$$

We can now build an extended matrix X_d , where row i equals $(x_i, x_i^2, \dots, x_i^d)$. our job now is to estimate the vector $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_d)$ so as to minimize

$$\|X_d \alpha - y\|_2^2 = \sum_{i=1}^m \left(\sum_j \alpha_j x_i^j - y_i \right)^2$$

The obtained fit with $d = 10$, depicted in figure 6.3, is indeed better in the sense that the sum of squared errors decreases. However, if we attempt to use this model to make predictions outside the range of our observations, the result is preposterous. By considering high-order interactions, we have bestowed the model with significant “wobble room”, enabling it to model noisy behavior. In other words, our model is

Figure 6.3: 10-th degree polynomial fit

too complex.

This problem can be dealt with by regularizing the model. Roughly, regularization consists in somehow favoring simpler models—in some sense—at the expense of goodness-of-fit. It is therefore a specific way to put the Ockham’s razor principle to work.

In linear regression, one way to accomplish this is with the addition of a penalty term to the objective function.

$$\min_{\alpha} \|X_d \alpha - y\|_2^2 + \lambda \|\alpha\|_2^2$$

where $\lambda \in \mathbb{R}$ is chosen to tune the amount of regularization. This encourages the choice of “small” coefficients, neglecting high-order interactions if their contributions to the minimization of the error is not significant. A regularized fit of the previously discussed data with $\lambda = 1$ is shown in figure 6.4. Even though the data are explained with lower precision, the quadratic nature of the true model is recovered more faithfully.

Regularization methods as the one depicted above are instrumental to successful

Figure 6.4: Regularized polynomial fit

data processing in many domains. It is closely connected to Bayesian data analysis [33, 14] and is central to the success of machine learning methods [60, 72, 50]. We therefore argue that the application of column subset selection methods in practice is unlikely to be successful without the introduction of some form of regularization. In the rest of the chapter we propose an approach to accomplish this.

6.1 Regularization in column subset selection

An important drawback of low-rank approximation via column subset selection is that the matrix CC^+A is unregularized, and therefore the coefficient matrix C^+A can grow unbounded without incurring any penalty. This means that if the matrix A contains contingent quirks (e.g. noisy measurements), any algorithm might yield spuriously expressive column subsets, which might later perform badly on new data.

Consider the following instance of GCSSP.

$$A = \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 1 & 0 \\ 0 & \epsilon & 1 \end{pmatrix}$$

Figure 6.5: GCSSP instance

and $k = 2$. Denote the columns of B as v_1, v_2, v_3 . Figure 6.5 illustrates this example. It is immediately apparent that matrix A can be expressed naturally in terms of columns v_1 and v_3 . However, in the objective function, the column subset $\{v_1, v_2\}$ attains the same value — A is approximated perfectly — even though $\{v_1, v_3\}$ seems somehow a better choice. The difference between the two choices is revealed by the coefficients of $B_S^+ A$, where S is either $\{v_1, v_2\}$ or $\{v_1, v_3\}$. We have

$$B_{\{v_1, v_2\}}^+ A = \begin{pmatrix} -\frac{1}{2}(\frac{1}{\epsilon} - 1) \\ \frac{1}{2\epsilon} \end{pmatrix} \quad B_{\{v_1, v_3\}}^+ A = \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix}$$

If ϵ is small enough, the coefficients of $B_{\{v_1, v_2\}}^+ A$ become extremely sensitive to small variations. If these data are the product of natural measurements, the approximation of future observations using this model is thus likely to incur significant error, even though the GCSSP objective is indifferent to either column subset. The model obtained using $\{v_1, v_3\}$ is in some sense more in tune with the Ockham's razor

principle, and is in fact likely to be much more robust for making predictions.

Our purpose in this chapter is to endow column subset selection with some form of regularization so that subsets leading to parsimonious approximations are favored. In addition, we want to derive an algorithm that is comparably efficient to the existing greedy method for the CSSP, and correctly optimizes each subproblem corresponding to the greedy approximation.

The main contributions of this section can be summarized as follows:

- We propose a regularized formulation of the column subset selection problem.
- We show that assuming certain conditions are met, there exists an algorithm that correctly computes a greedy step in essentially $O(\min\{mn, n^2\})$ time complexity. In addition, we show that a procedure can be developed to ensure that said conditions are met as we build the set S incrementally, thus allowing us to derive an efficient, correct greedy algorithm for problem.
- We discuss how this formulation can be inadequate for feature selection and propose an alternative objective, as well as an algorithm, to overcome this drawback.
- We offer a lower bound for the error of the proposed problems, which can serve to inform a stopping criterion.

6.2 Regularized greedy column subset selection

In this section we propose a modification of the CSSP objective to enable regularization. The same principles are applicable to the other variants of the problem.

The approach is as follows. If we observe that $CC^+A = C(C^TC)^{-1}C^TA$ (whenever the inverse exists), we can consider Tikhonov regularization for linear regression (ridge regression) and reformulate the CSSP as follows:

Problem 5. *Given a matrix $A \in \mathbb{R}^{m \times n}$, a positive integer $k \leq n$ and $\lambda \in \mathbb{R}$, let $\mathcal{A}_k$ denote the set of $m \times k$ matrices comprised of k columns of A . Find*

$$\operatorname{argmin}_{C \in \mathcal{A}_k} \|A - C(C^TC + \lambda I)^{-1}C^TA\|_F \quad (6.1)$$

This problem is equivalent to the CSSP, but using a regularized approximation of the target matrix. By introducing the term λ , we penalize subsets that would require very large coefficient matrices. We can therefore tune its value to trade off between goodness of fit and model complexity.

The problem of this approach is that the greedy algorithm described in section 4.4.1 is no longer applicable. The reason is that this greedy method, as described by Farahat et al. [28], relies heavily on the fact that CC^+ is a projection. The corresponding term of the regularized formulation, $(C^TC + \lambda I)^{-1}C^T$, ceases to be a projection whenever $\lambda > 0$. Therefore, equations 4.2 and 4.3 no longer hold.

Before we proceed, we will define the following function in order to simplify our notation. For a matrix $A \in \mathbb{R}^{m \times n}$,

$$\begin{aligned} f_A : \mathcal{P}([n]) \times \mathbb{R} &\rightarrow \mathbb{R}^{m \times n} \\ S, \lambda &\mapsto f_A(S, \lambda) = A_S(A_S^T A_S + \lambda I)^{-1} A_S^T A \end{aligned}$$

where $\mathcal{P}([n])$ is the power set of $[n]$.

Our goal in this chapter is to derive an efficient greedy algorithm for problem 5. In essence, we need an algorithm that solves the following sub-problem:

Problem 6. Given a matrix $A \in \mathbb{R}^{m \times n}$ and a set $S \subset [n]$, find

$$\operatorname{argmin}_i \|A - f_A(S \cup \{i\}, \lambda)\|_F^2 \quad (6.2)$$

Given a column subset S , this problem is solved by finding the best addition to said subset. Therefore, a correct greedy algorithm should solve this problem at each iteration. Obviously, we can solve problem 6 simply by inspecting the value of the objective function for all possible choices of i . This method, however, involves considerable computations, as each candidate requires (for $|S| = t$) matrix products of complexity $O(mnt)$ and $O(nt^2)$, as well as a matrix inversion of complexity $O(t^3)$, in addition to the matrix subtraction and the computation of the norm. In sum, each iteration of the resulting algorithm would take $O(\max\{mn^2t, n^2t^2\})$ operations. The algorithm we propose here solves problem 6 in $O(\min\{mp, n^2\})$ time, where $p = \max\{m, t\}$.

6.2.1 Algorithms for regularized greedy column subset selection

Let us assume we are running a greedy algorithm that has so far completed t iterations, i.e., we have a set S of t elements. We want to solve problem 6, that is,

$$\operatorname{argmin}_i \|A - f_A(S \cup \{i\}, \lambda)\|_F^2$$

We can of course do this by inspecting the value of this objective for all possible choices. However, this would be computationally costly. If we define $C_i = A_{S \cup \{i\}}$, in order to do this we would need to compute the inverse of $C_i^T C_i + \lambda I$ for all choices of i . We can start our derivation by attempting to circumvent these matrix inversions,

which we can accomplish as follows. We define

$$\hat{A} = \begin{pmatrix} A \\ \sqrt{\lambda}I \end{pmatrix}$$

and $\hat{C}_i = \hat{A}_{S \cup \{i\}}$.

If we observe that

$$C_i^T C_i + \lambda I = \hat{C}_i^T \hat{C}_i$$

then we can take advantage of the following fact. Let us denote $w = A_{:w}$, $\hat{w} = \hat{A}_{:w}$.

Since $\hat{C}_w = \hat{A}_{S \cup \{w\}} = (C|w)$, then it is well known [56] that

$$(C_w^T C_w + \lambda I)^{-1} = (\hat{C}_w^T \hat{C}_w)^{-1} = \left(\begin{array}{c|c} (C^T C + \lambda I)^{-1} + \frac{v v^T}{\alpha_w} & -\frac{v}{\alpha_w} \\ \hline -\frac{v^T}{\alpha_w} & \frac{1}{\alpha_w} \end{array} \right)$$

where $\alpha_w = \hat{w}^T \hat{w} - \hat{w}^T \hat{C} (C^T C + \lambda I)^{-1} \hat{C}^T \hat{w}$ and $v = (C^T C + \lambda I)^{-1} C^T w$. (Note that $C^T w = \hat{C}^T \hat{w}$ because the extension of w is multiplied by zero).

Relying on this expression for the inverse of $C_w^T C_w + \lambda I$, we can look for a solution to problem 6 that does not need to explicitly compute the value of the objective for all choices of i . For brevity, let us define $A^{(t)} = f_A(S, \lambda)$ and $A^{(t+1)} = f_A(S \cup \{w\}, \lambda)$. First, observe that (detailed derivations can be found in appendix B)

$$\begin{aligned} A^{(t+1)} &= C_w (C_w^T C_w + \lambda I)^{-1} C_w^T A \\ &= A^{(t)} + \frac{1}{\alpha_w} ((A_{:w}^{(t)} - w)(A_{:w}^{(t)} - w)^T) A \end{aligned} \tag{6.3}$$

Let $d_i = (A_{:,i}^{(t)} - A_{:,i}) \in \mathbb{R}^m$. Then the next column choice is yielded by

$$\begin{aligned}
& \operatorname{argmin} \|A - A^{(t+1)}\|_F^2 \\
&= \operatorname{argmin}_i \operatorname{tr}(A^T A) - \operatorname{tr}(A^T A^{(t)}) - \operatorname{tr}\left(\frac{1}{\alpha_i} A^T d_i d_i^T A\right) \\
&\quad - \operatorname{tr}((A^T)^{(t)} A) - \operatorname{tr}\left(\frac{1}{\alpha_i} A^T d_i d_i^T A\right) \\
&\quad + \operatorname{tr}((A^T)^{(t)} A^{(t)}) + \operatorname{tr}\left(\frac{1}{\alpha_i} (A^T)^{(t)} d_i d_i^T A\right) + \operatorname{tr}\left(\frac{1}{\alpha_i} A^T d_i d_i^T A^{(t)}\right) \\
&\quad + \operatorname{tr}\left(\frac{1}{\alpha_i^2} A^T d_i d_i^T d_i d_i^T A\right) \tag{6.4}
\end{aligned}$$

We define $x_i = A^T d_i \in \mathbb{R}^n$, $\tilde{x}_i = (A^{(t)})^T d_i \in \mathbb{R}^n$. Dropping irrelevant constants from the previous equality,

$$\begin{aligned}
& \operatorname{argmin} \|A - A^{(t+1)}\|_F^2 \\
&= -2\operatorname{tr}\left(\frac{1}{\alpha_i} A^T d_i d_i^T A\right) + 2\operatorname{tr}\left(\frac{1}{\alpha_i} (A^T)^{(t)} d_i d_i^T A\right) + \operatorname{tr}\left(\frac{1}{\alpha_i^2} A^T d_i d_i^T d_i d_i^T A\right) \\
&= \operatorname{argmin}_i \frac{2}{\alpha_i} \tilde{x}_i^T x_i - \frac{2}{\alpha_i} \|x_i\|_2^2 + \frac{1}{\alpha_i^2} \|d_i\|_2^2 \|x_i\|_2^2 \\
&= \operatorname{argmin}_i \frac{2}{\alpha_i} \tilde{x}_i^T x_i + \left(\frac{-2}{\alpha_i} + \frac{1}{\alpha_i^2} (\tilde{x}_{ii} - x_{ii})\right) x_i^T x_i + \lambda \tag{6.5}
\end{aligned}$$

Equality (6.5) provides a surrogate of problem 6, i.e., the problem we need to solve at each iteration in order to implement a correct greedy algorithm for problem 5. In particular, equality (6.5) shows that the solution to problem 6 can be found as a function of $x_i, \tilde{x}_i$ and α_i for $i = 1, \dots, n$. Our concern now is to develop a procedure to find the value of these variables at each iteration without incurring too much computational cost.

We have

$$\begin{aligned}\tilde{x}_i^T x_i &= d_i^T A^{(t)} A^T d_i \\ x_i^T x_i &= d_i^T A A^T d_i\end{aligned}$$

For notational convenience, we define the matrices X , $\tilde{X}$ and D , whose columns are the vectors x_i , $\tilde{x}_i$ and d_i , $i = 1, \dots, n$ respectively. First, observe from equality (6.3) that

$$A^{(t+1)} = A^{(t)} + \frac{1}{\alpha_w} d_w d_w^T A$$

Hence,

$$A^T A^{(t+1)} = A^T A^{(t)} + \frac{1}{\alpha_w} x_w x_w^T \quad (6.6)$$

And

$$(A^T A)^{(t+1)} = (A^T A)^{(t)} + \frac{1}{\alpha_w} \tilde{x}_w x_w^T + \frac{1}{\alpha_w} x_w \tilde{x}_w^T + \frac{1}{\alpha_w^2} x_w d_w^T d_w x_w^T \quad (6.7)$$

So finally,

$$\begin{aligned}X^{(t+1)} &= A^T D^{(t+1)} \\ &= A^T D^{(t)} + A^T A + \frac{1}{\alpha_w} x_w x_w^T - A^T A \\ &= X^{(t)} + \frac{1}{\alpha_w} x_w x_w^T = X^{(0)} + \sum_{i=0}^t \left(\frac{1}{\alpha_w} x_w x_w^T \right)^{(i)}\end{aligned} \quad (6.8)$$

And

$$\begin{aligned}
\tilde{X}^{(t+1)} &= (A^T D)^{(t+1)} \\
&= (A^T D)^{(t)} + \frac{1}{\alpha_w} \tilde{x}_w x_w^T + \frac{1}{\alpha_w} x_w \tilde{x}_w^T + \frac{1}{\alpha_w^2} x_w d_w^T d_w x_w^T - \frac{1}{\alpha_w} x_w x_w^T \\
&= \sum_{i=0}^t \left(\frac{1}{\alpha_w} \tilde{x}_w x_w^T + \frac{1}{\alpha_w} x_w \tilde{x}_w^T + \frac{1}{\alpha_w^2} x_w d_w^T d_w x_w^T - \frac{1}{\alpha_w} x_w x_w^T \right)^{(i)} \quad (6.9)
\end{aligned}$$

Expressions 6.8 and 6.9 allow us to compute the variables involved in problem 6.5 efficiently as more columns are greedily added to the final set. Specifically, it is easy to see that

$$\begin{aligned}
(x_i^T \tilde{x}_i)^{(t+1)} &= x_i^T \tilde{x}_i \\
&+ \left(x_i + \frac{1}{\alpha_w} x_w x_{wi} \right)^T \left(\tilde{x}_i + \frac{1}{\alpha_w} \tilde{x}_w x_w^T + \frac{1}{\alpha_w} x_w \tilde{x}_w^T + \frac{1}{\alpha_w^2} x_w x_w^T (\tilde{x}_{ww} - x_{ww}) - \frac{1}{\alpha_w} x_w x_w^T \right) \\
&= x_i^T \tilde{x}_i + \frac{1}{\alpha_w} (x_i^T \tilde{x}_w x_{wi} + x_i^T x_w (\tilde{x}_{wi} - x_{wi}) + \tilde{x}_i^T x_w x_{wi}) \\
&+ \frac{1}{\alpha_w^2} (x_i^T x_w x_{wi} (\tilde{x}_{ww} - x_{ww}) + x_w^T x_w x_{wi} \tilde{x}_{wi} + x_w^T \tilde{x}_w (x_{wi})^2 - x_w^T x_w (x_{wi})^2) \\
&+ \frac{1}{\alpha_w^3} x_w^T x_w (x_{wi})^2 (\tilde{x}_{ww} - x_{ww}) \quad (6.10)
\end{aligned}$$

$$(x_i^T x_i)^{(t+1)} = x_i^T x_i + \frac{2x_{wi}}{\alpha_w} x_i^T x_w + \left(\frac{x_{wi}}{\alpha_w} \right)^2 x_w^T x_w \quad (6.11)$$

All elements on the r.h.s. of equalities (6.11) and (6.10) correspond to iteration t . The superindex has been omitted for clarity. Furthermore, in equation (6.11) we consider the values for the already chosen indices to be irrelevant in order to simplify the expression.

The inner products involved in these updates can also be computed efficiently by making use of the recursive formulae in 6.8 and 6.9 as follows:

$$\begin{aligned}
x_i^T x_w &:= \left(x_i^{(0)}\right)^T (x_w)^{(t)} + \left(\sum_{j=0}^{t-1} \left(\frac{1}{\alpha_w} x_w x_{wi}\right)^{(j)}\right)^T (x_w)^{(t)} \\
x_i^T \tilde{x}_w &:= \left(x_i^{(0)}\right)^T (\tilde{x}_w)^{(t)} + \left(\sum_{j=0}^{t-1} \left(\frac{1}{\alpha_w} x_w x_{wi}\right)^{(j)}\right)^T (\tilde{x}_w)^{(t)} \\
\tilde{x}_i^T x_w &:= \sum_{j=0}^{t-1} \left(\frac{1}{\alpha_w} \tilde{x}_w x_{wi} + \frac{1}{\alpha_w} x_w \tilde{x}_{wi} + \frac{\tilde{x}_{ww} - x_{ww}}{\alpha_w^2} x_w x_{wi} - \frac{1}{\alpha_w} x_w x_{wi}\right)^{(j)} (x_w)^{(t)}
\end{aligned}$$

Based on the equalities presented above, we can state and prove our main result regarding the existence of an efficient algorithm for problem 6. The proof is constructive and provides the necessary equalities for implementing the algorithm.

Theorem 4. *Given a matrix $A \in \mathbb{R}^{m \times n}$, $\lambda \in \mathbb{R}$ and a set S of cardinality $t \leq n$, let us assume the following values, as defined above, are known:*

1. $X^{(0)} = -A^T A$
2. $x_w^{(i)}, \tilde{x}_w^{(i)}, \alpha_w^{(i)}, i = 1, \dots, t$

Then there exists an algorithm that solves problem 6 in $O(\min\{np, n^2\})$ time, where $p = \max\{m, t\}$.

By theorem 4, if we store the value of $x_w, \tilde{x}_w$ and α_w at each iteration, we can efficiently find the best column addition for the regularized column subset selection formulation. This allows us to derive a greedy algorithm for problem 5 (Algorithm 3).

Algorithm 3 Greedy algorithm for regularized column subset selection

---

```

1: procedure RegGreedy( $A, k, \lambda$ )
2:    $X := -A^T A$ 
3:   Compute  $x_i^T x_i, i = 1, \dots, n$ 
4:   Choose the first column,  $w := \operatorname{argmin}_i \frac{-2}{\alpha_i} - \frac{x_{ii}}{\alpha_i^2} x_i^T x_i + \lambda$ 
5:    $S := \{w\}$ 
6:   for  $t = 1 \dots k - 1$  do
7:      $x_w := X_{:w} + \sum_{j=0}^{t-1} \left( \frac{1}{\alpha_w} x_w x_{ww}^T \right)^{(j)}$ 
8:      $\tilde{x}_w := \sum_{j=0}^{t-1} \left( \frac{x_{ww}}{\alpha_w} \tilde{x}_w + \frac{\tilde{x}_{ww}}{\alpha_w} x_w + \frac{\tilde{x}_{ww} - x_{ww}}{\alpha_w^2} x_w x_{ww} - \frac{x_{ww}}{\alpha_w} x_w \right)^{(j)}$ 
9:      $\beta := X x_w + W W_\alpha^T x_w$ 
10:     $\gamma := X \tilde{x}_w + W W_\alpha^T \tilde{x}_w$ 
11:     $\delta := W \tilde{W}_\alpha^T x_w + \tilde{W} W_\alpha^T x_w + \alpha^{-1} (\tilde{x}_w - x_w) W W_\alpha^T x_w + W W_\alpha^T x_w$ 
12:     $\omega := \omega + \alpha_w^{-1} (\gamma \circ x_w + \beta \circ (\tilde{x}_w - x_w) + \delta \circ x_w)$ 
13:     $\quad + \alpha_w^{-2} (\beta \circ x_w (\tilde{x}_{ww} - x_{ww}) + \beta_w x_w \circ \tilde{x}_w + (\gamma_w - \beta_w) x_w^2)$ 
14:     $\quad + \alpha_w^{-3} \beta_w x_w \circ x_w (\tilde{x}_{ww} - x_{ww})$ 
15:     $\psi := \psi + \frac{2}{\alpha_w} x_w \circ \beta + \frac{\beta_w}{\alpha_w^2} x_w \circ x_w$ 
16:    for  $i = 1, \dots, n$ 
17:       $x_{ii} := x_{ii} + \frac{1}{\alpha_w} x_{wi}^2$ 
18:       $\tilde{x}_{ii} := \tilde{x}_{ii} + \frac{2}{\alpha_w} \tilde{x}_{wi} x_{wi} + \frac{\tilde{x}_{ii} - x_{ii}}{\alpha_w^2} x_{wi}^2 - \frac{1}{\alpha_w} x_{wi}$ 
19:       $\alpha_i := \lambda - x_{ii}$ 
20:       $w := \operatorname{argmin}_i \frac{2}{\alpha_i} \omega_i + \left( \frac{-2}{\alpha_i} + \frac{1}{\alpha_i^2} (\tilde{x}_{ii} - x_{ii}) \right) \psi_i + \lambda$ 
21:       $S := S \cup \{w\}$ 
22:    end for
23:  end procedure

```

An appropriate formulation for feature selection

The purpose of this algorithm is that of selecting a few variables and approximating the rest. In the context of practical applications of feature selection, we can consider that the chosen variables are available and do not need to be approximated. However, the penalty of the regularized formulation causes the chosen variables to be imperfectly estimated. This means that in optimizing the objective in problem 6, we are taking into account an error that should not be made in reality, thus providing a potentially mistaken choice.

This can be illustrated with an example. Consider the following matrix.

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

and the set $S = \{1, 2\}$. The solution to problem 6 is given by adding column 4 to S . However, if we don't consider the error made in the approximation of the chosen columns, the best choice is column 3. As stated above, in a feature selection setting we would not want to consider the approximation error of the chosen columns, as we can assume the corresponding variables to be available.

For this reason, we propose the following alternative problem formulation.

Problem 7. *Given a matrix $A \in \mathbb{R}^{m \times n}$ and a positive integer k smaller than the rank of A , and defining $\bar{S} = [n] \setminus S$, find*

$$\operatorname{argmin}_{S, |S|=k} \|A_{\bar{S}} - A_S(A_S^T A_S + \lambda I)^{-1} A_S^T A_{\bar{S}}\|_F^2 \quad (6.12)$$

This is equivalent to problem 5, but it discards the error made in approximating the chosen variables. Fortunately, the proposed algorithm can be easily modified to greedily optimize this objective. We show how by induction on t .

At iteration t , consider equation (6.3).

$$A^{(t+1)} = A^{(t)} + \frac{1}{\alpha_w} d_w x^T$$

If we assume that the chosen columns (those in S) are approximated exactly in $A^{(t)}$, i.e. $A_S = A_S^{(t)}$, these need not be modified. Therefore, we can simply set the corresponding positions of x to zero, and these columns will not be altered. For the next iteration, however, we are adding a new column to S , which we can denote w . Thus, we need that $A_{S \cup \{w\}} = A_{S \cup \{w\}}^{(t+1)}$.

We have $d_w = A_{:w}^{(t)} - w$. Therefore, if $x_w = -\alpha_w$,

$$A_{:w}^{(t)} + \frac{1}{\alpha_w} d_w x_{w,w} = A_{:w}^{(t)} - d_w = w$$

That is, by setting $x_S = 0$ and $x_w = -\alpha_w$ we ensure that the chosen columns are considered to be perfectly approximated. If we set $x_w^{(0)} = -\alpha_w^{(0)}$, then we ensure that the first chosen column is perfectly approximated, providing the basis for our inductive argument.

By defining $(x^*)^{(t)}$ to be equal to $x^{(t)}$ but with the previously described replacements, we can easily modify our algorithm to ensure that the column chosen at each iteration is the one that greedily optimizes problem 7. We now detail the necessary modifications:

1. $x_w^* := x_w$; $x_{wi}^* = 0, i \in S$; $x_{ww}^* = -\alpha_w$
2. $\tilde{x}_w := \sum_{i=0}^t \left(\frac{1}{\alpha_w} \tilde{x}_w x_w^T + \frac{1}{\alpha_w} x_w^* \tilde{x}_w^T + \frac{1}{\alpha_w^2} x_w^* d_w^T d_w x_w^T - \frac{1}{\alpha_w} x_w^* x_w^T \right)^{(i)}$

$$3. \ x_i^T x_w^* := \left(x_i^{(0)}\right)^T (\tilde{x}_w^*)^{(t)} + \left(\sum_{i=0}^{t-1} \left(\frac{1}{\alpha_w} x_w x_w^T\right)^{(i)}\right)^T (\tilde{x}_w^*)^{(t)}$$

$$4. \ \tilde{x}_i^T x_w := \sum_{i=0}^{t-1} \left(\frac{1}{\alpha_w} \tilde{x}_w x_w^T + \frac{1}{\alpha_w} x_w \tilde{x}_w^T + \frac{1}{\alpha_w^2} x_w d_w^T d_w x_w^T - \frac{1}{\alpha_w} x_w x_w^T\right)^{(i)} (x_w^*)^{(t)}$$

Equality (6.10):

$$\begin{aligned} (x_i^T \tilde{x}_i)^{(t+1)} &= x_i^T \tilde{x}_i \\ &+ \left(x_i + \frac{1}{\alpha_w} x_w x_{wi}\right)^T \left(\tilde{x}_i + \frac{1}{\alpha_w} \tilde{x}_w x_w^T + \frac{1}{\alpha_w} x_w \tilde{x}_w^T + \frac{1}{\alpha_w^2} x_w x_w^T (\tilde{x}_{ww} - x_{ww}) - \frac{1}{\alpha_w} x_w x_w^T\right) \\ &= x_i^T \tilde{x}_i + \frac{1}{\alpha_w} (x_i^T \tilde{x}_w x_{wi} + x_i^T x_w^* (\tilde{x}_{wi} - x_{wi}) + \tilde{x}_i^T x_w x_{wi}) \\ &+ \frac{1}{\alpha_w^2} (x_i^T x_w^* x_{wi} (\tilde{x}_{ww} - x_{ww}) + x_w^T x_w^* x_{wi} \tilde{x}_{wi} + x_w^T \tilde{x}_w (x_{wi})^2 - x_w^T x_w^* (x_{wi})^2) \\ &+ \frac{1}{\alpha_w^3} x_w^T x_w^* (x_{wi})^2 (\tilde{x}_{ww} - x_{ww}) \end{aligned}$$

Equality (6.11):

$$\begin{aligned} (x_i^T x_i^*)^{(t+1)} & \tag{6.13} \\ &= x_i^T x_i^* + \frac{2x_{wi}^*}{\alpha_w} x_i^T \tilde{x}_w + \left(\frac{x_{wi}^*}{\alpha_w}\right)^2 x_w^T x_w - \left(\frac{x_{wi}}{\alpha_w} x_{wS}^T x_{wS}\right)^2 - (x_{wi} + x_{wi} \frac{x_{ww}}{\alpha_w})^2 \end{aligned}$$

Choosing the value of k

In order to decide when to stop the algorithm, we can monitor the loss of the approximation at each iteration, i.e. $\|A - A^{(t)}\|_F^2$. Computing this quantity at each iteration, however, can be costly. Fortunately, we can take advantage of the variables involved in the proposed algorithm to efficiently track the exact value of the loss.

By equation (6.3) we can easily see that

$$\begin{aligned}
\|A - A^{(t)}\|_F^2 &= \|A - A^{(t-1)} - \frac{1}{\alpha_w} d_w x_w^T\|_F^2 \\
&= \left(A - A^{(t-1)} - \frac{1}{\alpha_w} d_w x_w^T \right)^T \left(A - A^{(t-1)} - \frac{1}{\alpha_w} d_w x_w^T \right) \\
&= \|A - A^{(t-1)}\|_F^2 - \frac{2}{\alpha_w} x_w^T x_w + \frac{2}{\alpha_w} \tilde{x}_w^T x_w + \frac{\tilde{x}_{ww} - x_{ww}}{\alpha_w} x_w^T x_w
\end{aligned}$$

For problem 7, this equality becomes

$$\|A - A^{(t)}\|_F^2 = \|A - A^{(t-1)}\|_F^2 - \frac{2}{\alpha_w} x_w^T x_w^* + \frac{2}{\alpha_w} \tilde{x}_w^T x_w^* + \frac{\tilde{x}_{ww} - x_{ww}}{\alpha_w} (x_w^*)^T x_w^*$$

where x^* is defined as in the previous section.

This means that we can compute the loss at each iteration as a function of its previous value and some readily available variables. The loss before the first iteration is simply the squared norm of the data matrix, which can be computed as $\text{tr} A^T A = -\text{tr} X$.

In the next section we discuss how this quantity can be used to determine when to stop iterating.

Lower bound for the error

In the conventional formulation of the CSSP, the approximation error eventually reaches zero as we add columns to the basis subset. However, the introduction of the regularizing term bars the approximation from being perfect. For this reason, even if we can track the loss as detailed in the previous section, it can be difficult to determine how much of an improvement can still be made. In other words, in problem 3.4 we can evaluate the expressive power of our column subset by checking

how far the error is from zero. On the contrary, in the case of problem 5 we do not know what this ideal lower bound is.

In the case of problem 7, the loss can of course reach zero, but rather artificially (when $k = n$, we simply evaluate the error over zero columns of the matrix). Here we propose a lower bound for the objective function of problem 7 at each iteration, providing insight on the manner in which λ interacts with the approximation error. This bound can be easily extended to problem 5.

Lemma 4. *Given a matrix $A \in \mathbb{R}^{m \times n}$ and a set S such that $|S| = k \leq n$. let σ_i denote the i -th largest singular value of A . Then*

$$\|A_{\bar{S}} - A_S(A_S^T A_S + \lambda I)^{-1} A_S^T A_{\bar{S}}\|_F^2 \geq \lambda^2 \sum_{i=k+1}^n \left(\frac{\sigma_i}{\sigma_i^2 + \lambda} \right)^2 \quad (6.14)$$

Proof. Let us assume $A = A_S$, which corresponds to the best possible approximation we can obtain of the columns of A using a subset of its columns as a basis. If $A = U\Sigma V^T$ is the singular value decomposition of A ,

$$\begin{aligned} A - A(A^T A + \lambda I)^{-1} A^T A &= U\Sigma V^T - U\Sigma(\Sigma^2 + \lambda I)^{-1} \Sigma^2 V^T \\ &= U\Sigma V^T - U \begin{pmatrix} \frac{\sigma_1^3}{\sigma_1^2 + \lambda} & & \\ & \ddots & \\ & & \frac{\sigma_n^3}{\sigma_n^2 + \lambda} \end{pmatrix} V^T \\ &= U \begin{pmatrix} \frac{\sigma_1 \lambda}{\sigma_1^2 + \lambda} & & \\ & \ddots & \\ & & \frac{\sigma_n \lambda}{\sigma_n^2 + \lambda} \end{pmatrix} V^T \end{aligned}$$

The error incurred by approximating A by itself using the regularized formulation is

therefore exactly

$$\|A - f_A([n], \lambda)\|_F^2 = \lambda^2 \sum_{i=1}^n \left(\frac{\sigma_i}{\sigma_i^2 + \lambda} \right)^2$$

Now consider that

$$\begin{aligned} \|A_{\bar{S}} - A_S(A_S^T A_S + \lambda I)^{-1} A_S^T A_{\bar{S}}\|_F^2 &\geq \|A_{\bar{S}} - A(A^T A + \lambda I)^{-1} A^T A_{\bar{S}}\|_F^2 \\ &\geq \lambda^2 \sum_{i=k+1}^n \left(\frac{\sigma_i}{\sigma_i^2 + \lambda} \right)^2 \end{aligned}$$

The second inequality holds because of the interlacing inequalities of the singular values [71], and because $A_{\bar{S}} - A(A^T A + \lambda I)^{-1} A^T A_{\bar{S}}$ is an $m \times (n - k)$ submatrix of $A - f_A([n], \lambda)$. $\square$

To adapt this bound to problem 5, we simply need to extend the summation of the last inequality over all singular values.

This expression for the error vanishes when $\lambda = 0$ and approaches $\sum_i \sigma_i^2$ as $\lambda \rightarrow \infty$. This is of course consistent with the problem formulation. Observe that in the first case, we are measuring the error incurred by approximating A_S using its full span. In the second case, we approximate A with a vanishing matrix, thus making the error equal to $\|A\|_F^2 = \sum_i \sigma_i^2$.

This bound can be used to choose the value of k if the input data set is suitable. If at some point the algorithm attains a value of the objective in problem 7 that is close to this bound, then the present matrix has almost as much representative power as the full column set, thus making the addition of more columns unnecessary.

Chapter 7

Numerical experiments

In order to validate our claims regarding the two approaches for column subset selection discussed in this thesis, we carried out a series of numerical experiments on real and synthetic data. This chapter is devoted to the description of the experimental setup and the analysis of the observed results.

7.1 Local search

In order to evaluate the efficiency and effectiveness of the `EfficientLocalSearch` algorithm, we tested it on various benchmark datasets, comparing it to other well-known algorithms for the column subset selection problem. These experiments are intended not only to test how the algorithm performs, but also to gain insight about its behaviour. For this reason, we have designed a wide array of experiments which we now enumerate briefly.

- Objective function: We measure the values `EfficientLocalSearch` obtains in the objective function of the CSSP, compared to other existing methods.

- **Running time:** We measure how long the initial phase and each iteration of `EfficientLocalSearch` take.
- **Reconstruction capabilities:** We test the capabilities of the model produced by `EfficientLocalSearch` to approximate individual data instances.
- **Benefits of randomized initialization:** We examine if repeated runs of the algorithm using different initial random subsets yield better results.
- **Iterations:** We measure how many iterations `EfficientLocalSearch` takes to finish on average, and how much progress it makes at each step.
- **Approximation error:** We evaluate the number of columns that `EfficientLocalSearch`, as well as other algorithms, need to match the approximation error of the best low rank approximation.
- **Kahan matrix:** We test our algorithm using a Kahan matrix as input, which is designed to evaluate the robustness of rank-revealing QR factorizations.

Algorithms for feature selection such as the ones presented in [42] [79] [11], e.g., do not explicitly target the same objective function as our method, generally yielding much worse linear approximations and therefore not being comparable in this regard.

We therefore employ the following algorithms for comparison:

- **TwoStage:** The two-stage algorithm described in [6]. As suggested by the authors, we draw 40 candidate subsets and pick the best one. We use our own Matlab implementation.
- **Greedy:** The algorithm described in [28]. We employ the Matlab code provided by the authors ¹.

---

¹<http://www.afarahat.com/code>

- **EfficientLocalSearch**: Our method².
- **EfficientLocalSearch-1st**: Our method, limited to one iteration.

We run all the experiments on a machine equipped with a 12-core Xeon CPU and 32 GB of RAM.

7.1.1 Data

We employ the following datasets: the Columbia University Image library (**COIL20**)³ [61]; the **ORL** database of faces⁴ provided by the AT&T Laboratories Cambridge [67]; the Sheffield face database (**UMIST**) provided by the University of Sheffield⁵; the **Binary Alpha** Digits dataset⁶; the Extended Yale Face database B (**YaleB**)⁷ [34, 53]; the **USPS** handwritten digits dataset, taking both the training and the test set⁸; the **Online News** popularity dataset⁹ [29]; the **BlogFeedback** dataset¹⁰ [10] and the **YearPredictionMSD** dataset¹¹ [4]. Some of these datasets are available at the UCI machine learning repository [55]. Table 7.1 provides a summary.

All datasets are standardized to zero mean and unit variance before running the algorithms (i.e. each column is transformed by subtracting its mean from it and then dividing it by its standard deviation).

---

²<https://github.com/brunez/IterFS>

³<http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php>

⁴<http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>

⁵<http://www.sheffield.ac.uk/eee/research/iel/research/face>

⁶<http://www.cs.nyu.edu/~roweis/data.html>

⁷<http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html>

⁸www.imo.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/PublicDatasets

⁹<https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity>

¹⁰<https://archive.ics.uci.edu/ml/datasets/BlogFeedback>

¹¹<https://archive.ics.uci.edu/ml/datasets/YearPredictionMSD>

Table 7.1: Employed datasets.

Dataset	Rows	Columns
COIL20	1440	1024
ORL	400	1024
UMIST	575	10304
BinaryAlpha	1404	320
YaleB	2414	1024
USPS	11100	256
OnlineNews	39644	60
BlogFeedback	52397	281
YearPredictionMSD	515345	90

7.1.2 Objective function

In order to evaluate the accuracy of our algorithm in the objective function we measure its error with respect to the best possible rank- k approximation, which can be obtained using the singular value decomposition and provides a loose lower bound for the CSSP. Specifically, given a data matrix A let A_k be its best rank- k approximation, and let C be the submatrix of A comprised of the k columns chosen by one of the considered algorithms. We define the **error ratio** as

$$\frac{\|A - CC^+A\|_F^2}{\|A - A_k\|_F^2} \quad (7.1)$$

The closer this is to 1, the better. The error ratio is always greater than one, unless a subset of features matches the top left singular vectors, which is of course extremely unlikely. Tables 7.2 and 7.3 report the error ratio for all the algorithms on all datasets, for increasing values of k (the number of chosen features). In this table, TS is **TwoStage**, GFS is **Greedy**, ELS is **EfficientLocalSearch** and ELS-1 is **EfficientLocalSearch** limited to one iteration. Since **TwoStage**, ELS and ELS-1 have a random component, we report the average of 10 runs in Tables 7.2 and 7.3 and

comment on the standard deviation below. These results show how our algorithm consistently outperforms both **TwoStage** and **Greedy**. Remarkably, one iteration of our method is enough to provide better approximation errors than the other two algorithms in most cases.

To avoid excess clutter we do not include the standard deviation in Tables 7.2 and 7.3, although we comment on it briefly. The results of **EfficientLocalSearch** presented a small standard deviation, generally below 10^{-3} . We did observe a higher deviation in the case of the OnlineNews dataset for $k = 30, 40$, reaching values of 0,033 and 0,024 respectively. Our algorithm only showed noticeable instability in the case of the BlogFeedback dataset for values of $k > 100$, reaching values close to 0.1 and even 0.3 in one instance. As expected, the standard deviation shown by **EfficientLocalSearch-1st** is slightly higher, but not significantly in general. The **TwoStage** algorithm shows negligible deviations in most cases. We remark that even though our method presents a certain amount of variety in its results, the produced subset is almost always better than those found by the other algorithms, regardless of the initial random subset.

7.1.3 Running time

We evaluate the running time of our algorithm compared to the other methods. Since our algorithm iterates until no further improvement is achieved, its running times can vary noticeably from one execution to another. Since a single iteration is often enough to outperform the other algorithms (Tables 7.2 and 7.3), we limit our algorithm to just the first iteration. Figure 7.1 shows the results. In the case of **TwoStage**, when $m > n$ we take advantage of Theorem 2 to significantly speed up the second phase. Our algorithm is as fast as **Greedy** in the case of USPS and faster

Table 7.2: Approximation error with respect to the best rank- k approximation.

Dataset	TS	GFS	ELS-1	ELS	TS	GFS	ELS-1	ELS
	k=20				k=40			
COIL20	1.578	1.499	1.488	1.459	1.770	1.619	1.587	1.552
ORL	1.542	1.421	1.419	1.387	1.673	1.548	1.529	1.499
UMIST	1.697	1.469	1.449	1.420	1.783	1.574	1.559	1.515
B.Alpha	1.426	1.427	1.427	1.411	1.600	1.564	1.559	1.540
YaleB	1.672	1.400	1.385	1.356	1.678	1.508	1.500	1.471
USPS	1.479	1.412	1.401	1.392	1.641	1.536	1.539	1.512
OnlineNews	1.357	1.291	1.302	1.290	1.846	1.797	1.746	1.729
BlogFeedback	1.281	1.058	1.054	1.055	1.175	1.092	1.087	1.087
YearPredMSD	1.418	1.350	1.342	1.338	1.653	1.608	1.578	1.562
Dataset	TS	GFS	ELS-1	ELS	TS	GFS	ELS-1	ELS
	k=60				k=80			
COIL20	1.838	1.681	1.660	1.618	1.896	1.744	1.717	1.672
ORL	1.747	1.625	1.601	1.567	1.894	1.702	1.672	1.637
UMIST	1.823	1.637	1.622	1.569	1.908	1.695	1.666	1.617
B.Alpha	1.657	1.637	1.617	1.594	1.645	1.665	1.637	1.620
YaleB	1.817	1.583	1.581	1.546	1.867	1.657	1.642	1.608
USPS	1.670	1.682	1.654	1.628	1.707	1.795	1.752	1.702
BlogFeedback	1.238	1.124	1.119	1.116	1.348	1.153	1.144	1.150
YearPredMSD	1.856	1.865	1.796	1.766	2.004	2.126	1.907	1.891

in BlogFeedback, OnlineNews and YearPredictionMSD. In some cases the higher sensitivity of our method to the value of k is significant, as is the case of Coil20, ORL and YaleB. In other instances, though noticeable, the impact of this sensitivity is not so notable, as is the case of UMIST and BinaryAlpha and our algorithm can still select tens of features in a few tens of milliseconds. In the case of the former, our algorithm is faster for small values of k . The sensitivity to the value of k seems to be particularly noticeable when the input matrix is large in both dimensions.

Table 7.3: Approximation error with respect to the best rank- k approximation.

Dataset	TS	GFS	ELS-1	ELS	TS	GFS	ELS-1	ELS
	k=100				k=120			
COIL20	1.914	1.790	1.762	1.713	1.896	1.834	1.795	1.742
ORL	1.924	1.777	1.743	1.695	2.032	1.853	1.818	1.766
UMIST	2.017	1.748	1.718	1.665	2.024	1.792	1.758	1.700
B.Alpha	1.680	1.671	1.659	1.636	1.687	1.689	1.680	1.654
YaleB	1.913	1.724	1.701	1.665	1.962	1.782	1.753	1.709
USPS	1.808	1.889	1.817	1.756	1.923	1.974	1.905	1.848
BlogFeedback	1.413	1.179	1.168	1.197	1.626	1.202	1.240	1.209
Dataset	TS	GFS	ELS-1	ELS	TS	GFS	ELS-1	ELS
	k=140				k=160			
COIL20	1.912	1.869	1.828	1.778	1.976	1.897	1.864	1.808
ORL	2.140	1.931	1.891	1.838	2.240	2.012	1.966	1.903
UMIST	2.074	1.830	1.786	1.733	2.118	1.865	1.819	1.756
B.Alpha	1.722	1.724	1.707	1.686	1.767	1.763	1.742	1.717
YaleB	1.944	1.829	1.797	1.749	2.012	1.862	1.826	1.783
USPS	2.017	2.025	1.979	1.931	2.135	2.127	2.078	2.044
BlogFeedback	1.817	1.227	1.226	1.228	2.089	1.261	1.230	1.261

7.1.4 Reconstruction capabilities

Since our algorithm achieves good approximation errors, we evaluate its capabilities to reconstruct single data instances. Given a data instance $x \in \mathbb{R}^n$, which corresponds to a row of the input matrix $A \in \mathbb{R}^{m \times n}$, we consider the vector $x' \in \mathbb{R}^k$ comprised only of the features chosen by the studied algorithms (i.e., x'^T is a row of the submatrix of chosen columns C). The complete instance x can be approximated as

$$x^T \approx x'^T C^+ A. \quad (7.2)$$

We pick various instances from the Yale faces dataset¹² with the following criterion. For $k = 10, 20, 30, 40, 50$ we compute the approximation error for the whole dataset

¹²<http://vision.ucsd.edu/content/yale-face-database>

using our algorithm (`EfficientLocalSearch`). We compute the per-instance average approximation error (i.e. the total error divided by the number of instances) and pick the individual instance whose approximation error (i.e. the sum of squared errors between x and the value approximated by (7.2)) is closest to this average. This can be interpreted as the instance that is closest to the expected approximation error if one were to pick a random data point.

Figure 7.2 shows the reconstruction achieved for five such images. Each row shows one image as reconstructed by `TwoStage`, `Greedy`, `EfficientLocalSearch`, the SVD and the original image. Below each image the reconstruction error with respect to the best rank- k approximation is shown. It might be shocking that in one instance the error attained by our algorithm is better than the one yielded by the SVD. Though unusual, this is entirely possible. The SVD yields the best approximation of the entire matrix, but an individual row might be approximated better using a different model. It should be noted that even though our algorithm yields better matrix approximations, it does not necessarily approximate all individual instances better than the other two algorithms. In this dataset, however, our algorithm consistently outperforms `TwoStage` and `Greedy` when it comes to the number of individual images that are approximated better than the other two. In Table 7.4 we report, for different values of k , the number of instances that each algorithm approximates better than the other two. We run `EfficientLocalSearch` and `TwoStage` 10 times and report the average and the standard deviation. This dataset totals 161 images.

7.1.5 Benefits of randomized initialization

One of the key properties of `EfficientLocalSearch` is that it starts from a random subset. This brings certain advantages over other approaches such as the greedy al-

k	TwoStage	Greedy	EfficientLocalSearch
10	10.8 ± 4.38	67.2 ± 2.32	87 ± 2.5
20	14.4 ± 2.22	62 ± 6.26	88.6 ± 7.04
30	23.5 ± 3.06	51.6 ± 3.72	89.9 ± 3.76
40	18.8 ± 2.86	52 ± 3.86	94.2 ± 4.78
50	16.8 ± 3.48	46.7 ± 3.02	101.5 ± 4.6

Table 7.4: Number of instances that each algorithm approximates better than the other two on average.

gorithm. If one has enough computational resources, **EfficientLocalSearch** can be run using various initial subsets and the best result can be kept. In our next experiment we intend to verify if this approach does indeed bring significant improvements. We run the algorithm on all data sets with $k = 10, 20$ using $2^i, i = 0, \dots, 5$ initial random subsets. For each run we keep the best result of the 2^i attempts. For each value of k and each number of initial subsets we run the experiment 10 times and report the mean of the obtained errors with respect to the best rank- k approximation (expression 7.1).

Figures 7.3, 7.4 and 7.5 show the results of these experiments. It is clear that running the algorithm on more than one initial subset yields noticeable (though small) improvements in general, although the benefits become less significant beyond 4 or 8 subsets. It is important to note that in the experiment described in section 7.1.2 we only used one initial subset, so the results shown in tables 7.2 and 7.3 can be improved if the algorithm is run various times.

7.1.6 Iterations

A reasonable concern to be raised about **EfficientLocalSearch** is the number of iterations it requires until it stops or until it reaches a good enough subset. Even

though it can iterate efficiently, the algorithm could be considered inefficient if the number of required iterations is generally too high. In this experiment we attempt to gain insight on this issue. We run the algorithm on all data sets with $k = 20 \times i, i = 1, \dots, 10$ and report the number of iterations required, as well as the error ratio obtained at each iteration. We run the algorithm 50 times for each data set and value of k and report the average values. Figure 7.6 shows the average number of iterations before **EfficientLocalSearch** stops for different values of k on all data sets, as well as the maximum and minimum number observed across all runs. Figure 7.7 shows the average error ratio attained at each iteration. It is clear that even though sometimes **EfficientLocalSearch** can take a fairly high number of iterations before it stops, the most significant improvement is achieved at the first one. This reinforces the conclusions drawn from tables 7.2 and 7.3, which show that one iteration is enough to attain a good column subset, while further iterations are only necessary for refinement.

7.1.7 Approximation error

In essence, **EfficientLocalSearch** is an algorithm for obtaining a low-rank approximation of a matrix. In general, no subset of k columns will attain an error as small as the best possible rank- k approximation. However, attaining such an error is possible by picking a larger number of columns. Therefore, we conduct an experiment to try to determine the number of columns that are required by **EfficientLocalSearch** to achieve that goal. Ultimately, it would be interesting to determine whether or not it is possible to guarantee an arbitrarily close relative approximation (by a factor of $1 + \epsilon$) by picking a number of columns determined by a small function of ϵ . With this experiment we attempt to shed some light on the feasibility of that endeavour.

We set the target rank to $\rho = 2, \dots, 9, 10, 20, \dots, 100$ and run the algorithm with increasing values of k until the best rank- ρ approximation error is attained. For this experiment we compare `EfficientLocalSearch` to the `TwoStage` and the greedy algorithms. Table 7.5 shows, for each target rank ρ , the number of columns required by `TwoStage` (TS), Greedy (Gr) and `EfficientLocalSearch` (ELS) to match the best rank- ρ approximation error. Greedy and `EfficientLocalSearch` perform similarly on various data sets, although on some of them (COIL20, ORL and UMIST) `EfficientLocalSearch` requires a significantly smaller number of columns.

7.1.8 Kahan matrix

In our final experiment, we test the algorithms on a Kahan matrix. Kahan matrices are specifically designed to pose problems to rank-revealing factorization algorithms. An $n \times n$ Kahan matrix is defined as follows:

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & s & 0 & \dots & 0 \\ 0 & 0 & s^2 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \dots & s^{n-1} \end{pmatrix} \begin{pmatrix} 1 & -c & -c & \dots & -c \\ 0 & 1 & -c & \dots & -c \\ 0 & 0 & 1 & \dots & -c \\ 0 & 0 & 0 & \ddots & -c \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}$$

where $s^2 + c^2 = 1, c, s > 0$. With a Kahan matrix as input, classic algorithms for QR factorizations with column pivoting [9] will fail to perform any column interchange at all.

We run the three algorithms on a 100×100 Kahan matrix for different values of k and report the error ratio with respect to the best rank- k approximation. In the case of `EfficientLocalSearch` and `TwoStage`, we run the algorithm 100 times and

Table 7.5: Number of columns required to match the best rank- ρ approximation.

	ρ	2	3	4	5	6	7	8	9	10	20	30	40	50	60	70	80	90	100
COIL20	TS	7	8	11	12	13	15	17	18	21	42	63	81	95	111	126	140	155	171
	Gr	4	6	8	10	12	14	15	17	19	39	57	74	90	106	121	137	151	165
	ELS	4	6	8	10	11	13	14	16	18	36	54	70	86	101	117	131	146	159
ORL	TS	8	9	11	12	16	18	20	22	24	45	62	82	97	113	124	139	153	164
	Gr	4	6	8	10	12	14	16	18	20	39	57	73	88	103	117	130	143	156
	ELS	4	6	8	10	12	13	15	17	18	37	54	70	85	99	112	126	139	151
UMIST	TS	10	13	15	17	18	19	21	24	25	43	67	83	103	119	138	152	169	185
	Gr	4	6	8	10	12	14	15	18	19	38	56	74	90	106	122	137	153	167
	ELS	4	6	7	9	11	13	14	17	18	36	54	69	86	101	116	132	145	159
BA	TS	6	7	9	11	14	16	17	18	20	41	58	75	91	105	118	130	143	153
	Gr	4	6	9	11	13	15	16	18	20	39	56	73	89	103	117	128	139	150
	ELS	4	6	9	11	13	15	16	18	20	39	56	73	89	103	117	128	139	150
YaleB	TS	9	10	12	14	19	21	25	25	27	49	67	86	101	118	133	148	163	178
	Gr	4	6	8	10	12	14	16	17	19	37	55	71	88	104	119	134	149	163
	ELS	4	6	8	10	12	14	16	17	19	37	55	71	88	104	119	134	149	163
USPS	TS	6	8	10	11	13	15	17	20	21	37	52	64	78	90	103	116	128	142
	Gr	5	7	8	10	12	14	16	17	19	34	48	63	76	89	102	113	127	138
	ELS	5	7	8	10	12	14	16	17	19	34	48	63	76	89	102	113	127	138
OnlNews	TS	4	6	7	8	9	10	11	12	13	26	36	44	52	-	-	-	-	-
	Gr	3	4	6	7	8	9	10	12	13	25	35	44	52	-	-	-	-	-
	ELS	3	4	6	7	8	9	10	12	13	25	35	44	52	-	-	-	-	-
BF	TS	4	9	11	11	14	14	17	25	26	40	54	68	84	91	105	114	131	138
	Gr	3	4	6	8	9	10	11	13	14	28	40	52	63	74	84	94	104	114
	ELS	3	4	6	8	9	10	11	13	14	28	40	52	63	74	84	94	104	114
YearPr	TS	5	9	11	12	13	15	18	19	22	33	44	54	62	70	78	84	-	-
	Gr	4	6	9	11	12	14	15	17	18	31	43	53	62	70	77	84	-	-
	ELS	4	6	9	11	12	14	15	17	18	31	43	53	62	70	77	84	-	-

Table 7.6: Error ratio for a 100×100 Kahan matrix.

k	No changes	Greedy	EfficientLocalSearch	TwoStage
2	7.028992	1.088793	1.088793	1.088793
3	6.959325	1.089115	1.089115	1.089115
4	6.889070	1.089350	1.089350	1.089350
5	6.818729	1.089577	1.089577	1.089577
6	6.748374	1.089806	1.089806	1.089806
7	6.678020	1.090040	1.090040	1.090040
8	6.607675	1.090281	1.090281	1.090281
9	6.537339	1.090528	1.090528	1.090528
10	6.467015	1.090783	1.090783	1.090783
20	5.764529	1.093816	1.118009	1.093816
30	5.063975	1.098087	1.219694	1.237866
40	4.366622	1.104401	1.245019	1.245019
50	3.674927	1.114186	1.256198	1.256198

report the average. We also show the error ratio without performing any column interchange, which is equivalent to the result of classic QR with pivoting. Table 7.6 shows the results. All three algorithms perform well on this task, although for some values of k the greedy approach is better. It would be interesting to investigate the source of this deficiency in the case of `EfficientLocalSearch`.

7.2 Regularized column subset selection

In this section we report the results for the proposal described in chapter 6. Specifically, we aim to assess the following aspects:

- *Generalization ability.* We test the ability of the proposed algorithm to choose variables that can approximate well not only the input data (training data) but also future observations (test data).
- *Stability.* We test how robust our algorithm is to noisy variations in the data.

- *Conditioning.* We measure the conditioning of the selected submatrix, defined as the ratio between the largest and the smallest singular values.
- *Running time.* We evaluate the running time of our algorithm with respect to different input parameters.
- *Applications.* We evaluate the effectiveness of our algorithm as a preprocessing step for clustering and its ability to reconstruct partially observed images.

To this end, we employed a variety of well-known data sets. We now briefly describe them, and indicate the preprocessing operations and the training/test splits for each of them.

- Isolet [26]. This data set consists of a collection of spoken letter recordings by various individuals, each represented by a set of features. The data were used as distributed. The variables are real-valued between -1 and 1. For the test set, we respected the split proposed by the authors.
- MNIST [52]. Images of handwritten digits. The data were divided by 255 to ensure that all values be between 0 and 1. The training/test split provided by the authors was respected.
- Yale Face Extended [34]. Images of faces. The data were divided by 255 to ensure that all values be between 0 and 1. The first 1200 instances were used for training. The rest for testing.
- ORL [67]. Ten different images of each of 40 distinct subjects. The data were divided by 255 to ensure that all values be between 0 and 1. The first 300 instances were used for training. The rest for testing.

- COIL-20 [61]. Images of objects of different categories. The data were divided by 255 to ensure that all values be between 0 and 1. The data were split into two halves for training and testing, ensuring class balance between both sets. We reduce the size of the images to 64×64 .
- Online News Popularity [29]. Statistics associated to news articles. All variables were standardized to zero mean and unit variance. The first 30,000 instances were used for training. The rest for testing.

Table 7.7 summarizes the employed data sets.

Table 7.7: Employed datasets.

Dataset	Variables	Train	Test
ORL	1024	300	100
MNIST	784	60,000	10,000
YaleB	1024	1200	1214
OnlineNews	58	30,000	30,000
IsoLet	617	5200	1559
COIL-20	4096	300	100

We consider two algorithms:

- **Greedy**: The unregularized greedy algorithm for problem 3.4. We use a Python implementation of the algorithm described by Farahat et al. [28].
- **RegGreedy**: The algorithm proposed in chapter 6 to solve problem 7 (iterating until the desired number of columns is chosen). We use a Python implementation. In all experiments we use the algorithm adapted to optimize problem 7.

7.2.1 Generalization ability

In our first set of experiments, we evaluate whether the regularized formulation and the corresponding algorithm select columns that produce models with better generalization ability. To this end, we run the algorithms, both **Greedy** and **RegGreedy**, on small samples of the training splits of the data sets, and then measure the ability of the resulting models to approximate the rest of the features of the test split.

Given an input matrix A , a test matrix B , a number $k \in \mathbb{N}$ and a value of $\lambda \in \mathbb{R}$, assume the algorithm being run outputs the set S . Then we measure the loss as

$$L(S) = \|B - B_S(A_S^T A_S + \lambda I)^{-1} A_S^T A\|_F^2$$

Notice that for both the unregularized and the regularized algorithms, we measure the loss using a regularized approximation. This is because even though the unregularized version does not attempt to optimize this objective, the approximation on the test split will generally be much better if we introduce the regularization term λ , thus providing a fairer comparison. In doing so, we set the bar higher for our algorithm.

For all data sets, we run the algorithms on $2^i\%$ of the training data, $i = 0, \dots, 4$, for $k = 2^i, i = 4, \dots, 9$, and then measure the loss on the test split. It should be noted that we simply set $\lambda = 1$ for all cases. However, better results might be obtained by fine-tuning this parameter. In order to assess the improvement brought by the regularized variant, we measure the relative improvement as follows. Let S_N be the set output by the **Greedy**, and S_R the set output by **RegGreedy**. Then the relative improvement is defined as

$$100 \times \frac{L(S_N) - L(S_R)}{L(S_N)}$$

In order to provide a better estimate of this value, we run the algorithms 50 times on different random samples of the training set and replace $L(S_N)$ and $L(S_R)$ with their averages.

Figure 7.8 illustrates the results. For each data set, we show the relative improvement for the different fractions of the training set. It can be seen that the regularizing penalty yields significant improvements, especially when training data are scarce and k is large. Notice the different scale on the plots corresponding to OnlineNews and MNIST, where the improvement was more moderate.

7.2.2 Stability

As discussed above, instances of the unregularized formulation of the column subset selection problem (problem 3.4) where $n \gg m$ are inconvenient. In this set of experiments, we aim to verify whether the regularized formulation and the corresponding algorithm improves the stability of the results. In order to tests this, we perturb the input data to see how the algorithms behave in the face of noise.

To measure the stability of each of the algorithms, we run them on s different instances of the perturbed matrix and measure the average pairwise Jaccard index, which we define below. Given two sets S_1, S_2 , the Jaccard index is measured as

$$J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

Given a collection of sets $S_1, \dots, S_s$, we define the average pairwise Jaccard index as

$$\bar{J}(S_1, \dots, S_s) = \frac{1}{s(s-1)} \sum_i^s \sum_j^s J(S_i, S_j) \mathbb{I}\{i \neq j\}$$

where $\mathbb{I}\{i \neq j\}$ is 1 if $i \neq j$, 0 otherwise. To make this index more meaningful, we

calculate its expected value assuming the column subsets are chosen uniformly at random. Given a matrix of n columns, assume we want to select a subset of size k . If we pick two subsets of $[n]$ at random, S_1 and S_2 , there are $\binom{n}{k}^2$ possible outcomes. Out of these, the number of pairs that have $k - p$ elements in common is

$$\binom{n}{k} \binom{k}{p} \binom{n-k}{p}$$

To see this, observe that for each of the $\binom{n}{k}$ possible values of S_1 , S_2 must have p out of k elements that are not in S_1 , and those can be any of the remaining $n - k$ ones. Therefore, the expected value of the size of the intersection between two subsets drawn uniformly at random is

$$\mathbb{E}[S_1 \cap S_2] = \sum_{p=0}^k \frac{(k-p) \binom{k}{p} \binom{n-k}{p}}{\binom{n}{k}}$$

Now, the Jaccard index of each of those pairs is the size of the intersection divided by the size of the union. Hence, given n and k ,

$$\mathbb{E}[J] = \sum_{p=0}^k \frac{(k-p) \binom{k}{p} \binom{n-k}{p}}{\binom{n}{k} (k+p)}$$

We can now measure the stability of the algorithms by running them on different perturbations of the input matrix, and then comparing the average pairwise Jaccard index of the resulting subsets with the expected value of the Jaccard index. We consider the case where $n > m$, that is, the input matrix has more columns than rows. To this end, we take random samples of 100 rows of each training data set and set $k = m$. In the case of OnlineNews, since $n < 100$, we take $k = m = n/2$. Note that the case $k > m$ becomes pathological in the unregularized formulation, since

Table 7.8: Average pairwise Jaccard index and loss.

	$\bar{J}$			
Dataset	$\lambda = 0$	$\lambda = 1$	$\lambda = 10$	$\mathbb{E}[J]$
ORL	0.177	0.245	0.408	0.012
MNIST	0.278	0.951	0.774	0.068
YaleB	0.252	0.632	0.852	0.051
OnlineNews	0.782	1.0	1.0	0.336
Isolet	0.422	0.6	0.838	0.088

any column choice once the span of the data has been covered is equally innocuous. Therefore, as k grows beyond the value of m , the Jaccard index for the unregularized formulation will approach $\mathbb{E}[J]$ if ties are broken arbitrarily.

We take the input data set and perturb it with a matrix whose entries are independently sampled from a Gaussian distribution with zero mean and a standard deviation of 10^{-3} . As explained above, we apply s different perturbations to the input data and run the algorithms on each of them, thus obtaining s different subsets for each algorithm. We set $s = 100$ and measure the average pairwise Jaccard index. Table 7.8 shows the results. We also show the expected value of the Jaccard index to know how close to a random choice each algorithm is. We run this experiment for $\lambda = 0$ (i.e. the unregularized algorithm by Farahat et al. [28]), $\lambda = 1$ and $\lambda = 10$. It can be seen that the regularized formulation significantly improves the stability of the results.

7.2.3 Conditioning

The conditioning of a matrix can be loosely understood as a measure of numerical rank deficiency. Formally, given a matrix C of rank k , we define its conditioning, or its condition number, as $\kappa(C) = \frac{\sigma_1(C)}{\sigma_k(C)}$. Ill-conditioned matrices, that is, with a

large condition number, are prone to significant numerical errors when involved in the solution of linear systems.

We compute the condition number of the submatrices of the training set selected by both algorithms, regularized and unregularized. We run the algorithm 50 times on different random samplings of the training set and report the minimum, average and maximum across all runs in tables 7.9 and 7.10. The results clearly reveal that the regularization term encourages the selection of significantly better-conditioned column subsets.

In order to avoid overcrowding the table we only report the results for $k = 16, 32$, as they are illustrative of the general behaviour of the algorithms in this regard.

An interesting fact revealed by our experiments is the following: in cases where $k > m$, the unregularized formulation of the problem is ill-posed. In terms of the objective function, once the m -dimensional subspace spanned by the matrix has been covered, any subsequent column choice is equally good. The unregularized algorithm therefore yields particularly poorly conditioned subsets (see e.g. YaleB, $0.01 * m$, $k = 16$, $\lambda = 0$ in tables 7.9 and 7.10. What is surprising is that in these situations, the regularized variant produces column subsets that lead to well-conditioned matrices *even in the test set*. An example of the obtained condition numbers on submatrices of the test set is shown in table 7.11.

7.2.4 Clustering

We test the effectiveness of our algorithm as a preprocessing step for clustering. Dimensionality reduction is often essential for these tasks, because the distance computations employed by most clustering algorithms are particularly sensitive to large numbers of variables.

Table 7.9: Condition number of the matrices output by the unregularized and the regularized algorithms. For each experiment we report the minimum, average and maximum (min / avg / max) of 50 runs on different random samplings.

Sample size	Dataset	$k = 16$	
		Greedy	RegGreedy
0.01*m	ORL	13.07 / 338.94 / 3127.57	4.45 / 6.28 / 10.71
	MNIST	130.88 / 142.34 / 153.50	5.74 / 6.13 / 6.56
	YaleB	243.81 / 12186.18 / 48241.03	31.97 / 183.84 / 1019.68
	OnlineNews	2.37 / 3.63 / 5.33	2.37 / 3.39 / 4.59
	Isolet	12.66 / 19.91 / 29.78	11.58 / 14.75 / 17.96
0.04*m	ORL	119.35 / 2331.63 / 20602.93	55.78 / 90.22 / 187.40
	MNIST	136.33 / 147.26 / 161.20	5.61 / 5.90 / 6.27
	YaleB	29.62 / 35.36 / 46.36	24.32 / 32.24 / 41.23
	OnlineNews	2.46 / 3.17 / 4.07	2.46 / 3.01 / 3.91
	Isolet	10.82 / 13.24 / 16.66	10.42 / 12.52 / 16.53
0.16*m	ORL	61.95 / 74.08 / 90.82	41.89 / 60.14 / 86.10
	MNIST	130.80 / 146.03 / 151.36	5.55 / 18.69 / 134.06
	YaleB	22.11 / 24.50 / 28.54	20.93 / 24.29 / 28.38
	OnlineNews	2.45 / 2.80 / 3.34	2.45 / 2.80 / 3.34
	Isolet	9.58 / 10.74 / 11.93	9.58 / 10.74 / 11.93

In order to evaluate the ability of our methods to produce robust feature subsets, we proceed as follows: we run the algorithm on a small portion of the training set (of varying size) and then reduce the test set so as to keep the chosen variables only. We then run the k -means clustering algorithm on this reduced data set. For reference, we also consider the case where $k = n$, that is, using the whole feature set for clustering. Note that in this case, the training split does not play a part in the result. The results for the different test set sizes, equal to $m - (\text{training set size})$, are thus expected to be similar.

We considered the data sets ORL, COIL20 and IsoLet. We discarded MNIST and YaleB, where the k -means algorithm did not produce acceptable results, and OnlineNews, whose target values are better suited to a regression task.

To measure the quality of the result, we compute the normalized mutual infor-

Table 7.10: Condition number of the matrices output by the unregularized and the regularized algorithms. For each experiment we report the minimum, average and maximum (min / avg / max) of 50 runs on different random samplings.

Sample size	Dataset	$k = 32$	
		Greedy	RegGreedy
0.01*m	ORL	33.92 / 116.77 / 240.82	3.90 / 5.10 / 7.68
	MNIST	10.22 / 146.46 / 241.65	10.22 / 10.90 / 11.80
	YaleB	214.28 / 1218.04 / 2575.45	10.10 / 63.30 / 272.59
	OnlineNews	4.67 / 5.86 / 7.43	4.58 / 5.71 / 7.32
	Isolet	43.04 / 77.20 / 133.39	31.35 / 39.31 / 58.30
0.04*m	ORL	183.43 / 985.98 / 4140.42	20.59 / 27.88 / 42.15
	MNIST	227.12 / 235.81 / 247.82	9.36 / 10.03 / 10.40
	YaleB	99.93 / 134.32 / 185.35	76.28 / 103.24 / 139.18
	OnlineNews	3.99 / 4.82 / 5.86	3.98 / 4.72 / 5.51
	Isolet	20.40 / 24.28 / 28.62	20.38 / 23.58 / 27.96
0.16*m	ORL	178.82 / 233.01 / 304.82	131.69 / 175.00 / 210.30
	MNIST	225.88 / 235.61 / 244.84	9.95 / 55.10 / 241.62
	YaleB	40.26 / 43.60 / 51.41	38.01 / 41.91 / 48.99
	OnlineNews	3.78 / 4.38 / 5.12	3.77 / 4.24 / 5.12
	Isolet	17.82 / 20.30 / 21.42	17.82 / 20.30 / 21.42

Table 7.11: Condition number of the test submatrices chosen by the unregularized and the regularized algorithms, in a case where $k > m$. In this situation, the unregularized version of the problem becomes ill-posed, and the algorithm unstable.

Sample size	Dataset	$k = 16$	
		$\lambda = 0$	$\lambda = 1$
0.01*m	YaleB	6.22e+16 / 8.76e+16 / 2.37e+17	17.43 / 26.07 / 45.11
		$k = 32$	
		1.21e+16 / 1.74e+16 / 2.84e+16	28.34 / 46.74 / 72.87

mation (NMI) of the ground truth labels and the obtained partition. The NMI is defined as follows. Given two discrete random variables X, Y , the NMI is defined as follows:

$$NMI(X, Y) = \frac{MI(X, Y)}{\sqrt{H(X)H(Y)}}$$

where $MI(X, Y)$ is the mutual information of X and Y , and $H(X)$ is the entropy of X .

Cluster centroids were initialized using the k -means++ scheme, and the best result out of 10 in terms of the objective function was kept. The process was repeated 50 times, running the column subset selection algorithms on different random samples of the training set each time. We report the average of the obtained NMI values.

The results are shown in table 7.12, using from 1% to 16% of the training data. In all 3 data sets, **RegGreedy** shows superior performance. Some of the results warrant further discussion. First, an interesting property of **RegGreedy** is that the NMI is remarkably stable with respect to the amount of training data used, while **Greedy** generally only starts obtaining good results when a sizeable portion is employed. Second, in a few instances, **RegGreedy** did show clearly poorer performance (COIL20, $k = 16, 32$). It would be interesting to determine the cause of this deficiency.

This results provide evidence of the clear advantages of using the regularized variant of column subset selection for practical applications. In particular, note how the quality of the clustering improves when using feature subsets of size 128 or more rather than the entire feature set. In the case of **RegGreedy**, this improvement is present even when the feature subset was chosen using only 1% of the training data.

7.2.5 Image reconstruction

In order to provide a visual account of the improved generalization ability of the proposed method, we evaluate its ability to reconstruct instances of unseen images. To this end, we run **Greedy** and **RegGreedy** on a small portion of a 64×64 version of the ORL data set.

We proceed as follows. Let A be the portion of the training data set used, and let S be the subset output by the employed algorithm. We define $C = A_S$ and

Table 7.12: NMI for clustering results on the test set using the feature subset chosen by each algorithm. G=Greedy, RG=RegGreedy

ORL										
	0.01 * m		0.02 * m		0.04 * m		0.08 * m		0.16 * m	
k	G	RG	G	RG	G	RG	G	RG	G	RG
16	0.67	0.768	0.733	0.777	0.786	0.787	0.78	0.785	0.797	0.796
32	0.653	0.791	0.713	0.798	0.76	0.806	0.809	0.809	0.817	0.805
64	0.632	0.808	0.685	0.808	0.734	0.816	0.786	0.817	0.821	0.817
128	0.614	0.819	0.657	0.817	0.7	0.815	0.754	0.818	0.8	0.819
256	0.588	0.821	0.625	0.82	0.672	0.82	0.721	0.819	0.767	0.821
512	0.565	0.824	0.596	0.823	0.639	0.825	0.69	0.821	0.739	0.821
n	0.81		0.808		0.803		0.811		0.809	
COIL20										
	0.01 * m		0.02 * m		0.04 * m		0.08 * m		0.16 * m	
k	G	RG	G	RG	G	RG	G	RG	G	RG
16	0.644	0.636	0.696	0.636	0.718	0.623	0.733	0.629	0.739	0.604
32	0.642	0.686	0.703	0.67	0.744	0.684	0.762	0.676	0.769	0.672
64	0.654	0.721	0.701	0.724	0.739	0.726	0.774	0.727	0.783	0.723
128	0.639	0.76	0.701	0.769	0.744	0.78	0.774	0.772	0.789	0.769
256	0.646	0.782	0.707	0.782	0.74	0.787	0.772	0.794	0.79	0.795
512	0.662	0.782	0.698	0.779	0.746	0.777	0.77	0.779	0.793	0.783
n	0.758		0.755		0.755		0.754		0.756	
IsoLet										
	0.01 * m		0.02 * m		0.04 * m		0.08 * m		0.16 * m	
k	G	RG	G	RG	G	RG	G	RG	G	RG
16	0.573	0.571	0.576	0.575	0.579	0.577	0.583	0.583	0.589	0.589
32	0.604	0.602	0.608	0.61	0.604	0.608	0.601	0.603	0.596	0.597
64	0.642	0.655	0.66	0.665	0.656	0.662	0.666	0.665	0.662	0.663
128	0.577	0.703	0.698	0.709	0.707	0.715	0.712	0.71	0.711	0.712
256	0.5	0.725	0.613	0.733	0.728	0.735	0.734	0.733	0.733	0.736
n	0.697		0.699		0.704		0.704		0.701	

$W = (C^T C + \lambda I)^{-1} C^T A$. To rebuild an instance of the test set x , we compute

$$x_S^T W$$

Three examples are shown in figure 7.9. The reconstructions were done for $k = 128, 256, 512, 1024, 2048$. The shown images were selected as follows. We compute

the average reconstruction error attained by each algorithm at each value of k . The first image we show is the one whose reconstruction error was the closest to the average for $k = 2048$ (which was casually the same for both algorithms). The second image was the closest to the average error using **Greedy** for $k = 128$. The third, the closest to the average error using **RegGreedy** for $k = 128$. The chosen images should therefore be close to the expected reconstruction error incurred by each algorithm for $k = 128$ and $k = 2048$.

These images demonstrate how the models obtained using **RegGreedy** exhibit a better ability to recover certain visual characteristics. In particular, this highlights a previously discussed issue (see section 7.2.2). When k becomes larger than the rank of the input matrix, the unregularized formulation of the problem no longer constitutes a suitable objective to judiciously choose additional columns. This is visible in this example when $k > 512$. Increasing the value of k does not provide improvements when using **Greedy**, while **RegGreedy** manages to recover additional nuance when the dimensionality of the model increases (even if the result is visually subtle, the reconstruction error continues to decrease using **RegGreedy**, while it ceases to do so with **Greedy**).

7.2.6 Running time

In order to evaluate the running time of our algorithm, we generated synthetic matrices of increasing dimensions and ran the algorithms for increasing values of k . Figure 7.10 shows the resulting times for the unregularized (**Greedy**) and the regularized (**RegGreedy**) algorithms. To measure the sensitivity with respect to each parameter, we fixed the other two. Specifically, we run the following experiments:

- $n = 1000, k = 128, m \in [10^3, 10^5]$

- $m = 1000, k = 1, n \in [100, 5000]$
- $m = 1000, n = 1024, k \in [10, 1020]$

The behavior of both algorithms with respect to m and n (the number of rows and columns respectively) is very similar. For large values of k , the regularized variant does exhibit noticeably larger running times. It should be noted, however, that the ratio between the time required by the two algorithms converges to a constant factor. This ratio is shown in the plot for varying values of k to support this claim.

Figure 7.1: Running times for one iteration of our algorithm compared to other methods.

TS	Greedy	ELS	SVD	Original
$k = 10$				
2.2064	1.4374	1.1603	1.000	
$k = 20$				
2.7925	1.6557	1.3819	1.000	
$k = 30$				
2.7705	1.3594	1.0023	1.000	
$k = 40$				
1.8073	1.6981	0.9838	1.000	
$k = 50$				
2.2121	1.5641	1.3917	1.000	

Figure 7.2: Reconstructions achieved with different algorithms. Below each image is the approximation error with respect to the SVD.

Figure 7.3: Average error attained by EfficientLocalSearch as the number of attempts increases

Figure 7.4: Average error attained by EfficientLocalSearch as the number of attempts increases

Figure 7.5: Average error attained by `EfficientLocalSearch` as the number of attempts increases

Figure 7.6: Average number of iterations before `EfficientLocalSearch` stops.

Figure 7.7: Average error attained by `EfficientLocalSearch` at each iteration

Figure 7.8: Relative improvement of the regularized variant

(a)

(b)

(c)

Figure 7.9: Reconstruction of test set images. For both (a), (b) and (c): Top: reconstruction by **RegGreedy**. Bottom: reconstruction by **Greedy**. Right: Original image. We show reconstructions for $k = 128, 256, 512, 1024, 2048$.

Figure 7.10: Running times with respect to the different input parameters. The ratio between the running time of the two algorithms is shown in the plot for the value of k .

Chapter 8

Conclusions & future work

In this thesis we have addressed the problem of column subset selection in practice. We have first set out to overcome potential drawbacks of existing methods in practical settings, leading to the choice of a local-search heuristic. Our contributions in this regard can be summarized as follows:

- We have derived an efficient implementation of local search taking advantage of the nature of column substitutions in the linear-algebraic context. This has resulted in an algorithm of significantly reduced complexity and running time with respect to a naive implementation.
- We have derived results that enable the application of this algorithm to very large data sets.
- We have derived elementary approximation guarantees for the resulting algorithm. In particular, we have provided an approximation of factor k with respect to the optimal solution of the maximization version of the problem, scaled by the smallest singular value of the corresponding optimal matrix.

Second, we have addressed the problem of regularization in column subset selection. Our contributions in this regard can be summarized as follows:

- We have proposed regularized formulations of the problem, including variants that are adequate for feature selection in practice.
- We have produced a correct and efficient greedy algorithm for these formulations.
- We have derived a lower bound on the error of the proposed problems.

In addition, we have performed a wide variety of experiments to demonstrate the validity of our claims, to assess the performance of our algorithms in practice and to gain insight on their behaviour.

The work described in this thesis has brought our attention to various avenues of possible further inquiry.

One of the most interesting open questions relating to column subset selection is the approximability of the combinatorial objective. As we have seen, most existing works regard column subsets as low-rank approximations, and thus the employed yardstick is the ratio to the best possible factorization of a given rank. Using exactly k columns we cannot hope for a ratio better than $k + 1$. However, if we replace the best rank- k approximation with simply an optimal subset, the extent to which we can improve this result is not yet known. Given the vast amount of research already devoted to these problems, we think this could turn out to be a fertile endeavour. In particular, can existing approximation algorithms for column subset selection problems be shown to achieve guarantees for optimal solutions? How do both ratios—to best rank- k approximations and optimal solutions—relate?

To the best of our knowledge, only one work proposes an approximation scheme to optimal solutions of GCSSP. However, the impact of the smallest singular value of the optimal matrix in the number of required columns bars the algorithm from being a PTAS—it is easy to come up with examples where the optimal solution is arbitrarily ill-conditioned. It remains to be seen whether we can eliminate the presence of such value in the number of columns required to achieve multiplicative or additive approximations. To this end, can we leverage methods from the numerical linear algebra literature to isolate well-conditioned blocks? An equally interesting challenge arises for the case where we are restricted to exactly k columns.

The algorithms and problem formulations proposed in this dissertation also give rise to several questions of independent interest. For instance, why does a single iteration of local search seem to perform remarkably well? Can we take advantage of this behaviour to derive initialization schemes that lead to stronger approximation guarantees?

Our expedition into regularization has also revealed some compelling questions. First of all, it is apparent that an efficient local-search algorithm could be derived with similar techniques as the ones used for the greedy approach, which might lead to better performance in practice. Other puzzles that arise immediately are, for instance, the possibility of deriving approximation guarantees and tighter lower bounds. Additionally, the incorporation of regularizing penalties implicitly regards the resulting models as the product of a statistical inference task. Therefore, it would be interesting to identify maximum-a-posteriori objectives that lead to an equivalent problem formulation, in order to reveal the implicit assumptions about the nature of the data that the proposed approaches make.

Finally, it is of undoubted interest to explore column subset selection and, more generally, unsupervised feature selection beyond the boundaries of linear models.

Existing methods for non-linear inference —e.g. kernel ridge regression— suggest that many of the ideas discussed in this thesis could be extended in this direction.

Bibliography

- [1] J. Altschuler, A. Bhaskara, G. Fu, V. Mirrokni, A. Rostamizadeh, and M. Zadi-moghaddam. Greedy column subset selection: New bounds and distributed algorithms. In *International Conference on Machine Learning*, pages 2539–2548, 2016.
- [2] H. Arai, C. Maung, and H. Schweitzer. Optimal column subset selection by a-star search. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
- [3] R. E. Bellman. *Adaptive control processes: a guided tour*. Princeton university press, 2015.
- [4] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.
- [5] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal column-based matrix reconstruction. *SIAM Journal on Computing*, 43(2):687–717, 2014.
- [6] C. Boutsidis, M. W. Mahoney, and P. Drineas. Unsupervised feature selection for principal components analysis. In *Proceedings of the 14th ACM SIGKDD*

- international conference on Knowledge discovery and data mining*, pages 61–69. ACM, 2008.
- [7] C. Boutsidis, M. W. Mahoney, and P. 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. Society for Industrial and Applied Mathematics, 2009.
  - [8] P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, volume 98, pages 82–90, 1998.
  - [9] P. Businger and G. H. Golub. Linear least squares solutions by householder transformations. *Numerische Mathematik*, 7(3):269–276, 1965.
  - [10] K. Buza. Feedback prediction for blogs. In *Data analysis, machine learning and knowledge discovery*, pages 145–152. Springer, 2014.
  - [11] D. Cai, C. Zhang, and X. He. Unsupervised feature selection for multi-cluster data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 333–342. ACM, 2010.
  - [12] T. F. Chan. Rank revealing qr factorizations. *Linear algebra and its applications*, 88, 1987.
  - [13] T. F. Chan and P. C. Hansen. Some applications of the rank revealing qr factorization. *SIAM Journal on Scientific and Statistical Computing*, 13(3):727–741, 1992.
  - [14] M. B. Christopher. *Pattern Recognition and machine learning*. Springer-Verlag New York, 2016.

- [15] A. Çivril. Column subset selection problem is ug-hard. *Journal of Computer and System Sciences*, 80(4):849–859, 2014.
- [16] A. Civril and M. Magdon-Ismail. Column subset selection via sparse approximation of svd. *Theoretical Computer Science*, 421:1–14, 2012.
- [17] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.
- [18] A. Deshpande and L. Rademacher. Efficient volume sampling for row/column subset selection. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 329–338. IEEE, 2010.
- [19] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In *Proceedings of the seventeenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1117–1126. Society for Industrial and Applied Mathematics, 2006.
- [20] A. Deshpande and S. Vempala. Adaptive sampling and fast low-rank matrix approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 292–303. Springer, 2006.
- [21] P. Drineas and R. Kannan. Fast monte-carlo algorithms for approximate matrix multiplication. In *FoCS*, volume 1, pages 452–459, 2001.
- [22] P. Drineas, R. Kannan, and M. W. Mahoney. Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. *SIAM Journal on Computing*, 36(1):184–206, 2006.

- [23] L. Du and Y.-D. Shen. Unsupervised feature selection with adaptive structure learning. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 209–218. ACM, 2015.
- [24] J. G. Dy and C. E. Brodley. Feature selection for unsupervised learning. *The Journal of Machine Learning Research*, 5:845–889, 2004.
- [25] Z. et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proc. of the 9th USENIX conf. on Networked Systems Design and Implementation*, pages 2–2. USENIX Assoc., 2012.
- [26] M. Fanty and R. Cole. Spoken letter recognition. In *Advances in Neural Information Processing Systems*, pages 220–226, 1991.
- [27] A. K. Farahat, A. Elgohary, A. Ghodsi, and M. S. Kamel. Distributed column subset selection on mapreduce. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 171–180. IEEE, 2013.
- [28] A. K. Farahat, A. Ghodsi, and M. S. Kamel. An efficient greedy method for unsupervised feature selection. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 161–170. IEEE, 2011.
- [29] K. Fernandes, P. Vinagre, and P. Cortez. A proactive intelligent decision support system for predicting the popularity of online news. In *Progress in Artificial Intelligence*, pages 535–546. Springer, 2015.
- [30] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.
- [31] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

- [32] G. M. Fung and O. L. Mangasarian. A feature selection newton method for support vector machine classification. *Computational optimization and applications*, 28(2):185–202, 2004.
- [33] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian data analysis*. CRC press, 2013.
- [34] A. S. Georgiades, P. N. Belhumeur, and D. J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(6):643–660, 2001.
- [35] G. Golub. Numerical methods for solving linear least squares problems. *Numerische Mathematik*, 7(3):206–216, 1965.
- [36] G. Golub, V. Klema, and G. W. Stewart. Rank degeneracy and least squares problems. Technical report, DTIC Document, 1976.
- [37] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420, 1970.
- [38] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.
- [39] M. Gu and S. C. Eisenstat. Efficient algorithms for computing a strong rank-revealing qr factorization. *SIAM Journal on Scientific Computing*, 17(4):848–869, 1996.
- [40] V. Guruswami and A. K. Sinop. Optimal column-based low-rank matrix reconstruction. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1207–1214. SIAM, 2012.

- [41] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.
- [42] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. In *Advances in neural information processing systems*, pages 507–514, 2005.
- [43] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- [44] C. Hou, F. Nie, X. Li, D. Yi, and Y. Wu. Joint embedding learning and sparse regression: A framework for unsupervised feature selection. *IEEE Transactions on Cybernetics*, 44(6):793–804, 2014.
- [45] C. Hou, F. Nie, D. Yi, and Y. Wu. Feature selection via joint embedding learning and sparse regression. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1324, 2011.
- [46] A. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(2):153–158, 1997.
- [47] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.
- [48] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.
- [49] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [50] A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

- [51] Q. V. Le. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8595–8598. IEEE, 2013.
- [52] Y. LeCun, C. Cortes, and C. J. Burges. Mnist handwritten digit database. *AT&T Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, 2, 2010.
- [53] K.-C. Lee, J. Ho, and D. J. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(5):684–698, 2005.
- [54] Z. Li, Y. Yang, J. Liu, X. Zhou, H. Lu, et al. Unsupervised feature selection using nonnegative spectral analysis. In *AAAI*, 2012.
- [55] M. Lichman. UCI machine learning repository, 2013.
- [56] H. Lutkepohl. Handbook of matrices. *Computational Statistics and Data Analysis*, 2(25):243, 1997.
- [57] C. D. Meyer, Jr. Generalized inversion of modified matrices. *SIAM Journal on Applied Mathematics*, 24(3):315–323, 1973.
- [58] P. Mitra, C. Murthy, and S. K. Pal. Unsupervised feature selection using feature similarity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):301–312, 2002.
- [59] L. C. Molina, L. Belanche, and À. Nebot. Feature selection algorithms: A survey and experimental evaluation. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 306–313. IEEE, 2002.

- [60] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
- [61] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-20). Technical Report CUCS-005-96, Columbia University, February 1996.
- [62] B. Ordozgoiti, S. G. Canaval, and A. Mozo. A fast iterative algorithm for improved unsupervised feature selection. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 390–399. IEEE, 2016.
- [63] B. Ordozgoiti, S. G. Canaval, and A. Mozo. Iterative column subset selection. *Knowledge and Information Systems*, pages 1–30, 2017.
- [64] C.-T. Pan. On the existence and computation of rank-revealing lu factorizations. *Linear Algebra and its Applications*, 316(1):199–222, 2000.
- [65] M. Qian and C. Zhai. Robust unsupervised feature selection. In *IJCAI*, pages 1621–1627, 2013.
- [66] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.
- [67] F. S. Samaria and A. C. Harter. Parameterisation of a stochastic model for human face identification. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 138–142. IEEE, 1994.
- [68] Y. Shitov. Column subset selection is np-complete. *arXiv preprint arXiv:1701.02764*, 2017.
- [69] C. Small. A global analysis of mid-ocean ridge axial topography. *Geophysical Journal International*, 116(1):64–84, 1994.

- [70] G. W. Stewart. On the early history of the singular value decomposition. *SIAM review*, 35(4):551–566, 1993.
- [71] R. C. Thompson. Principal submatrices ix: Interlacing inequalities for singular values of submatrices. *Linear Algebra and its Applications*, 5(1):1–12, 1972.
- [72] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
- [73] V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.
- [74] A. Verikas and M. Bacauskiene. Feature selection with neural networks. *Pattern Recognition Letters*, 23(11):1323–1335, 2002.
- [75] S. Wang, W. Pedrycz, Q. Zhu, and W. Zhu. Unsupervised feature selection via maximum projection and minimum redundancy. *Knowledge-Based Systems*, 75:19–29, 2015.
- [76] S. Wang, J. Tang, and H. Liu. Embedded unsupervised feature selection. In *AAAI*, pages 470–476, 2015.
- [77] Z. Xu, I. King, M. R.-T. Lyu, and R. Jin. Discriminative semi-supervised feature selection via manifold regularization. *IEEE Transactions on Neural networks*, 21(7):1033–1047, 2010.
- [78] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, pages 412–420, 1997.

- [79] Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th international conference on Machine learning*, pages 1151–1157. ACM, 2007.
- [80] Z. Zhao, L. Wang, H. Liu, et al. Efficient spectral feature selection with minimum redundancy. In *AAAI*, pages 673–678, 2010.

Appendix A

Proofs

Proof of proposition 1

Proof. Zeroing out the column i of matrix C can be seen as the following rank-1 update of C :

$$\tilde{C} = C + cd^T$$

where $c = C_{:,i}$ and $d = -e_i$ is minus the i -th vector of the canonical basis of a k -dimensional vector space (all zeros and a -1 in position i). It is obvious that c lies in the column space of C . Additionally, $\text{rank}(C) = k$. Therefore, $d^T \in \mathbb{R}^{1 \times k}$ lies in the row space of C . Let us define $\beta = 1 + d^T C^+ c$. Then

$$\beta = 1 - (C^+)_{i,:} c = 1 - ((C^T C)^{-1} C^T)_{i,:} c = 1 - 1 = 0$$

This means that this update always corresponds to case (vi) of [57] and the generalized inverse can be updated as

$$\tilde{C}^+ = C^+ - k k^+ C^+ - C^+ h^+ h + k^+ C^+ h^+ k h$$

where $k = C^+c$ and $h = d^T C^+ = (C^+)_{i:} = \rho^T$. Now, we have the three following equalities:

$$\begin{aligned} kk^+ C^+ &= C^+ c (C^+ c)^+ C^+ = e_i e_i^T C^+ = (C^+)_{i:} \\ C^+ h^+ h &= C^+ \frac{h^T}{\|h\|_2^2} h = \|\rho\|_2^{-2} C^+ \rho \rho^T \\ k^+ C^+ h^+ k h &= (C^+ c)^+ C^+ (\rho^T)^+ k h = k h = (C^+)_{i:} \end{aligned}$$

Hence,

$$\tilde{C}^+ = C^+ - \|\rho\|_2^{-2} C^+ \rho \rho^T$$

□

Proof of proposition 2

Proof. Let $G = -\|\rho\|_2^{-2} C^+ \rho \rho^T$ (Proposition 1). Then

$$\begin{aligned} \tilde{E} &= A - \tilde{C} \tilde{C}^+ A \\ &= A - \tilde{C} (C^+ + G) A \\ &= A - \tilde{C} C^+ A - \tilde{C} G A \end{aligned}$$

Since $\tilde{C} C^+ A = C C^+ A - C_{:i} (C^+ A)_{i:}$

$$\begin{aligned} \tilde{E} &= E + C_{:i} (C^+ A)_{i:} - \tilde{C} G A \\ &= E + C_{:i} \rho^T A - \tilde{C} (-\|\rho\|_2^{-2} C^+ \rho \rho^T) A \end{aligned}$$

□

Proof of proposition 3.

Proof. From corollary 3 in [28],

$$\tilde{F} = F + \frac{\delta\delta^T}{\delta_j}$$

Therefore, for $k = 1, \dots, n$ we have

$$\begin{aligned} \tilde{f}_k &= \|\tilde{F}_{:k}\|_2^2 = \sum_i (F_{ik} + \frac{\delta_i\delta_k}{\delta_j})^2 \\ &= \sum_i F_{ik}^2 + \left(\frac{\delta_i\delta_k}{\delta_j}\right)^2 + 2F_{ik}\frac{\delta_i\delta_k}{\delta_j} \\ &= \|F_{:k}\|_2^2 + \frac{\delta_k^2}{\delta_j^2}\|\delta\|_2^2 + 2\sum_i F_{ik}\frac{\delta_i\delta_k}{\delta_j} \end{aligned}$$

Additionally, we have the following:

$$\sum_i F_{ik}\frac{\delta_i\delta_k}{\delta_j} = \frac{\delta_k}{\delta_j} \sum_i F_{ik}\delta_i = \frac{\delta_k}{\delta_j}(F_{:k})^T\delta = \frac{\delta_k}{\delta_j}\gamma_k$$

Therefore

$$\tilde{f}_k = \|F_{:k}\|_2^2 + \frac{\delta_k^2}{\delta_j^2}\|\delta\|_2^2 + 2\frac{\delta_k}{\delta_j}\gamma_k = f_k + \frac{\delta_k^2}{\delta_j^2}\|\delta\|_2^2 + 2\frac{\delta_k}{\delta_j}\gamma_k$$

Considering the whole vector $\tilde{f}$, we retrieve the first equality we wanted to prove.

The case of $\tilde{g}$ is trivial, given that

$$\tilde{g}_k = F_{kk} + \frac{\delta_k\delta_k}{\delta_j}$$

□

Proof of proposition 4.

Proof. Adding the column $A_{:w}$ to matrix $\tilde{C}$ can be seen as the following rank-1 update:

$$C = \tilde{C} + cd^T$$

where $c = A_{:w}$ and $d = e_i$ is the i -th vector of the canonical basis of a k -dimensional vector space (all zeros and a 1 in position i).

We assume that the addition of column $A_{:w}$ increases the rank of $\tilde{C}$, which implies that c is not in the column space of $\tilde{C}$. In addition, since $\tilde{C}_{:,i} = 0$, $d^T = e_i^T$ is not in the rowspace of $\tilde{C}$. Therefore, we are in case (i) of [57]. We define $\beta = 1 + d^T \tilde{C}^+ c$. The generalized inverse can thus be updated as

$$C^+ = \tilde{C}^+ - ku^+ - v^+h + \beta v^+u^+$$

where $k = \tilde{C}^+ c$, $h = d^T \tilde{C}^+ = (\tilde{C}^+)_{i,:}$, $u = (I - \tilde{C} \tilde{C}^+)c$ and $v = d^T(I - \tilde{C}^+ \tilde{C})$. Since $\text{rank}(\tilde{C}) = k - 1$ and $\tilde{C}_{:,i} = 0$, $I - \tilde{C}^+ \tilde{C}$ is comprised exclusively of zeros, except for $\tilde{C}_{ii} = 1$. Therefore, $v = e_i^T$. We therefore have

$$C^+ = \tilde{C}^+ - \frac{\tilde{C}^+ c(c - \tilde{C} \tilde{C}^+ c)^T}{\|c - \tilde{C} \tilde{C}^+ c\|_2^2} - H + \frac{\beta e_i(c - \tilde{C} \tilde{C}^+ c)^T}{\|c - \tilde{C} \tilde{C}^+ c\|_2^2}.$$

where H is a zero matrix with h in its i -th row. Now, since $\tilde{C}_{:,i} = 0$, then $h = (\tilde{C}^+)_{i,:} = 0$ and $\beta = 1$. Also, note that $c - \tilde{C} \tilde{C}^+ c = \tilde{E}_{:w} = \omega$. Therefore,

$$C^+ = \tilde{C}^+ - \|\omega\|_2^{-2}(\tilde{C}^+ A_{:w} \omega^T - e_i \omega^T)$$

□

Proof of theorem 2.

Proof. For some $S \in \mathcal{S}^{n \times k}$, let $C = AS$ and $\tilde{C} = \Sigma V^T S$. Since U is orthogonal and its first n columns span the column space of A , for all $S \in \mathcal{S}^{n \times k}$

$$\begin{aligned} \|A - CC^+A\|_F^2 &= \|U^T(A - CC^+A)\|_F^2 \\ &= \|U^T A - U^T CC^+A\|_F^2 \\ &= \|\Sigma V^T - \tilde{C}C^+A\|_F^2 \end{aligned}$$

The first equality holds also for the “thin” SVD because U is column-wise orthogonal and CC^+A lies in the column space of A , and thus so does $A - CC^+A$. We have that

$$\begin{aligned} \tilde{C}^T \tilde{C} &= (\Sigma V^T S)^T \Sigma V^T S = S^T V \Sigma \Sigma V^T S \\ &= S^T A^T U U^T A S = (AS)^T A S = C^T C \end{aligned}$$

Now, since $C^+ = (C^T C)^{-1} C^T$, $\tilde{C}^+ = (\tilde{C}^T \tilde{C})^{-1} \tilde{C}^T$ and $\tilde{C} = \Sigma V^T S = U^T A S = U^T C$, we have that

$$\begin{aligned} \tilde{C}^+ \Sigma V^T &= (\tilde{C}^T \tilde{C})^{-1} \tilde{C}^T \Sigma V^T = (C^T C)^{-1} \tilde{C}^T \Sigma V^T \\ &= (C^T C)^{-1} (U^T C)^T \Sigma V^T = (C^T C)^{-1} C^T U \Sigma V^T = C^+ A \end{aligned}$$

Which implies that

$$\begin{aligned} \|A - (AS)(AS)^+A\|_F^2 &= \|A - CC^+A\|_F^2 \\ &= \|\Sigma V^T - \tilde{C}C^+A\|_F^2 = \|\Sigma V^T - \tilde{C}\tilde{C}^+\Sigma V^T\|_F^2 \\ &= \|\Sigma V^T - \Sigma V^T S(\Sigma V^T S)^+\Sigma V^T\|_F^2 \end{aligned}$$

Note that $\text{rank}(A) = n$ and $n < m$, so $\text{rank}(C) = k$ and $(C^T C)^{-1}$ exists. Therefore,

the equality in (5.4) holds for all values of S , which means that the minimum will be attained at the same argument. $\square$

Proof of theorem 3.

Proof.

$$\begin{aligned} f_i &= \|(E^T E)_{:i}\|_2^2 = \|(V \Sigma U^T U \Sigma V^T)_{:i}\|_2^2 \\ &= \|(V \Sigma^2 V^T)_{:i}\|_2^2 = \|(\Sigma^2 V^T)_{:i}\|_2^2 \end{aligned}$$

The last equality holds because V is orthogonal. On the other hand,

$$\begin{aligned} g_i &= \|E_{i:}\|_2^2 = (E_{i:})^T E_{i:} = (V \Sigma U^T)_{i:} (U \Sigma V^T)_{:i} \\ &= (V \Sigma)_{i:} (\Sigma V^T)_{:i} = \|(\Sigma V^T)_{:i}\|_2^2 \end{aligned}$$

$\square$

Proof of lemma 2.

Proof. Let us consider that at the beginning of some iteration (Algorithm 1, line 6), we are working with matrix $C \in \mathbb{R}^{m \times k}$ (column subset of A).

Let us denote $\tilde{C}_i = (C \setminus i)$. For all i , we can decompose CC^+A as

$$CC^+A = \hat{C}_i + \bar{C}_i$$

where $\hat{C}_i = \tilde{C}_i \tilde{C}_i^+ A$ is the part that lies in the span of the columns of $\tilde{C}_i$, and $\bar{C}_i = CC^+A - \tilde{C}_i \tilde{C}_i^+ A$ is the part that is orthogonal to them. Note that $\|\bar{C}_i\|_F^2$ is what we lose from $\|CC^+A\|_F^2$ if we remove column i from C .

Of course,

$$\|\bar{C}_i\|_F^2 \leq \|CC^+A\|_F^2$$

We can further tighten this bound by taking into account that after removing any column i , we keep the $k - 1$ columns of $\tilde{C}_i$, whose norm is kept intact. That is, the quantity that could be lost by the removal of a column i is bounded as

$$\|\bar{C}_i\|_F^2 \leq \|CC^+A\|_F^2 - \|\tilde{C}_i\|_F^2$$

To make this bound independent of the particular column that was removed, we proceed as follows. Since $\tilde{C}_i$ is an $m \times k - 1$ submatrix of A , we invoke the interlacing property of the singular values to see that $\sigma_i(\tilde{C}_i) \geq \sigma_{n-k+i+1}(A)$ for $i = 1, \dots, k - 1$. Since for every matrix W it is true that $\|W\|_F^2 = \sum_i \sigma_i^2(W)$, we can state that $\|\tilde{C}_i\|_F^2 \geq \sum_{i=n-k+2}^n \sigma_i^2(A)$. Furthermore, we can assume there is a column w in C such that the loss is a k^{-1} -factor of the total (in the alternative case, we can easily derive an approximation equivalent to that of lemma 3, so here we focus on this worst-case condition). Therefore,

$$\|\bar{C}_w\|_F^2 \leq \frac{1}{k} \left(\|CC^+A\|_F^2 - \sum_{i=n-k+2}^n \sigma_i^2(A) \right)$$

Combining the above result with lemma 1, we can conclude that there is at least one column in C whose replacement will yield at least the following improvement:

$$\sigma_k^2(B) \frac{(\|BB^+A\|_F^2 - \|\tilde{C}\tilde{C}^+A\|_F^2)^2}{4k\|BB^+A\|_F^2} - \frac{1}{k} \left(\|CC^+A\|_F^2 - \sum_{i=n-k+2}^n \sigma_i^2(A) \right) \quad (\text{A.1})$$

That is, if the above quantity is greater than zero for some column of C , the algorithm will not stop. This implies that, once the algorithm has stopped, the

above quantity will be less than or equal to zero. That is, after some elementary manipulations we can state the following:

$$\|CC^+A\|_F^2 \geq \sigma_k^2(B) \frac{(\|BB^+A\|_F^2 - \|\tilde{C}\tilde{C}^+A\|_F^2)^2}{4\|BB^+A\|_F^2} + \sum_{i=n-k+2}^n \sigma_i^2(A)$$

Finally, since $\tilde{C}\tilde{C}^+A$ is a (likely suboptimal) rank $k-1$ approximation of BB^+A ,

$$\|CC^+A\|_F^2 \geq \sigma_k^2(B) \frac{\sigma_k^4(BB^+A)}{4\|BB^+A\|_F^2} + \sum_{i=n-k+2}^n \sigma_i^2(A)$$

From which our result follows. □

Proof of lemma 3.

Proof. Define $\tilde{C}_i = C \setminus i$. Assume $\|\tilde{C}_i\tilde{C}_i^+A\| < c\|BB^+A\|_F^2$ for some i and some constant c such that

$$0 < c \leq \frac{\sigma_k^2(B)}{k} \leq \frac{1}{k}$$

Then

$$\begin{aligned}
& \|CC^+A\|_F^2 - \|\tilde{C}_i\tilde{C}_i^+A\|_F^2 \geq \sigma_k^2(B) \frac{\left(\|BB^+A\|_F^2 - \|\tilde{C}_i\tilde{C}_i^+A\|_F^2\right)^2}{4k\|BB^+A\|_F^2} \\
& \Rightarrow \|CC^+A\|_F^2 - \|\tilde{C}_i\tilde{C}_i^+A\|_F^2 \geq \sigma_k^2(B) \frac{\left(\|BB^+A\|_F^2 - \frac{1}{k}\|BB^+A\|_F^2\right)^2}{4k\|BB^+A\|_F^2} \\
& \Rightarrow \|CC^+A\|_F^2 - \|\tilde{C}_i\tilde{C}_i^+A\|_F^2 \geq \sigma_k^2(B) \frac{\left(\frac{k-1}{k}\|BB^+A\|_F^2\right)^2}{4k\|BB^+A\|_F^2} \\
& \Rightarrow \|CC^+A\|_F^2 - \|\tilde{C}_i\tilde{C}_i^+A\|_F^2 \geq \sigma_k^2(B) \frac{\left(\frac{k-1}{k}\right)^2\|BB^+A\|_F^2}{4k} \\
& \Rightarrow \|CC^+A\|_F^2 - \|\tilde{C}_i\tilde{C}_i^+A\|_F^2 \geq \sigma_k^2(B) \left(\frac{(k^2 + 1 - 2k)\|BB^+A\|_F^2}{4k^3}\right) \\
& \Rightarrow \|CC^+A\|_F^2 \geq O\left(\frac{\sigma_k^2(B)}{k}\right) \|BB^+A\|_F^2
\end{aligned}$$

□

Proof of theorem 4.

Proof. Equality 6.5 gives an expression that reveals the optimum of the objective function of problem 6. We now show that this expression can be computed in $O(\min\{np, n^2\})$ time.

First, observe from equalities 6.8 and 6.9 that the values of x_w and $\tilde{x}_w$ for this iteration can be computed in $O(nt)$ time complexity as follows:

$$\begin{aligned}
x_w &:= X_{:w}^{(0)} + \sum_{j=0}^{t-1} \left(\frac{1}{a_w} x_w x_{ww}^T \right)^{(j)} \\
\tilde{x}_w &:= \sum_{j=0}^{t-1} \left(\frac{x_{ww}}{\alpha_w} \tilde{x}_w + \frac{\tilde{x}_{ww}}{\alpha_w} x_w + \frac{\tilde{x}_{ww} - x_{ww}}{\alpha_w^2} x_w x_{ww} - \frac{x_{ww}}{\alpha_w} x_w \right)^{(j)}
\end{aligned}$$

We now define the matrices $W, \tilde{W}$ whose columns are respectively $x_w^{(0)}, \dots, x_w^{(t-1)}$ and $\tilde{x}_w^{(0)}, \dots, \tilde{x}_w^{(t-1)}$. We also define the diagonal matrix

$$B = \begin{pmatrix} \frac{1}{\alpha_w^{(0)}} & & \\ & \ddots & \\ & & \frac{1}{\alpha_w^{(t-1)}} \end{pmatrix}$$

and the column-scaled matrices $W_\alpha = WB, \tilde{W}_\alpha = \tilde{W}B$. The following equalities can be easily verified.

$$\begin{aligned} \left(\sum_{j=0}^{t-1} \left(\frac{1}{a_w} x_w x_{wi} \right)^{(j)} \right)^T (x_w)^{(t)} &= W_{i:} W_\alpha^T x_w \\ \left(\sum_{j=0}^{t-1} \left(\frac{1}{a_w} x_w x_{wi} \right)^{(j)} \right)^T (\tilde{x}_w)^{(t)} &= W_{i:} W_\alpha^T \tilde{x}_w \end{aligned}$$

Combined with equalities 6.8 and 6.9, this implies the following:

$$(x_1^T x_w, \dots, x_n^T x_w)^T = X^{(0)} x_w + WW_\alpha^T x_w$$

$$(x_1^T \tilde{x}_w, \dots, x_n^T \tilde{x}_w)^T = X^{(0)} \tilde{x}_w + WW_\alpha^T \tilde{x}_w$$

And similarly,

$$\begin{aligned} (\tilde{x}_1^T x_w, \dots, \tilde{x}_n^T x_w)^T &= \tilde{X}^T x_w \\ &= W\tilde{W}_\alpha^T x_w + \tilde{W}W_\alpha^T x_w + \alpha^{-1}(\tilde{x}_w - x_w)WW_\alpha^T x_w + WW_\alpha^T x_w \end{aligned}$$

Since $X^{(0)} \in \mathbb{R}^{n \times n}$, $W, \tilde{W} \in \mathbb{R}^{n \times t}$ and $x_w, \tilde{x}_w \in \mathbb{R}^{n \times 1}$, these equalities can be computed in $O(2n^2 + 8nt) = O(n^2)$. If $n > m$, instead of storing $X^{(0)}$ we can explicitly

compute $-A^T A x_w$ and $-A^T A \tilde{x}_w$, resulting in $O(\max\{mn, nt\})$ time complexity.

Using the variables $x_i^T x_w$, $x_i^T \tilde{x}_w$ and $\tilde{x}_i^T x_w$, which we have computed for $i = 1, \dots, n$, we can now compute $\tilde{x}_i^T x_i$ and $x_i^T x_i$ as shown in equalities 6.10 and 6.11. Finally, from equalities 6.8 and 6.9 and the definition of α_i , it is easily verified that

$$(x_{ii})^{(t)} = (x_{ii})^{(t-1)} + \frac{1}{(\alpha_w)^{(t-1)}} x_{wi}^2$$

$$(\tilde{x}_{ii})^{(t)} = (\tilde{x}_{ii})^{(t-1)} + \frac{2\tilde{x}_{wi}x_{wi} - x_{wi}^2}{(\alpha_w)^{(t-1)}} + x_{wi}^2 \left(\frac{\tilde{x}_{ii} - x_{ii}}{\alpha_w^2} \right)^{(t-1)}$$

$$\alpha_i = \lambda - x_{ii}$$

These operations are easily seen to require $O(n)$ time complexity if computed for all i .

Having computed all these variables, we can now compute the value of expression 6.5 for all i in a straightforward manner, thus completing the proof.

□

Appendix B

Detailed derivations

Here we show more detailed derivations of some of the equalities in the dissertation.

Equality 6.3:

$$\begin{aligned} A^{(t+1)} &= C_w(C_w^T C_w + \lambda I)^{-1} C_w^T A \\ &= (C \ w)(C_w^T C_w + \lambda I)^{-1} (C \ w)^T A \\ &= C(C^T C + \lambda I)^{-1} C^T A + C \frac{vv^T}{\alpha_w} C^T A - w \frac{v^T}{\alpha_w} C^T A - C \frac{vw^T}{\alpha_w} A + \frac{ww^T}{\alpha_w} A \\ &= A^{(t)} + \frac{(A_{:,w} A_{:,w}^T)^{(t)}}{\alpha_w} A - w \frac{(A_{:,w}^T)^{(t)}}{\alpha_w} A - \frac{A_{:,w}^{(t)} w^T}{\alpha_w} A + \frac{ww^T}{\alpha_w} A \\ &= A^{(t)} + \frac{1}{\alpha_w} ((A_{:,w}^{(t)} - w)(A_{:,w}^{(t)} - w)^T) A \end{aligned}$$

Equality 6.4:

$$\begin{aligned}
& \operatorname{argmin}_i \|A - A^{(t+1)}\|_F^2 \\
&= \operatorname{argmin}_i \operatorname{tr}((A - A^{(t+1)})^T (A - A^{(t+1)})) \\
&= \operatorname{argmin}_i \operatorname{tr}(A^T A) - \operatorname{tr}(A^T A^{(t+1)}) - \operatorname{tr}((A^T)^{(t+1)} A^T) + \operatorname{tr}((A^T A)^{(t+1)}) \\
&= \operatorname{argmin}_i \operatorname{tr}(A^T A) - \operatorname{tr}(A^T (A^{(t)} + \frac{1}{\alpha_i} d_i d_i^T A)) \\
&\quad - \operatorname{tr}((A^{(t)} + \frac{1}{\alpha_i} d_i d_i^T A)^T A) + \operatorname{tr}((A^{(t)} + \frac{1}{\alpha_i} d_i d_i^T A)^T (A^{(t)} + \frac{1}{\alpha_i} d_i d_i^T A)) \\
&= \operatorname{argmin}_i \operatorname{tr}(A^T A) - \operatorname{tr}(A^T A^{(t)}) - \operatorname{tr}(\frac{1}{\alpha_i} A^T d_i d_i^T A) \\
&\quad - \operatorname{tr}((A^T)^{(t)} A) - \operatorname{tr}(\frac{1}{\alpha_i} A^T d_i d_i^T A) \\
&\quad + \operatorname{tr}((A^T)^{(t)} A^{(t)}) + \operatorname{tr}(\frac{1}{\alpha_i} (A^T)^{(t)} d_i d_i^T A) + \operatorname{tr}(\frac{1}{\alpha_i} A^T d_i d_i^T A^{(t)}) \\
&\quad + \operatorname{tr}(\frac{1}{\alpha_i^2} A^T d_i d_i^T d_i d_i^T A)
\end{aligned}$$

Equality 6.5:

$$\begin{aligned}
& \operatorname{argmin}_i \|A - A^{(t+1)}\|_F^2 \\
&= -2\operatorname{tr}(\frac{1}{\alpha_i} A^T d_i d_i^T A) + 2\operatorname{tr}(\frac{1}{\alpha_i} (A^T)^{(t)} d_i d_i^T A) + \operatorname{tr}(\frac{1}{\alpha_i^2} A^T d_i d_i^T d_i d_i^T A) \\
&= \operatorname{argmin}_i \frac{2}{\alpha_i} \operatorname{tr}((A^T)^{(t)} d_i d_i^T A) - \frac{2}{\alpha_i} \operatorname{tr}(A^T d_i d_i^T A) + \frac{1}{\alpha_i^2} \|d_i d_i^T A\|_F^2 \\
&= \operatorname{argmin}_i \frac{2}{\alpha_i} \tilde{x}_i^T x_i - \frac{2}{\alpha_i} \|x_i\|_2^2 + \frac{1}{\alpha_i^2} \|d_i\|_2^2 \|x_i\|_2^2 \\
&= \operatorname{argmin}_i \frac{2}{\alpha_i} (\tilde{x}_i - x_i)^T x_i + \frac{1}{\alpha_i^2} \|x_i\|_2^2 (\tilde{x}_{ii} - x_{ii}) \\
&= \operatorname{argmin}_i \frac{2}{\alpha_i} \tilde{x}_i^T x_i + (\frac{-2}{\alpha_i} + \frac{1}{\alpha_i^2} (\tilde{x}_{ii} - x_{ii})) x_i^T x_i + \lambda
\end{aligned}$$

Equality 6.6

$$\begin{aligned}
A^T A^{(t+1)} &= A^T (A^{(t)} + \frac{1}{\alpha_w} d_w d_w^T A) \\
&= A^T A^{(t)} + \frac{1}{\alpha_w} A^T d_w d_w^T A \\
&= A^T A^{(t)} + \frac{1}{\alpha_w} x_w x_w^T
\end{aligned}$$

Equality 6.7

$$\begin{aligned}
(A^T A)^{(t+1)} &= (A^{(t)} + \frac{1}{\alpha_w} d_w d_w^T A)^T (A^{(t)} + \frac{1}{\alpha_w} d_w d_w^T A) \\
&= (A^T A)^{(t)} + \frac{1}{\alpha_w} (A^T)^{(t)} d_w d_w^T A + \frac{1}{\alpha_w} A^T d_w d_w^T A^{(t)} + \frac{1}{\alpha_w^2} A^T d_w d_w^T d_w d_w^T A \\
&= (A^T A)^{(t)} + \frac{1}{\alpha_w} \tilde{x}_w x_w^T + \frac{1}{\alpha_w} x_w \tilde{x}_w^T + \frac{1}{\alpha_w^2} x_w d_w^T d_w x_w^T
\end{aligned}$$

Equality 6.8

$$\begin{aligned}
X^{(t+1)} &= A^T D^{(t+1)} \\
&= A^T A^{(t+1)} - A^T A \\
&= A^T A^{(t)} + \frac{1}{\alpha_w} x_w x_w^T - A^T A \\
&= A^T D^{(t)} + A^T A + \frac{1}{\alpha_w} x_w x_w^T - A^T A \\
&= X^{(t)} + \frac{1}{\alpha_w} x_w x_w^T = X^{(0)} + \sum_{i=0}^t \left(\frac{1}{\alpha_w} x_w x_w^T \right)^{(i)}
\end{aligned}$$

Equality 6.9

$$\begin{aligned}
\tilde{X}^{(t+1)} &= (A^T D)^{(t+1)} \\
&= (A^T A)^{(t+1)} - (A^T)^{(t+1)} A \\
&= (A^T A)^{(t)} + \frac{1}{\alpha_w} \tilde{x}_w x_w^T + \frac{1}{\alpha_w} x_w \tilde{x}_w^T + \frac{1}{\alpha_w^2} x_w d_w^T d_w x_w^T - (A^T)^{(t)} A - \frac{1}{\alpha_w} x_w x_w^T \\
&= (A^T A)^{(t)} - (A^T)^{(t)} A + \frac{1}{\alpha_w} \tilde{x}_w x_w^T + \frac{1}{\alpha_w} x_w \tilde{x}_w^T + \frac{1}{\alpha_w^2} x_w d_w^T d_w x_w^T - \frac{1}{\alpha_w} x_w x_w^T \\
&= (A^T D)^{(t)} + \frac{1}{\alpha_w} \tilde{x}_w x_w^T + \frac{1}{\alpha_w} x_w \tilde{x}_w^T + \frac{1}{\alpha_w^2} x_w d_w^T d_w x_w^T - \frac{1}{\alpha_w} x_w x_w^T \\
&= X^{(0)} + \sum_{i=0}^t \left(\frac{1}{\alpha_w} \tilde{x}_w x_w^T + \frac{1}{\alpha_w} x_w \tilde{x}_w^T + \frac{1}{\alpha_w^2} x_w d_w^T d_w x_w^T - \frac{1}{\alpha_w} x_w x_w^T \right)^{(i)}
\end{aligned}$$