ABSTRACT

Title of Dissertation: CUR MATRIX APPROXIMATION THROUGH CONVEX OPTIMIZATION

Kathryn Linehan
Doctor of Philosophy, 2024

Dissertation Directed by: Professor Radu V. Balan
Department of Mathematics,
Center for Scientific Computation and
Mathematical Modeling

In this dissertation we present work on the CUR matrix approximation. Specifically, we present 1) an approximation of the proximal operator of the ℓ_∞ norm using a neural network, 2) a novel deterministic CUR formulation and algorithm, and 3) a novel application of CUR as a feature selection method to determine discriminant proteins when clustering protein expression data in a self-organizing map (SOM). The proximal operator of the ℓ_∞ norm arises in our CUR algorithm.

Since the computation of the proximal operator of the ℓ_∞ norm requires a sort of the input data (or at least a partial sort similar to quicksort), we present a neural network to approximate the proximal operator. A novel aspect of the network is that it is able to accept vectors of varying lengths due to a feature selection process that uses moments of the input data. We present results on the accuracy of the approximation, feature importance, and computational efficiency of the approach, and present an algorithm to calculate the proximal operator of the ℓ_∞ norm exactly,

relate it to the Moreau decomposition, and compare its computational efficiency to that of the approximation.

Next, we present a novel deterministic CUR formulation that uses convex optimization to form the matrices $\mathbf{C}$ and $\mathbf{R}$, and a corresponding algorithm that uses bisection to ensure that the user selected number of columns appear in $\mathbf{C}$ and the user selected number of rows appear in $\mathbf{R}$. We implement the algorithm using the surrogate functional technique of Daubechies et al. [*Communications on Pure and Applied Mathematics*, 57.11 (2004)] and extend the theory of this approach to apply to our CUR formulation. Numerical results are presented that demonstrate the effectiveness of our CUR algorithm as compared to the singular value decomposition (SVD) and other CUR algorithms.

Last, we use our CUR approximation as a feature selection method in the application by Higuera et al. [*PLOS ONE*, 10(6) (2015)] to determine discriminant proteins when clustering protein expression data in an SOM. This is a novel application of CUR and to the best of our knowledge, this is the first use of CUR on protein expression data. We compare the performance of our CUR algorithm to other CUR algorithms and the Wilcoxon rank-sum test (the original feature selection method in the work).

CUR MATRIX APPROXIMATION THROUGH CONVEX OPTIMIZATION

by

Kathryn Joyce Linehan

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2024

Advisory Committee:
Professor Radu V. Balan, Chair/Advisor
Professor Howard Elman
Professor Maria Cameron
Professor Tom Goldstein
Professor Millard Alexander, Dean's Representative

Acknowledgments

I would like to thank my advisor, Radu Balan, for his guidance throughout this work. Radu has been an exceptional advisor from whom I have learned a great deal. I would like to thank Millard Alexander, Howard Elman, Maria Cameron and Tom Goldstein for serving on my committee, and Sallie Keller for encouraging me to complete this degree. I thank my undergraduate mentor and now friend, Betty Mayfield, for her support throughout my career.

I am grateful for the support of my family, friends, and colleagues throughout this process and especially want to thank my boyfriend Aaron for his love and support. I would not have been able to complete this without him. And last but not least, I thank my cat for his loving companionship.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Low rank matrix approximations are common tools in many applications including principal component analysis (PCA), signal denoising, and least squares. While the truncated singular value decomposition (SVD) is the optimal approximation in terms of matrix reconstruction (Eckart-Young theorem), the singular vectors cannot be interpreted in terms of the original data. Mahoney and Drineas [1] provided an example of this: $[\frac{1}{2}\text{age} - \frac{1}{\sqrt{2}}\text{height} + \frac{1}{2}\text{income}]$ is an eigenvector for a dataset of features about people that "is not particularly informative or meaningful". In addition, several applications exist that seek important matrix columns or rows [2], e.g., selecting important genes from gene expression data in order to cluster patients by cancer type [1]. Hence, in this thesis we are interested in the approximate CUR matrix factorization which can be interpreted in terms of the original data.

The approximate CUR factorization of $\mathbf{X} \in \mathbb{R}^{m \times n}$ is generally computed in three steps: 1) select $c \in \mathbb{R}$ columns of $\mathbf{X}$ and let $\mathbf{C} \in \mathbb{R}^{m \times c}$ contain these columns, 2) select $r \in \mathbb{R}$ rows of $\mathbf{X}$ and let $\mathbf{R} \in \mathbb{R}^{r \times n}$ contain these rows, and 3) compute $\mathbf{U} \in \mathbb{R}^{c \times r}$ so that $\mathbf{CUR}$ is a good approximation to $\mathbf{X}$. The result is a matrix approximation

$$\underset{m \times n}{\mathbf{X}} \approx \underset{m \times c}{\mathbf{C}} \; \underset{c \times r}{\mathbf{U}} \; \underset{r \times n}{\mathbf{R}},$$

where generally $c \ll n$ and $r \ll m$. $\mathbf{C}$ and $\mathbf{R}$ can be viewed as containing the most important

columns and rows of the original data, respectively.

While Hamm and Huang [3] provide a history of CUR, they also mention that recent work on the CUR approximation most likely began with developments in the mid-to-late 1990s by Goreinov, Tyrtyshnikov, and Zamarashkin, e.g., [4]. Since then, several CUR algorithms have been developed; some are randomized, e.g., [1,5,6], and others are deterministic, e.g., [2,7]. Work on CUR includes proving accuracy and/or other theoretical guarantees for algorithms, e.g., [1,5], and also performance of CUR algorithms in practice without theoretical guarantees, e.g., [8]. In addition, CUR has been used for a variety of applications such as clustering documents [1], classification using genetic data [2], image feature selection for classification problems [9], video background-foreground separation [10], and sensor selection and channel assignment [11].

The focus of this thesis is a new CUR algorithm that utilizes convex optimization to select columns and rows for inclusion in $\mathbf{C}$ and $\mathbf{R}$. However, in Chapter 2 we begin by researching the proximal operator of the ℓ_∞ norm, $\mathbf{prox}_{\alpha||\cdot||_\infty} : \mathbb{R}^m \to \mathbb{R}^m$, defined as

$$\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x}) = \underset{\mathbf{y}\in\mathbb{R}^m}{\operatorname{argmin}} \left(\frac{1}{2}||\mathbf{y}-\mathbf{x}||_2^2 + \alpha||\mathbf{y}||_\infty \right)$$

for $\mathbf{x} \in \mathbb{R}^m$ and $\alpha \geq 0$. This proximal operator occurs as a repeated calculation in our CUR algorithm. Algorithms exist to compute $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ that are based on the Moreau decomposition and projection onto the simplex (see e.g., [12]). However, we present properties of $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ and an algorithm for its computation, which motivate a neural network approach to approximate $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$. The network is novel because we use a preprocessing and feature selection process that allows the network to accept vectors of varying lengths. The network approach has $O(m)$ complexity and we provide numerical experiments that demonstrate its computationally

efficiency. We also show numerical results on the approximation accuracy.

In Chapter 3, we present our CUR algorithm that uses convex optimization. While there is prior work in using convex optimization for CUR, e.g., [8, 13], our formulation is novel, and our algorithm solves for $\mathbf{C}$ and $\mathbf{R}$ separately and allows the user to select the number of columns and rows to include in $\mathbf{C}$ and $\mathbf{R}$. In addition, we make contributions in the implementation of our algorithm by utilizing the "surrogate functional" technique [14], which we adapt for use with a new penalty function by proving convergence of our algorithm to a minimum of the objective function. Furthermore, we provide numerical results that compare our CUR algorithm with the SVD and other deterministic CUR algorithms. Specifically, we show that our CUR algorithm performs very well as a feature selection method in an experiment from [2] on gene expression data.

In Chapter 4, we present a novel application of CUR for feature selection. We adapt the experiments of Higuera et al. [15] in which Self-Organizing Maps (SOMs) and the Wilcoxon rank-sum test were used to determine proteins that critically affect learning in wild type and trisomic (Down syndrome) mice. Specifically, we use CUR as the feature selection method instead of the Wilcoxon rank-sum test. We compare the performance of our CUR algorithm to that of other deterministic CUR algorithms, and show that CUR can be used effectively in this application. This is not only a novel application of CUR, but to the best of our knowledge, also the first use of CUR on protein expression data. In Chapter 5, we provide concluding remarks for the thesis.

Chapter 2: Approximation of the Proximal Operator of the ℓ_∞ Norm Using a Neural Network

2.1 Introduction

Proximal operators frequently serve as building blocks for solving complex optimization problems. For example, proximal operators commonly arise in the method of alternating direction method of multipliers (ADMM) [16, 17]. In this chapter we focus on the proximal operator of the ℓ_∞ norm, $\mathbf{prox}_{\alpha||\cdot||_\infty} : \mathbb{R}^m \to \mathbb{R}^m$, defined as

$$\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x}) = \underset{\mathbf{y}\in\mathbb{R}^m}{\operatorname{argmin}} \left(\frac{1}{2}||\mathbf{y}-\mathbf{x}||_2^2 + \alpha||\mathbf{y}||_\infty \right) \tag{2.1}$$

for $\mathbf{x} \in \mathbb{R}^m$ and $\alpha \geq 0$. We discuss theoretical properties of $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$, which lead to an $O(m \log m)$ algorithm for its exact computation. We relate this approach to that which uses the Moreau decomposition to compute $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$. Computing $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ requires a sort of the input data, or at least a partial sort similar to quicksort. We present a neural network approach to approximate $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ which does not require a sort and is computationally efficient with $O(m)$ complexity. A novel piece of this work is that we use a preprocessing and feature selection process that allows the network to accept vectors of varying lengths. This is a desired property since 1) $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ can be computed for a vector of any length, and 2) this eliminates the need

for multiple networks, i.e. one for each unique vector length in the input data.

The remainder of this chapter is organized as follows: related work is presented in Section 2.2; the theoretical properties, algorithm to exactly compute $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$, and the relationship between this algorithm and the Moreau decomposition are provided in Section 2.3; the neural network approximation of $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ and results on the approximation accuracy, feature importance, and computational efficiency including a comparison with the exact algorithm are given in Section 2.4; and the chapter concludes with Section 2.5. Throughout this work vectors are denoted by bold, lowercase letters, the vector $(|\mathbf{v}_1|, |\mathbf{v}_2|, \ldots, |\mathbf{v}_m|)$ is denoted as $|\mathbf{v}|$, and the set $\{1, 2, \ldots, m\}$ is denoted as $[m]$. The notation $\mathbf{x} = \mathbf{x} + \mu$ denotes element-wise addition, i.e., $\mathbf{x}_i = \mathbf{x}_i + \mu$ for $i \in [m]$.

2.2 Related Work

A thorough treatment of proximal operators including properties, examples, applications, and history is given in [16]. In this section we will focus on work directly related to computing $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ rather than proximal operators more broadly. Typically, for $\alpha > 0$, $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ is calculated using the Moreau decomposition:

$$\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x}) = \mathbf{x} - \alpha\mathcal{P}_{||\cdot||_1 \leq 1}(\mathbf{x}/\alpha),$$

where $\mathcal{P}_{||\cdot||_1 \leq 1} : \mathbb{R}^m \to \mathbb{R}^m$ is projection onto the ℓ_1 ball, i.e.,

$$\mathcal{P}_{||\cdot||_1 \leq 1}(\mathbf{v}) = \underset{||\mathbf{y}||_1 \leq 1}{\operatorname{argmin}} \frac{1}{2}||\mathbf{y} - \mathbf{v}||_2^2.$$

Projection onto the ℓ_1 ball can be easily calculated from projection onto the simplex [18]. Projection onto the simplex can be calculated (naively) using an $O(m \log m)$ algorithm originally published in [19] and rediscovered in [20]. The algorithm performs soft-thresholding on the input vector using an iteratively determined threshold. Naively implemented, the iterative algorithm to determine the threshold requires a sort of the entries of the input vector; however, in [18] an expected linear time, divide and conquer algorithm is presented which leverages the fact that a full sort of the input vector is not actually necessary to find the threshold. In addition, other fast algorithms for projection onto the simplex exist; see e.g., [12, 21]. A review of algorithms for projection onto the simplex is also given in [12].

In the next section we present an approach for calculating $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ that does not make use of the Moreau decomposition, but does use a strategy of iteratively computing a threshold similar to the projection onto the simplex algorithms presented in [18–20] and Algorithms 1 and 3 in [12].

2.3 Computing $\mathbf{prox}_{\alpha||\cdot||_\infty}$

To compute $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$, we first present a key insight. Let $\tau = ||\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})||_\infty$ and note that the elements of $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ are determined from the threshold value τ as follows:

$$[\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})]_k = \sigma_\tau(\mathbf{x}_k) := \begin{cases} \tau, & \text{if } \mathbf{x}_k \geq \tau \\ \mathbf{x}_k, & \text{if } |\mathbf{x}_k| < \tau \\ -\tau, & \text{if } \mathbf{x}_k \leq -\tau, \end{cases} \tag{2.2}$$

where $\sigma_\tau : \mathbb{R} \to \mathbb{R}$. Hence, to calculate $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ in Equation 2.1, we solve

$$\tau = \underset{t \geq 0}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{k=1}^{m} (\sigma_t(\mathbf{x}_k) - \mathbf{x}_k)^2 + \alpha t \right) \tag{2.3}$$

and then use σ_τ, defined in Equation 2.2, to find the entries of $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$. There is not a closed form solution to Equation 2.3; thus, in the remainder of this section, we analyze and provide an algorithm to solve Equation 2.3.

Let $\psi : \mathbb{R}^+ \cup \{0\} \to \mathbb{R}$ be defined as

$$\psi(t) = \frac{1}{2} \sum_{k=1}^{m} (\sigma_t(\mathbf{x}_k) - \mathbf{x}_k)^2 + \alpha t,$$

and let $I_t = \{k \mid |\mathbf{x}_k| \geq t\}$. Since

$$\sigma_t(\mathbf{x}_k) - \mathbf{x}_k = \begin{cases} t - \mathbf{x}_k, & \text{if } \mathbf{x}_k \geq t \\ 0, & \text{if } |\mathbf{x}_k| < t \\ -t - \mathbf{x}_k, & \text{if } \mathbf{x}_k \leq -t, \end{cases}$$

we can simplify ψ to

$$\psi(t) = \frac{1}{2} \sum_{k \in I_t} (|\mathbf{x}_k| - t)^2 + \alpha t.$$

To establish properties of ψ, we will use $\mathbf{s}$, a permutation of $|\mathbf{x}|$ such that $\mathbf{s}_1 \geq \mathbf{s}_2 \geq \cdots \geq \mathbf{s}_m \geq 0$. Let $\mathbf{s}_{m+1} = 0$.

Lemma 1. *ψ is continuous over $t \geq 0$.*

Proof. We first show that ψ is piecewise continuous. Suppose $t > \mathbf{s}_1$. Then $\psi(t) = \alpha t$ is a linear

function. In addition, for each index $i \in [m]$ such that $\mathbf{s}_{i+1} < t < \mathbf{s}_i$, $|I_t| = i$ and ψ is a quadratic. Hence ψ is piecewise continuous. Now, suppose $t = \mathbf{s}_1$. Then

$$\lim_{t \to \mathbf{s}_1^+} \psi(t) = \lim_{t \to \mathbf{s}_1^+} [\alpha t] = \psi(\mathbf{s}_1) = \lim_{t \to \mathbf{s}_1^-} \left[\frac{1}{2} \sum_{\substack{k: \\ \mathbf{s}_k = \mathbf{s}_1}} (\mathbf{s}_k - t)^2 + \alpha t \right] = \lim_{t \to \mathbf{s}_1^-} \psi(t).$$

Similarly for $t = \mathbf{s}_{m+1} = 0$,

$$\lim_{t \to \mathbf{s}_{m+1}^+} \psi(t) = \lim_{t \to \mathbf{s}_{m+1}^+} \left[\frac{1}{2} \sum_{k=1}^{m} (\mathbf{s}_k - t)^2 + \alpha t \right] = \psi(\mathbf{s}_{m+1}).$$

Next suppose $t = \mathbf{s}_i$ for some $i \in [m] \setminus \{1\}$. Let $S_i = \{k : \mathbf{s}_k = \mathbf{s}_i\}$. Then

$$\begin{aligned} \lim_{t \to \mathbf{s}_i^+} \psi(t) &= \lim_{t \to \mathbf{s}_i^+} \left[\frac{1}{2} \sum_{k \in [i] \setminus S_i} (\mathbf{s}_k - t)^2 + \alpha t \right] = \psi(\mathbf{s}_i) \\ &= \lim_{t \to \mathbf{s}_i^-} \left[\frac{1}{2} \sum_{k \in [i] \cup S_i} (\mathbf{s}_k - t)^2 + \alpha t \right] = \lim_{t \to \mathbf{s}_i^-} \psi(t). \end{aligned}$$

Thus, ψ is continuous over $t \geq 0$. □

Theorem 2. *ψ is $\mathcal{C}^1$ over $t > 0$.*

Proof. By Lemma 1, ψ is continuous, linear over $t > \mathbf{s}_1$, and quadratic over $\mathbf{s}_{k+1} < t < \mathbf{s}_k$ for $k \in [m]$ such that $\mathbf{s}_{k+1} \neq \mathbf{s}_k$. Hence ψ is $\mathcal{C}^1$ on $t > \mathbf{s}_1$, and $\mathbf{s}_{k+1} < t < \mathbf{s}_k$ for $k \in [m]$ such that $\mathbf{s}_{k+1} \neq \mathbf{s}_k$. We now turn our attention to the points $t = \mathbf{s}_k$ for $k \in [m]$. Suppose $t = \mathbf{s}_1$. Then

$$\lim_{t \to \mathbf{s}_1^+} \psi'(t) = \lim_{t \to \mathbf{s}_1^+} \alpha = \alpha = \lim_{t \to \mathbf{s}_1^-} \left[- \sum_{\substack{k: \\ \mathbf{s}_k = \mathbf{s}_1}} (\mathbf{s}_k - t) + \alpha \right] = \lim_{t \to \mathbf{s}_1^-} \psi'(t),$$

and thus $\psi'(\mathbf{s}_1) = \alpha$. Next fix an index $i \in [m] \setminus \{1\}$ and suppose $t = \mathbf{s}_i$. Let $S_i = \{k : \mathbf{s}_k = \mathbf{s}_i\}$.

Then

$$\lim_{t\to \mathbf{s}_i^+} \psi'(t) = \lim_{t\to \mathbf{s}_i^+} \left[-\sum_{k\in[i]\setminus S_i} (\mathbf{s}_k - t) + \alpha \right] = \lim_{t\to \mathbf{s}_i^-} \left[-\sum_{k\in[i]\cup S_i} (\mathbf{s}_k - t) + \alpha \right] = \lim_{t\to \mathbf{s}_i^-} \psi'(t),$$

and thus $\psi'(\mathbf{s}_i) = -\sum_{k\in[i]\setminus S_i}(\mathbf{s}_k - \mathbf{s}_i) + \alpha$. Hence ψ is $\mathcal{C}^1$ over $t > 0$. □

So, we can calculate $\psi' : \mathbb{R}^+ \to \mathbb{R}$,

$$\psi'(t) = -\sum_{k\in I_t} |\mathbf{x}_k| + |I_t|t + \alpha,$$

and leverage it to find $\tau = \operatorname{argmin}_{t\geq 0} \psi(t)$.

Theorem 3. $\tau \leq ||\mathbf{x}||_\infty = \mathbf{s}_1$.

Proof. Since $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{0}) = \mathbf{0}$, it is clear that if $\mathbf{x} = \mathbf{0}$, then $\tau = 0 = \mathbf{s}_1$. If $\mathbf{x} \neq \mathbf{0}$ and $\alpha > 0$, then $\psi'(t) = \alpha > 0$ for $t \geq \mathbf{s}_1$. Thus $\tau < \mathbf{s}_1 = ||\mathbf{x}||_\infty$. If $\mathbf{x} \neq \mathbf{0}$ and $\alpha = 0$, it is clear that $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x}) = \mathbf{x}$ and the smallest τ that would gaurantee this is $\tau = \mathbf{s}_1$. □

Now we will prove the uniqueness of τ by establishing the strict convexity of ψ.

Theorem 4. *ψ is strictly convex over* $0 \leq t \leq ||\mathbf{x}||_\infty = \mathbf{s}_1$.

Proof. We will show that ψ' is strictly increasing over $0 < t < \mathbf{s}_1$ thus proving the result [22, Theorems 12A, 12B]. Let $0 < t_1 < t_2 < \mathbf{s}_1$. This guarantees that $|I_{t_1}| > 0$ and $|I_{t_2}| > 0$. We proceed using two cases.

Case 1: Suppose t_1 and t_2 are such that $I_{t_2} = I_{t_1}$. Then

$$\psi'(t_1) = |I_{t_1}|t_1 - \sum_{k\in I_{t_1}} |\mathbf{x}_k| + \alpha < |I_{t_2}|t_2 - \sum_{k\in I_{t_2}} |\mathbf{x}_k| + \alpha = \psi'(t_2),$$

hence ψ' is strictly increasing.

Case 2: Suppose t_1 and t_2 are such that $I_{t_2} \subset I_{t_1}$. Then $t_1 \leq |\mathbf{x}_k|$ for $k \in I_{t_1} - I_{t_2}$ and

$$|I_{t_1} - I_{t_2}|t_1 - \sum_{k \in I_{t_1} - I_{t_2}} |\mathbf{x}_k| \leq 0.$$

Since

$$\begin{aligned}
|I_{t_2}|t_1 < |I_{t_2}|t_2 &\Rightarrow |I_{t_2}|t_1 + |I_{t_1} - I_{t_2}|t_1 - \sum_{k \in I_{t_1} - I_{t_2}} |\mathbf{x}_k| < |I_{t_2}|t_2 \\
&\Rightarrow (|I_{t_2}| + |I_{t_1} - I_{t_2}|)t_1 - \left(\sum_{k \in I_{t_2}} |\mathbf{x}_k| + \sum_{k \in I_{t_1} - I_{t_2}} |\mathbf{x}_k| \right) + \alpha < |I_{t_2}|t_2 - \sum_{k \in I_{t_2}} |\mathbf{x}_k| + \alpha \\
&\Rightarrow |I_{t_1}|t_1 - \sum_{k \in I_{t_1}} |\mathbf{x}_k| + \alpha < |I_{t_2}|t_2 - \sum_{k \in I_{t_2}} |\mathbf{x}_k| + \alpha \\
&\Rightarrow \psi'(t_1) < \psi'(t_2),
\end{aligned}$$

ψ' is strictly increasing. □

Theorem 5. $\tau = 0$ *if and only if* $||\mathbf{x}||_1 \leq \alpha$.

Proof. We will prove that if $\tau = 0$, then $||\mathbf{x}||_1 \leq \alpha$ by contradiction. Suppose $\tau = 0$ and $||\mathbf{x}||_1 > \alpha$. Then $\psi'(0) = -\|\mathbf{x}\|_1 + \alpha < 0$. In addition, $\psi'(\mathbf{s}_1) = \alpha \geq 0$. So, by Theorem 2 and the Intermediate Value Theorem, there exists a $0 < t \leq \mathbf{s}_1$ such that $\psi'(t) = 0$. By Theorem 4, this contradicts $\tau = 0$.

Now suppose $||\mathbf{x}||_1 \leq \alpha$. Then $\psi'(0) = -\|\mathbf{x}\|_1 + \alpha \geq 0$. By Theorem 4, this implies $\tau = 0$. □

Theorems 2 through 5 provide the basis for Algorithm 1 which computes $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$. The main iteration of this algorithm finds the value $0 < t \leq \mathbf{s}_1$ such that $\psi'(t) = 0$, i.e., $t = (\sum_{k \in I_t} |\mathbf{x}_k| - \alpha)/|I_t|$. Due to the strict convexity of ψ, there is at most one value of t for which

this is true. Clearly the algorithm is necessary since we do not know the elements of I_τ in advance. The complexity of Algorithm 1 is $O(m \log m)$ due to the required sort of the input vector. We can improve the complexity to $O(m)$ in expectation using a similar divide and conquer strategy to that used in [12,18] for computing the projection onto the simplex. The expected $O(m)$ algorithm is provided in Supplementary Section 2.6.

Algorithm 1 Proximal Operator of the ℓ_∞ Norm

Input: $\mathbf{x} \in \mathbb{R}^m$, $\alpha \geq 0$
Output: $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x}) \in \mathbb{R}^m$

1: **if** $||\mathbf{x}||_1 \leq \alpha$ **then**
2: $\quad \mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x}) = \mathbf{0}$
3: **else**
4: $\quad$ Let $\mathbf{s}$ be a permutation of $|\mathbf{x}|$ such that $\mathbf{s}_1 \geq \mathbf{s}_2 \geq \cdots \geq \mathbf{s}_m \geq 0$. Set $\mathbf{s}_{m+1} = 0$.
5: $\quad \gamma = 0$
6: $\quad i = 1$
7: $\quad$ **while** $i \leq m$ **do** ▷ Assume that $|I_t| = i$, i.e., $\mathbf{s}_{i+1} < t \leq \mathbf{s}_i$
8: $\quad\quad \gamma = \gamma + \mathbf{s}_i$
9: $\quad\quad j = 1$
10: $\quad\quad$ **while** $i + j \leq m$ and $\mathbf{s}_{i+j} == \mathbf{s}_i$ **do** ▷ Account for repeated elements in $\mathbf{s}$
11: $\quad\quad\quad \gamma = \gamma + \mathbf{s}_i$
12: $\quad\quad\quad j = j + 1$
13: $\quad\quad$ **end while**
14: $\quad\quad i = i + (j - 1)$
15: $\quad\quad t_0 = (\gamma - \alpha)/i$ ▷ Find the minimum of $\psi(t)$, $(\sum_{k=1}^{i} \mathbf{s}_k - \alpha)/i$
16: $\quad\quad$ **if** $\mathbf{s}_{i+1} < t_0 \leq \mathbf{s}_i$ **then** ▷ If t_0 satisfies the assumption from Line 8
17: $\quad\quad\quad \tau = t_0$
18: $\quad\quad\quad$ **break**
19: $\quad\quad$ **end if**
20: $\quad\quad i = i + 1$
21: $\quad$ **end while**
22: $\quad \forall k \in [m]$ compute $[\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})]_k = \sigma_\tau(\mathbf{x}_k)$
23: **end if**
24: **return** $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$

As mentioned in Section 2.2, a common method for calculating $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ when $\alpha > 0$

is to use the Moreau decomposition:

$$\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x}) = \mathbf{x} - \alpha\mathcal{P}_{||\cdot||_1\leq 1}(\mathbf{x}/\alpha),$$

where $\mathcal{P}_{||\cdot||_1\leq r}(\mathbf{v}) : \mathbb{R}^m \to \mathbb{R}^m$ is the projection of $\mathbf{v}$ onto the ℓ_1 ball of radius r,

$$\mathcal{P}_{||\cdot||_1\leq r}(\mathbf{v}) = \underset{||\mathbf{y}||_1\leq r}{\operatorname{argmin}} \frac{1}{2}||\mathbf{y} - \mathbf{v}||_2^2.$$

By the Moreau decomposition and [12, Proposition 2.1], which relates projection onto the ℓ_1 ball to projection onto the simplex, we have

$$\begin{aligned}
\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x}) &= \mathbf{x} - \alpha\mathcal{P}_{||\cdot||_1\leq 1}(\mathbf{x}/\alpha) \\
&= \mathbf{x} - \mathcal{P}_{||\cdot||_1\leq \alpha}(\mathbf{x}), \\
&= \mathbf{x} - \begin{cases} \mathbf{x}, & \text{if } ||\mathbf{x}||_1 \leq \alpha \\ \operatorname{sign}(\mathbf{x}) \circ \mathcal{P}_{||\cdot||_1=\alpha}(|\mathbf{x}|), & \text{else} \end{cases}
\end{aligned}$$

where $\mathcal{P}_{||\cdot||_1=\alpha}(\mathbf{v}) : \mathbb{R}^m \to \mathbb{R}^m$ is the projection of $\mathbf{v}$ onto the simplex, i.e.,

$$\mathcal{P}_{||\cdot||_1=r}(\mathbf{v}) = \underset{\substack{||\mathbf{y}||_1=r \\ \mathbf{y}_i\geq 0}}{\operatorname{argmin}} \frac{1}{2}||\mathbf{y} - \mathbf{v}||_2^2,$$

and element-wise vector multiplication is denoted using $\circ$, e.g., $\mathbf{a} \circ \mathbf{b} = (\mathbf{a}_1\mathbf{b}_1, \ldots, \mathbf{a}_m\mathbf{b}_m)$. Hence, computing $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ using the Moreau decomposition is dependent on $\mathcal{P}_{||\cdot||_1=\alpha}(|\mathbf{x}|)$. The $O(m \log m)$ algorithm for projection onto the simplex, mentioned in Section 2.2, is provided for reference in Algorithm 2.

Algorithm 2 Projection onto the Simplex [19, 20]

Input: $\mathbf{x} \in \mathbb{R}^m$, $\alpha > 0$
Output: $\mathcal{P}_{||\cdot||_1=\alpha}(\mathbf{x}) \in \mathbb{R}^m$

Sort $\mathbf{x}$ into $\mathbf{v}$ such that $\mathbf{v}_1 \geq \mathbf{v}_2 \geq \cdots \geq \mathbf{v}_m$.
$\omega = 0$
$i^* = 0$
for $i = 1$ to m **do** ▷ Find $i^* = \max_{1 \leq i \leq m}\{i \mid (\sum_{k=1}^{i} \mathbf{v}_k - \alpha)/i < \mathbf{v}_i\}$
    $\omega = \omega + \mathbf{v}_i$
    $t_0 = (\omega - \alpha)/i$
    if $t_0 < \mathbf{v}_i$ **then**
        $i^* = i$
        $\tau_{\text{simplex}} = t_0$ ▷ $\tau_{\text{simplex}} = (\sum_{k=1}^{i^*} \mathbf{v}_k - \alpha)/i^*$
    end if
end for
$\forall k \in [m]$ compute $[\mathcal{P}_{||\cdot||_1=\alpha}(\mathbf{x})]_k = \max\{\mathbf{x}_k - \tau_{\text{simplex}}, 0\}$
return $\mathcal{P}_{||\cdot||_1=\alpha}(\mathbf{x})$

For $\mathbf{x} \in \mathbb{R}^m$ such that $||\mathbf{x}||_1 > \alpha$, Algorithm 1 to compute $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ and Algorithm 2 to compute $\mathcal{P}_{||\cdot||_1=\alpha}(|\mathbf{x}|)$ are similar in that a threshold is iteratively computed and then used to compute the appropriate quantity. We show that these thresholds are actually equal, and first present a helpful lemma.

Lemma 6. *For $\alpha > 0$, let $\mathbf{x} \in \mathbb{R}^m$ be such that $||\mathbf{x}||_1 > \alpha$, and τ be calculated as in Algorithm 1. If $\tau = |\mathbf{x}_r| = \mathbf{s}_r$ and $\mathbf{s}_r$ appears j times in $\mathbf{s}$, i.e., $\mathbf{s}_r = \mathbf{s}_{r-1} = \cdots = \mathbf{s}_{r-(j-1)}$, then*

$$\tau = \left(\sum_{k=1}^{r} \mathbf{s}_k - \alpha\right)/r = \left(\sum_{k=1}^{r-1} \mathbf{s}_k - \alpha\right)/(r-1) = \cdots = \left(\sum_{k=1}^{r-j} \mathbf{s}_k - \alpha\right)/(r-j).$$

Proof. By Algorithm 1, $\tau = (\sum_{k=1}^{r} \mathbf{s}_k - \alpha)/r$. To show the second part of the result, we use induction. Let $i = 1$. Then,

$$\mathbf{s}_r = \tau = \left(\sum_{k=1}^{r} \mathbf{s}_k - \alpha\right)/r \Leftrightarrow \left(\sum_{k=1}^{r-1} \mathbf{s}_k - \alpha\right)/(r-1) = \mathbf{s}_r = \mathbf{s}_{r-1} = \tau.$$

Now, assume that $\tau = (\sum_{k=1}^{r} \mathbf{s}_k - \alpha)/r = (\sum_{k=1}^{r-i} \mathbf{s}_k - \alpha)/(r-i)$ for some $i \in [j-1]$. Then,

$$\tau = \left(\sum_{k=1}^{r-i} \mathbf{s}_k - \alpha\right)/(r-i) = \mathbf{s}_{r-i} \Leftrightarrow \left(\sum_{k=1}^{r-(i+1)} \mathbf{s}_k - \alpha\right)/(r-(i+1)) = \mathbf{s}_{r-(i+1)} = \tau.$$

Hence, by induction,

$$\tau = \left(\sum_{k=1}^{r} \mathbf{s}_k - \alpha\right)/r = \left(\sum_{k=1}^{r-(j-1)} \mathbf{s}_k - \alpha\right)/(r-(j-1)) = \mathbf{s}_{r-(j-1)},$$

and as a final step,

$$\tau = \left(\sum_{k=1}^{r-(j-1)} \mathbf{s}_k - \alpha\right)/(r-(j-1)) = \mathbf{s}_{r-(j-1)} \Leftrightarrow \left(\sum_{k=1}^{r-j} \mathbf{s}_k - \alpha\right)/(r-j) = \mathbf{s}_{r-(j-1)} = \tau.$$

□

Theorem 7. *For $\alpha > 0$, let $\mathbf{x} \in \mathbb{R}^m$ be such that $||\mathbf{x}||_1 > \alpha$, and τ be calculated as in Algorithm 1 for finding $\textbf{\textit{prox}}_{\alpha||\cdot||_\infty}(\mathbf{x})$, and $\tau_{simplex}$ as in Algorithm 2 for finding $\mathcal{P}_{||\cdot||_1=\alpha}(|\mathbf{x}|)$. Then $\tau = \tau_{simplex}$.*

Proof. We will prove the result using two cases.

Case 1: Suppose $\mathbf{s}_{i+1} < \tau < \mathbf{s}_i$ for some $i \in [m]$ as found by Algorithm 1. Then, $\tau = (\sum_{k=1}^{i} \mathbf{s}_k - \alpha)/i$. Hence $i^* \geq i$ for Algorithm 2. We will show by induction that for all $j \in [m]$ such that $i < j \leq m$, $(\sum_{k=1}^{j} \mathbf{s}_k - \alpha)/j \geq \mathbf{s}_j$ and therefore $i^* = i$ and thus $\tau = \tau_{\text{simplex}}$. For $j = i+1$, we have

$$\left(\sum_{k=1}^{i} \mathbf{s}_k - \alpha\right)/i > \mathbf{s}_{i+1} \Leftrightarrow \left(\sum_{k=1}^{i+1} \mathbf{s}_k - \alpha\right)/(i+1) > \mathbf{s}_{i+1}.$$

Next, assume that $(\sum_{k=1}^{j} \mathbf{s}_k - \alpha)/j \geq \mathbf{s}_j$ for some $j \in [m]$ such that $i+1 \leq j \leq m$. By the

same argument above and the fact that $\mathbf{s}_j \geq \mathbf{s}_{j+1}$, we have $(\sum_{k=1}^{j+1} \mathbf{s}_k - \alpha)/(j+1) \geq \mathbf{s}_{j+1}$, and the result is proven.

Case 2: Suppose $\tau = \mathbf{s}_i$ for some $i \in [m]$ as found by Algorithm 1. Then $\mathbf{s}_{i+1} < \tau \leq \mathbf{s}_i$ and $\tau = (\sum_{k=1}^{i} \mathbf{s}_k - \alpha)/i = \mathbf{s}_i$. Let $\hat{i} < i$ be the largest index such that $\mathbf{s}_{\hat{i}} > \mathbf{s}_i$. Then by Lemma 6,

$$\mathbf{s}_{\hat{i}} > \mathbf{s}_i = \tau = \left(\sum_{k=1}^{i} \mathbf{s}_k - \alpha\right)/i = \left(\sum_{k=1}^{\hat{i}} \mathbf{s}_k - \alpha\right)/\hat{i}.$$

By an argument similar to that of Case 1, for all $j \in [m]$ such that $\hat{i} < j \leq m$, $(\sum_{k=1}^{j} \mathbf{s}_k - \alpha)/j \geq \mathbf{s}_j$ and thus $i^* = \hat{i}$ for Algorithm 2 and $\tau = \tau_{\text{simplex}}$. □

So, the same threshold is used to compute $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ in Algorithm 1 and $\mathcal{P}_{||\cdot||_1=\alpha}(|\mathbf{x}|)$ in Algorithm 2; however, how the threshold is applied to the input vector to compute the final result differs between these two algorithms. Let the common threshold be τ. By the Moreau decomposition and Theorem 7, for $\alpha > 0$ we have

$$[\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})]_k = \mathbf{x}_k - \begin{cases} \mathbf{x}_k, & \text{if } ||\mathbf{x}||_1 \leq \alpha \\ \text{sign}(\mathbf{x}_k)\max\{|\mathbf{x}_k| - \tau, 0\}, & \text{else} \end{cases}$$

$$= \begin{cases} 0, & \text{if } ||\mathbf{x}||_1 \leq \alpha \\ \sigma_\tau(\mathbf{x}_k), & \text{else,} \end{cases}$$

which is our approach to computing $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$.

We can also relate the threshold computed in Algorithm 1 to the solution to the projection

onto the ℓ_1 ball. The solution to $\mathcal{P}_{||\cdot||_1 \leq 1}(\mathbf{v})$ is

$$[\mathcal{P}_{||\cdot||_1 \leq 1}(\mathbf{v})]_k = \text{sign}(\mathbf{v}_k)\max(|\mathbf{v}_k| - \lambda, 0) = \begin{cases} \mathbf{v}_k - \lambda, & \text{if } \mathbf{v}_k \geq \lambda \\ 0, & \text{if } |\mathbf{v}_k| < \lambda \\ \mathbf{v}_k + \lambda, & \text{if } \mathbf{v}_k \leq -\lambda, \end{cases}$$

where the threshold λ is computed using the equation $\sum_{k=1}^{m} \max(|\mathbf{v}_k| - \lambda, 0) = 1$. Thus, by the Moreau decomposition, for $\alpha > 0$ we have

$$\begin{aligned} [\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})]_k &= \mathbf{x}_k - \alpha \begin{cases} \mathbf{x}_k/\alpha - \lambda, & \text{if } \mathbf{x}_k/\alpha \geq \lambda \\ 0, & \text{if } |\mathbf{x}_k/\alpha| < \lambda \\ \mathbf{x}_k/\alpha + \lambda, & \text{if } \mathbf{x}_k/\alpha \leq -\lambda, \end{cases} \\ &= \begin{cases} \alpha\lambda, & \text{if } \mathbf{x}_k \geq \alpha\lambda \\ \mathbf{x}_k, & \text{if } |\mathbf{x}_k| < \alpha\lambda \\ -\alpha\lambda, & \text{if } \mathbf{x}_k \leq -\alpha\lambda \end{cases} \\ &= \sigma_{\alpha\lambda}(\mathbf{x}_k), \end{aligned}$$

which is Equation 2.2 with $\tau = \alpha\lambda$. Hence the threshold computed using Algorithm 1, τ, and that for computing $\mathcal{P}_{||\cdot||_1 \leq 1}(\mathbf{x})$, λ, are related as well.

2.4 Computation by Neural Network

The previous algorithms presented to compute $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ (including Algorithm 5 in Supplementary Section 2.6) require a sort of the input vector or at least a partial sort of the input vector similar to quicksort. For $\mathbf{x} \in \mathbb{R}^m$ this leads to $O(m \log m)$ or expected $O(m)$ and worst case $O(m^2)$ computation time. In this section we present an $O(m)$ method to approximate $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ using a neural network. Specifically, we use the neural network to approximate τ without requiring a sort of the input vector. A novel aspect of the network is that it is able to accept vectors of varying lengths due to a feature selection process that uses moments of the input data. This property is desired since 1) $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ can be computed for a vector of any length and 2) it allows the user to train one network instead of one per vector length in the dataset. We present results on the accuracy of the approximation, feature importance, and computational efficiency of the approach. We also compare the network loss to that of a "vanilla neural network", which naively solves the problem and serves as a baseline comparison.

2.4.1 Data Preprocessing and Feature Selection

For this approach, data consists of $\mathbf{x}$-α-τ triples, where $\mathbf{x}$ can be any length and $\tau = ||\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})||_\infty$. To start, we present a helpful theorem for data preprocessing.

Theorem 8. *Let $\alpha > 0$, $\mu \in \mathbb{R}$, and $||\mathbf{x}||_1 > \alpha$. Since $\tau = ||\boldsymbol{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})||_\infty$, it is also true that $||\boldsymbol{prox}_{||\cdot||_\infty}(|\mathbf{x}|/\alpha + \mu)||_\infty = \tau/\alpha + \mu$.*

Proof. Let $\widehat{\mathbf{x}} = |\mathbf{x}|/\alpha + \mu$ and $\widehat{\mathbf{s}} = \mathbf{s}/\alpha + \mu$, where $\mathbf{s}$ is a permutation of $|\mathbf{x}|$ such that $\mathbf{s}_1 \geq \mathbf{s}_2 \geq \cdots \geq \mathbf{s}_m \geq 0$ and $\mathbf{s}_{m+1} = 0$. By Algorithm 1 there exists an i such that $\mathbf{s}_{i+1} < \tau \leq \mathbf{s}_i$, where

$\tau = (\sum_{k=1}^{i} \mathbf{s}_k - \alpha)/i$. Since

$$\begin{aligned} \mathbf{s}_{i+1} < \tau \leq \mathbf{s}_i &\Leftrightarrow \mathbf{s}_{i+1}/\alpha + \mu < \tau/\alpha + \mu \leq \mathbf{s}_i/\alpha + \mu \\ &\Leftrightarrow \widehat{\mathbf{s}}_{i+1} < \tau/\alpha + \mu \leq \widehat{\mathbf{s}}_i, \end{aligned} \tag{2.4}$$

and $\tau/\alpha + \mu = (\sum_{k=1}^{i} \widehat{\mathbf{s}}_k - 1)/i$ it is clear that $||\mathbf{prox}_{||\cdot||_\infty}(|\mathbf{x}|/\alpha + \mu)||_\infty = \tau/\alpha + \mu$. □

We note that τ is a function of two variables, $\tau(|\mathbf{x}|, \alpha)$. By Theorem 8, we can effectively remove α as a parameter of the problem by scaling $\mathbf{x}$ by $1/\alpha$. Thus τ can be viewed as a function of only $\mathbf{x}$, i.e., $\tau(|\mathbf{x}|)$, that is permutation invariant with respect to the order of elements in $\mathbf{x}$.

Theorem 9. *$\tau(|\mathbf{x}|)$: $\mathbb{R}^m \to \mathbb{R}$ is continuous.*

Proof. Let $\hat{\mathbf{x}} \in \mathbb{R}^m$ be chosen arbitrarily and let $\hat{\mathbf{s}}$ be a permutation of $|\hat{\mathbf{x}}|$ such that $\hat{\mathbf{s}}_1 \geq \hat{\mathbf{s}}_2 \geq \cdots \geq \hat{\mathbf{s}}_m \geq 0$ and $\hat{\mathbf{s}}_{m+1} = 0$, and $\hat{\tau} = \tau(\hat{\mathbf{s}})$. Fix $\epsilon > 0$, and let $\mathbf{s}$ be the vector $\hat{\mathbf{s}}$ with small perturbations in its elements, i.e., $\forall j \in [m]$, $|\mathbf{s}_j - \hat{\mathbf{s}}_j| < \delta$. We shall find $\delta = \delta(\epsilon) > 0$ so that $|\tau(\mathbf{s}) - \tau(\hat{\mathbf{s}})| < \epsilon$. We prove the result in two cases.

Case 1: $\hat{\mathbf{s}}_{i+1} < \hat{\tau} < \hat{\mathbf{s}}_i$ for some $i \in [m]$. Choose $\delta = \min(\epsilon, \frac{1}{2}(\hat{\tau} - \hat{\mathbf{s}}_{i+1}), \frac{1}{2}(\hat{\mathbf{s}}_i - \hat{\tau}))$. Then $\max(\mathbf{s}_{i+1}, \hat{\mathbf{s}}_{i+1}) < \tau(\mathbf{s}) < \min(\mathbf{s}_i, \hat{\mathbf{s}}_i)$, and thus,

$$|\tau(\mathbf{s}) - \tau(\hat{\mathbf{s}})| = \frac{1}{i}\left|\sum_{k=1}^{i} \mathbf{s}_k - \alpha - \sum_{k=1}^{i} \hat{\mathbf{s}}_k + \alpha\right| \leq \frac{1}{i}\sum_{k=1}^{i} |\mathbf{s}_k - \hat{\mathbf{s}}_k| < \delta \leq \epsilon.$$

Case 2: $\hat{\tau} = \hat{\mathbf{s}}_i$ for some $i \in [m] \setminus 1$.

Subcase 2.1: $\hat{\mathbf{s}}_{i+r+1} < \hat{\mathbf{s}}_{i+r} = \cdots = \hat{\mathbf{s}}_{i+1} = \hat{\mathbf{s}}_i = \hat{\tau} < \hat{\mathbf{s}}_{i-1}$. Let $\delta_{\min} = \min(\hat{\mathbf{s}}_{i-1} - \hat{\mathbf{s}}_i, \hat{\mathbf{s}}_{i+r} - \hat{\mathbf{s}}_{i+r+1})$ and choose $\delta = \min(\epsilon, \frac{1}{5}\delta_{\min})$. By Theorem 7, we can consider the index i^* in Algorithm 2 associated to vector $\mathbf{s}$, the perturbation of $\hat{\mathbf{s}}$. We claim that $i - 1 \leq i^* \leq i + r$. By

Lemma 6,

$$\hat{\mathbf{s}}_{i+r} = \hat{\mathbf{s}}_i = \hat{\tau} = \frac{1}{i+r}\left(\sum_{k=1}^{i+r} \hat{\mathbf{s}}_k - \alpha\right) = \frac{1}{i-1}\left(\sum_{k=1}^{i-1} \hat{\mathbf{s}}_k - \alpha\right).$$

Hence,

$$\begin{aligned}
\mathbf{s}_{i-1} - \frac{1}{i-1}\left(\sum_{k=1}^{i-1} \mathbf{s}_k - \alpha\right) &= \hat{\mathbf{s}}_{i-1} - \frac{1}{i-1}\left(\sum_{k=1}^{i-1} \hat{\mathbf{s}}_k - \alpha\right) + (\mathbf{s}_{i-1} - \hat{\mathbf{s}}_{i-1}) - \frac{1}{i-1}\sum_{k=1}^{i-1}(\mathbf{s}_k - \hat{\mathbf{s}}_k) \\
&\geq \hat{\mathbf{s}}_{i-1} - \hat{\mathbf{s}}_i - 2\delta \\
&\geq \delta_{\text{min}} - 2\delta \\
&> 0.
\end{aligned}$$

Thus, by the condition on line 7 of Algorithm 2, $i^* \geq i - 1$.

Similarly,

$$\begin{aligned}
&\mathbf{s}_{i+r+1} - \frac{1}{i+r+1}\left(\sum_{k=1}^{i+r+1} \mathbf{s}_k - \alpha\right) = \\
&\hat{\mathbf{s}}_{i+r+1} - \frac{1}{i+r+1}\left(\sum_{k=1}^{i+r+1} \hat{\mathbf{s}}_k - \alpha\right) + (\mathbf{s}_{i+r+1} - \hat{\mathbf{s}}_{i+r+1}) - \frac{1}{i+r+1}\sum_{k=1}^{i+r+1}(\mathbf{s}_k - \hat{\mathbf{s}}_k) \\
&\qquad \leq \frac{i+r}{i+r+1}\hat{\mathbf{s}}_{i+r+1} - \frac{i+r}{i+r+1}\hat{\mathbf{s}}_{i+r} + 2\delta \qquad (2.5) \\
&\qquad \leq -\frac{i+r}{i+r+1}\delta_{\text{min}} + 2\delta \\
&\qquad \leq -\frac{1}{2}\delta_{\text{min}} + 2\delta \\
&\qquad < 0,
\end{aligned}$$

and thus $i^* < i + r + 1$. Consequently,

$$\tau(\mathbf{s}) = \frac{1}{i-1+p}\left(\sum_{k=1}^{i-1+p} \mathbf{s}_k - \alpha\right) \qquad (2.6)$$

for some $p \in \{0, 1, 2, \ldots, r+1\}$. Hence, by Lemma 6,

$$|\tau(\mathbf{s}) - \tau(\hat{\mathbf{s}})| \leq \frac{1}{i-1+p} \sum_{k=1}^{i-1+p} |\mathbf{s}_k - \hat{\mathbf{s}}_k| < \delta \leq \epsilon. \tag{2.7}$$

Subcase 2.2: $\hat{\mathbf{s}}_{2+r} < \hat{\mathbf{s}}_{1+r} = \cdots = \hat{\mathbf{s}}_2 = \hat{\mathbf{s}}_1 = \hat{\tau}$. Let $\delta_{\text{min}} = \hat{\mathbf{s}}_{1+r} - \hat{\mathbf{s}}_{2+r}$ and choose $\delta = \min(\epsilon, \frac{1}{5}\delta_{\text{min}})$. Again we consider the index i^* in Algorithm 2 associated to vector $\mathbf{s}$. We claim that $1 \leq i^* \leq 1+r$. The logic of Subcase 2.1 applies for $i = 1$ except for 1) establishing the lower bound on i^*, which is true by Theorem 3, and 2) Equations 2.6 and 2.7 apply for $p \in [r+1]$.

Subcase 2.3: $0 = \hat{\mathbf{s}}_{m+1} = \hat{\mathbf{s}}_m = \cdots = \hat{\mathbf{s}}_i = \hat{\tau} < \hat{\mathbf{s}}_{i-1}$. Let $\delta_{\text{min}} = \hat{\mathbf{s}}_{i-1} - \hat{\mathbf{s}}_i$ and choose $\delta = \min(\epsilon, \frac{1}{3}\delta_{\text{min}})$. We claim that $i - 1 \leq i^* \leq m$ for the index i^* in Algorithm 2 associated to vector $\mathbf{s}$. The logic of Subcase 2.1 applies for $r = m - i$ except for establishing the upper bound on i^*, which is trivially true.

Subcase 2.4: $0 = \hat{\mathbf{s}}_{m+1} = \hat{\mathbf{s}}_m = \cdots = \hat{\mathbf{s}}_1 = \hat{\tau}$, i.e. $\hat{\mathbf{s}} = \mathbf{0}$. Choose $\delta = \min(\epsilon, \frac{\alpha}{m})$. Then $||\mathbf{s}||_1 \leq \alpha$, and by Theorem 5, $\tau(\mathbf{s}) = 0$. Thus,

$$|\tau(\mathbf{s}) - \tau(\hat{\mathbf{s}})| = 0 < \delta \leq \epsilon.$$

Since $\hat{\mathbf{x}}$ was chosen arbitrarily in each case, $\tau(|\mathbf{x}|)$ is continuous. □

By Theorem 9 and the Stone-Weierstrass Theorem, we can approximate $\tau(|\mathbf{x}|)$ by a polynomial. Since $\tau(|\mathbf{x}|)$ is permutation invariant, this polynomial is symmetric. Hence, it can be written uniquely as a polynomial in the elementary symmetric polynomials by the Fundamental Theorem of Symmetric Polynomials, and thus uniquely as a polynomial in power sums by the Newton-Girard formulas [23]. For $\mathbf{x} \in \mathbb{R}^m$, we refer to the k-th power sum, $\sum_{i=1}^m \mathbf{x}_i^k$, as the k-th

moment of $\mathbf{x}$.

This reasoning motivated our choices of preprocessing and feature selection for each $\mathbf{x}$-α-τ triple as seen in Algorithm 3. We note that for any vector $\mathbf{x}$ such that $\|\widehat{\mathbf{x}}\|_1 \leq 1$ (or equivalently, $\|\mathbf{x}\|_1 \leq \alpha$, where $\widehat{\mathbf{x}} = |\mathbf{x}|/\alpha$), empty values are returned. Since $\tau = 0$ for these vectors, we effectively remove them from the dataset since a prediction of τ is not needed.

Algorithm 3 Data Preprocessing and Feature Generation

Input: $\mathbf{x} \in \mathbb{R}^m$, α, τ, k ▷ k is the number of moments to compute.
Output: $\mathbf{w} \in \mathbb{R}^{k+3}$, $\widehat{\tau} \in \mathbb{R}$

$\widehat{\mathbf{x}} = |\mathbf{x}|/\alpha$
if $\|\widehat{\mathbf{x}}\|_1 \leq 1$ **then**
    $\mathbf{w} = \emptyset$, $\widehat{\tau} = \emptyset$ ▷ An empty vector and scalar are returned.
else
    $\mu = ||\widehat{\mathbf{x}}||_1/m$
    $\widehat{\mathbf{x}} = \widehat{\mathbf{x}} - \mu$ such that $\widehat{\mathbf{x}}_i = \widehat{\mathbf{x}}_i - \mu$ for $i \in [m]$. ▷ Center the data.
    $\mathbf{w}_1 = \min(\widehat{\mathbf{x}})$ and $\mathbf{w}_2 = \max(\widehat{\mathbf{x}})$
    for $j \in [k]$ **do** ▷ Calculate the moments.
        if $j == 1$ **then**
            $\mathbf{w}_3 = \frac{1}{m}\sum_{i=1}^{m} |\widehat{\mathbf{x}}_i|$
        else
            $\mathbf{w}_{j+2} = \sqrt[j]{\frac{1}{m}\sum_{i=1}^{m} \widehat{\mathbf{x}}_i^j}$
        end if
    end for
    $\mathbf{w}_{k+3} = \ln(m)$
    $\widehat{\tau} = \tau/\alpha - \mu$
end if
return $\mathbf{w}$, $\widehat{\tau}$

For a vector $\mathbf{x}$ of any length, Algorithm 3 returns a set of features $\mathbf{w} \in \mathbb{R}^{k+3}$, where k, the number of moments to compute, is constant for all $\mathbf{x}$-α-τ triples in the dataset. In summary, for each $\mathbf{x}$-α-τ triple, we scale and center $|\mathbf{x}|$ and τ, creating $\widehat{\mathbf{x}}$ and $\widehat{\tau}$, and then compute the following features of $\widehat{\mathbf{x}}$: the minimum, maximum, k moments, and the natural log of the length. In order to keep all features on a similar scale, moments are scaled by $1/m$ and an appropriate root is taken, and the natural log of the length is used. We note that for the first moment, we

use $\frac{1}{m}\sum_{i=1}^{m}|\widehat{\mathbf{x}}_i|$ instead of $\frac{1}{m}\sum_{i=1}^{m}\widehat{\mathbf{x}}_i$ since $\frac{1}{m}\sum_{i=1}^{m}\widehat{\mathbf{x}}_i = 0$ due to the centering of the data. The output of Algorithm 3, $\mathbf{w}$ and $\widehat{\tau}$, become input to the network during training. The network will output an approximation to $\widehat{\tau}$.

2.4.2 Numerical Experiments

Experiments were run on the University of Virginia's High-Performance Computing system, Rivanna. Rivanna's hardware includes multiple nodes and cores per node, and GPUs. We used one (CPU) node, using 8 cores of an Intel(R) Xeon(R) Gold 6248 CPU at 2.50GHz and 72GB of RAM for all experiments unless otherwise noted.

We performed six experiments using a neural network to predict $\widehat{\tau}$. The network architecture used in the experiments is given in Table 2.1. This network is small enough that it did not require nor benefit from training on a GPU (the training time per iteration was longer when trained on the GPU rather than the CPU). In each of the following experiments, the network was trained and tested using single precision PyTorch. To train the network, we used the Adam optimizer with a learning rate of 0.001, and a batch size of 32. The loss function used was mean squared error (MSE) as defined below, where $\widetilde{\tau}$ is the output of the network, i.e., the approximation to $\widehat{\tau}$, and b is the batch size:

$$MSE = \frac{1}{b}\sum_{i=1}^{b}(\widetilde{\tau}_i - \widehat{\tau}_i)^2.$$

In each experiment, Algorithm 3 was used with $k = 10$ moments to preprocess and generate features for each $\mathbf{x}$-α-τ triple. The architecture and number of moments were chosen through experimentation with the goal of having high performance and low complexity. We found that increasing the number of hidden layers, and/or the number of neurons in each layer, and/or the

number of moments did not lead to significant performance gains.

Layer	**Input**	**1st Hidden**	**2nd Hidden**	**Output**
Number of Neurons	$k+3$	25	10	1
Activation Function	-	ReLU	ReLU	-

Table 2.1: Neural network architecture. k is the number of moments computed in Algorithm 3.

Data for each experiment consisted of 10,000 $\mathbf{x}$-α-τ triples where the vector data was either Gaussian $\mathcal{N}(0,1)$ or uniformly distributed $\mathcal{U}(0,1)$ and vectors were of varying lengths (see Table 2.2 for details), α's were sampled from the $\mathcal{U}[1,6)$ distribution, and τ's were computed using Algorithm 1. Vector lengths were sampled from the discrete uniform distribution over [1,000, 2,000] or [1,000, 100,000].

Exp. #	$\mathcal{N}(0,1)$ **Vectors**	$\mathcal{U}(0,1)$ **Vectors**	**Vector Lengths**
1	10,000	0	1,000 - 2,000
2	10,000	0	1,000 - 100,000
3	0	10,000	1,000 - 2,000
4	0	10,000	1,000 - 100,000
5	5,000	5,000	1,000 - 2,000
6	5,000	5,000	1,000 - 100,000

Table 2.2: Vector data summary.

Of the 10,000 $\mathbf{w}$-$\widehat{\tau}$ pairs generated by Algorithm 3, we used an 80-20 random train-test split for experiments 1-4 and an 80-20 random train-test split stratified on vector data distribution for experiments 5-6.

2.4.2.1 Learning Curves and Comparison with "Vanilla Neural Network"

The learning curves for experiments 1-2, 3-4, and 5-6 are presented in Figures 2.1, 2.2, and 2.3 respectively. The training (red) and testing (blue) errors for $\widehat{\tau}$ are the errors for the network output. For a given epoch, the reported training error is the average error per batch. Testing was

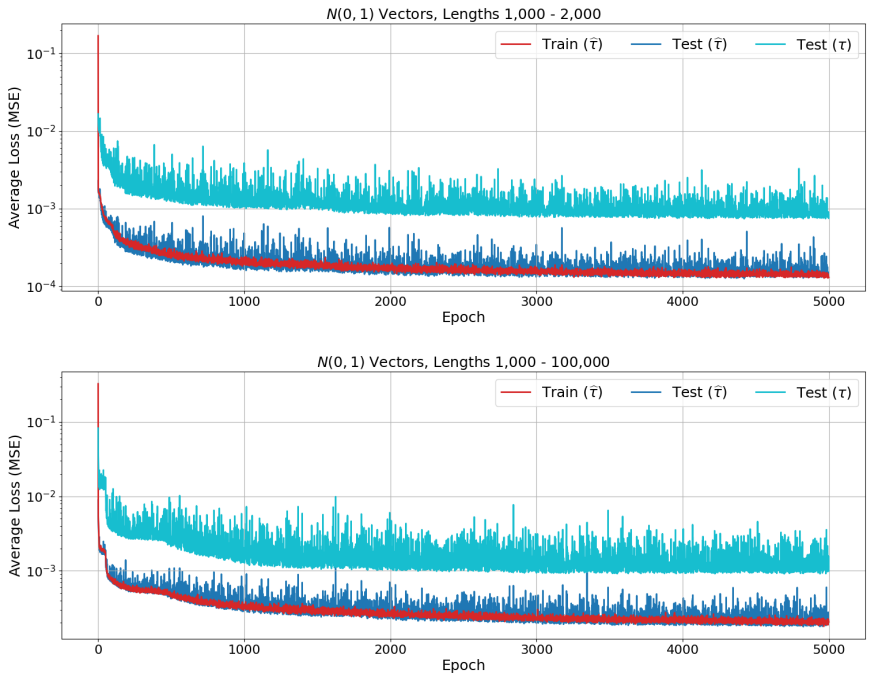


Figure 2.1: Learning curves for experiments 1-2.

completed in a single batch, so the reported testing error is the MSE for the testing set. In addition, we present the testing error on τ (light blue). For a given $\mathbf{x}$-α-τ triple, an approximation to τ is $\alpha(\widetilde{\tau}+\mu)$, where $\widetilde{\tau}$ is the network output, and μ is computed during Algorithm 3. For experiments 5-6, Figure 2.3 also includes the $\widehat{\tau}$ testing error for the Gaussian $\mathcal{N}(0,1)$ vector data (purple) and the uniformly distributed $\mathcal{U}(0,1)$ vector data (orange) separately.

In all experiments the $\widehat{\tau}$ testing error is about one order of magnitude lower than that for τ, which is to be expected since $\alpha \in [1,6)$. In experiments 1-4, we see that the network performs better on $\mathcal{U}(0,1)$ vector data than $\mathcal{N}(0,1)$ vector data. For vectors of length 1,000-2,000, the τ testing error for $\mathcal{U}(0,1)$ data is two orders of magnitude lower than that for $\mathcal{N}(0,1)$ data (10^{-6} vs. 10^{-4}), and for vectors of length 1,000-100,000 the difference is four orders of magnitude (10^{-7} vs. 10^{-3}). This suggests that performance of the network is highly dependent on the vector data distribution. See Supplementary Section 2.7 for additional experiments that explore the

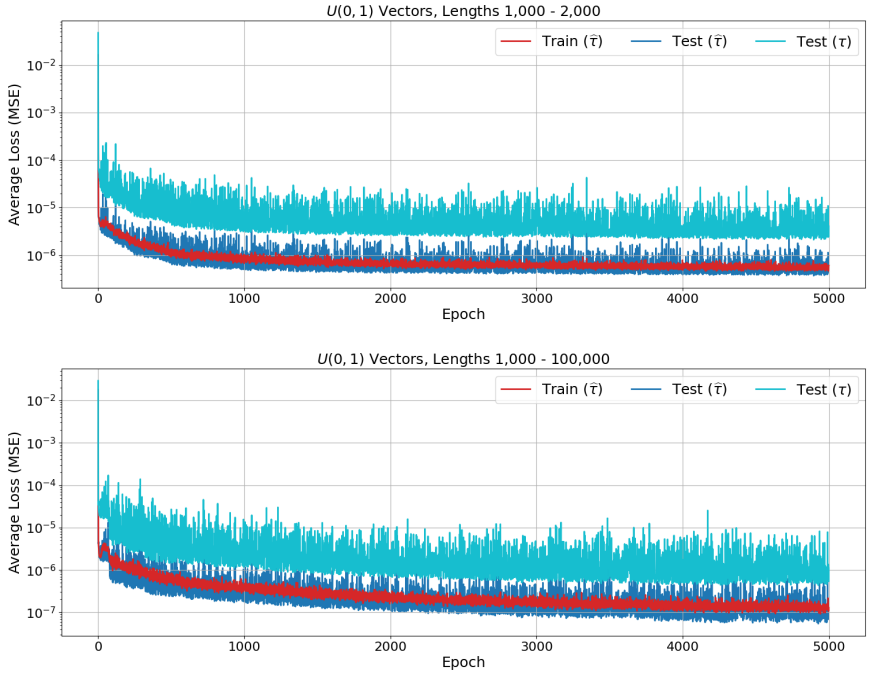


Figure 2.2: Learning curves for experiments 3-4.

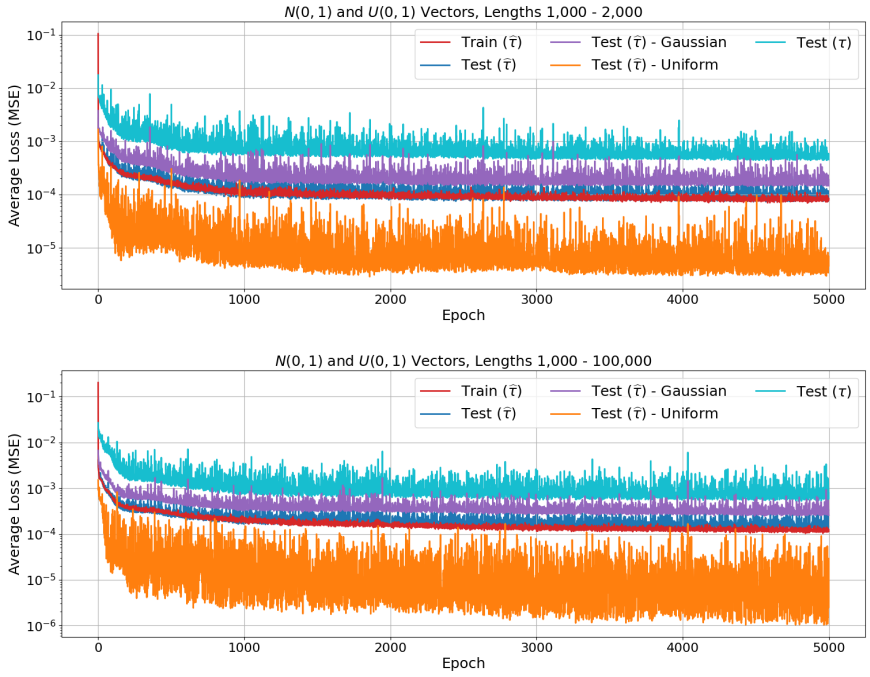


Figure 2.3: Learning curves for experiments 5-6.

difference in network performance for uniformly distributed and Gaussian vector data.

We also note from experiments 3-4 that for $\mathcal{U}(0,1)$ vector data, a larger range of lengths in the dataset improves the τ testing error by an order of magnitude, i.e., 10^{-7} vs. 10^{-6}. For $\mathcal{N}(0,1)$ vector data, a larger range of lengths in the dataset does not significantly change the τ testing error. These patterns generally hold for experiments 5-6 when the dataset includes both $\mathcal{N}(0,1)$ and $\mathcal{U}(0,1)$ vector data. In addition, in experiments 5-6, the $\widehat{\tau}$ testing error is on the order of 10^{-4} for $\mathcal{N}(0,1)$ vector data, which is the same as in experiments 1-2. For $\mathcal{U}(0,1)$ vector data in experiments 5-6, the $\widehat{\tau}$ testing error is on the order of 10^{-6}, which is one to two orders of magnitude higher as compared to experiments 3-4, i.e., 10^{-6} vs. 10^{-7}, for vector lengths 1,000-2,000 and 10^{-6} vs. 10^{-8}, for vector lengths 1,000-100,000.

To gauge the performance of the network, we compare it with a "vanilla neural network" that naively approximates τ using a neural network. The vanilla network processes input vectors of different lengths by zero-padding the end of each vector until all vectors in the dataset are the same length. By Theorem 10, this does not affect τ.

Theorem 10. *Suppose* $\mathbf{x} \in \mathbb{R}^m$ *and* $\tau = ||\boldsymbol{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})||_\infty$. *If* $\mathbf{x}_{\textit{zero-pad}}$ *is a zero padded version of* $\mathbf{x}$, *i.e.,* $\mathbf{x}_{\textit{zero-pad}} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m, 0, ..., 0)$, *then* $\tau_{\textit{zero-pad}} = ||\boldsymbol{prox}_{\alpha||\cdot||_\infty}(\mathbf{x}_{\textit{zero-pad}})||_\infty = \tau$.

Proof. Since $\mathbf{x}_{\text{zero-pad}} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m, 0, ..., 0)$, the computations in each iteration of Algorithm 1 will be the same for $\mathbf{x}_{\text{zero-pad}}$ as they are for $\mathbf{x}$ until the algorithm terminates. Thus, $\tau_{\text{zero-pad}} = ||\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x}_{\text{zero-pad}})||_\infty = \tau$. □

Vanilla network preprocessing is given in Algorithm 4, where ℓ is the maximum vector length in the dataset. The vanilla network architecture is given in Table 2.3. The architecture was chosen to create a fair comparison to the features based network presented previously.

Algorithm 4 Vanilla Network Preprocessing

Input: $\mathbf{x} \in \mathbb{R}^m$, $\alpha \geq 0$, τ, ℓ
Output: $\widehat{\mathbf{x}} \in \mathbb{R}^m$, $\widehat{\tau} \in \mathbb{R}$

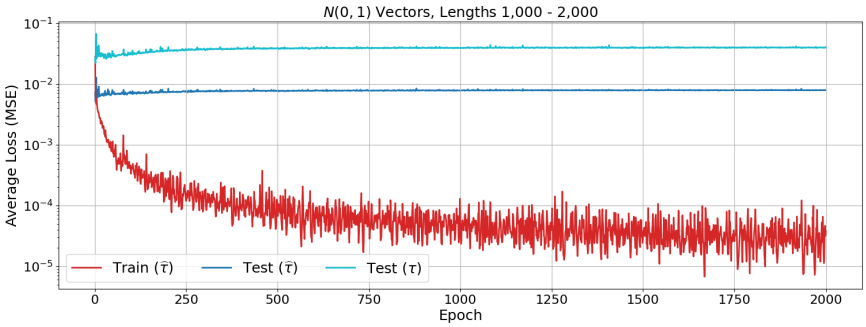
if length$(\mathbf{x}) < \ell$ **then**
    Zero-pad $\mathbf{x}$ so that length$(\mathbf{x}) = \ell$
end if
$\widehat{\mathbf{x}} = |\mathbf{x}|/\alpha$
if $\|\widehat{\mathbf{x}}\|_1 \leq 1$ **then**
    $\widehat{\mathbf{x}} = \emptyset$, $\widehat{\tau} = \emptyset$ ▷ An empty vector and scalar are returned.
else
    $\widehat{\tau} = \tau/\alpha$
end if
return $\widehat{\mathbf{x}}$, $\widehat{\tau}$

Layer	**Input**	**1st Hidden**	**2nd Hidden**	**3rd Hidden**	**Output**
Number of Neurons	ℓ	200	100	50	1
Activation Function	-	ReLU	ReLU	ReLU	-

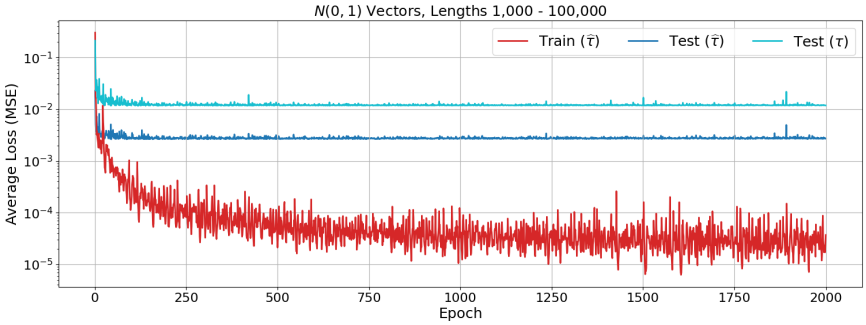
Table 2.3: Vanilla network architecture. ℓ is the maximum vector length in the dataset.

The vanilla network experiments are structured in the same way and use the same data as the features based network experiments. The only exceptions are that 1) due to the network size, the training and testing of the network were completed on an NVIDIA A100 SXM4 40GB GPU, and 2) the testing error on τ (light blue) is calculated using $\alpha\widetilde{\tau}$ as the approximation to τ for an $\mathbf{x}$-α-τ triple. We will denote these experiments V1, V2, ..., V6 to avoid confusion with experiments 1-6 on the features based network. The learning curves for experiments V1-V2, V3-V4, and V5-V6 are presented in Figures 2.4, 2.5, and 2.6 respectively.

The features based network outperforms the vanilla network. For $\mathcal{N}(0,1)$ vector data in experiments V1-V2, the τ testing error is one to two orders of magnitude higher than the features based network (10^{-2} vs. 10^{-4} for vector lengths 1,000-2,000 and 10^{-2} vs. 10^{-3} for vector lengths 1,000-100,000). For $\mathcal{U}(0,1)$ vector data in experiments V3-V4, it is two orders of magnitude higher (10^{-4} vs. 10^{-6} for vector lengths 1,000-2,000 and 10^{-5} vs. 10^{-7} for vector lengths



(a) $\ell = 2{,}000$.

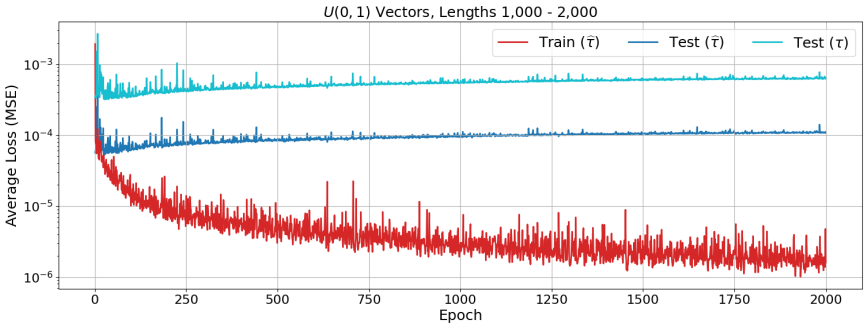


(b) $\ell = 100{,}000$.

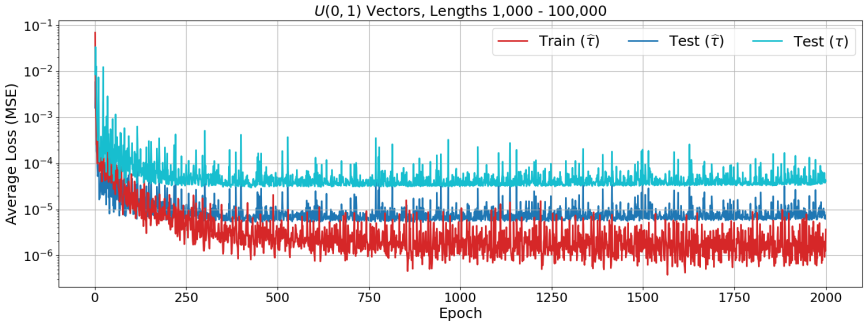
Figure 2.4: Vanilla network learning curves for experiments 1-2.

1,000-100,000). Similar to the features based network, the vanilla network performs better on $\mathcal{U}(0,1)$ vector data than $\mathcal{N}(0,1)$ vector data. However, in the vanilla network, we see overfitting in experiments V1 and V3 on vectors with a smaller range of lengths. Overfitting was not encountered with the features based network.

In experiments V5-V6, the vanilla network struggles to learn when both $\mathcal{N}(0,1)$ and $\mathcal{U}(0,1)$ vector data are included in the dataset. The $\widehat{\tau}$ testing error is one to two orders of magnitude higher for $\mathcal{N}(0,1)$ vector data as compared to experiments V1-V2 (10^{-2} vs. 10^{-3} for vector lengths 1,000-2,000 and 10^{-1} vs. 10^{-3} for vector lengths 1,000-100,000), and three to four orders of magnitude higher for $\mathcal{U}(0,1)$ vector data as compared to experiments V3-V4 (10^{-2} to 10^{-5} for vector lengths 1,000-2,000 and 10^{-2} to 10^{-6} for vector lengths 1,000-100,000).



(a) $\ell = 2{,}000$.

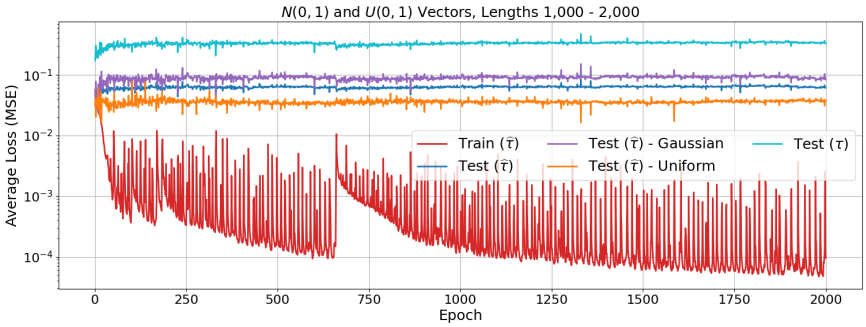


(b) $\ell = 100{,}000$.

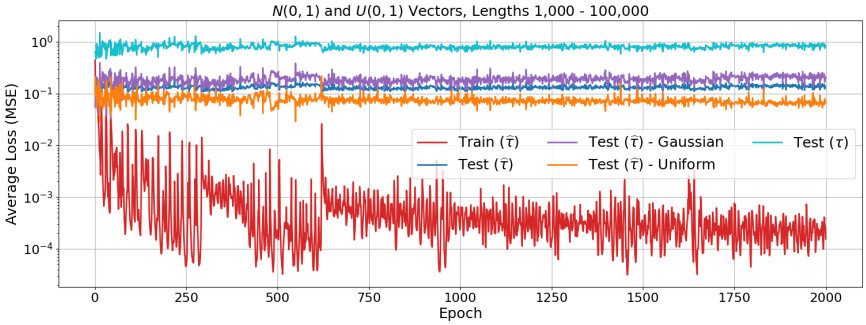
Figure 2.5: Vanilla network learning curves for experiments 3-4.

2.4.2.2 Proximal Operator Error

In each of experiments 1-6, we saved the features based network that achieved the lowest τ testing error. We call these the best models for experiments 1-6. For each best model, we compute the error in the proximal operator and the objective function (defined below) for each instance in the testing set. For $\mathbf{x} \in \mathbb{R}^m$, let $\mathbf{p}(\mathbf{x})$ be the exact proximal operator of $\mathbf{x}$, i.e., $[\mathbf{p}(\mathbf{x})]_k = \sigma_\tau(\mathbf{x}_k)$, and let $\widetilde{\mathbf{p}}(\mathbf{x})$ be the approximate proximal operator of $\mathbf{x}$ computed using $\widetilde{\tau}$, the output of the features based network, i.e., $[\widetilde{\mathbf{p}}(\mathbf{x})]_k = \sigma_{\alpha(\widetilde{\tau}+\mu)}(\mathbf{x}_k)$. Given $\mathbf{y} \in \mathbb{R}^m$, let the objective function f be defined as $f(\mathbf{x}, \mathbf{y}) = \frac{1}{2}||\mathbf{y} - \mathbf{x}||_2^2 + \alpha||\mathbf{y}||_\infty$. Define the proximal operator error, $\delta_{\mathbf{p}}(\mathbf{x})$, and



(a) $\ell = 2{,}000$.



(b) $\ell = 100{,}000$.

Figure 2.6: Vanilla network learning curves for experiments 5-6.

the objective function error, $\delta_f(\mathbf{x})$, as

$$\delta_{\mathbf{p}}(\mathbf{x}) = \frac{\|\mathbf{p}(\mathbf{x}) - \widetilde{\mathbf{p}}(\mathbf{x})\|_2}{\|\mathbf{p}(\mathbf{x})\|_2},$$

and

$$\delta_f(\mathbf{x}) = \frac{f(\mathbf{x}, \widetilde{\mathbf{p}}(\mathbf{x})) - f(\mathbf{x}, \mathbf{p}(\mathbf{x}))}{f(\mathbf{x}, \mathbf{p}(\mathbf{x}))}.$$

In Table 2.4, we report the median, average, and standard deviation for the proximal operator and objective function errors. As expected from the learning curves for experiments 1-6, lower median and average $\delta_{\mathbf{p}}$ and δ_f values are achieved for $\mathcal{U}(0,1)$ vector data than $\mathcal{N}(0,1)$ vector data, usually by an order of magnitude, for vectors of lengths 1,000-2,000 and 1,000-100,000. In addition, median and average $\delta_{\mathbf{p}}$ values for vectors of lengths 1,000-100,000 are less than those

for the corresponding vectors of lengths 1,000-2,000. Median and average $\delta_{\mathbf{p}}$ and δ_f values for experiments 5-6 with $\mathcal{N}(0,1)$ and $\mathcal{U}(0,1)$ vector data are higher than those of only $\mathcal{N}(0,1)$ or $\mathcal{U}(0,1)$ vector data in experiments 1-2 and 3-4, respectively.

			$\delta_{\mathbf{p}}$			δ_f		
Exp.	**Data**	**Vector Lengths**	**median**	**avg**	**sd**	**median**	**avg**	**sd**
1	$\mathcal{N}(0,1)$	1,000 - 2,000	1.5e-3	1.8e-3	1.5e-3	1.7e-4	4.1e-4	6.8e-4
2	$\mathcal{N}(0,1)$	1,000 - 100,000	3.7e-4	5.4e-4	7.5e-4	2.4e-4	5.6e-4	9.8e-4
3	$\mathcal{U}(0,1)$	1,000 - 2,000	4.8e-4	5.8e-4	4.7e-4	1.8e-5	4.4e-5	7.3e-5
4	$\mathcal{U}(0,1)$	1,000 - 100,000	7.1e-5	1.2e-4	1.9e-4	1.3e-5	2.9e-5	4.6e-5
5	$\mathcal{N}(0,1)$ & $\mathcal{U}(0,1)$	1,000 - 2,000	1.6e-3	1.9e-3	1.5e-3	2.0e-4	4.5e-4	7.1e-4
6	$\mathcal{N}(0,1)$ & $\mathcal{U}(0,1)$	1,000 - 100,000	7.5e-4	1.1e-3	1.3e-3	2.7e-3	4.1e-3	4.1e-3

Table 2.4: Median, average, and standard deviation of the proximal operator and objective function errors over the testing set for the best models from experiments 1-6.

2.4.2.3 Feature Importance

We used the best models from experiments 1-6 to measure feature importance by computing saliency for each feature, i.e., the gradient of the network output with respect to each feature. A feature with a higher saliency is considered more important to the network, since a small change in the value of the feature will result in a larger change in the output of the network. For each best model we computed the average saliency of each feature using the testing set feature vectors, i.e., the $\mathbf{w}$ vectors that resulted from inputting the testing set data vectors into Algorithm 3. The results are presented in Figure 2.7.

For $\mathcal{N}(0,1)$ vectors of lengths 1,000-2,000, the 5th-8th moments are relatively more important than other features; however, for $\mathcal{N}(0,1)$ vectors of lengths 1,000-100,000, the 10th moment is by far the most important feature. For $\mathcal{N}(0,1)$ vector data of both length ranges, the minimum is relatively unimportant. For $\mathcal{U}(0,1)$ vectors of lengths 1,000-2,000, the 10th moment is the most important feature, but the maximum, 6th and 8th moments are also relatively important.

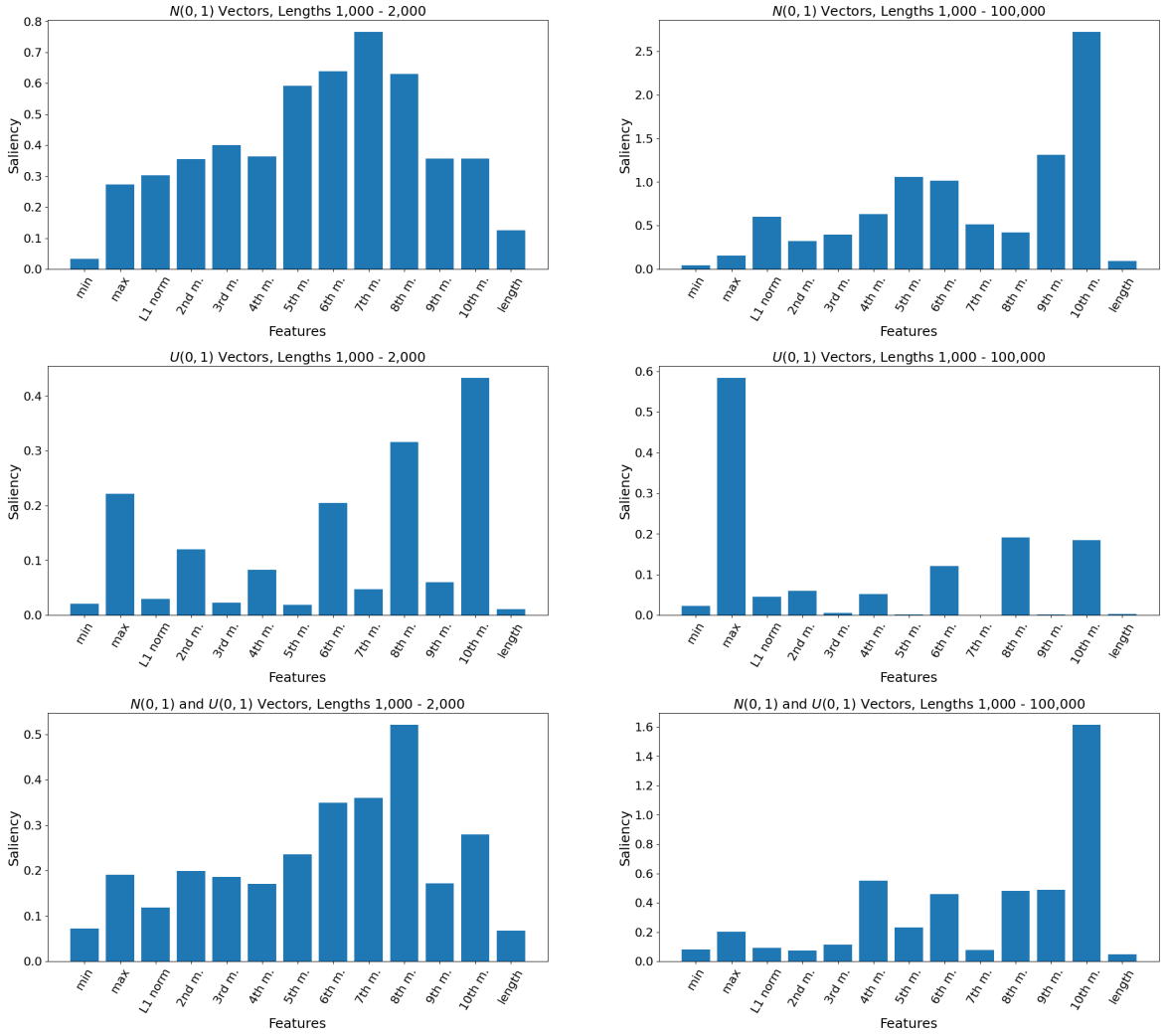


Figure 2.7: Feature importances for the best model from experiments 1-6 as measured by saliency, i.e., the gradient of the network output with respect to each feature.

This is in contrast to $\mathcal{U}(0,1)$ vectors of lengths 1,000-100,000 in which the maximum is by far the most important feature. For $\mathcal{U}(0,1)$ vector data of both length ranges, the even moments are more important than the odd moments, the minimum, and the log of the length.

Experiments 5-6 with $\mathcal{N}(0,1)$ and $\mathcal{U}(0,1)$ vector data exhibit patterns that are fairly similar to those in experiments 1-2 with $\mathcal{N}(0,1)$ vector data only. However, the 8th moment is the most important feature for $\mathcal{N}(0,1)$ and $\mathcal{U}(0,1)$ vectors with lengths 1,000-2,000, as opposed to the 7th moment for $\mathcal{N}(0,1)$ vectors of lengths 1,000-2,000.

2.4.2.4 Computational Efficiency

Table 2.5 gives the average times to compute 1) an approximate proximal operator using the features based network and 2) an exact proximal operator using Algorithm 1. The features based network approach is divided into preprocessing, which includes feature generation, i.e., Algorithm 3 without Line 16 (Line 16 computes a scaled, centered version of τ); inference; and proximal operator computation time (given $\widetilde{\tau}$). The times are averaged over 10,000 vectors of the given length. The lowest average time between the approximate and exact proximal operator is given in bold. For each vector length in Table 2.5, we used 5,000 $\mathcal{N}(0,1)$ vectors and 5,000 $\mathcal{U}(0,1)$ vectors of the given length. The best model from experiment 5 was used for inference for vectors of length 1,000 and the best model from experiment 6 was used for inference for vectors of lengths 10,000 and 100,000. All calculations are double precision except the inference step of the features based network approach, which is single precision. The preprocessing and proximal operator computations for the features based network approach and the exact proximal operator computations operate on one vector at a time via a loop. Since these are embarrassingly parallel computations, we used the numba package and jit function decorator in Python as a speed up.

	Features Based Network Average Times (sec)				
Vector Length	**Preprocessing**	**NN Inference**	**Prox. Op.**	**Total**	**Avg Exact Time**
1,000	2.6e-6	4.5e-6	1.2e-6	8.2e-6	**7.8e-6**
10,000	2.7e-5	4.6e-6	8.9e-6	**4.1e-5**	1.0e-4
100,000	2.6e-4	9.3e-6	1.2e-4	**3.9e-4**	1.4e-3

Table 2.5: Average times to compute 1) an approximate proximal operator using the features based network and 2) an exact proximal operator. Averages are computed over 10,000 vectors of the given length.

As vector length increases, the times for the features based network and exact approaches increase, and the approximate proximal operator is computed faster than the exact proximal op-

erator by an order of magnitude. Inference time is similar for all vector lengths since the network depends on the number of moments calculated for each vector, not vector length. The results in Table 2.5 are consistent with the fact that for $\mathbf{x} \in \mathbb{R}^m$, the features based network approach has complexity $O(m)$ and the exact approach in Algorithm 1 has complexity $O(m \log m)$. For the features based network approach, the constant for the $O(m)$ complexity depends on the number of moments chosen to compute, which motivates choosing a small value for the number of moments.

2.5 Conclusion

We theoretically developed an approach to compute $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$, presented the corresponding $O(m \log m)$ algorithm, and related it to the approach to compute $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ that uses the Moreau decomposition. We then developed a neural network to approximate τ, the threshold calculated in the computation of $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$. The network is novel in that it can accept vectors of varying lengths due to a feature selection process based on the moments of the input data. We showed through numerical experiments that the network can efficiently learn τ. Hence, using this network we can approximate $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$ in $O(m)$ complexity.

2.6 Supplementary Material: Divide and Conquer Algorithm

Similar to the motivation for the expected linear time algorithm of [18] for projection onto the simplex, the motivation for Algorithm 5 is that we actually do not need a full sort of the input vector to find τ; we only need to know the sum of the elements in the set I_τ and $|I_\tau|$. As in Algorithm 1, the main iteration finds a $0 < t \leq \mathbf{s}_1$ such that $\psi'(t) = 0$. Let $\widehat{\mathbf{x}} = \{|\widehat{\mathbf{x}}_i| \,|\, \widehat{\mathbf{x}}_i \neq 0\}$. In

Algorithm 5 Linear Time Computation of the Proximal Operator of the ℓ_∞ Norm

Input: $\mathbf{x} \in \mathbb{R}^m$, $\alpha \geq 0$
Output: $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x}) \in \mathbb{R}^m$

if $||\mathbf{x}||_1 \leq \alpha$ **then**
  $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x}) = \mathbf{0}$
else
  $\widehat{\mathbf{x}} = \{|\widehat{\mathbf{x}}_i| \mid \widehat{\mathbf{x}}_i \neq 0\}$
  $\boldsymbol{\ell} = \emptyset$, $\mathbf{u} = \emptyset$, $\boldsymbol{\ell}^{\text{max}} = 0$
  $\widetilde{\nu} = 0$, $n_{\widetilde{\nu}} = 0$
  while length$(\widehat{\mathbf{x}}) \neq 0$ **do** ▷ Find $\boldsymbol{\ell}^{\text{max}}$ and $\mathbf{u}^{\text{min}}$
    Randomly select a pivot, p, from the elements of $\widehat{\mathbf{x}}$.
    Partition $\widehat{\mathbf{x}}$ such that $\boldsymbol{\ell} = \{\widehat{\mathbf{x}}_i | \ \widehat{\mathbf{x}}_i < p\}$ and $\mathbf{u} = \{\widehat{\mathbf{x}}_i | \ \widehat{\mathbf{x}}_i > p\}$. Let $n_\nu = \text{length}(\mathbf{u})$, $\nu = \sum_{k=1}^{n_\nu} \mathbf{u}_k$, and $n_p = \text{length}(\{\widehat{\mathbf{x}}_i | \ \widehat{\mathbf{x}}_i = p\})$.
    $\psi'(p) = -(\widetilde{\nu} + \nu) + (n_{\widetilde{\nu}} + n_\nu)p + \alpha$ ▷ $\psi'(p) = -\sum_{k \in I_p} |\mathbf{x}_k| + |I_p|p + \alpha$
    if $\psi'(p) < 0$ **then**
      $\boldsymbol{\ell}^{\text{max}} = p$
      $\widehat{\mathbf{x}} = \mathbf{u}$
    else
      $\mathbf{u}^{\text{min}} = p$
      $\widehat{\mathbf{x}} = \boldsymbol{\ell}$
      $\widetilde{\nu} = \widetilde{\nu} + \nu + pn_p$
      $n_{\widetilde{\nu}} = n_{\widetilde{\nu}} + n_\nu + n_p$
    end if
  end while
  $t_0 = (\widetilde{\nu} - \alpha)/n_{\widetilde{\nu}}$ ▷ Find the minimum of $\psi(t)$
  if $\boldsymbol{\ell}^{\text{max}} < t_0 \leq \mathbf{u}^{\text{min}}$ **then**
    $\tau = t_0$
  end if
  $\forall k \in [m]$ compute $[\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})]_k = \sigma_\tau(\mathbf{x}_k)$
end if
return $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$

each while loop iteration of Algorithm 5 we choose an element of $\widehat{\mathbf{x}}$ as the pivot, p, and partition $\widehat{\mathbf{x}}$ by p. We use the strict convexity of ψ and the sign of $\psi(p)$ to determine in which subset of $\widehat{\mathbf{x}}$ to continue searching for τ. After execution of the while loop, there exists a $k \in [m]$ such that $\boldsymbol{\ell}^{\max} = \mathbf{s}_{k+1}$ and $\mathbf{u}^{\min} = \mathbf{s}_k$ for which $\boldsymbol{\ell}^{\max} < \tau \leq \mathbf{u}^{\min}$. At this point we are able to calculate τ since the number of elements greater than or equal to $\mathbf{u}^{\min}$ and the sum of these elements have been updated in each iteration of the while loop. Note that this is equivalent to how τ is computed in Algorithm 1; the difference is that in Algorithm 1 we arrive at the interval $[\mathbf{s}_{k+1}, \mathbf{s}_k)$ by a linear search and in Algorithm 5 we use a divide and conquer technique.

The complexity of Algorithm 5 is based on the pivot selected in line 9. The expected complexity is $O(m)$, but the worst case complexity is $O(m^2)$. We note that if the pivot is chosen to be an element of $\widehat{\mathbf{x}}$ that is approximately the median, the complexity is also $O(m)$, assuming a linear time approximate median finding routine such as median of medians.

2.7 Supplementary Material: Data Dependent Network Performance

In Section 2.4 we demonstrated that performance of the network is highly dependent on the vector data distribution. Specifically, uniformly distributed $\mathcal{U}(0,1)$ vector data performed better than Gaussian $\mathcal{N}(0,1)$ vector data. For vectors of length 1,000-2,000, the τ testing error for $\mathcal{U}(0,1)$ data is two orders of magnitude lower than that for $\mathcal{N}(0,1)$ data (10^{-6} vs. 10^{-4}), and for vectors of length 1,000-100,000 the difference is four orders of magnitude (10^{-7} vs. 10^{-3}). To explore why $\mathcal{U}(0,1)$ vector data performed better than $\mathcal{N}(0,1)$ vector data, we investigated network performance with uniformly distributed $\mathcal{U}(0,a)$ vector data for $a = 10$ and $a = 20$, and compared with that of $\mathcal{N}(0,1)$ vector data.

We performed four additional experiments, which we denote D1-D4. These experiments were structured exactly as experiments 1-6, with the only difference being the datasets used. Dataset information is given in Table 2.6.

Exp. No.	$\mathcal{U}(0, 10)$ **vectors**	$\mathcal{U}(0, 20)$ **vectors**	**Vector Lengths**
D1	10,000	0	1,000 - 2,000
D2	10,000	0	1,000 - 100,000
D3	0	10,000	1,000 - 2,000
D4	0	10,000	1,000 - 100,000

Table 2.6: Vector data summary for experiments D1-D4.

Learning curves are provided in Figures 2.8 and 2.9 for experiments on vectors of lengths 1,000-2,000 and experiments on vectors of lengths 1,000-100,000, respectively. In each figure we include two additional plots that were provided previously in Section 2.4: 1) the learning curves for $\mathcal{U}(0, 1)$ vector data, and 2) the learning curves for $\mathcal{N}(0, 1)$ vector data. In Figures 2.8 and 2.9 we see that for uniformly distributed vector data, performance decreases as variance increases. For vectors of length 1,000-2,000, performance is similar for $\mathcal{U}(0, 20)$ and $\mathcal{N}(0, 1)$ data. However, for vectors of length 1,000-100,000, the τ testing error for $\mathcal{U}(0, 20)$ data is lower than that for $\mathcal{N}(0, 1)$ data by two orders of magnitude (10^{-5} vs. 10^{-3}).

We also provide the feature importance for experiments D1-D4 in Figure 2.10 along with those from Section 2.4 for experiments on $\mathcal{U}(0, 1)$ and $\mathcal{N}(0, 1)$ vector data for comparison. For all uniformly distributed vector data experiments except that on $\mathcal{U}(0, 1)$ vectors of lengths 1,000-2,000, the maximum is by far the most important feature. For $\mathcal{N}(0, 1)$ vector data, the maximum is relatively important for vectors of lengths 1,000-2,000 and relatively unimportant for vectors of lengths 1,000-100,000. Since the computation of τ depends on the largest i elements of $|\mathbf{x}|$ (Algorithm 1), we hypothesize based on these numerical results that the performance of the net-

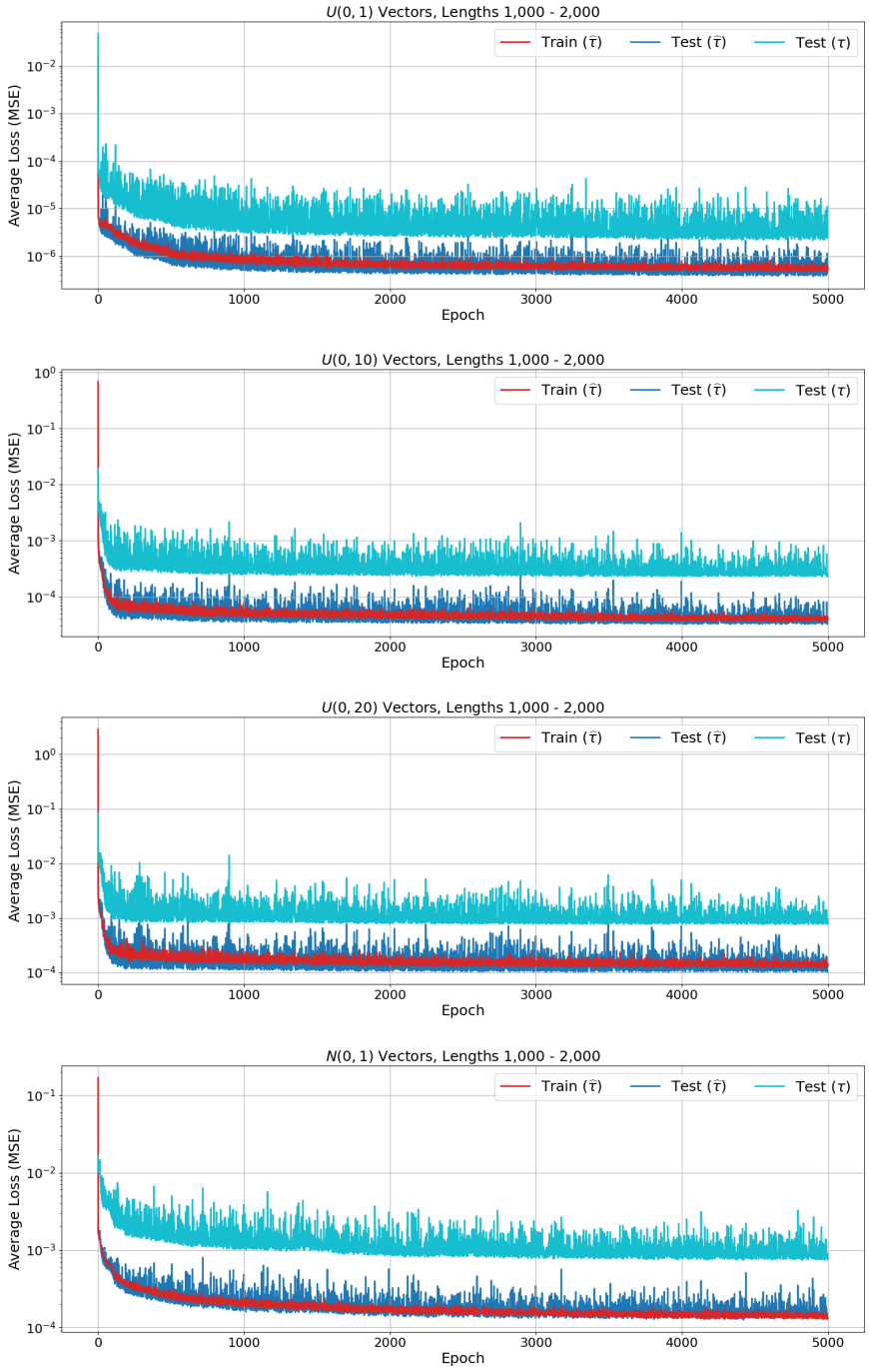


Figure 2.8: Learning curves for vector data with lengths 1,000-2,000, i.e., $\mathcal{U}(0,1)$ data (experiment 3), $\mathcal{U}(0,10)$ data (experiment D1), $\mathcal{U}(0,20)$ data (experiment D3), and $\mathcal{N}(0,1)$ data (experiment 1).

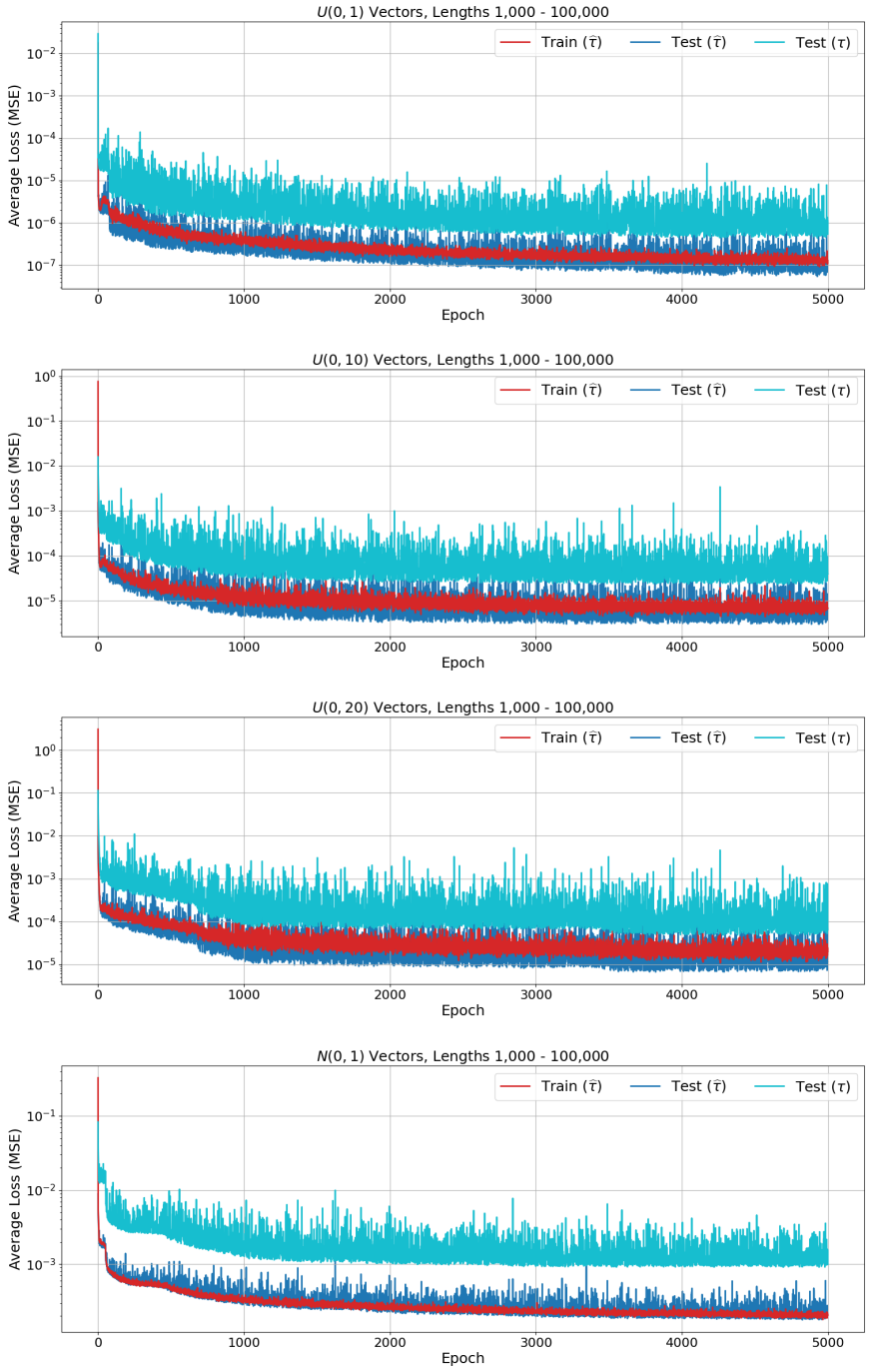


Figure 2.9: Learning curves for vector data with lengths 1,000-100,000, i.e., $\mathcal{U}(0,1)$ data (experiment 4), $\mathcal{U}(0,10)$ data (experiment D2), $\mathcal{U}(0,20)$ data (experiment D4), and $\mathcal{N}(0,1)$ data (experiment 2).

work for a given $\mathbf{x}$ could be related to the density of elements in $|\mathbf{x}|$ near the maximum of $|\mathbf{x}|$. However, more investigation is needed and is a potential area for future work.

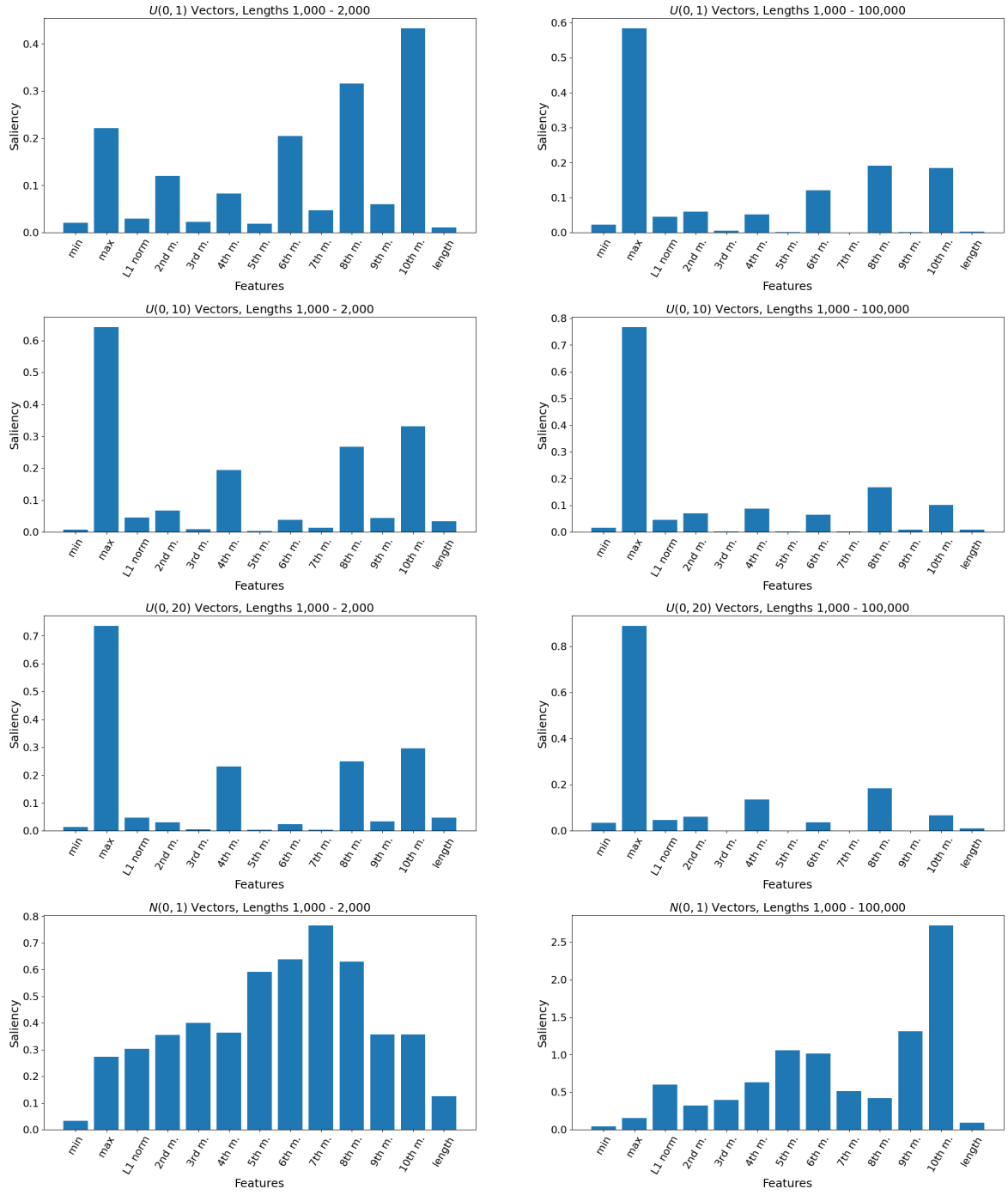


Figure 2.10: Feature importances for experiments 3-4, D1-D4, and 1-2 as measured by saliency, i.e., the gradient of the network output with respect to each input.

Chapter 3: CUR Algorithm Utilizing Convex Optimization

3.1 Introduction

Low rank matrix approximations are common tools in many applications including principal component analysis (PCA), signal denoising, and least squares. While the truncated singular value decomposition (SVD) is the optimal approximation in terms of matrix reconstruction (Eckart-Young theorem), the singular vectors cannot be interpreted in terms of the original data. However, the approximate CUR matrix factorization can be interpreted in terms of the original data, making it an attractive low-rank approximation option. In addition, CUR maintains the structure of the data, for example sparsity or nonnegativity.

The approximate CUR factorization of $\mathbf{X} \in \mathbb{R}^{m \times n}$ is generally computed in three steps: 1) select $c \in \mathbb{R}$ columns of $\mathbf{X}$ and let $\mathbf{C} \in \mathbb{R}^{m \times c}$ contain these columns, 2) select $r \in \mathbb{R}$ rows of $\mathbf{X}$ and let $\mathbf{R} \in \mathbb{R}^{r \times n}$ contain these rows, and 3) compute $\mathbf{U} \in \mathbb{R}^{c \times r}$ so that $\mathbf{CUR}$ is a good approximation to $\mathbf{X}$. The result is a matrix approximation

$$\underset{m \times n}{\mathbf{X}} \approx \underset{m \times c}{\mathbf{C}} \; \underset{c \times r}{\mathbf{U}} \; \underset{r \times n}{\mathbf{R}},$$

where generally $c \ll n$ and $r \ll m$. While various algorithms exist to compute a CUR approximation, we are particularly interested in deterministic CUR algorithms that can independently

select columns of $\mathbf{X}$ without simultaneous selection of its rows due to the fact that 1) several applications exist that seek important matrix columns or rows and not a full matrix factorization [2], and 2) for a practical application, a randomized CUR will likely produce a different set of important columns and/or rows in each run of the algorithm, which may not be desirable to the scientist [8, 13].

One deterministic approach to computing a CUR approximation is to select columns and rows of $\mathbf{X}$ for inclusion in $\mathbf{C}$ and $\mathbf{R}$ using convex optimization with regularization. In this work, we propose a novel CUR algorithm utilizing convex optimization with contributions in the formulation, implementation, and application of CUR. The main contributions of the work are 1) a novel convex optimization formulation for CUR, 2) an algorithm utilizing convex optimization that solves for $\mathbf{C}$ and $\mathbf{R}$ separately and allows the user to select c and r, and 3) an implementation utilizing the "surrogate functional" technique [14], which we adapt for use with a new penalty function. We also note that our CUR algorithm and implementation can accommodate a variety of penalty functions, allowing the user a flexible framework. In addition, we provide numerical results that compare our CUR algorithm with the SVD and other deterministic CUR algorithms that select $\mathbf{C}$ and $\mathbf{R}$ separately and allow the user to select c and r.

The remainder of this chapter is organized as follows: related work is given in Section 3.2, our novel CUR algorithm utilizing convex optimization is presented in Section 3.3, the theoretical foundations of the algorithm are given in Section 3.4, numerical experiments are provided in Section 3.5, and a conclusion is given in Section 3.6. Throughout this work we use MATLAB notation to denote rows and columns of matrices, e.g., row i of $\mathbf{X}$ is denoted $\mathbf{X}(i,:)$ and column j of $\mathbf{X}$ is denoted $\mathbf{X}(:,j)$. In addition, the set $\{1, 2, ..., n\}$ is denoted $[n]$.

3.2 Related Work

Work on the CUR approximation has generally focused on algorithm development, theoretical accuracy guarantees, and applications. There are several algorithms for computing the CUR approximation of a matrix; some are randomized, e.g., [1,5,6], and others are deterministic, e.g., [2, 7]. For a history of CUR see [3], and for a survey of algorithms see [24]. Applications of CUR include clustering documents [1], classification using genetic data [2], image feature selection for classification problems [9], video background-foreground separation [10], and sensor selection and channel assignment [11].

Deterministic CUR algorithms that solve for $\mathbf{C}$ and $\mathbf{R}$ separately and allow the user to select c and r include a leverage score approach [1], a discrete empirical interpolation method (DEIM) approach [2], and a pivoted QR approach [7]. The leverage score approach by Mahoney and Drineas [1] is often compared to in the CUR literature. This approximation is randomized and columns and rows are sampled based on their "normalized statistical leverage scores", which capture information on how much a column or row contributes to the optimal low-rank approximation to the data matrix, the rank-k SVD, where k is a rank parameter chosen by the user. However, a deterministic variant of this algorithm is to select the columns and rows with the largest leverage scores for inclusion in $\mathbf{C}$ and $\mathbf{R}$, respectively. In the DEIM approach by Sorenson and Embree [2], columns are chosen for inclusion in $\mathbf{C}$ and rows are chosen for inclusion in $\mathbf{R}$ using the discrete empirical interpolation method (DEIM) on the top-k right and left singular vectors of the data matrix, respectively, where $k = c = r$. In the pivoted QR approach by Stewart [7], columns are selected for inclusion in $\mathbf{C}$ and rows are selected for inclusion in $\mathbf{R}$ using a pivoted QR factorization of $\mathbf{X}$ and $\mathbf{X}^T$, respectively. Sorenson and Embree [2] present a slight

adaptation of this approach in which rows are selected for inclusion in $\mathbf{R}$ using a pivoted QR factorization of $\mathbf{C}^T$. In each of these CUR algorithms, $\mathbf{U}$ is computed as $\mathbf{U} = \mathbf{C}^+\mathbf{X}\mathbf{R}^+$, i.e., the minimizer of $\|\mathbf{X} - \mathbf{C}\mathbf{U}\mathbf{R}\|_F$ [7].

Since the CUR algorithm that we present in this work uses convex optimization with regularization to select columns and rows of the data matrix for inclusion in $\mathbf{C}$ and $\mathbf{R}$, we also note related work in this area. In [13] Bien, Xu, and Mahoney related CUR to sparse PCA and used the following convex minimization problem to find $\mathbf{C}$:

$$\mathbf{B}^* = \underset{\mathbf{B}\in\mathbb{R}^{n\times n}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{X}\mathbf{B}\|_F + \lambda \sum_{i=1}^{n} \|\mathbf{B}(i,:)\|_2, \tag{3.1}$$

where $\lambda > 0$ is a regularization parameter. The indices of nonzero rows of $\mathbf{B}^*$ are the indices of columns to choose from $\mathbf{X}$ for inclusion in $\mathbf{C}$. $\mathbf{R}$ can be found using a similar optimization problem; however, the computation of $\mathbf{U}$ is not discussed. Mairal et al. [8] formulated CUR as a convex optimization problem that selects columns and rows of $\mathbf{X}$ at the same time, i.e.,

$$\mathbf{W}^* = \underset{\mathbf{W}\in\mathbb{R}^{n\times m}}{\operatorname{argmin}} ||\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{X}||_F^2 + \lambda_{\text{row}} \sum_{i=1}^{m} \|\mathbf{W}(:,i)\|_\infty + \lambda_{\text{col}} \sum_{j=1}^{n} \|\mathbf{W}(j,:)\|_\infty, \tag{3.2}$$

where $\lambda_{\text{row}}, \lambda_{\text{col}} > 0$ are regularization parameters. Similar to [13], the nonzero row indices of $\mathbf{W}^*$ are the indices of columns to select from $\mathbf{X}$, and the nonzero column indices of $\mathbf{W}^*$ are the indices of rows to select from $\mathbf{X}$. $\mathbf{U}$ is computed as $\mathbf{U} = \mathbf{C}^+\mathbf{X}\mathbf{R}^+$. In both [8, 13], the convex optimization CUR algorithm achieves similar matrix reconstruction accuracy to that of the leverage score based CUR [1] in numerical experiments. In addition, Ida et al. [25] presented a method to speed up the coordinate descent algorithm for solving Equation 3.1 as presented

in [13] and claimed it can be extended to solve Equation 3.2 as well.

In [26], Peng et al. used an optimization problem with regularization terms that simultaneously performed a CUR approximation of a network node attribute matrix (to choose representative nodes and attributes at the same time) and residual analysis in order to detect anomalies on attributed networks. The part of the optimization formulation related to CUR is similar to Equation 3.2, but uses the ℓ_2 norm rather than the ℓ_∞ norm in the regularization terms. This optimization problem was solved using alternating convex optimizations, and parameters were chosen using a grid search in experimental results. In each of the convex optimization CUR approaches mentioned above [8, 13, 25, 26] there is not a built-in algorithmic control for selecting c columns and r rows of the data matrix.

Li et al. [27] used a convex optimization CUR to simultaneously select features and representative data samples in order to perform feature selection and active learning at the same time. The optimization problem used is similar to Equation 3.2 except the regularization terms use the ℓ_2 norm rather than the ℓ_∞ norm, and there is an additional regularization term that provides "local linear reconstruction". Parameters were grid searched in experimental results and after the optimization problem is solved, the indices of the c rows of $\mathbf{W}^*$ with the largest ℓ_2 norms are the indices of columns selected from the data matrix for inclusion in $\mathbf{C}$. The indices of r rows to include in $\mathbf{R}$ are found similarly.

3.3 CUR Algorithm

Let $\mathbf{X} \in \mathbb{R}^{m \times n}$ be the matrix we wish to approximate as $\mathbf{X} = \mathbf{CUR}$. Our formulation of CUR using convex optimization builds upon ideas from [13], [1], and [8]. To select a subset of

columns from $\mathbf{X}$ to form the matrix $\mathbf{C}$, we solve

$$\mathbf{W}^* = \underset{\mathbf{W}\in\mathbb{R}^{n\times m}}{\operatorname{argmin}} ||\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{X}||_F^2 + \lambda_C \sum_{i=1}^{n} \|\mathbf{W}(i,:)\|_\infty, \tag{3.3}$$

for a given $\lambda_C \in \mathbb{R} \geq 0$. Then $\mathbf{C} = \mathbf{X}(:, I_C)$, where I_C is the set of indices of nonzero rows in $\mathbf{W}^*$. Hence λ_C controls how many columns are selected from $\mathbf{X}$. i.e., the larger the value of λ_C, the more rows of $\mathbf{W}$ are forced to $\mathbf{0}$, and the fewer columns of $\mathbf{X}$ are selected.

After $\mathbf{C}$ has been calculated, we select a set of rows from $\mathbf{X}$ to form the matrix $\mathbf{R}$ by solving

$$\mathbf{W}^* = \underset{\mathbf{W}\in\mathbb{R}^{c\times m}}{\operatorname{argmin}} ||\mathbf{X} - \mathbf{C}\mathbf{W}\mathbf{X}||_F^2 + \lambda_R \sum_{j=1}^{m} \|\mathbf{W}(:,j)\|_\infty, \tag{3.4}$$

for a given $\lambda_R \in \mathbb{R} \geq 0$. Then $\mathbf{R} = \mathbf{X}(I_R, :)$, where I_R is the set of indices of nonzero columns in $\mathbf{W}^*$, and λ_R controls the number of rows of selected.

Input to our algorithm includes c and r, the number of columns and rows to be selected from $\mathbf{X}$ for $\mathbf{C}$ and $\mathbf{R}$, respectively. For a given λ_C, it is unknown in advance how many columns will be selected from $\mathbf{X}$ by the solution of Equation 3.3. Hence, we utilize bisection on λ_C with multiple iterates of the column selection procedure to find a selection of exactly c columns. We use a similar process with λ_R and the row selection procedure to find a selection of exactly r rows. To complete the algorithm, $\mathbf{U}$ is computed as $\mathbf{U} = \mathbf{C}^+\mathbf{X}\mathbf{R}^+$, where $\mathbf{X}^+$ denotes the Moore-Penrose generalized inverse or pseudoinverse of $\mathbf{X}$. The pseudocode for our CUR approximation is given in Algorithm 6.

The initial minimum value for λ_C in the bisection method is 0, which corresponds to potentially all columns of $\mathbf{X}$ being selected for the matrix $\mathbf{C}$. The initial maximum value for λ_C in

the bisection method is the smallest value of λ_C that forces zero columns of $\mathbf{X}$ to be selected, i.e., the solution to Equation 3.3 to be $\mathbf{W} = \mathbf{0}$. We call this the critical value of λ_C and denote it λ_C^*. The range of values for λ_R in the bisection method with Equation 3.4 is set similarly, with the critical value of λ_R being denoted λ_R^*. To prove the exact values of λ_C^* and λ_R^*, we first provide a helpful lemma and note that Equations 3.3 and 3.4 can be reshaped as

$$\mathbf{w}^* = \min_{\mathbf{w}\in\mathbb{R}^{mn}} ||(\mathbf{X}^\mathbf{T} \otimes \mathbf{X})\mathbf{w} - \mathbf{b}||_2^2 + \lambda_C \sum_{i=1}^{n} \max_{1\le j\le m} |\mathbf{w}_{i+(j-1)n}|, \tag{3.5}$$

and

$$\mathbf{w}^* = \min_{\mathbf{w}\in\mathbb{R}^{mc}} ||(\mathbf{X}^\mathbf{T} \otimes \mathbf{C})\mathbf{w} - \mathbf{b}||_2^2 + \lambda_R \sum_{j=1}^{m} \max_{1\le i\le c} |\mathbf{w}_{i+(j-1)n}|, \tag{3.6}$$

where $\otimes$ denotes the Kronecker product, $\mathbf{b} = \text{vec}(\mathbf{X}) \in \mathbb{R}^{mn}$, i.e. a column-stacked version of $\mathbf{X}$, $\mathbf{w} = \text{vec}(\mathbf{W}) \in \mathbb{R}^{mn}$ (Equation 3.5) or $\mathbf{w} = \text{vec}(\mathbf{W}) \in \mathbb{R}^{mc}$ (Equation 3.6), and $\mathbf{w}^*$ is defined similarly to $\mathbf{w}$.

Lemma 11. *Let* $\mathbf{v} \in \mathbb{R}^m$ *be fixed, and* $\lambda \in \mathbb{R}$. *If* $\langle \mathbf{v}, \mathbf{x}\rangle \le \frac{\lambda}{2}||\mathbf{x}||_\infty \; \forall \mathbf{x} \in \mathbb{R}^m$, *then* $\lambda \ge 2||\mathbf{v}||_1$.

Proof.

$$\langle \mathbf{v}, \mathbf{x}\rangle = \sum_{k=1}^{m} \mathbf{v}_k\mathbf{x}_k \le \left(\sum_{k=1}^{m} |\mathbf{v}_k|\right) \max_{1\le k\le m} |\mathbf{x}_k| = ||\mathbf{v}||_1||\mathbf{x}||_\infty.$$

Hence for $\mathbf{x}_k = \text{sign}(\mathbf{v}_k)$, $\langle \mathbf{v}, \mathbf{x}\rangle = ||\mathbf{v}||_1||\mathbf{x}||_\infty$ and $\lambda = 2||\mathbf{v}||_1$. Since the smallest value of λ which holds $\forall \mathbf{x} \in \mathbb{R}^m$ will occur when $\langle \mathbf{v}, \mathbf{x}\rangle = \frac{\lambda}{2}||\mathbf{x}||_\infty$, it is the case that $\forall \mathbf{x} \in \mathbb{R}^m$, $\lambda \ge 2||\mathbf{v}||_1$. □

Theorem 12. *Let* $\mathbf{M} = \text{reshape}((\mathbf{X}^\mathbf{T} \otimes \mathbf{X})^\mathbf{T}\mathbf{b}, n, m)$, *i.e.* $(\mathbf{X}^\mathbf{T} \otimes \mathbf{X})^\mathbf{T}\mathbf{b}$ *reshaped from* $\mathbb{R}^{mn}$ *to*

$\mathbb{R}^{n\times m}$. *Then*

$$\lambda_C^* = 2\max_{1\le i\le n} \|\mathbf{M}(i,:)\|_1 = 2||\mathbf{M}||_\infty.$$

Similarly, let $\mathbf{N} = \text{reshape}((\mathbf{X^T}\otimes\mathbf{C})^\mathbf{T}\mathbf{b}, c, m)$. *Then*

$$\lambda_R^* = 2\max_{1\le j\le m} \|\mathbf{N}(:,j)\|_1 = 2||\mathbf{N}||_1.$$

Proof. Let $\mathbf{A} = (\mathbf{X^T}\otimes\mathbf{X})$ and the objective function of Equation 3.5 be $J(\mathbf{w})$:

$$J(\mathbf{w}) = ||\mathbf{Aw}-\mathbf{b}||_2^2 + \lambda_C \sum_{i=1}^{n} \max_{1\le j\le m} |\mathbf{w}_{i+(j-1)n}|.$$

We want to find the smallest $\lambda_C^* > 0$ such that $\forall\lambda \ge \lambda_C^*$, $\text{argmin}_{\mathbf{w}\in\mathbb{R}^{mn}} J(\mathbf{w}) = \mathbf{0}$. We have

$$\begin{aligned}
J(\mathbf{w}) &= ||\mathbf{Aw}||_2^2 - 2\mathbf{w^T}\mathbf{A^T}\mathbf{b} + ||\mathbf{b}||_2^2 + \lambda_C \sum_{i=1}^{n} \max_{1\le j\le m} |\mathbf{w}_{i+(j-1)n}| \\
&= ||\mathbf{Aw}||_2^2 + ||\mathbf{b}||_2^2 + \lambda_C \sum_{i=1}^{n} \max_{1\le j\le m} |\mathbf{w}_{i+(j-1)n}| - 2\sum_{k=1}^{mn} (\mathbf{A^T}\mathbf{b})_k \mathbf{w}_k \\
&= ||\mathbf{Aw}||_2^2 + ||\mathbf{b}||_2^2 + \lambda_C \sum_{i=1}^{n} \max_{1\le j\le m} |\mathbf{w}_{i+(j-1)n}| - 2\sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{M}_{ij}\mathbf{W}_{ij},
\end{aligned}$$

where $\mathbf{W} = \text{reshape}(\mathbf{w}, n, m)$. If $\forall i$,

$$\frac{\lambda_C}{2} \max_{1\le j\le m} |\mathbf{w}_{i+(j-1)n}| - \sum_{j=1}^{m} \mathbf{M}_{ij}\mathbf{W}_{ij} \ge 0, \tag{3.7}$$

then

$$\sum_{i=1}^{n} \left[\frac{\lambda_C}{2} \max_{1\le j\le m} |\mathbf{w}_{i+(j-1)n}| - \sum_{j=1}^{m} \mathbf{M}_{ij}\mathbf{W}_{ij} \right] \ge 0,$$

and

$$\lambda_C \sum_{i=1}^{n} \max_{1\leq j\leq m} |\mathbf{w}_{i+(j-1)n}| - 2\sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{M}_{ij}\mathbf{W}_{ij} \geq 0.$$

Hence, $J(\mathbf{w}) \geq J(\mathbf{0}) = ||\mathbf{b}||_2^2$ for all $\mathbf{w}$, thus $\operatorname{argmin}_{\mathbf{w}\in\mathbb{R}^{mn}} J(\mathbf{w}) = \mathbf{0}$. By Lemma 11, assuming Equation 3.7 is true, $\forall i$, $\lambda_C \geq 2||\mathbf{M}(i,:)||_1$. Thus, $\lambda_C^* = 2\max_{1\leq i\leq n} \|\mathbf{M}(i,:)\|_1 = 2||\mathbf{M}||_\infty$.

A similar proof can be used to show the result for λ_R^*, letting $\mathbf{A} = (\mathbf{X}^\mathbf{T} \otimes \mathbf{C})$ and $J(\mathbf{w})$ be the objective function of Equation 3.6:

$$J(\mathbf{w}) = ||\mathbf{A}\mathbf{w} - \mathbf{b}||_2^2 + \lambda_R \sum_{j=1}^{m} \max_{1\leq i\leq c} |\mathbf{w}_{i+(j-1)n}|.$$

□

3.3.1 Implementation for Minimization Problems

For $\mathbf{X}$ of small[1] dimensions, we can solve the minimization problems on lines 5 and 19 of Algorithm 6 using the reshaped versions (Equations 3.5 and 3.6) and a convex programming solver such as the CVX package in MATLAB [28, 29]. However using a solver for $\mathbf{X}$ of larger dimensions becomes infeasible due to the use of the Kronecker product in the reshaped problems. For example, to store the genetics dataset referenced in Section 3.5, a dense double array $\mathbf{X} \in \mathbb{R}^{107\times 22,283}$, 19.07 MB are used; to store $\mathbf{X}^T \otimes \mathbf{X} \in \mathbb{R}^{2,384,281\times 2,384,281}$, 45.48 TB are used.

To accommodate large-scale problems we solve these minimizations in Algorithm 6 using an extension of an iterative method that utilizes a "surrogate functional" to solve regularized least squares minimization problems in which weighted ℓ_p-norm penalty functions are used, for $1 \leq$

[1] Small is relative to the user's computer memory size.

Algorithm 6 CUR through Convex Optimization

Input: $\mathbf{X} \in \mathbb{R}^{m \times n}$, $c > 0$, $r > 0$
Output: $\mathbf{C} \in \mathbb{R}^{m \times c}$, $\mathbf{U} \in \mathbb{R}^{c \times r}$, $\mathbf{R} \in \mathbb{R}^{r \times n}$ such that $\mathbf{X} \approx \mathbf{CUR}$

$\lambda_C^* = \|\mathbf{X}^T\mathbf{X}\mathbf{X}^T\|_\infty$
$n_c = 0$, $\lambda_{\text{min}} = 0$, $\lambda_{\text{max}} = \lambda_C^*$
while $n_c \neq c$ **do**
    $\lambda_C = (\lambda_{\text{max}} + \lambda_{\text{min}})/2$
    solve $\mathbf{W}^* = \text{argmin}_{\mathbf{W} \in \mathbb{R}^{n \times m}} ||\mathbf{X} - \mathbf{XWX}||_F^2 + \lambda_C \sum_{i=1}^{n} \|\mathbf{W}(i,:)\|_\infty$
    let I_C be the set of indices of nonzero rows of $\mathbf{W}^*$
    $n_c = |I_C|$
    if $c > n_c$ **then**
        $\lambda_{\text{min}} = \lambda_C$
    else if $c < n_c$ **then**
        $\lambda_{\text{max}} = \lambda_C$
    end if
end while
$\mathbf{C} = \mathbf{X}(:, I_C)$
$\lambda_R^* = \|\mathbf{C}^T\mathbf{X}\mathbf{X}^T\|_1$
$n_r = 0$, $\lambda_{\text{min}} = 0$, $\lambda_{\text{max}} = \lambda_R^*$
while $n_r \neq r$ **do**
    $\lambda_R = (\lambda_{\text{max}} + \lambda_{\text{min}})/2$
    solve $\mathbf{W}^* = \text{argmin}_{\mathbf{W} \in \mathbb{R}^{c \times m}} ||\mathbf{X} - \mathbf{CWX}||_F^2 + \lambda_R \sum_{j=1}^{m} \|\mathbf{W}(:,j)\|_\infty$
    let I_R be the set of indices of nonzero columns of $\mathbf{W}^*$
    $n_r = |I_R|$
    if $r > n_r$ **then**
        $\lambda_{\text{min}} = \lambda_R$
    else if $r < n_r$ **then**
        $\lambda_{\text{max}} = \lambda_R$
    end if
end while
$\mathbf{R} = \mathbf{X}(I_R, :)$
$\mathbf{U} = \mathbf{C}^+\mathbf{X}\mathbf{R}^+$
return $\mathbf{C}, \mathbf{U}, \mathbf{R}$

$p \leq 2$ [14]. This technique decouples a large minimization problem into smaller, easy-to-solve problems. We extend the results of [14] to apply to our penalty functions, e.g. $\sum_{i=1}^{n} \|\mathbf{W}(i,:)\|_\infty$. We demonstrate the method for the line 5 minimization in Algorithm 6. The line 19 minimization is handled similarly; details can be found in Supplementary Section 3.7.

Let the objective function of Equation 3.3 be denoted by

$$J(\mathbf{W}) = ||\mathbf{X} - \mathbf{XWX}||_F^2 + \lambda_C \sum_{i=1}^{n} \|\mathbf{W}(i,:)\|_\infty,$$

and the corresponding surrogate functional by

$$\widehat{J}(\mathbf{W},\mathbf{Z}) = ||\mathbf{X} - \mathbf{XWX}||_F^2 + \lambda_C \sum_{i=1}^{n} \|\mathbf{W}(i,:)\|_\infty + \mu||\mathbf{W} - \mathbf{Z}||_F^2 - ||\mathbf{XWX} - \mathbf{XZX}||_F^2,$$

where $\mathbf{Z} \in \mathbb{R}^{n\times m}$ and $\mu > 0$. For any $\mathbf{Z} \in \mathbb{R}^{n\times m}$ and $\mu > 0$ we have

$$\begin{aligned}
\widehat{J}(\mathbf{W},\mathbf{Z}) &= \mu||\mathbf{W}||_F^2 - 2\,\mathrm{tr}\{\mathbf{W}(\mu\mathbf{Z}^\mathbf{T} + \mathbf{XX}^\mathbf{T}\mathbf{X} - \mathbf{XX}^\mathbf{T}\mathbf{Z}^\mathbf{T}\mathbf{X}^\mathbf{T}\mathbf{X})\} + \lambda_C \sum_{i=1}^{n} \|\mathbf{W}(i,:)\|_\infty \\
&\quad + ||\mathbf{X}||_F^2 + \mu||\mathbf{Z}||_F^2 - ||\mathbf{XZX}||_F^2 \\
&= \sum_{i=1}^{n} \left[\mu||\mathbf{W}(i,:)||_2^2 - 2\langle \mathbf{W}(i,:), \mathbf{L}^\mathbf{T}(i,:)\rangle + \lambda_C||\mathbf{W}(i,:)||_\infty\right] \\
&\quad + ||\mathbf{X}||_F^2 + \mu||\mathbf{Z}||_F^2 - ||\mathbf{XZX}||_F^2,
\end{aligned}$$

where $\mathbf{L} = \mu\mathbf{Z}^\mathbf{T} + \mathbf{XX}^\mathbf{T}\mathbf{X} - \mathbf{XX}^\mathbf{T}\mathbf{Z}^\mathbf{T}\mathbf{X}^\mathbf{T}\mathbf{X}$. Hence,

$$\begin{aligned}
\underset{\mathbf{W}\in\mathbb{R}^{n\times m}}{\mathrm{argmin}}\ \widehat{J}(\mathbf{W},\mathbf{Z}) &= \underset{\mathbf{W}\in\mathbb{R}^{n\times m}}{\mathrm{argmin}}\ \mu \sum_{i=1}^{n} \left[||\mathbf{W}(i,:) - \frac{1}{\mu}\mathbf{L}^\mathbf{T}(i,:)||_2^2 - ||\frac{1}{\mu}\mathbf{L}^\mathbf{T}(i,:)||_2^2 + \frac{\lambda_C}{\mu}||\mathbf{W}(i,:)||_\infty\right] \\
&= \underset{\mathbf{W}\in\mathbb{R}^{n\times m}}{\mathrm{argmin}}\ 2\mu \sum_{i=1}^{n} \left[\frac{1}{2}||\mathbf{W}(i,:) - \frac{1}{\mu}\mathbf{L}^\mathbf{T}(i,:)||_2^2 + \frac{\lambda_C}{2\mu}||\mathbf{W}(i,:)||_\infty\right].
\end{aligned}$$

Thus, we can easily minimize $\widehat{J}$ over $\mathbf{W}$ by computing the proximal operator of the ℓ_∞ norm,

$$\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x}) = \underset{\mathbf{y}\in\mathbb{R}^m}{\operatorname{argmin}} \left(\frac{1}{2}||\mathbf{y}-\mathbf{x}||_2^2 + \alpha||\mathbf{y}||_\infty \right) \tag{3.8}$$

for $\mathbf{x} \in \mathbb{R}^m$ and $\alpha \geq 0$, for each row of $\mathbf{W}$. To find a minimizer of J, we utilize the minimization of $\widehat{J}$ in the iterative process in Algorithm 7. In Section 3.4, we will show that $\mathbf{W}^* = \operatorname{argmin}_{\mathbf{W}\in\mathbb{R}^{n\times m}} J(\mathbf{W})$, where $\mathbf{W}^*$ is the output of Algorithm 7.

Algorithm 7 Convex Optimization Using the Surrogate Functional

Input: $\mathbf{X} \in \mathbb{R}^{m\times n}, \lambda_C \geq 0, \mu > ||\mathbf{X}||_2^4$
Output: $\mathbf{W}^* \in \mathbb{R}^{n\times m}$ s.t. $\mathbf{W}^* = \operatorname{argmin}_{\mathbf{W}\in\mathbb{R}^{n\times m}} ||\mathbf{X}-\mathbf{X}\mathbf{W}\mathbf{X}||_F^2 + \lambda_C \sum_{i=1}^n \|\mathbf{W}(i,:)\|_\infty$

1: $\mathbf{W}^0 = \mathbf{0}$
2: $k = 0$
3: **repeat** ▷ each iteration solves $\mathbf{W^k} = \operatorname{argmin}_{\mathbf{Y}\in\mathbb{R}^{n\times m}} \widehat{J}(\mathbf{Y}, \mathbf{W^{k-1}})$.
4: $k = k+1$
5: $\mathbf{L} = \mu(\mathbf{W^{k-1}})^\mathbf{T} + \mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{X} - \mathbf{X}\mathbf{X}^\mathbf{T}(\mathbf{W^{k-1}})^\mathbf{T}\mathbf{X}^\mathbf{T}\mathbf{X}$
6: **for** $i = 1$ to n **do**
7: $\mathbf{W^k}(i,:) = \operatorname{argmin}_{\mathbf{y}\in\mathbb{R}^m} \left[\frac{1}{2}||\mathbf{y} - \frac{1}{\mu}\mathbf{L}^\mathbf{T}(i,:)||_2^2 + \frac{\lambda_C}{2\mu}||\mathbf{y}||_\infty \right]$
8: **end for**
9: **until** convergence of the sequence $\{\mathbf{W^k}\}$
10: $\mathbf{W}^* = \mathbf{W^k}$ ▷ $\mathbf{W}^* = \lim_{k\to\infty}\{\mathbf{W^k}\}$
11: **return** $\mathbf{W}^*$

3.3.2 Complexity

For this analysis we will assume that $r \ll m$ and $c \ll n$. We provide computational complexities in Table 3.1 that help determine the overall complexity of Algorithm 6. Quantities that are used in each iteration of one of the bisection loops should be computed once before the loop begins. For the first bisection loop on lines 3-13, this includes $\mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{X}$ (which is the transpose of the matrix computed to find λ_C^* in line 1), $\mathbf{X}\mathbf{X}^\mathbf{T}$, $\mathbf{X}^\mathbf{T}\mathbf{X}$, and $\mu_C > \|\mathbf{X}\|_2^4$ which is an input to Algorithm 7. For the second bisection loop on lines 17-27, this includes $\mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{C}$

(which is the transpose of the matrix computed to find λ_R^* in line 15), $\mathbf{X}\mathbf{X}^\mathbf{T}$ (which was already computed before the first bisection loop), $\mathbf{C}^\mathbf{T}\mathbf{C}$, and $\mu_R > \|\mathbf{X}\|_2^2\|\mathbf{C}\|_2^2$ (an input to Algorithm 8 found in Supplementary Section 3.7), in which $\|\mathbf{X}\|_2$ was already computed as well.

Computation	**Complexity**
λ_C^*	$O(mn^2)$ if $m \geq n$ or $O(m^2n)$ if $m < n$
$\mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{X}$	$O(mn)$ utilizing λ_C^* computation
$\mathbf{X}\mathbf{X}^\mathbf{T}$	$O(m^2n)$
$\mathbf{X}^\mathbf{T}\mathbf{X}$	$O(mn^2)$
μ_C	$O(mn)$
λ_R^*	$O(cmn)$ if $m \geq c$ or $O(cm^2)$ (utilizing $\mathbf{X}\mathbf{X}^\mathbf{T}$ computation) if $m < c$
$\mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{C}$	$O(cm)$ utilizing λ_R^* computation
$\mathbf{C}^\mathbf{T}\mathbf{C}$	$O(c^2m)$
μ_R	$O(cm)$

Table 3.1: Computational complexities of quantities used in Algorithm 6.

Before we analyze the entirety of Algorithm 6, we will analyze the complexity of Algorithm 7, which solves the minimization problem on line 5 of Algorithm 6. In each iteration, the most expensive steps are the computation of $\mathbf{L}$ and the n proximal operators. $\mathbf{L}$ can be computed in $O(mn(m+n))$ time and each proximal operator can be computed in $O(m \log m)$ time. Determining if the sequence has converged in line 9 can be completed in $O(mn)$ time. Hence the total time for Algorithm 7, assuming k iterations are completed, is $O(kmn(m+n))$. A similar analysis shows that the complexity of Algorithm 8 (Supplementary Section 3.7) for solving the minimization problem on line 19 of Algorithm 6 is $O(\hat{k}cm(m+c))$, assuming $\hat{k}$ iterations are completed. For each minimization problem, we have found that using a small maximum number of iterations in practice, e.g., 20, is sufficient for use in the CUR algorithm, Algorithm 6. For the remainder of this analysis, we will assume that the number of iterations for each minimization problem is a small constant.

In Algorithm 6, the bisection loop in lines 3-13 dominates the computational complexity.

This loop runs in $O(\ell mn(m+n))$ time, assuming ℓ iterations. This is due to the computation time of Algorithm 7 and that finding the set I_C can be completed in $O(mn)$ time. Using a similar analysis, the second bisection loop on lines 17-27 is an order of magnitude less expensive, i.e., $O(\hat{\ell}cm(m+c))$ assuming $\hat{\ell}$ iterations. The computation of $\mathbf{U}$ involves two pseudoinverses, $\mathbf{C}^+$ and $\mathbf{R}^+$, which can be computed in time $O(cm\min(c,m))$ and $O(nr\min(n,r))$, respectively. The product $\mathbf{U} = \mathbf{C}^+\mathbf{X}\mathbf{R}^+$ can be computed in time $O(cn(m+r))$ time if $m \geq n$, or $O(mr(n+c))$ time if $m < n$. Hence the total computational complexity for Algorithm 6 is $O(\ell mn(m+n))$.

3.3.3 Generalizations

To form the matrix $\mathbf{C}$, our algorithm and implementation can also accommodate objective functions of the form

$$||\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{X}||_F^2 + \lambda_C \sum_{i=1}^{n} ||\mathbf{W}(i,:)||_p^p,$$

for $1 \leq p \leq 2$. The theory for using the surrogate functional technique with these choices of objective functions is already complete in [14], and the only change to the implementation detailed above is that the proximal operator in Algorithm 7 would be of the ℓ_p-norm. Closed form solutions for the proximal operator of the ℓ_1 and ℓ_2 norms exist, making these easy choices to implement. Similar penalty function adaptations can be made to the objective function used to form the matrix $\mathbf{R}$. Hence, our algorithm and implementation provide a CUR framework.

We also note that the objective function could be generalized as

$$\min_{\mathbf{W}\in\mathbb{R}^{n\times m}} ||\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{X}||_p^p + \lambda_C \sum_{i=1}^{n} ||\mathbf{W}(i,:)||_\infty,$$

where $p \in [1, \infty]$. We have thought about this problem, specifically solving it using a mixture of ADMM and the surrogate functional technique. While outside the scope of this thesis, numerical experimentation and proving convergence of this method are potential areas for future work.

3.4 Theoretical Foundation

In this section, we follow the theoretical approach of [14] to prove the correctness of Algorithm 7 for solving the minimization on line 5 of Algorithm 6, i.e., that

$$\mathbf{W}^* = \underset{\mathbf{W} \in \mathbb{R}^{n \times m}}{\operatorname{argmin}} J(\mathbf{W}),$$

where $\mathbf{W}^*$ is the output of Algorithm 7. The correctness of Algorithm 8, for solving the minimization on line 19 of Algorithm 6, can be proved similarly.

Let the nonlinear operator $\mathbf{T} : \mathbb{R}^{n \times m} \to \mathbb{R}^{n \times m}$ be defined as $\mathbf{Z} \mapsto \mathbf{T}(\mathbf{Z}) = \mathbf{W}$ and given by:

1. construct $\mathbf{L}(\mathbf{Z}) = \mu \mathbf{Z}^\mathbf{T} + \mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{X} - \mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{Z}^\mathbf{T}\mathbf{X}^\mathbf{T}\mathbf{X}$,

2. For each row of $\mathbf{L}^\mathbf{T}$ (i.e., $\forall i \in [n]$), solve

$$\mathbf{W}(i,:) = \underset{\mathbf{y} \in \mathbb{R}^m}{\operatorname{argmin}} \left[\frac{1}{2} ||\mathbf{y} - \frac{1}{\mu} \mathbf{L}^\mathbf{T}(i,:)||_2^2 + \frac{\lambda_C}{2\mu} ||\mathbf{y}||_\infty \right],$$

   and

3. Reassemble $\mathbf{W}$ from its rows.

Using the operator $\mathbf{T}$, we can write each $\mathbf{W}^k$ produced in Algorithm 7 as $\mathbf{W}^k = \mathbf{T}^k(\mathbf{W}^0)$. The

primary result that will we prove in this section is that the sequence $\{\mathbf{W}^k\}_{k\in\mathbb{N}}$ converges to a minimizer of J. We will build to this result throughout the section.

3.4.1 Convergence to a Fixed Point of $\mathbf{T}$

We first show that the sequence $\{\mathbf{W}^k\}_{k\in\mathbb{N}}$ converges to a fixed point of $\mathbf{T}$.

Lemma 13. $(\mathbf{X^T X}) \otimes (\mathbf{X X^T})$ *is symmetric and positive semidefinite.*

Proof. $(\mathbf{X^T X}) \otimes (\mathbf{X X^T})$ is clearly symmetric. We will show that $\langle((\mathbf{X^T X}) \otimes (\mathbf{X X^T}))\mathbf{v}, \mathbf{v}\rangle \geq 0, \forall \mathbf{v} \in \mathbb{R}^{mn}$, thus proving the lemma. Fix $\mathbf{v} \in \mathbb{R}^{mn}$ and $\mathbf{V} \in \mathbb{R}^{m\times n}$ such that $\text{vec}(\mathbf{V}) = \mathbf{v}$.

$$\begin{aligned}
\langle((\mathbf{X^T X}) \otimes (\mathbf{X X^T}))\mathbf{v}, \mathbf{v}\rangle &= \langle \text{vec}((\mathbf{X X^T})\mathbf{V}(\mathbf{X^T X})), \text{vec}(\mathbf{V})\rangle \\
&= \langle (\mathbf{X X^T})\mathbf{V}(\mathbf{X^T X}), \mathbf{V}\rangle_{\text{F}} \\
&= \text{tr}\{(\mathbf{X X^T})\mathbf{V}(\mathbf{X^T X})\mathbf{V^T}\} \\
&= \text{tr}\{(\mathbf{X X^T})^{\frac{1}{2}}\mathbf{V}(\mathbf{X^T X})^{\frac{1}{2}}(\mathbf{X^T X})^{\frac{1}{2}}\mathbf{V^T}(\mathbf{X X^T})^{\frac{1}{2}}\} \\
&= ||(\mathbf{X X^T})^{\frac{1}{2}}\mathbf{V}(\mathbf{X^T X})^{\frac{1}{2}}||_F^2 \geq 0,
\end{aligned}$$

where $\langle .\,, .\rangle_{\text{F}}$ is the Frobenius inner product.

□

Lemma 14. $||(\mathbf{X^T X}) \otimes (\mathbf{X X^T})||_2 \leq (\sigma_{max}(\mathbf{X}))^4 = ||\mathbf{X}||_2^4$, *where* $\sigma_{max}(\mathbf{X})$ *is the largest singular value of the matrix* $\mathbf{X}$.

Proof. By Lemma 13, the matrix $(\mathbf{X^T X}) \otimes (\mathbf{X X^T})$ is symmetric and positive semidefinite.

Thus, the eigenvalues and singular values of $(\mathbf{X^T X}) \otimes (\mathbf{X X^T})$ are the same. Hence

$$\begin{aligned}
||(\mathbf{X^T X}) \otimes (\mathbf{X X^T})||_2 &= \sigma_{\max}((\mathbf{X^T X}) \otimes (\mathbf{X X^T})) \\
&= \max_{||\mathbf{v}||_2=1} \langle((\mathbf{X^T X}) \otimes (\mathbf{X X^T}))\mathbf{v}, \mathbf{v}\rangle \\
&= \max_{||\mathbf{V}||_F=1} ||(\mathbf{X X^T})^{\frac{1}{2}} \mathbf{V} (\mathbf{X^T X})^{\frac{1}{2}}||_F^2,
\end{aligned}$$

where the last equation follows from the proof of Lemma 13.

$$\begin{aligned}
||(\mathbf{X X^T})^{\frac{1}{2}} \mathbf{V} (\mathbf{X^T X})^{\frac{1}{2}}||_F &\leq ||(\mathbf{X X^T})^{\frac{1}{2}}||_2 ||\mathbf{V} (\mathbf{X^T X})^{\frac{1}{2}}||_F \\
&= ||(\mathbf{X X^T})^{\frac{1}{2}}||_2 ||(\mathbf{X^T X})^{\frac{1}{2}} \mathbf{V^T}||_F \\
&\leq ||(\mathbf{X X^T})^{\frac{1}{2}}||_2 ||(\mathbf{X^T X})^{\frac{1}{2}}||_2 ||\mathbf{V^T}||_F \\
&\leq \sigma_{\max}((\mathbf{X X^T})^{\frac{1}{2}}) \sigma_{\max}((\mathbf{X^T X})^{\frac{1}{2}}) ||\mathbf{V^T}||_F \\
&= (\sigma_{\max}(\mathbf{X}))^2 ||\mathbf{V^T}||_F.
\end{aligned}$$

Thus

$$\max_{||\mathbf{V}||_F=1} ||(\mathbf{X X^T})^{\frac{1}{2}} \mathbf{V} (\mathbf{X^T X})^{\frac{1}{2}}||_F^2 \leq (\sigma_{\max}(\mathbf{X}))^4,$$

proving the result. □

Lemma 15. *Let* $\mu \geq ||\mathbf{X}||_2^4$ *and define the operator* $\mathscr{L} : \mathbf{U} \mapsto \mu \mathbf{U} - \mathbf{X X^T U X^T X}$. *Then* $||\mathscr{L}|| \leq \mu$.

Proof. The operator norm of $\mathscr{L}$ is

$$\begin{aligned}
||\mathscr{L}|| &= \max_{||\mathbf{U}||_F=1} ||\mathscr{L}(\mathbf{U})||_F \\
&= \max_{||\mathbf{U}||_F=1} ||\text{vec}(\mu\mathbf{U} - \mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{U}\mathbf{X}^\mathbf{T}\mathbf{X})||_2 \\
&= \max_{||\mathbf{U}||_F=1} ||(\mu\mathbf{I} - (\mathbf{X}^\mathbf{T}\mathbf{X}) \otimes (\mathbf{X}\mathbf{X}^\mathbf{T}))\text{vec}(\mathbf{U})||_2 \\
&= \sigma_{\max}(\mu\mathbf{I} - (\mathbf{X}^\mathbf{T}\mathbf{X}) \otimes (\mathbf{X}\mathbf{X}^\mathbf{T})).
\end{aligned}$$

By Lemmas 13 and 14, $(\mathbf{X}^\mathbf{T}\mathbf{X}) \otimes (\mathbf{X}\mathbf{X}^\mathbf{T})$ is a symmetric positive semidefinite matrix with 2-norm bounded above by $||\mathbf{X}||_2^4$. Hence

$$||\mathscr{L}|| = \sigma_{\max}(\mu\mathbf{I} - (\mathbf{X}^\mathbf{T}\mathbf{X}) \otimes (\mathbf{X}\mathbf{X}^\mathbf{T})) \leq \mu.$$

□

Lemma 16. *Let $\mu \geq ||\mathbf{X}||_2^4$. Then the mapping $\mathbf{T}$ is nonexpansive, i.e., $\forall\, \mathbf{Z}, \mathbf{Z}' \in \mathbb{R}^{n\times m}$,*

$$||\mathbf{T}(\mathbf{Z}) - \mathbf{T}(\mathbf{Z}')||_F \leq ||\mathbf{Z} - \mathbf{Z}'||_F.$$

Proof. By the fact that $\mathbf{prox}_{\frac{\lambda}{2\mu}||\cdot||_\infty}$ is nonexpansive [16] and Lemma 15, we have

$$\begin{aligned}
||\mathbf{T}(\mathbf{Z}) - \mathbf{T}(\mathbf{Z}')||_F^2 &= \sum_{i=1}^{n} \left\| \mathbf{prox}_{\frac{\lambda}{2\mu}||\cdot||_\infty}\left(\frac{1}{\mu}[\mathbf{L}(\mathbf{Z})]^{\mathbf{T}}(i,:)\right) - \mathbf{prox}_{\frac{\lambda}{2\mu}||\cdot||_\infty}\left(\frac{1}{\mu}[\mathbf{L}(\mathbf{Z}')]^{\mathbf{T}}(i,:)\right) \right\|_2^2 \\
&\leq \sum_{i=1}^{n} \left\| \frac{1}{\mu}[\mathbf{L}(\mathbf{Z})]^{\mathbf{T}}(i,:) - \frac{1}{\mu}[\mathbf{L}(\mathbf{Z}')]^{\mathbf{T}}(i,:) \right\|_2^2 \\
&= \frac{1}{\mu^2}||\mathbf{L}(\mathbf{Z}) - \mathbf{L}(\mathbf{Z}')||_F^2 \\
&= \frac{1}{\mu^2}||\mathscr{L}(\mathbf{Z} - \mathbf{Z}')||_F^2 \\
&\leq \frac{1}{\mu^2}||\mathscr{L}||^2||\mathbf{Z} - \mathbf{Z}'||_F^2 \\
&\leq ||\mathbf{Z} - \mathbf{Z}'||_F^2.
\end{aligned}$$

□

We provide proofs of the next lemma and theorem in Appendices 3.8 and 3.9, respectively, as they largely mirror those of [14].

Lemma 17. *Let $\mu > ||\mathbf{X}||_2^4$. Then the mapping $\mathbf{T}$ is asymptotically regular, i.e. $\forall\, \mathbf{Z} \in \mathbb{R}^{n\times m}$,*

$$\lim_{k\to\infty} ||\mathbf{T}^{k+1}(\mathbf{Z}) - \mathbf{T}^k(\mathbf{Z})||_F = 0.$$

Theorem 18. *Let the mapping $\mathbf{T} : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$ be nonexpansive and asymptotically regular. Then the sequence $\{\mathbf{W}^k = \mathbf{T}^k(\mathbf{W}^0)\}_{k\in\mathbb{N}}$ converges to a fixed point of $\mathbf{T}$.*

3.4.2 Convergence to a Minimizer of J

We now show that the sequence $\{\mathbf{W}^k\}_{k\in\mathbb{N}}$ converges to a minimizer of J. We begin by establishing three lemmas.

Lemma 19. *Let $f : I \to \mathbb{R}$ be convex, with $I = (a, b)$. If $f(x) = f_1(x) + \alpha x^2$, where $\alpha > 0$ and $f_1(x)$ is piecewise linear, then f_1 is convex and therefore f is strictly convex.*

Proof. Let $x_0, x_1, x_2, ..., x_n, x_{n+1} \in I$ such that $a = x_0 < x_1 < x_2 < ... < x_n < x_{n+1} = b$, and $\forall k \in [n+1]$, f_1 restricted to (x_{k-1}, x_k) is linear and given by $f_1(x) = a_k x + b_k$. Due to the convexity of f, f_1 is continuous. Hence $a_k x_k + b_k = a_{k+1} x_k + b_{k+1}$, for every $k \in [n]$.

We show that $a_1 \leq a_2 \leq ... \leq a_{n+1}$, which implies that f_1 is convex. Consider the inequality $a_k \leq a_{k+1}$ for any $k \in [n]$. Let $h \in \mathbb{R}$ such that $0 \leq h < \min(x_k - x_{k-1}, x_{k+1} - x_k)$. The convexity of f implies that

$$\frac{f(x_k - h) + f(x_k + h)}{2} \geq f(x_k).$$

Thus,

$$\frac{a_k(x_k - h) + b_k + \alpha(x_k - h)^2 + a_{k+1}(x_k + h) + b_{k+1} + \alpha(x_k + h)^2}{2} \geq \frac{a_k x_k + b_k + \alpha x_k^2 + a_{k+1} x_k + b_{k+1} + \alpha x_k^2}{2},$$

where we have used the continuity of f_1 at x_k on the right hand side of the inequality. Hence

$$(a_{k+1} - a_k)h + 2\alpha h^2 \geq 0. \tag{3.9}$$

$(a_{k+1} - a_k)h + 2\alpha h^2$ is a convex quadratic in h with roots 0 and $\frac{-(a_{k+1} - a_k)}{2\alpha}$. This leads to three cases:

1. The root $\frac{-(a_{k+1} - a_k)}{2\alpha} = 0$, i.e. 0 is a root of multiplicity two. This implies $a_k = a_{k+1}$ and Equation 3.9 holds.

2. The root $\frac{-(a_{k+1}-a_k)}{2\alpha} < 0$. This implies $a_k < a_{k+1}$ and Equation 3.9 holds for $h \geq 0$.

3. The root $\frac{-(a_{k+1}-a_k)}{2\alpha} > 0$. This implies $a_k > a_{k+1}$. For h such that $0 < h < \min(x_k - x_{k-1}, x_{k+1} - x_k, \frac{-(a_{k+1}-a_k)}{2\alpha})$, Equation 3.9 does not hold.

Thus, Equation 3.9 holds $\forall h$ such that $0 \leq h < \min(x_k - x_{k-1}, x_{k+1} - x_k)$ if and only if $a_k \leq a_{k+1}$. Hence $a_1 \leq a_2 \leq ... \leq a_{n+1}$, which implies that f_1 is convex. □

Lemma 20. *Let $f : I \to \mathbb{R}$ with $I = (a, b)$ such that $0 \in I$ be defined as $f(\alpha) = F(\alpha) + \frac{1}{2}\alpha^2 \geq 0$. If F is continuous, piecewise linear, convex, and $F(0) = 0$, then $F(\alpha) \geq 0$.*

Proof. Let $\alpha_0, \alpha_1, \alpha_2, ..., \alpha_n, \alpha_{n+1} \in I$ such that $a = \alpha_0 < \alpha_1 < \alpha_2 < ... < \alpha_n < \alpha_{n+1} = b$, and $\forall k \in [n+1]$, $F(\alpha) = a_k\alpha + b_k$, if $\alpha_{k-1} \leq \alpha \leq \alpha_k$. We will use two cases to prove the result.

Case 1: Suppose $0 \in (\alpha_{k-1}, \alpha_k)$ for some $k \in [n+1]$. Since $F(0) = 0$, $b_k = 0$. In addition, $a_k = 0$, which we will show by contradiction. Suppose $a_k > 0$ and $\alpha_{k-1} < \epsilon < 0$ such that $|\epsilon| < 2a_k$. Then $f(\epsilon) = a_k\epsilon + \frac{1}{2}\epsilon^2 < 0$, which is a contradiction. The case in which $a_k < 0$ similarly leads to a contradiction. Hence $F(\alpha) = 0$ on $[\alpha_{k-1}, \alpha_k]$. Due to the convexity of F, $a_1 \leq a_2 \leq \cdots \leq a_n \leq a_{n+1}$. Therefore $F'(\alpha) \leq 0$ for $\alpha \leq \alpha_{k-1}$ and $F'(\alpha) \geq 0$ for $\alpha \geq \alpha_k$. Hence $F(\alpha) \geq 0$.

Case 2: Suppose $\alpha_k = 0$ for some $k \in [n]$. Since $F(0) = 0$, $F(\alpha) = a_k\alpha$ and $f(\alpha) = a_k\alpha + \frac{1}{2}\alpha^2$ on $\alpha_{k-1} \leq \alpha \leq \alpha_k = 0$. We will show that $a_k \leq 0$. Suppose $a_k > 0$. Then $f(\alpha) = \alpha(a_k + \frac{1}{2}\alpha) < 0$ on $-2a_k < \alpha < 0$, which is a contradiction. Similarly, $F(\alpha) = \alpha a_{k+1}$ and $f(\alpha) = \alpha a_{k+1} + \frac{1}{2}\alpha^2$ on $0 = \alpha_k \leq \alpha \leq \alpha_{k+1}$. We will show that $a_{k+1} \geq 0$. Suppose $a_{k+1} < 0$. Then $f(\alpha) = \alpha(a_{k+1} + \frac{1}{2}\alpha) < 0$ on $0 < \alpha < 2|a_{k+1}|$, which is a contradiction. Due to the convexity of F, $a_1 \leq a_2 \leq \cdots \leq a_n \leq a_{n+1}$. Therefore $F'(\alpha) \leq a_k \leq 0$ for $\alpha \leq \alpha_k$ and $F'(\alpha) \geq a_{k+1} \geq 0$ for $\alpha \geq \alpha_k$. Hence $F(\alpha) \geq 0$.

□

Lemma 21. *Let* $\mathbf{X} \in \mathbb{R}^{m\times n}$, $\mathbf{W}, \mathbf{Z} \in \mathbb{R}^{n\times m}$, $\mu > 0$, $\lambda_C \geq 0$, *and* $\forall i \in [n]$

$$\mathbf{W}(i,:) = \underset{\mathbf{y}\in\mathbb{R}^m}{\operatorname{argmin}} \left[\frac{1}{2}\|\mathbf{y} - \frac{1}{\mu}\mathbf{L}^{\mathbf{T}}(i,:)\|_2^2 + \frac{\lambda_C}{2\mu}\|\mathbf{y}\|_\infty\right],$$

where $\mathbf{L}(\mathbf{Z}) = \mu\mathbf{Z}^{\mathbf{T}} + \mathbf{X}\mathbf{X}^{\mathbf{T}}\mathbf{X} - \mathbf{X}\mathbf{X}^{\mathbf{T}}\mathbf{Z}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{X}$. *Then for every* $\mathbf{H} \in \mathbb{R}^{n\times m}$,

$$\widehat{J}(\mathbf{W}+\mathbf{H};\mathbf{Z}) \geq \widehat{J}(\mathbf{W};\mathbf{Z}) + \mu||\mathbf{H}||_F^2.$$

Proof. By definition,

$$\begin{aligned}
\widehat{J}(\mathbf{W}+\mathbf{H},\mathbf{Z}) &= ||\mathbf{X} - \mathbf{X}(\mathbf{W}+\mathbf{H})\mathbf{X}||_F^2 + \lambda_C \sum_{i=1}^{n} \|(\mathbf{W}+\mathbf{H})(i,:)\|_\infty \\
&\quad + \mu||\mathbf{W}+\mathbf{H}-\mathbf{Z}||_F^2 - ||\mathbf{X}(\mathbf{W}+\mathbf{H})\mathbf{X} - \mathbf{X}\mathbf{Z}\mathbf{X}||_F^2 \\
&= \widehat{J}(\mathbf{W},\mathbf{Z}) + \lambda_C \sum_{i=1}^{n} (||(\mathbf{W}+\mathbf{H})(i,:)||_\infty - ||\mathbf{W}(i,:)||_\infty) + 2\,\mathrm{tr}\{\mathbf{H}(\mu\mathbf{W}^{\mathbf{T}} - \mathbf{L})\} + \mu||\mathbf{H}||_F^2 \\
&= \widehat{J}(\mathbf{W},\mathbf{Z}) + \sum_{i=1}^{n} \left[\lambda_C(||(\mathbf{W}+\mathbf{H})(i,:)||_\infty - ||\mathbf{W}(i,:)||_\infty) + 2\mu \left\langle \mathbf{H}(i,:), (\mathbf{W} - \frac{1}{\mu}\mathbf{L}^{\mathbf{T}})(i,:) \right\rangle\right] \\
&\quad + \mu||\mathbf{H}||_F^2.
\end{aligned}$$

To prove the result we need to show that

$$\sum_{i=1}^{n} \left[\lambda_C(||(\mathbf{W}+\mathbf{H})(i,:)||_\infty - ||\mathbf{W}(i,:)||_\infty) + 2\mu \left\langle \mathbf{H}(i,:), (\mathbf{W} - \frac{1}{\mu}\mathbf{L}^{\mathbf{T}})(i,:) \right\rangle\right] \geq 0.$$

Hence, it suffices to show that $\forall i \in [n]$,

$$\frac{\lambda_C}{2\mu}(||(\mathbf{W}+\mathbf{H})(i,:)||_\infty - ||\mathbf{W}(i,:)||_\infty) + \left\langle \mathbf{H}(i,:), (\mathbf{W} - \frac{1}{\mu}\mathbf{L}^\mathbf{T})(i,:) \right\rangle \geq 0. \tag{3.10}$$

To simplify notation, let $\mathbf{w} = \mathbf{W}(i,:)$, $\mathbf{h} = \mathbf{H}(i,:)$ and $\boldsymbol{\ell} = \mathbf{L}^\mathbf{T}(i,:)$. Let

$$F(\mathbf{h}) = \frac{\lambda_C}{2\mu}(||\mathbf{w}+\mathbf{h}||_\infty - ||\mathbf{w}||_\infty) + \left\langle \mathbf{h}, \mathbf{w} - \frac{1}{\mu}\boldsymbol{\ell} \right\rangle,$$

where $\mathbf{h} = \alpha\mathbf{u}$ and $\mathbf{u} \in \mathbb{R}^m$ is a random unit vector. Then we can denote $F(\mathbf{h}) = F(\alpha, \mathbf{u})$, and fix a $\mathbf{u}$ so that F is only a function of α:

$$F(\alpha) = \frac{\lambda_C}{2\mu}(||\mathbf{w}+\alpha\mathbf{u}||_\infty - ||\mathbf{w}||_\infty) + \alpha\left\langle \mathbf{u}, \mathbf{w} - \frac{1}{\mu}\boldsymbol{\ell} \right\rangle.$$

Let $G(\mathbf{w}) = \frac{1}{2}||\mathbf{w} - \frac{1}{\mu}\boldsymbol{\ell}||_2^2 + \frac{\lambda_C}{2\mu}||\mathbf{w}||_\infty$. Then

$$G(\mathbf{w}+\alpha\mathbf{u}) = \frac{1}{2}||\mathbf{w}+\alpha\mathbf{u} - \frac{1}{\mu}\boldsymbol{\ell}||_2^2 + \frac{\lambda_C}{2\mu}||\mathbf{w}+\alpha\mathbf{u}||_\infty,$$

and

$$\begin{aligned}
G(\mathbf{w}+\alpha\mathbf{u}) - G(\mathbf{w}) &= \frac{1}{2}||\mathbf{w}+\alpha\mathbf{u} - \frac{1}{\mu}\boldsymbol{\ell}||_2^2 - \frac{1}{2}||\mathbf{w} - \frac{1}{\mu}\boldsymbol{\ell}||_2^2 + \frac{\lambda_C}{2\mu}||\mathbf{w}+\alpha\mathbf{u}||_\infty - \frac{\lambda_C}{2\mu}||\mathbf{w}||_\infty \\
&= \frac{1}{2}\left(||\mathbf{w} - \frac{1}{\mu}\boldsymbol{\ell}||_2^2 + 2\left\langle \alpha\mathbf{u}, \mathbf{w} - \frac{1}{\mu}\boldsymbol{\ell} \right\rangle + ||\alpha\mathbf{u}||_2^2 - ||\mathbf{w} - \frac{1}{\mu}\boldsymbol{\ell}||_2^2\right) \\
&\quad + \frac{\lambda_C}{2\mu}(||\mathbf{w}+\alpha\mathbf{u}||_\infty - ||\mathbf{w}||_\infty) \\
&= \frac{\lambda_C}{2\mu}(||\mathbf{w}+\alpha\mathbf{u}||_\infty - ||\mathbf{w}||_\infty) + \alpha\left\langle \mathbf{u}, \mathbf{w} - \frac{1}{\mu}\boldsymbol{\ell} \right\rangle + \frac{1}{2}\alpha^2.
\end{aligned}$$

Thus, $F(\alpha) = G(\mathbf{w} + \alpha\mathbf{u}) - G(\mathbf{w}) - \frac{1}{2}\alpha^2$. Let $f(\alpha) = F(\alpha) + \frac{1}{2}\alpha^2$. We note that f is convex since $f(\alpha) = G(\mathbf{w} + \alpha\mathbf{u}) - G(\mathbf{w})$. In order to use Lemma 19, we also need to show that $F(\alpha)$ is piecewise linear in α. There is a constant term of $F(\alpha)$, $\frac{-\lambda_C}{2\mu}||\mathbf{w}||_\infty$, and a linear term, $\alpha\langle \mathbf{u}, \mathbf{w} - \frac{1}{\mu}\boldsymbol{\ell}\rangle$. The remaining term, $\frac{\lambda_C}{2\mu}||\mathbf{w} + \alpha\mathbf{u}||_\infty$ is piecewise linear in α, since as α increases

$$||\mathbf{w} + \alpha\mathbf{u}||_\infty = \max(\mathbf{w}_1 + \alpha\mathbf{u}_1, \dots, \mathbf{w}_m + \alpha\mathbf{u}_m, -\mathbf{w}_1 - \alpha\mathbf{u}_1, \dots, -\mathbf{w}_m - \alpha\mathbf{u}_m),$$

and the maximum of a set of linear functions is piecewise linear. Thus, $F(\alpha)$ is piecewise linear, and by Lemma 19, $F(\alpha)$ is convex.

The remaining step is to show that $F(\alpha) \geq 0$, which will establish the claim in Equation 3.10 and thus the result. Since $\mathbf{w} = \operatorname{argmin}_{\mathbf{y}} G(\mathbf{y})$, we have $f(\alpha) = G(\mathbf{w} + \alpha\mathbf{u}) - G(\mathbf{w}) \geq 0$. We also know that $F(\alpha)$ is continuous and piecewise linear in α, convex, and $F(0) = 0$. Hence by Lemma 20, $F(\alpha) \geq 0$.

□

The proof of Theorem 22 mirrors that of [14], so is included in Supplementary Section 3.10.

Theorem 22. *A fixed point of* $\mathbf{T}$ *is a minimizer of* $J(\mathbf{W})$.

Hence by Theorems 18 and 22, the sequence $\{\mathbf{W}^k\}_{k\in\mathbb{N}}$ converges to a minimizer of J.

Theorem 23. $J(\mathbf{W})$ *has a unique minimizer if* $\mathbf{X}$ *is square and full rank.*

Proof. The minimizer of $J(\mathbf{W})$ is unique if $\|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{X}\|_F^2$ is strictly convex in $\mathbf{W}$. Since

$$\|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{X}\|_F^2 = ||(\mathbf{X}^\mathbf{T} \otimes \mathbf{X})\mathbf{w} - \mathbf{b}||_2^2,$$

where $\mathbf{b} = \text{vec}(\mathbf{X}) \in \mathbb{R}^{mn}$ and $\mathbf{w} = \text{vec}(\mathbf{W}) \in \mathbb{R}^{mn}$, we need to guarantee that $(\mathbf{X^T} \otimes \mathbf{X})$ has full rank. Since $\text{rank}(\mathbf{X^T} \otimes \mathbf{X}) = \text{rank}(\mathbf{X}^T)\text{rank}(\mathbf{X}) = (\text{rank}(\mathbf{X}))^2$, $(\mathbf{X^T} \otimes \mathbf{X})$ has full rank if $\mathbf{X}$ is square and full rank, proving the result. □

3.5 Numerical Experiments

We demonstrate our CUR algorithm on two datasets: 1) a document-term matrix, and 2) a gene expression dataset. CUR has been applied to both of these types of datasets before, e.g. [1, 2, 6] for document-term matrices and [1, 8] for gene expression data. We compare our CUR algorithm to the leverage score CUR [1] (the randomized version and deterministic variant), the DEIM CUR [2], the QR CUR variant described in [2], and the SVD. In these experiments we denote our CUR algorithm as SF CUR (for surrogate functional CUR), and the leverage score CUR as LS CUR. For a brief summary of these methods, see Table 3.2, and for more details see Section 3.2. These experiments were performed in MATLAB on the University of Virginia's High-Performance Computing system, Rivanna. We used one (CPU) node, using 8 cores of an Intel(R) Xeon(R) Gold 6248 CPU at 2.50GHz and 72GB of RAM.

Method	**Complexity**	**Computation of C and R**
Randomized LS CUR	$O(k'mn) + O(rmn)$	columns (rows) sampled from a probability distribution based on leverage scores computed from the leading k' right (left) singular vectors. k' is a rank parameter.
Deterministic LS CUR	$O(k'mn) + O(rmn)$	columns (rows) with largest leverage scores computed from the leading k' right (left) singular vectors. k' is a rank parameter.
DEIM CUR	$O(kmn)$	columns (rows) chosen using DEIM point selection algorithm on the leading k right (left) singular vectors.
QR CUR	$O(mn^2)$	columns (rows) chosen using pivoted QR of $\mathbf{X}$ ($\mathbf{C}^T$).
rank-k SVD	$O(kmn)$	-

Table 3.2: Summary of methods that we compare to SF CUR in this section. The complexity given is for $\mathbf{X} \in \mathbb{R}^{m \times n}$, letting c be the number of columns in $\mathbf{C}$, r the number of rows of $\mathbf{R}$, and $c = r = k$. We assume that $m < n$ and $c, r \leq m$. For each CUR method, $\mathbf{U} = \mathbf{C}^{+}\mathbf{X}\mathbf{R}^{+}$.

3.5.1 Document-Term Matrix

We compare the accuracy of the SF CUR algorithm with that of other CUR algorithms and the SVD on a document-term matrix, $\mathbf{T} \in \mathbb{R}^{2,389 \times 21,238}$. $\mathbf{T}$ is sparse with 0.23% non-zero entries and was downloaded and created from the 20 newsgroups dataset using the scikit-learn package [30]. The documents include the training set documents for the four recreation categories and headers, footers, quotes, and a list of English stop words were removed from the text. The documents were vectorized using TFIDF and the resulting matrix rows were normalized using the ℓ_2 norm[2]. $\mathbf{T}$ is a rank-deficient matrix; $\text{rank}(\mathbf{T}) = 2,295$.

Let c be the number of columns chosen for $\mathbf{C}$, and r, the number of rows chosen for $\mathbf{R}$. Figure 3.1 presents the reconstruction accuracy of $\mathbf{T}$ and Figure 3.2 presents the computation time for each CUR approximation in which $c = r$ and c, r vary over $\{200, 400, ..., 2,200, 2,295\}$, and for the rank-$k$ SVD in which $k = c = r$. Since the randomized LS CUR is randomized, we ran five experiments for each value of c, r and reported the average accuracy and time with the standard deviations given by error bars. We note that due to sampling, the randomized LS CUR may have chosen a number of columns and/or rows slightly more or less than c and/or r. For both the deterministic and randomized LS CUR, the rank parameter for leverage score computation was 10.

[2]The data was downloaded using the function sklearn.datasets.fetch_20newsgroups and vectorized using sklearn.feature_extraction.text.TfidfVectorizer. The processing was completed as described using options in these functions. The stop word list used was the built-in list provided in scikit-learn, and the TFIDF calculations were based on the default parameter settings.

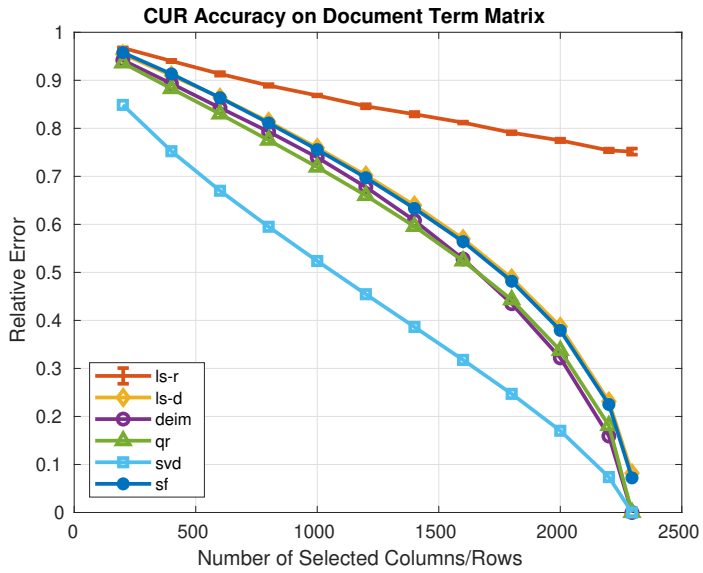


Figure 3.1: Relative error of CUR approximations and the rank-k SVD on a document-term matrix. The rank of the SVD approximation is the same as the number of selected columns/rows for the CUR approximations.

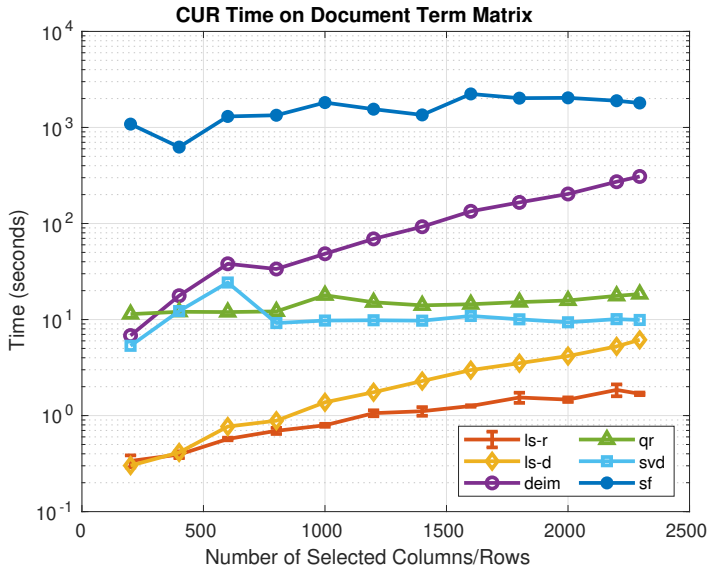


Figure 3.2: Time of CUR approximations and the rank-k SVD on a document-term matrix. The rank of the SVD approximation is the same as the number of selected columns/rows for the CUR approximations.

In general, the SF CUR error is similar to that of the deterministic LS CUR, and the DEIM CUR error is similar to that of the QR CUR. Additionally, the DEIM CUR and QR CUR achieve lower error than that of the SF CUR and deterministic LS CUR. However, for $c, r \geq 400$, the

DEIM CUR is more expensive than the QR CUR with computation times larger than 100 seconds for $c = r \geq 1,600$ as compared to computation times of less than 20 seconds for all values of c, r for the QR CUR. Furthermore, the SF CUR is the most expensive method with computation times generally larger than 1,000 seconds, and the deterministic LS CUR is relatively fast with computation times all less than those of the SF CUR, DEIM CUR, QR CUR, and SVD. The randomized LS CUR does not perform well in accuracy as c, r increase; however, it takes the least amount of time to compute for $c = r \geq 400$. Lastly, the SVD clearly achieves the lowest error and has computation times generally around 10 seconds.

3.5.2 Gene Expression Data

We compare the accuracy and classification performance of the SF CUR algorithm with that of other CUR algorithms and the SVD on the National Institutes of Health (NIH) gene expression dataset, GSE10072. Soreson and Embree [2] compared the results of the DEIM CUR and the deterministic LS CUR on this dataset. We repeat their experiments and 1) add results using the SF CUR, QR CUR, randomized LS CUR, and SVD, and 2) add a metric to compare the classification performance of these methods. The GSE10072 dataset, $\mathbf{G} \in \mathbb{R}^{22,283 \times 107}$, contains gene expression data for 107 patients, of which 58 have a lung tumor and 49 do not. All entries of $\mathbf{G}$ are positive and larger entries represent a greater reaction to a probe. Each row of $\mathbf{G}$ is centered using its mean. The experiments conducted in [2] are:

1. calculating the reconstruction accuracy of each CUR approximation, and
2. assessing if the probes selected by CUR separate the patients into those with and without a tumor.

We approximate $\mathbf{G}^T$ (so that probes, i.e., columns, are selected first in the SF algorithm) and use $\|\mathbf{G}^T - \mathbf{CUR}\|_F / \|\mathbf{G}^T\|_F$ as the relative error of the CUR instead of $\|\mathbf{G} - \mathbf{CUR}\|_2$ as in [2]. To assess how well a probe separates the patients into two classes, the number of patients in each class with a (mean-centered) entry in $\mathbf{G}^T$ greater than one for that probe are counted. As mentioned in [2], there are 23 probes for which at least 30 patients with a tumor have an entry greater than one, and 95 probes for which at least 30 patients without a tumor have an entry greater than one. No probe is included in both of these sets.

Figure 3.3 presents the reconstruction accuracy of $\mathbf{G}^T$ and Figure 3.4 presents the computation time for each CUR and SVD approximation. Values of c, r vary over $\{5, 10, ..., 105, 107\}$ and for each CUR approximation $c = r$. For the rank-k SVD, $k = c = r$. We reported the average accuracy and time over five runs for the randomized LS CUR along with the corresponding standard deviations. For both the deterministic and randomized LS CUR, the rank parameter for leverage score computation was 2. The SF CUR and deterministic LS CUR have similar errors for all values of c, r; however, the SF CUR takes longer to compute (generally about 5 seconds) than the deterministic LS CUR (generally about 0.1 seconds). The QR CUR achieves lower error than the SF CUR for $c, r \geq 20$, and the DEIM CUR achieves the lowest error of the CUR approximations for every c, r value. The randomized LS CUR has an average error lower than that of the SF CUR for $c, r \leq 65$. The SF CUR takes the longest to compute, while the other methods have relatively similar computation times under 0.5 seconds.

Next, we determine classification performance of each CUR and SVD approximation. For each CUR approximation, we set $c = r = 15$ and report the selected probes (i.e., columns of $\mathbf{G}^T$). For the rank-k SVD, we perform principal component analysis (PCA) using a rank-2 SVD since the two classes (tumor and no tumor) are separated well when the data is projected onto the

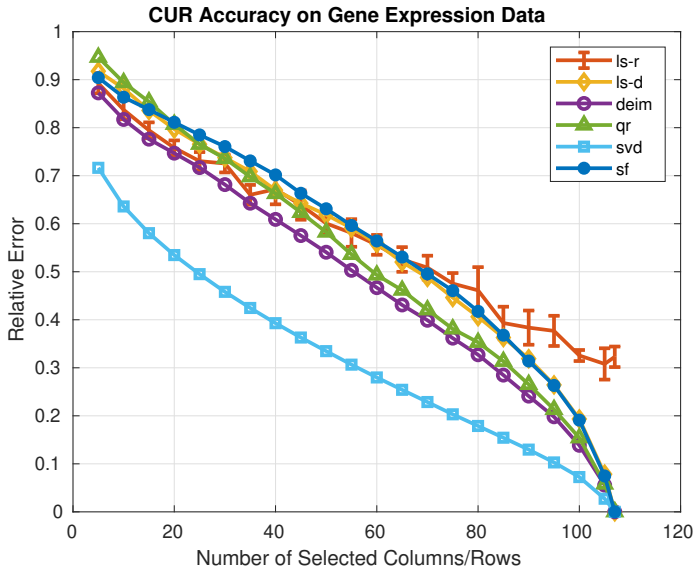


Figure 3.3: Relative error of CUR approximations and the rank-k SVD on gene expression data. The rank of the SVD approximation is the same as the number of selected columns/rows for the CUR approximations.

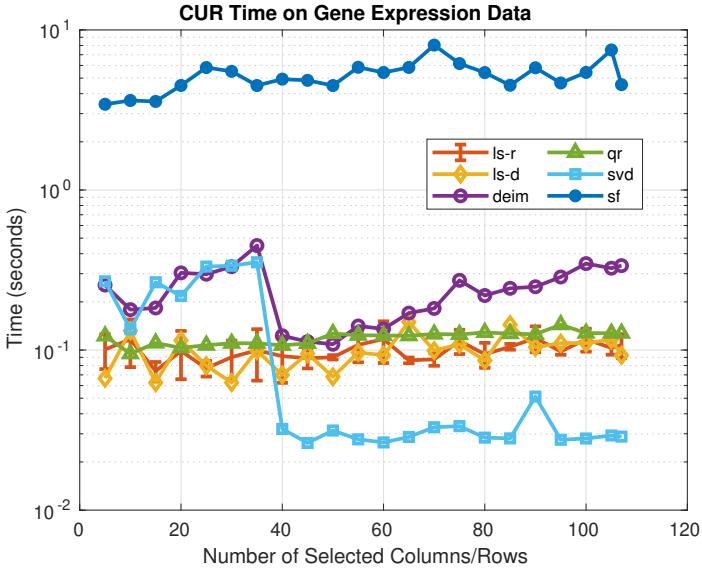


Figure 3.4: Time of CUR approximations and the rank-k SVD on gene expression data. The rank of the SVD approximation is the same as the number of selected columns/rows for the CUR approximations.

leading two principal axes [2][3], and then select the 15 probes that have the largest correlation (in absolute value) with either the first or second principal component. Table 3.3 presents results for the SF CUR and deterministic LS CUR, where probes are listed in alphabetical order. These two methods select the same 15 probes. Since the deterministic LS CUR returns selected columns in a ranked order by leverage score, the fourth column of Table 3.3 provides the rank for each of the selected columns. Table 3.4 presents results for the DEIM CUR, QR CUR, and SVD. Since each of these methods return selected columns in a ranked order, tables a-c) in Table 3.4 list the selected probes in ranked order (1 to 15). In order to compare the performance of these methods, for each selected probe we compute the absolute value of the difference between the number of entries greater than one in $\mathbf{G}^T$ for patients with and without a tumor. For each method we report the median of these 15 differences in Table 3.5.

	# of Entries > 1 in $\mathbf{G}^T$		**LS CUR Leverage**
Probe	**No Tumor**	**Tumor**	**Score Rank**
203980_at	44	2	10
204712_at	43	5	9
205200_at	39	0	15
205866_at	39	0	14
205982_x_at	48	5	5
209612_s_at	46	2	12
209613_s_at	47	2	6
209875_s_at	2	50	4
210081_at	45	2	1
210096_at	44	6	8
211735_x_at	48	5	3
214135_at	47	3	13
214387_x_at	48	6	2
215454_x_at	46	0	7
219230_at	38	2	11

Table 3.3: SF CUR and deterministic LS CUR classification results.

[3]Sorenson and Embree cite A. Kundu, S. Nambirajan, and P. Drineas [*Identifying influential entries in a matrix*, preprint arXiv:1310.3556 [cs.nA], 2013] for this result; however, this paper was withdrawn from the arXiv.

Probe	# of Entries > 1 in $\mathbf{G}^T$	
	No Tumor	**Tumor**
210081_at	45	2
214895_s_at	3	8
209156_s_at	6	5
211653_x_at	1	18
214777_at	3	27
219612_s_at	17	17
204304_s_at	4	16
203824_at	4	17
204748_at	14	18
201909_at	34	21
214774_x_at	0	34
211074_at	4	7
210096_at	44	6
204475_at	0	27
214612_x_at	0	16

(a) DEIM CUR.

Probe	# of Entries > 1 in $\mathbf{G}^T$	
	No Tumor	**Tumor**
214387_x_at	48	6
201909_at	34	21
206239_s_at	2	30
205725_at	33	7
219612_s_at	17	17
203290_at	29	19
213831_at	28	18
213674_x_at	8	17
204475_at	0	27
201884_at	0	30
209278_s_at	0	21
204885_s_at	12	18
207430_s_at	6	9
205476_at	11	16
209309_at	7	17

(b) QR CUR.

Probe	# of Entries > 1 in $\mathbf{G}^T$	
	No Tumor	**Tumor**
204931_at	36	0
206702_at	34	0
201540_at	43	0
209074_s_at	45	1
206742_at	38	0
205200_at	39	0
219213_at	19	0
204894_s_at	29	0
213103_at	5	0
206209_s_at	36	0
219719_at	13	0
204719_at	42	0
208981_at	22	0
210081_at	45	2
204396_s_at	36	0

(c) SVD.

Table 3.4: DEIM CUR, QR CUR and SVD classification results.

Method	Median Difference Between Classes
SF CUR	43
Deterministic LS CUR	43
DEIM CUR	13
QR CUR	10
SVD	36

Table 3.5: Comparison of classification methods. A larger median difference implies better performance, i.e., better class separation.

The 15 probes selected by SF CUR and deterministic LS CUR perform the best in separating patients with a tumor from those without a tumor. The probes selected by SVD also do fairly well at this task. However, less than half of the probes selected by DEIM CUR and QR CUR separate the two classes well, resulting in much lower values for the median difference between classes in Table 3.5. This is interesting since the DEIM CUR for $c = r = 15$ achieves the best reconstruction accuracy for the matrix $\mathbf{T}$ among the CUR algorithms.

3.6 Conclusion

We have presented a novel deterministic CUR formulation using convex optimization to solve for $\mathbf{C}$ and $\mathbf{R}$ separately, and a corresponding algorithm that utilizes bisection in order to allow the user to select the number of columns chosen for $\mathbf{C}$ and the number of rows chosen for $\mathbf{R}$. We implemented this algorithm using the surrogate functional [14], and extended the surrogate functional theory to include the penalty functions used in our CUR formulation in order to prove the correctness of our algorithm. We showed through numerical examples that our CUR algorithm performs similarly to the deterministic LS CUR in matrix reconstruction and classification feature selection, in which it outperformed the DEIM CUR, QR CUR and SVD. Hence, we have shown that the SF CUR is a viable option for low-rank matrix approximation

and applications.

3.7 Supplementary Material: Minimization to find $\mathbf{R}$

The minimization problem on line 19 of Algorithm 6,

$$\mathbf{W}^* = \underset{\mathbf{W}\in\mathbb{R}^{c\times m}}{\operatorname{argmin}} ||\mathbf{X} - \mathbf{CWX}||_F^2 + \lambda_R \sum_{j=1}^{m} \|\mathbf{W}(:,j)\|_\infty,$$

can be solved using the surrogate functional [14] similarly to the minimization problem on line 5 that was analyzed in Section 3.3.1. Let $J(\mathbf{W})$ be the objective function in Equation 3.4,

$$J(\mathbf{W}) = ||\mathbf{X} - \mathbf{CWX}||_F^2 + \lambda_R \sum_{j=1}^{m} \|\mathbf{W}(:,j)\|_\infty,$$

and $\widehat{J}$ be the surrogate functional. For any $\mathbf{Z} \in \mathbb{R}^{c\times m}$ and $\mu > 0$ we have

$$\begin{aligned}
\widehat{J}(\mathbf{W},\mathbf{Z}) &= ||\mathbf{X} - \mathbf{CWX}||_F^2 + \lambda_R \sum_{j=1}^{m} \|\mathbf{W}(:,j)\|_\infty + \mu||\mathbf{W} - \mathbf{Z}||_F^2 - ||\mathbf{CWX} - \mathbf{CZX}||_F^2 \\
&= \mu||\mathbf{W}||_F^2 - 2\,\mathrm{tr}\{\mathbf{W}(\mu\mathbf{Z}^\mathbf{T} + \mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{C} - \mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{Z}^\mathbf{T}\mathbf{C}^\mathbf{T}\mathbf{C})\} + \lambda_R \sum_{j=1}^{m} \|\mathbf{W}(:,j)\|_\infty \\
&\quad + ||\mathbf{X}||_F^2 + \mu||\mathbf{Z}||_F^2 - ||\mathbf{CZX}||_F^2 \\
&= \sum_{j=1}^{m} \left[\mu||\mathbf{W}(:,j)||_2^2 - 2\langle \mathbf{W}(:,j), \mathbf{L}^\mathbf{T}(:,j)\rangle + \lambda_R||\mathbf{W}(:,j)||_\infty\right] \\
&\quad + ||\mathbf{X}||_F^2 + \mu||\mathbf{Z}||_F^2 - ||\mathbf{CZX}||_F^2,
\end{aligned} \tag{3.11}$$

where $\mathbf{L} = \mu\mathbf{Z}^{\mathbf{T}} + \mathbf{X}\mathbf{X}^{\mathbf{T}}\mathbf{C} - \mathbf{X}\mathbf{X}^{\mathbf{T}}\mathbf{Z}^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}\mathbf{C}$. Hence,

$$\begin{aligned}\underset{\mathbf{W}\in\mathbb{R}^{c\times m}}{\operatorname{argmin}} = \widehat{J}(\mathbf{W},\mathbf{Z}) &= \mu\sum_{j=1}^{m}\left[||\mathbf{W}(:,j) - \frac{1}{\mu}\mathbf{L}^{\mathbf{T}}(:,j)||^2 - ||\frac{1}{\mu}\mathbf{L}^{\mathbf{T}}(:,j)||^2 + \frac{\lambda_R}{\mu}||\mathbf{W}(:,j)||_\infty\right] \\ &= 2\mu\sum_{j=1}^{m}\left[\frac{1}{2}||\mathbf{W}(:,j) - \frac{1}{\mu}\mathbf{L}^{\mathbf{T}}(:,j)||^2 + \frac{\lambda_R}{2\mu}||\mathbf{W}(:,j)||_\infty\right].\end{aligned} \tag{3.12}$$

Thus, we can minimize $\widehat{J}$ over $\mathbf{W}$ by computing the proximal operator of the ℓ_∞ norm for each column of $\mathbf{W}$. To find a minimizer of J, we utilize the minimization of $\widehat{J}$ in the iterative process in Algorithm 8.

Algorithm 8 Convex Optimization Using the Surrogate Functional

Input: $\mathbf{X}\in\mathbb{R}^{m\times n}, \mathbf{C}\in\mathbb{R}^{m\times c}, \lambda > 0, \mu > \|\mathbf{X}\|_2^2\|\mathbf{C}\|_2^2$
Output: $\mathbf{W}^*\in\mathbb{R}^{c\times m}$ s.t. $\mathbf{W}^* = \min_{\mathbf{W}\in\mathbb{R}^{c\times m}} ||\mathbf{X} - \mathbf{C}\mathbf{W}\mathbf{X}||_F^2 + \lambda\sum_{i=1}^{m}\|\mathbf{W}(:,j)\|_\infty$

$\mathbf{W}_0 = \mathbf{0}$
$k = 0$
repeat ▷ each iteration solves $\mathbf{W}^k = \operatorname{argmin}_{\mathbf{Y}\in\mathbb{R}^{c\times m}} \widehat{J}(\mathbf{Y}, \mathbf{W}^{k-1})$.
  $k = k + 1$
  $\mathbf{L} = \mu(\mathbf{W}^{\mathbf{k-1}})^{\mathbf{T}} + \mathbf{X}\mathbf{X}^{\mathbf{T}}\mathbf{C} - \mathbf{X}\mathbf{X}^{\mathbf{T}}(\mathbf{W}^{\mathbf{k-1}})^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}\mathbf{C}$
  for $j = 1$ to m **do**
    $\mathbf{W}^{\mathbf{k}}(:,j) = \operatorname{argmin}_{\mathbf{y}\in\mathbb{R}^c}\left[\frac{1}{2}||\mathbf{y} - \frac{1}{\mu}\mathbf{L}^{\mathbf{T}}(:,j)||_2^2 + \frac{\lambda}{2\mu}||\mathbf{y}||_\infty\right]$
  end for
until convergence of the sequence $\{\mathbf{W}^{\mathbf{k}}\}$
$\mathbf{W}^* = \mathbf{W}^{\mathbf{k}}$ ▷ $\mathbf{W}^* = \lim_{k\to\infty}\{\mathbf{W}^{\mathbf{k}}\}$
return $\{\mathbf{W}^*\}$

3.8 Supplementary Material: Asymptotic Regularity Proof

In this section we prove Theorem 24, which was stated in the main text as Theorem 17. This is done through three lemmas and is provided as supplementary material since the proofs mainly mirror those given in [14].

Theorem 24. *Let $\mu > ||\mathbf{X}||_2^4$. Then the mapping $\mathbf{T}$ is asymptotically regular, i.e. $\forall\, \mathbf{Z} \in \mathbb{R}^{n\times m}$,*

$$\lim_{k\to\infty} ||\mathbf{T}^{k+1}(\mathbf{Z}) - \mathbf{T}^k(\mathbf{Z})||_F = 0.$$

We first provide the three lemmas.

Lemma 25. *Let $\mu \geq ||\mathbf{X}||_2^4$. Then $[J(\mathbf{W}^k)]_{n\in\mathbb{N}}$ and $[\widehat{J}(\mathbf{W}^{k+1}, \mathbf{W}^k)]_{k\in\mathbb{N}}$ are nonincreasing sequences.*

Proof. Since $\mathbf{W}^{k+1} = \operatorname{argmin}_{\mathbf{Y}} \widehat{J}(\mathbf{Y}, \mathbf{W}^k)$, we have

$$\begin{aligned} J(\mathbf{W}^{k+1}) + \mu||\mathbf{W}^{k+1} - \mathbf{W}^k||_F^2 - ||\mathbf{X}\mathbf{W}^{k+1}\mathbf{X} - \mathbf{X}\mathbf{W}^k\mathbf{X}||_F^2 &= \widehat{J}(\mathbf{W}^{k+1}, \mathbf{W}^k) \\ &\leq \widehat{J}(\mathbf{W}^k, \mathbf{W}^k) \\ &= J(\mathbf{W}^k). \end{aligned} \tag{3.13}$$

If we show that $\mu||\mathbf{W}^{k+1} - \mathbf{W}^k||_F^2 - ||\mathbf{X}\mathbf{W}^{k+1}\mathbf{X} - \mathbf{X}\mathbf{W}^k\mathbf{X}||_F^2 \geq 0$, we will have proved that $J(\mathbf{W}^{k+1}) \leq J(\mathbf{W}^k)$. Let $\mathbf{H} = \mathbf{W}^{k+1} - \mathbf{W}^k$, $\mathbf{h} = \text{vec}(\mathbf{H})$, and

$$\mathbf{M}^2 = \mu\mathbf{I} - (\mathbf{X}^\mathbf{T} \otimes \mathbf{X})^\mathbf{T}(\mathbf{X}^\mathbf{T} \otimes \mathbf{X}) = \mu\mathbf{I} - (\mathbf{X}\mathbf{X}^\mathbf{T}) \otimes (\mathbf{X}^\mathbf{T}\mathbf{X}).$$

Since $\mu \geq ||\mathbf{X}||_2^4$, $\mathbf{M}^2$ is symmetric, positive semi-definite and we can define

$$\mathbf{M} = (\mathbf{M}^2)^{1/2} = [\mu\mathbf{I} - (\mathbf{X}\mathbf{X}^\mathbf{T}) \otimes (\mathbf{X}^\mathbf{T}\mathbf{X})]^{1/2}.$$

Then,

$$\begin{aligned}
\mu||\mathbf{W}^{k+1}-\mathbf{W}^k||_F^2-||\mathbf{X}\mathbf{W}^{k+1}\mathbf{X}-\mathbf{X}\mathbf{W}^k\mathbf{X}||_F^2 &= \mu||\mathbf{H}||_F^2-||\mathbf{X}\mathbf{H}\mathbf{X}||_F^2 \\
&= \mu||\mathbf{h}||_2^2-||\text{vec}(\mathbf{X}\mathbf{H}\mathbf{X})||_2^2 \\
&= \mu||\mathbf{h}||_2^2-||(\mathbf{X}^\mathbf{T}\otimes\mathbf{X})\mathbf{h}||_2^2 \\
&= \text{tr}\{\mu\mathbf{h}^\mathbf{T}\mathbf{h}-\mathbf{h}^\mathbf{T}(\mathbf{X}\otimes\mathbf{X}^\mathbf{T})(\mathbf{X}^\mathbf{T}\otimes\mathbf{X})\mathbf{h}\} \\
&= \text{tr}\{\mathbf{h}^\mathbf{T}(\mu\mathbf{I}-(\mathbf{X}\mathbf{X}^\mathbf{T})\otimes(\mathbf{X}^\mathbf{T}\mathbf{X}))\mathbf{h}\} \\
&= \text{tr}\{\mathbf{h}^\mathbf{T}\mathbf{M}^\mathbf{T}\mathbf{M}\mathbf{h}\} \\
&= ||\mathbf{M}\mathbf{h}||_2^2 \\
&\geq 0.
\end{aligned}$$

Thus, $J(\mathbf{W}^{k+1})\leq J(\mathbf{W}^k)$.

To show the second part of the claim, we use Equation 3.13.

$$\begin{aligned}
\widehat{J}(\mathbf{W}^{k+2},\mathbf{W}^{k+1}) &\leq J(\mathbf{W}^{k+1}) \\
&\leq J(\mathbf{W}^{k+1})+||\mathbf{M}\mathbf{h}||_2^2 \\
&= \widehat{J}(\mathbf{W}^{k+1},\mathbf{W}^k).
\end{aligned}$$

□

Lemma 26. *Let $\mu\geq||\mathbf{X}||_2^4$. Then the $||\mathbf{W}^k||_F$ are bounded uniformly.*

Proof. Define

$$|||\mathbf{W}^k|||_{\lambda,\infty}=\lambda\sum_{i=1}^{n}||\mathbf{W}(i,:)||_\infty.$$

Then

$$|||\mathbf{W}^k|||_{\lambda,\infty} \leq J(\mathbf{W}^k) \leq J(\mathbf{W}^0),$$

where the second inequality follows from Lemma 25. Since

$$\begin{aligned}
||\mathbf{W}||_F^2 &= \sum_{i=1}^{n}\sum_{j=1}^{m}|\mathbf{W}_{ij}|^2 = \frac{1}{\lambda^2}\sum_{i=1}^{n}\lambda^2\sum_{j=1}^{m}|\mathbf{W}_{ij}|^2 \\
&\leq \frac{m}{\lambda^2}\sum_{i=1}^{n}\lambda^2 \max_{1\leq j\leq m}|\mathbf{W}_{ij}|^2 \\
&\leq \frac{m}{\lambda^2}\left(\lambda\sum_{i=1}^{n}||\mathbf{W}(i,:)||_\infty\right)^2 \\
&= \frac{m}{\lambda^2}|||\mathbf{W}|||_{\lambda,\infty}^2,
\end{aligned}$$

we have

$$||\mathbf{W}^k||_F \leq \frac{\sqrt{m}}{\lambda}|||\mathbf{W}^k|||_{\lambda,\infty} \leq \frac{\sqrt{m}}{\lambda}J(\mathbf{W}^0).$$

Thus the $||\mathbf{W}^k||_F$ are bounded uniformly. □

Lemma 27. *Let $\mu > ||\mathbf{X}||_2^4$. Then $\sum_{k=0}^{\infty}||\mathbf{W}^{k+1}-\mathbf{W}^k||_F^2$ is a convergent series.*

Proof. Since $\forall k\ ||\mathbf{W}^{k+1}-\mathbf{W}^k||_F^2 \geq 0$, the sequence of partial sums for this series is monotonically increasing. We will show that this sequence is uniformly bounded and thus converges by the Monotone Convergence Theorem. Using $\mathbf{M}$ and $\mathbf{h}$ as defined in the proof of Lemma 25, we have

$$\begin{aligned}
||\mathbf{M}\mathbf{h}||_2^2 &= \mu||\mathbf{h}||_2^2 - ||(\mathbf{X}^\mathbf{T}\otimes\mathbf{X})\mathbf{h}||_2^2 \\
&= \left(\mu - ||\mathbf{X}||_2^4\right)||\mathbf{h}||_2^2 + ||\mathbf{X}||_2^4\,||\mathbf{h}||_2^2 - ||(\mathbf{X}^\mathbf{T}\otimes\mathbf{X})\mathbf{h}||_2^2.
\end{aligned}$$

Since

$$||(\mathbf{X}^\mathbf{T} \otimes \mathbf{X})\mathbf{h}||_2^2 \leq ||(\mathbf{X}^\mathbf{T} \otimes \mathbf{X})||_2^2 \, ||\mathbf{h}||_2^2 = ||\mathbf{X}||_2^4 \, ||\mathbf{h}||_2^2,$$

we have

$$(\mu - ||\mathbf{X}||_2^4) \, ||\mathbf{h}||_2^2 \leq ||\mathbf{M}\mathbf{h}||_2^2.$$

Hence

$$||\mathbf{h}||_2^2 \leq \frac{1}{(\mu - ||\mathbf{X}||_2^4)} ||\mathbf{M}\mathbf{h}||_2^2,$$

and

$$||\mathbf{W}^{k+1} - \mathbf{W}^k||_F^2 \leq \frac{1}{(\mu - ||\mathbf{X}||_2^4)} (\mu ||\mathbf{W}^{k+1} - \mathbf{W}^k||_F^2 - ||\mathbf{X}\mathbf{W}^{k+1}\mathbf{X} - \mathbf{X}\mathbf{W}^k\mathbf{X}||_F^2).$$

Fix a $K \in \mathbb{N}$. Then

$$\begin{aligned}
\sum_{k=0}^{K} ||\mathbf{W}^{k+1} - \mathbf{W}^k||_F^2 &\leq \sum_{k=0}^{K} \left[\frac{1}{(\mu - ||\mathbf{X}||_2^4)} (\mu ||\mathbf{W}^{k+1} - \mathbf{W}^k||_F^2 - ||\mathbf{X}\mathbf{W}^{k+1}\mathbf{X} - \mathbf{X}\mathbf{W}^k\mathbf{X}||_F^2) \right] \\
&\leq \frac{1}{(\mu - ||\mathbf{X}||_2^4)} \sum_{k=0}^{K} \left[J(\mathbf{W}^k) - J(\mathbf{W}^{k+1}) \right] \\
&= \frac{1}{(\mu - ||\mathbf{X}||_2^4)} \left[J(\mathbf{W}^0) - J(\mathbf{W}^{K+1}) \right] \\
&\leq \frac{J(\mathbf{W}^0)}{(\mu - ||\mathbf{X}||_2^4)},
\end{aligned}$$

where the second line follows from the proof of Lemma 25. Thus the sequence of partial sums is monotone and uniformly bounded, and converges by the Monotone Convergence Theorem. Therefore, the series converges as well. □

We can now prove Theorem 24.

Proof of Theorem 24. By Lemma 27 we have

$$\lim_{k\to\infty} ||\mathbf{T}^{k+1}(\mathbf{W}^0) - \mathbf{T}^k(\mathbf{W}^0)||_F = \lim_{k\to\infty} ||\mathbf{W}^{k+1} - \mathbf{W}^k||_F = 0.$$

This proves the result since $\mathbf{W}^0$ is arbitrarily chosen in $\mathbb{R}^{n\times m}$. □

3.9 Supplementary Material: Convergence Proof

In this section we prove Theorem 28, which was stated in the main text as Theorem 18. This is done using two lemmas and is provided as supplementary material since the proofs are slight adaptations of those given in [14]. To keep this section self contained we review two definitions from the main text before presenting the theorem.

Definition 3.9.1. *A mapping* $\mathbf{T}$ *is called nonexpansive if* $\forall\, \mathbf{Z}, \mathbf{Z}' \in \mathbb{R}^{n\times m}$,

$$||\mathbf{T}(\mathbf{Z}) - \mathbf{T}(\mathbf{Z}')||_F \leq ||\mathbf{Z} - \mathbf{Z}'||_F.$$

Definition 3.9.2. *A mapping* $\mathbf{T}$ *is called asymptotically regular if* $\forall\, \mathbf{Z} \in \mathbb{R}^{n\times m}$,

$$\lim_{k\to\infty} ||\mathbf{T}^{k+1}(\mathbf{Z}) - \mathbf{T}^k(\mathbf{Z})||_F = 0.$$

Theorem 28. *Let the mapping* $\mathbf{T} : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$ *be nonexpansive and asymptotically regular. Then the sequence* $\{\mathbf{W}^k = \mathbf{T}^k(\mathbf{W}^0)\}_{k\in\mathbb{N}}$ *converges to a fixed point of* $\mathbf{T}$.

We first provide the two lemmas.

Lemma 29. *Let the mapping* $\mathbf{T} : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$ *be nonexpansive and asymptotically regular. If a subsequence of* $\{\mathbf{T}^k(\mathbf{Z})\}_{k\in\mathbb{N}}$ *converges, then its limit is a fixed point of* $\mathbf{T}$.

Proof. Suppose that $\{\mathbf{T}^{k_a}(\mathbf{Z})\}_{k_a\in\mathbb{N}}$ is a subsequence of $\{\mathbf{T}^k(\mathbf{Z})\}_{k\in\mathbb{N}}$ and $\lim_{a\to\infty} \mathbf{T}^{k_a}(\mathbf{Z}) = \mathbf{V}$. Since $\mathbf{T}$ is asymptotically regular,

$$\lim_{k\to\infty} ||\mathbf{T}^k(\mathbf{Z}) - \mathbf{T}\mathbf{T}^k(\mathbf{Z})||_F = 0.$$

Hence

$$\lim_{a\to\infty} ||\mathbf{T}^{k_a}(\mathbf{Z}) - \mathbf{T}\mathbf{T}^{k_a}(\mathbf{Z})||_F = 0,$$

and

$$||\mathbf{V} - \mathbf{T}(\mathbf{V})||_F = 0,$$

where the last line follows because $\mathbf{T}$ is nonexpansive and thus continuous. Therefore $\mathbf{V} = \mathbf{T}(\mathbf{V})$, i.e., $\mathbf{V}$ is a fixed point of $\mathbf{T}$.

□

Lemma 30. *Let the mapping* $\mathbf{T} : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$ *be nonexpansive and have at least one fixed point. For a fixed point* $\mathbf{V}$*, the sequence* $\{||\mathbf{T}^k(\mathbf{Z}) - \mathbf{V}||_F\}_{k\in\mathbb{N}}$ *is nonincreasing and thus has a limit.*

Proof. We will show that the sequence is nonincreasing first.

$$||\mathbf{T}^{k+1}(\mathbf{Z}) - \mathbf{V}||_F = ||\mathbf{T}\mathbf{T}^k(\mathbf{Z}) - \mathbf{T}(\mathbf{V})||_F \leq ||\mathbf{T}^k(\mathbf{Z}) - \mathbf{V}||_F,$$

where the inequality follows from the fact that $\mathbf{T}$ is nonexpansive. The sequence $\{||\mathbf{T}^k(\mathbf{Z}) -$

$\mathbf{V}||_F\}_{k\in\mathbb{N}}$ is clearly bounded below by 0, so converges by the Monotone Convergence Theorem. This proves the result. □

Using these two lemmas, we can now prove Theorem 28.

Proof of Theorem 28. By Lemma 26, the sequence $\{\mathbf{W}^k\}_{k\in\mathbb{N}}$ is bounded. Hence a convergent subsequence exists by the Bolzano-Weierstrass Theorem. Call this convergent subsequence $\{\mathbf{W}^{k_a}\}_{k_a\in\mathbb{N}}$ and let $\lim_{a\to\infty}\mathbf{W}^{k_a} = \mathbf{V}$. Will we show that this limit point $\mathbf{V}$ is unique, thus proving that the sequence $\{\mathbf{W}^k\}_{k\in\mathbb{N}}$ converges to $\mathbf{V}$. So, suppose this limit point is not unique and there exists another convergent subsequence $\{\mathbf{W}^{k_b}\}_{k_b\in\mathbb{N}}$ such that $\lim_{b\to\infty}\mathbf{W}^{k_b} = \widehat{\mathbf{V}}$, where $\mathbf{V}\neq\widehat{\mathbf{V}}$. By Lemma 29, $\mathbf{V}$ and $\widehat{\mathbf{V}}$ are fixed points of $\mathbf{T}$, and thus by Lemma 30, the limits $\lim_{k\to\infty}||\mathbf{W}^k - \mathbf{V}||_F$ and $\lim_{k\to\infty}||\mathbf{W}^k - \widehat{\mathbf{V}}||_F$ both exist. Since $\{||\mathbf{W}^{k_a} - \mathbf{V}||_F\}_{k_a\in\mathbb{N}}$ and $\{||\mathbf{W}^{k_b} - \widehat{\mathbf{V}}||_F\}_{k_b\in\mathbb{N}}$ are subsequences of convergent sequences, we have

$$\lim_{k\to\infty}||\mathbf{W}^k - \mathbf{V}||_F = \lim_{a\to\infty}||\mathbf{W}^{k_a} - \mathbf{V}||_F = 0$$

and

$$\lim_{k\to\infty}||\mathbf{W}^k - \widehat{\mathbf{V}}||_F = \lim_{b\to\infty}||\mathbf{W}^{k_b} - \widehat{\mathbf{V}}||_F = 0.$$

Hence, $\lim_{k\to\infty}\mathbf{W}^k = \mathbf{V}$, and $\lim_{k\to\infty}\mathbf{W}^k = \widehat{\mathbf{V}}$. Thus, $\mathbf{V} = \widehat{\mathbf{V}}$ and the limit point is unique. This proves the result. □

3.10 Supplementary Material: Minimizer of J Proof

Theorem 31. *A fixed point of* $\mathbf{T}$ *is a minimizer of* $J(\mathbf{W})$.

Proof. Let $\mathbf{V}$ be a fixed point of $\mathbf{T}$. Then $\mathbf{V} = \mathbf{T}(\mathbf{V}) = \operatorname{argmin}_{\mathbf{Y} \in \mathbb{R}^{n \times m}} \widehat{J}(\mathbf{Y}, \mathbf{V})$. Hence, by Lemma 21, for every $\mathbf{H} \in \mathbb{R}^{n \times m}$,

$$\widehat{J}(\mathbf{V} + \mathbf{H}; \mathbf{V}) \geq \widehat{J}(\mathbf{V}; \mathbf{V}) + \mu ||\mathbf{H}||_F^2.$$

Noting that $\widehat{J}(\mathbf{V} + \mathbf{H}; \mathbf{V}) = J(\mathbf{V} + \mathbf{H}) + \mu ||\mathbf{H}||_F^2 - ||\mathbf{X}\mathbf{H}\mathbf{X}||_F^2$ and $\widehat{J}(\mathbf{V}; \mathbf{V}) = J(\mathbf{V})$, we have

$$J(\mathbf{V} + \mathbf{H}) \geq J(\mathbf{V}) + ||\mathbf{X}\mathbf{H}\mathbf{X}||_F^2$$

for all $\mathbf{H} \in \mathbb{R}^{n \times m}$. Thus, $\mathbf{V}$ is a minimizer of J. □

3.11 Supplementary Material: Approximate Proximal Operator

We experimented with using the neural network approximation of the proximal operator (see Chapter 1) instead of using the exact solution to the proximal operator in Algorithm 7. We found that the theory developed in Section 3.4 does not always hold when the approximate proximal operator is used. Specifically, we tracked the sequences $[J(\mathbf{W}^k)]_{n \in \mathbb{N}}$ and $[\widehat{J}(\mathbf{W}^{k+1}, \mathbf{W}^k)]_{k \in \mathbb{N}}$ during the surrogate functional iterations in runs of Algorithm 6, and observed that Lemma 25 is not always true. Details are provided below.

In order to produce an accurate proximal operator approximation, we created the training set from data that arises in runs of Algorithm 6. Since a goal of the network is to approximate the proximal operator for vectors of varying lengths, we ran Algorithm 6 on 1) a $1,000 \times 2000$ matrix with entries sampled from the standard normal distribution, and 2) a $1,100 \times 2000$ matrix with entries sampled from the standard normal distribution. The $\mathbf{x}$-α-τ triples for the training set

arise in the proximal operator problem of Algorithm 7. We randomly selected 5,000 $\mathbf{x}$-α-τ triples for vectors of length 1,000 (out of 34,599) and 5,000 for vectors of length 1,100 (out of 37,794). The network was trained on these 10,000 $\mathbf{x}$-α-τ triples.

We then used this network to approximate the proximal operators in Algorithm 7 on one of the matrices that produced data for the training set. Even for data that the network was trained on, the sequences $[J(\mathbf{W}^k)]_{n\in\mathbb{N}}$ and $[\widehat{J}(\mathbf{W}^{k+1}, \mathbf{W}^k)]_{k\in\mathbb{N}}$ were not nonincreasing. Hence, if we use the neural network approximation to the proximal operator in Algorithm 7, we are not guaranteed that the sequence $\{\mathbf{W}^k = \mathbf{T}^k(\mathbf{W}^0)\}_{k\in\mathbb{N}}$ converges to a fixed point of $\mathbf{T}$ nor that the sequence $\{\mathbf{W}^k\}_{k\in\mathbb{N}}$ converges to a minimizer of J. Thus, we did not use the approximation of the proximal operator in experiments in the main text.

Chapter 4: Protein Expression Discriminant Analysis with CUR

4.1 Introduction

CUR matrix approximation has been successfully used for feature selection in clustering applications. We provided one example in Chapter 2, in which we extended the experiment of Sorenson and Embree [2] of selecting important genes to cluster patients into two classes - those with and without a lung tumor. Mahoney and Drineas [1] provided other examples: selecting important terms from a document-term matrix in order to cluster documents, and selecting important genes from gene expression data in order to cluster patients by cancer type. In addition, Li et al. [27] used a convex formulation of CUR that jointly chooses $\mathbf{C}$ and $\mathbf{R}$ to perform feature and sample selection on gene expression data, molecular data, image and video data, and human activity recognition data recorded by a smartphone in order to achieve better classification accuracy.

In this work, we are interested in a novel application of CUR for feature selection. Higuera et al. [15] used Self-Organizing Maps (SOMs)[1] and the Wilcoxon rank-sum test to determine proteins that critically affect learning in wild type and trisomic (Down syndrome) mice. Specifically, discriminant proteins were discovered in biologically relevant pairwise class comparisons with the Wilcoxon rank-sum test functioning as a feature selection process. In this work, we explore

[1] also known as Kohonen Maps.

the use of CUR as the feature selection method in a subset of these computational experiments. This is not only a novel application of CUR, but to the best of our knowledge, also the first use of CUR on protein expression data.

4.2 Prior Computational Experiments

In this section we provide a summary of the dataset used and computational experiments performed in [15].

4.2.1 Data

The protein expression data used in these experiments was measured from two groups of mice, control and trisomic. Each mouse was exposed to one option from each of two treatments:

1. Context fear conditioning (CFC), an associative learning assessment task, of either context-shock (CS) or shock-context (SC), and

2. an injection of memantine, a drug known to treat learning impairment, or saline.

The CFC task consisted of placing a mouse in a novel cage and allowing it to explore. The context-shock option involves giving the mouse an electric shock after a few minutes of cage exploration, whereas the shock-context option involves an immediate shock to the mouse before exploration. Control mice given the context-shock option will learn an association between cage and shock, whereas those given the shock-context option will not. However, trisomic mice given the context-shock option will not learn the association between cage and shock. Thus, the second treatment, an injection of memantine or saline, is given prior to the CFC task. Trisomic mice injected with memantine prior to the CFC context-shock task will learn the association as the

Class Name	Genotype	CFC (Stimulation to Learn)	Injection	Learning	Class Size
c-CS-s	control	context-shock (yes)	saline	Normal	9
c-CS-m	control	context-shock (yes)	memantine	Normal	10
c-SC-s	control	shock-context (no)	saline	None	9
c-SC-m	control	shock-context (no)	memantine	None	10
t-CS-s	trisomic	context-shock (yes)	saline	Failed	7
t-CS-m	trisomic	context-shock (yes)	memantine	Rescued	9
t-SC-s	trisomic	shock-context (no)	saline	None	9
t-SC-m	trisomic	shock-context (no)	memantine	None	9

Table 4.1: Classes of 72 total mice.

control mice do. Learning in control mice is not affected by memantine injection. Table 4.1 summarizes the eight classes of mice and presents class size and type of learning. For the remainder of this work, we will refer to classes by their names in Table 4.1.

Data consist of expression levels for 77 proteins measured from the brains of the 72 mice represented in Table 4.1. Each protein was measured 15 times for each mouse giving a total of 1,080 measurements of 77 proteins, resulting in a data matrix that is 1,080 $\times$ 77. For each of the 1,080 observations, the mouse id and class of the mouse that produced the measurements are also provided. The dataset is available in the supporting information of [15] and in the University of California, Irvine Machine Learning Repository[2].

Missing data arises as a consequence of the protein measurement process. Higuera et al. processed the data by 1) removing an outlier mouse with mainly missing data, 2) filling in missing entries, and 3) normalizing the data. For any mice in class c missing data for protein p, the missing entries were replaced with the average expression level of protein p from the reported (non-missing) entries for mice in class c. Min-max normalization was then applied to each column of the data, i.e. for each protein. When we analyzed the raw data, we could not identify the outlier mouse and suspect the data for this mouse was already removed from the dataset before it was

[2] `https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression`

posted for download. We found that 1.7% of entries in the data were missing with only 6 out of 77 proteins missing 20 or more measurements out of the 1,080 total. We also discovered that two columns of the raw data are equal; these columns correspond to the proteins ARC_N and pS6_N. These columns are clearly the same after the data is processed as well, and thus make the matrix of protein expression data rank deficient.

4.2.2 Methodology

We first give a high-level overview of the methodology and then follow with more details. SOMs and the Wilcoxon rank-sum test[3] were used to discover discriminant proteins in biologically relevant pairwise class comparisons. An SOM is an unsupervised neural network clustering method that can identify the topology and distribution of data such that clusters that exist close together in the topology should cluster similar data points. It is useful for dimension reduction and provides 2D visualization of the data. An SOM is used in this case to cluster mice with similar protein expression levels in order to discover protein expression patterns among classes. The data provided to the SOM include the protein expression data, but not the class of each mouse. The Wilcoxon rank-sum test is then used to find discriminant proteins between pairs of SOM "class-specific clusters" of mice (details below). This nonparametric test checks for equal medians between two independent samples of data that are not necessarily the same size.

Higuera et al. used this method on 1) data from the four control classes, 2) data from the four trisomic classes, and 3) a mixture of the two (c-CS-s, c-CS-m, t-CS-m, t-CS-s, and c-SC-s). We explore feature selection with CUR instead of the Wilcoxon rank-sum test on data from the four control classes only. The particular experiments are detailed below.

[3] also called the Mann Whitey U test.

Initially, ten 7×7 SOMs are computed on the processed data from the four control classes, a 570×77 dataset. Since SOM neuron weights are initialized randomly, each SOM instance will most likely be different. The average quantization error, q, is measured for each SOM as

$$q = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{d}_i - \mathbf{w}_{BMU(\mathbf{d}_i)}\|_2,$$

where n is the number of observations, $\mathbf{d}_i \in \mathbb{R}^{77}$ is an observation (a row of the data matrix), and $\mathbf{w}_{BMU(\mathbf{d}_i)} \in \mathbb{R}^{77}$ is the weight vector of the Best Matching Unit (BMU) or closest neuron to $\mathbf{d}_i$. The SOM with the smallest average quantization error is then used to identify "class-specific clusters", defined in [15] as "(i) two or more adjacent [neurons] that contain mice of the same class and no mice from other classes, or (ii) a single [neuron] that contains $\geq 80\%$ (or ≥ 12 of 15) of the measurements of one mouse and no measurements of mice from any other class." While the class of each mouse was not used in the learning process of the SOM, the class of each mouse is used to determine the class-specific clusters. Two class-specific clusters can be compared using the weight vectors of the neurons included in the cluster and 77 instances of the Wilcoxon rank-sum test, one for each protein (each neuron weight vector is length 77). For example, to compare levels of the protein in column 5 of the dataset, the Wilcoxon rank-sum test would use two samples - one created from the fifth element of each neuron weight vector for the neurons in the first class-specific cluster, and one created from the fifth element of each neuron weight vector for the neurons in the second class-specific cluster. Those proteins for which the Wilcoxon rank-sum test returns a p-value of less than 0.05 are considered to be the discriminant proteins between the two class-specific clusters and thus the two classes they represent.

A new 7×7 SOM is then created using data for the discriminant proteins only (a $570 \times$

k matrix where k is the number of discriminant proteins). Class-specific clusters are identified in this SOM. The discriminant proteins are validated through a qualitative analysis of this SOM, which includes the number of mixed class neurons and the number of observations in mixed class neurons as metrics to determine how well the discriminant proteins clustered the data.

In particular, Higuera et al. found the common discriminant proteins to four pairwise comparisons involving successful learning, c-CS-s vs. c-SC-s, c-CS-m vs. c-SC-m, c-CS-m vs. c-SC-s, and c-CS-s vs. c-SC-m and then used the results in two other experiments.

1. Experiment 1: Finding the discriminant proteins between c-CS-m and c-CS-s and taking the union of these proteins with the common discriminant proteins between the four successful learning comparisons.

2. Experiment 2: Finding the discriminant proteins between c-SC-m and c-SC-s and taking the union of these proteins with the common discriminant proteins between the four successful learning comparisons.

Each experiment produced an SOM that was qualitatively analyzed in order to validate the selection of discriminant proteins. In the next two sections of this work we describe how we used CUR as a feature selection method in this methodology and provide results for its use in Experiments 1 and 2.

4.3 Feature Selection Using CUR

To use CUR as a feature selection method between two class-specific clusters, we construct a matrix $\mathbf{D}$ that contains pairwise differences between neuron weight vectors in opposite clusters. For example if cluster A contains a neurons (call their weight vectors $\mathbf{A}_1, ..., \mathbf{A}_a$) and cluster B

contains b neurons (call their weight vectors $\mathbf{B}_1, ..., \mathbf{B}_b$), then $\mathbf{D} \in \mathbb{R}^{ab \times 77}$, and

$$\mathbf{D}(j + b(i-1), :) = \mathbf{A}_i - \mathbf{B}_j,$$

for $i \in [a]$ and $j \in [b]$. We then compute 77 CUR approximations of $\mathbf{D}$ - one for each possible number of columns to select for the matrix $\mathbf{C}$, i.e., $1, 2, ..., 77$[4]. For this particular application, we are only interested in the chosen subset of columns selected for $\mathbf{C}$. In addition, in each CUR approximation that we use, the columns are chosen first, independently of the rows. Hence, we could select any number of rows for the matrix $\mathbf{R}$. We chose to use all rows and set $\mathbf{R} = \mathbf{D}$. In order to select a CUR approximation from the 77 calculated, we compute the Akaike information criterion (AIC) and the Bayesian information criterion (BIC) for each CUR model as given in the formulas below. Let $\mathbf{D} \in \mathbb{R}^{m \times 77}$, and $\mathbf{CUR}$ be the CUR approximation to $\mathbf{D}$ where $\mathbf{C} \in \mathbb{R}^{m \times c}$. Then,

$$\text{AIC} = 2(mc + 3) + 77m \ln\left(\frac{\|\mathbf{D} - \mathbf{CUR}\|_F^2}{77m}\right),$$

and

$$\text{BIC} = (mc + 3)\ln(77m) + 77m \ln\left(\frac{\|\mathbf{D} - \mathbf{CUR}\|_F^2}{77m}\right).$$

The CUR model with the lowest AIC score and the CUR model with the lowest BIC score are selected. The columns chosen to be in the matrix $\mathbf{C}$ of each CUR correspond to the discriminant proteins. We then train two SOMs - the first using the discriminant proteins from the CUR model with the lowest AIC and the second using the discriminant proteins from the CUR model with

[4]Since two columns of the raw protein expression data are equal, $\mathbf{D}$ also has this property. Thus, the SF CUR may fail to produce a selection of columns for particular values of the number of columns to select. In these cases, we ignore these values of the number of columns to select.

the lowest BIC. Hence, using CUR for feature selection will result in two possibilities for the set of discriminant proteins, which can be compared through a qualitative assessment of the SOMs trained on each set.

4.4 Results

We repeated Experiments 1 and 2 by Higuera et al. with two exceptions: 1) we used CUR as the feature selection process instead of the Wilcoxon rank-sum test, and 2) each time we needed to use an SOM, we trained 10 SOMs and chose the one with the minimum number of mixed-class neurons. If multiple SOMs had the minimum number of mixed-class neurons, we chose the SOM with the minimum number of observations in mixed-class neurons. We focused on these metrics based on their importance in the qualitative analysis of the discriminant protein SOMs in [15]. We compared the performance of four CUR algorithms in this feature selection task: SF CUR, deterministic LS CUR, DEIM CUR, and QR CUR. (Each of these CUR algorithms were described in Chapter 2.) In addition, we used the Wilcoxon rank-sum test for feature selection in order to compare to the performance of the CUR algorithms. In all experiments, the deterministic LS CUR rank parameter for leverage score computation was 2. All experiments were run in MATLAB on a laptop with 16GB RAM and a 2.20 GHz Intel Core i7-1360P processor[5].

Figure 4.1 presents the SOM using all 77 proteins. Neurons are labeled with the classes of mice they contain in sorted order, i.e. the first class listed is the majority class, and the number of observations contained in the neuron. Neurons are colored by their majority class - c-CS-m: yellow, c-CS-s: green, c-SC-m: tan, c-SC-s: orange - and a bold outline of a neuron represents

[5]The SOM implementation in the Deep Learning Toolbox was used with the default parameters except the SOM size.

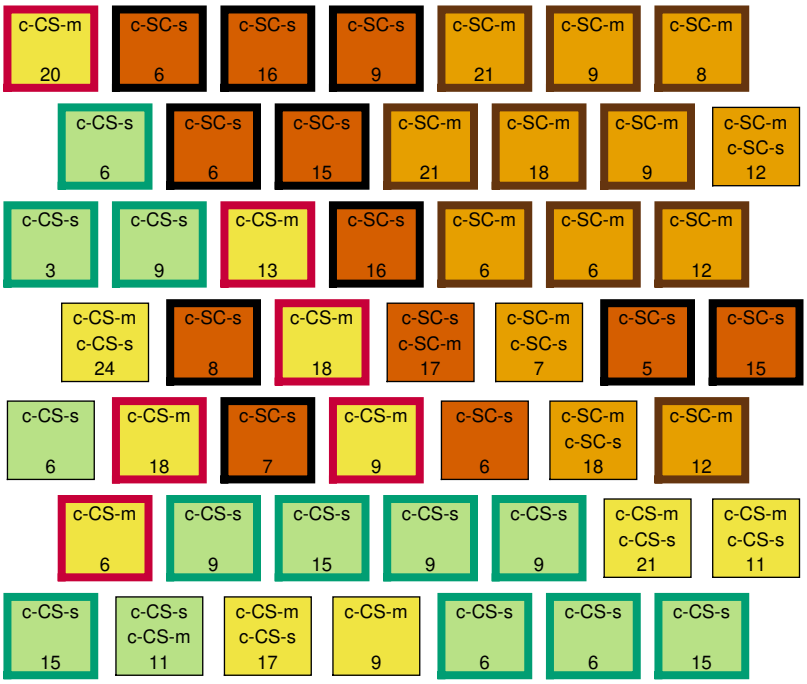


Figure 4.1: SOM using all 77 proteins.

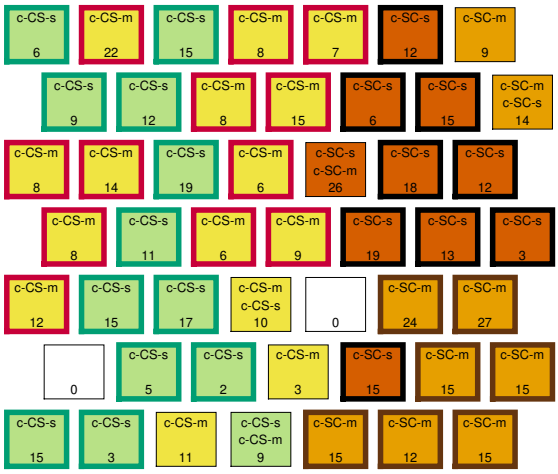
class-specific cluster membership - c-CS-m class cluster: red outline, c-CS-s class cluster: green outline, c-SC-m class cluster: brown outline, c-SC-s class cluster: black outline. This color scheme is similar to that in [15].

We define a mixed-CS-class neuron as a mixed-class neuron that includes either c-CS-m or c-CS-s observations, and a mixed-SC-class neuron as a mixed-class neuron that includes either c-SC-m or c-SC-s observations. As a reference for comparison in the results of Experiments 1 and 2, the SOM in Figure 4.1 has five mixed-CS-class neurons which contain 84 observations and four mixed-SC-class neurons which contain 54 observations. The goal of Experiment 1 is to select discriminant proteins such that when an SOM is trained on the discriminant protein data only, the SOM improves, i.e. has a smaller number of mixed-CS-class neurons and observations contained within those neurons, as compared to the SOM in Figure 4.1 that was trained on all protein data. The goal of Experiment 2 is similar except that the discriminant protein SOM should have

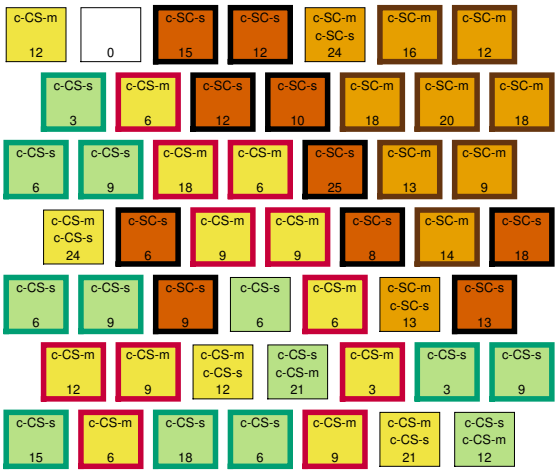
a smaller number of mixed-SC-class neurons and observations contained within those neurons. Since each CUR algorithm results in two potential sets of discriminant proteins (one for the CUR with minimum AIC and one for the CUR with minimum BIC), we present results for nine feature selection methods: 1) Wilcoxon rank-sum test, 2-3) SF CUR AIC/BIC, 4-5) deterministic LS CUR AIC/BIC, 6-7) DEIM CUR AIC/BIC, and 8-9) QR CUR AIC/BIC.

4.4.1 Experiment 1

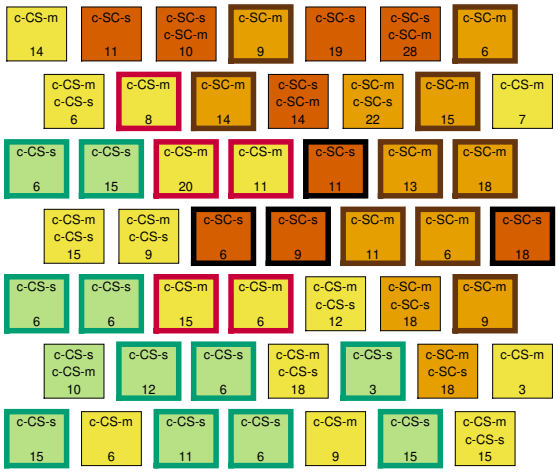
For each feature selection method we 1) identified the common discriminant proteins between the four successful learning comparisons, c-CS-s vs. c-SC-s, c-CS-m vs. c-SC-m, c-CS-m vs. c-SC-s, and c-CS-s vs. c-SC-m, and 2) identified the discriminant proteins between the c-CS-m and c-CS-s classes. The union of these two sets of proteins is the set of discriminant proteins. We present the results of Experiment 1 in Table 4.2. In addition, Figure 4.2 contains the discriminant protein SOMs for the Wilcoxon rank-sum test and each CUR algorithm using the AIC model selection criteria. Since the CUR algorithms using the AIC criteria generally outperformed those using the BIC criteria, the discriminant protein SOMs for the CUR algorithms using the BIC model selection criteria are found in Supplementary Section 4.6.



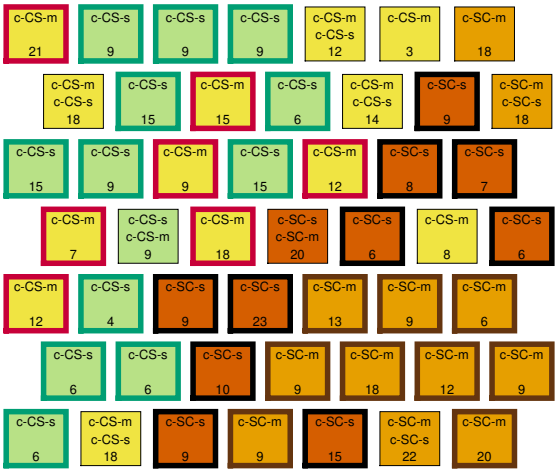
(a) Wilcoxon rank-sum test



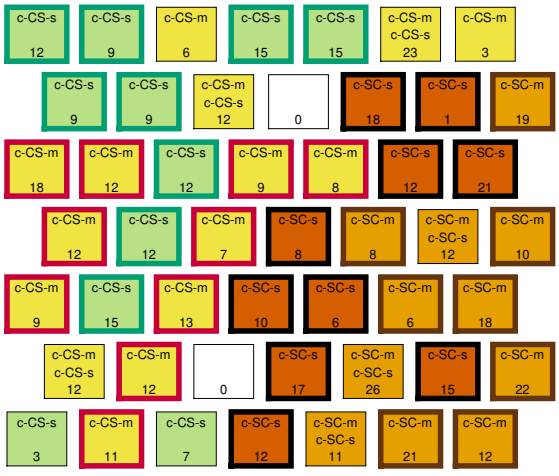
(b) SF CUR, AIC



(c) LS CUR, AIC



(d) DEIM CUR, AIC



(e) QR CUR, AIC

Figure 4.2: Experiment 1 discriminant protein SOMs for the Wilcoxon rank-sum test and CUR algorithms using the AIC model selection criteria.

Feature Selection Method	# Mixed-CS Neurons	# Observations in Mixed-CS Neurons	# Discriminant Proteins
Wilcoxon rank-sum test	**2**	**19**	15
SF CUR - AIC	5	90	35
SF CUR - BIC	7	91	21
LS CUR - AIC	7	85	49
LS CUR - BIC	6	105	28
DEIM CUR - AIC	5	71	20
DEIM CUR - BIC	7	90	18
QR CUR - AIC	3	47	18
QR CUR - BIC	7	77	16

Table 4.2: Experiment 1 results. The minimum numbers of mixed-CS-class neurons and observations are in bold. When feature selection was not used and all 77 proteins were used to train an SOM, this resulted in 5 mixed-CS-class neurons which contain 84 observations.

The Wilcoxon rank-sum test performs the best of the feature selection methods, resulting in two mixed-CS-class neurons and 19 observations in those neurons, which is by far the minimum number of observations in mixed-CS-class neurons. It is interesting to note that not only does the Wilcoxon rank-sum test perform the best, but it also selects the fewest number of discriminant proteins. Amongst CUR algorithms, the QR CUR - AIC performs the best, resulting in only three mixed-CS-class neurons containing 47 observations. All CUR algorithms except the QR CUR - AIC, QR CUR - BIC, and DEIM CUR - AIC perform worse than the baseline comparison when no feature selection is performed (5 mixed-CS-class neurons which contain 84 observations). Although, the QR CUR - BIC results are mixed: there are two more mixed-CS-class neurons, but these neurons contain less observations than that of the baseline.

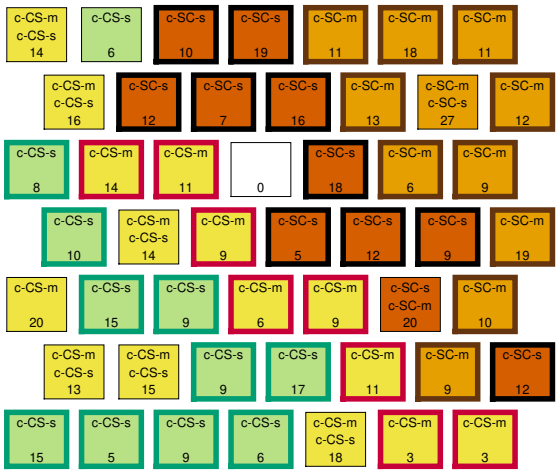
4.4.2 Experiment 2

For each feature selection method we 1) again identified the common discriminant proteins between the four successful learning comparisons, c-CS-s vs. c-SC-s, c-CS-m vs. c-SC-m, c-

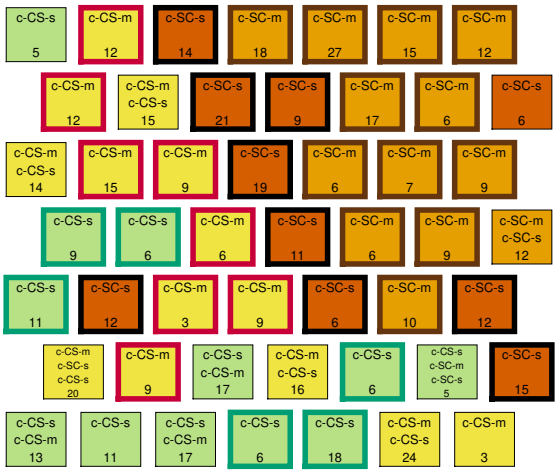
CS-m vs. c-SC-s, and c-CS-s vs. c-SC-m, and 2) identified the discriminant proteins between the c-SC-m and c-SC-s classes. The union of these two sets of proteins is the set of discriminant proteins. Results for Experiment 2 are presented in Table 4.3. The discriminant protein SOMs for the Wilcoxon rank-sum test and CUR algorithms using the AIC model selection criteria are given in Figure 4.3. The discriminant protein SOMs for CUR algorithms using the BIC model selection criteria are given in Supplementary Section 4.6.

Feature Selection Method	**# Mixed-SC Neurons**	**# Observations in Mixed-SC Neurons**	**# Discriminant Proteins**
Wilcoxon Rank Sum Test	**2**	47	27
SF CUR - AIC	3	**37**	49
SF CUR - BIC	7	99	27
LS CUR - AIC	6	89	29
LS CUR - BIC	8	109	28
DEIM CUR - AIC	**2**	42	26
DEIM CUR - BIC	**2**	42	26
QR CUR - AIC	3	52	25
QR CUR - BIC	4	67	20

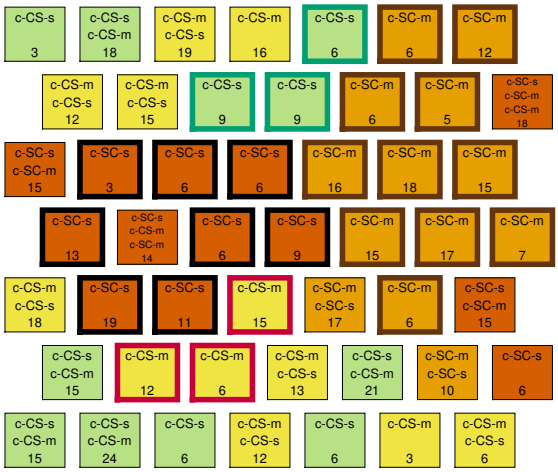
Table 4.3: Experiment 2 results. The minimum numbers of mixed-SC-class neurons and observations are in bold. When feature selection was not used and all 77 proteins were used to train an SOM, this resulted in 4 mixed-SC-class neurons which contain 54 observations.



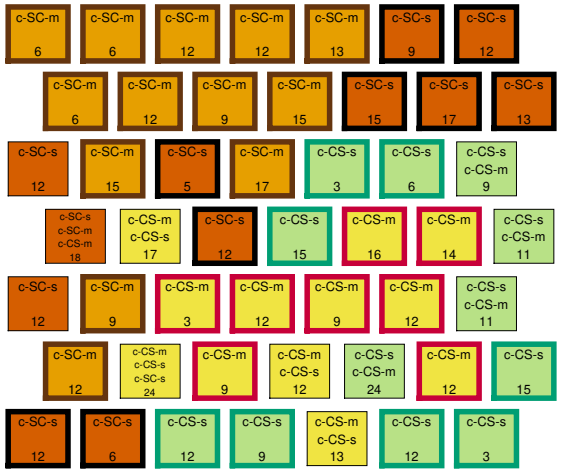
(a) Wilcoxon rank-sum test



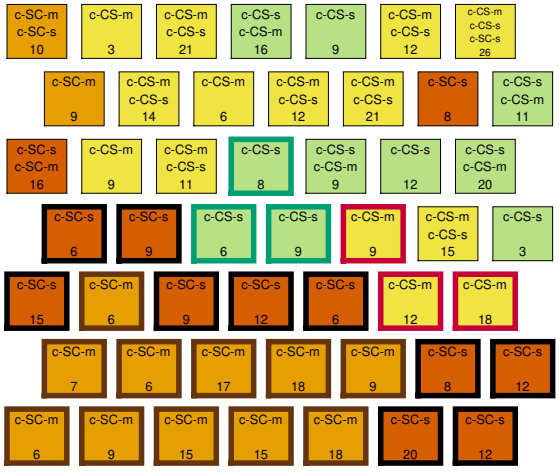
(b) SF CUR, AIC



(c) LS CUR, AIC



(d) DEIM CUR, AIC



(e) QR CUR, AIC

Figure 4.3: Experiment 2 discriminant protein SOMs for the Wilcoxon rank-sum test and CUR algorithms using the AIC model selection criteria.

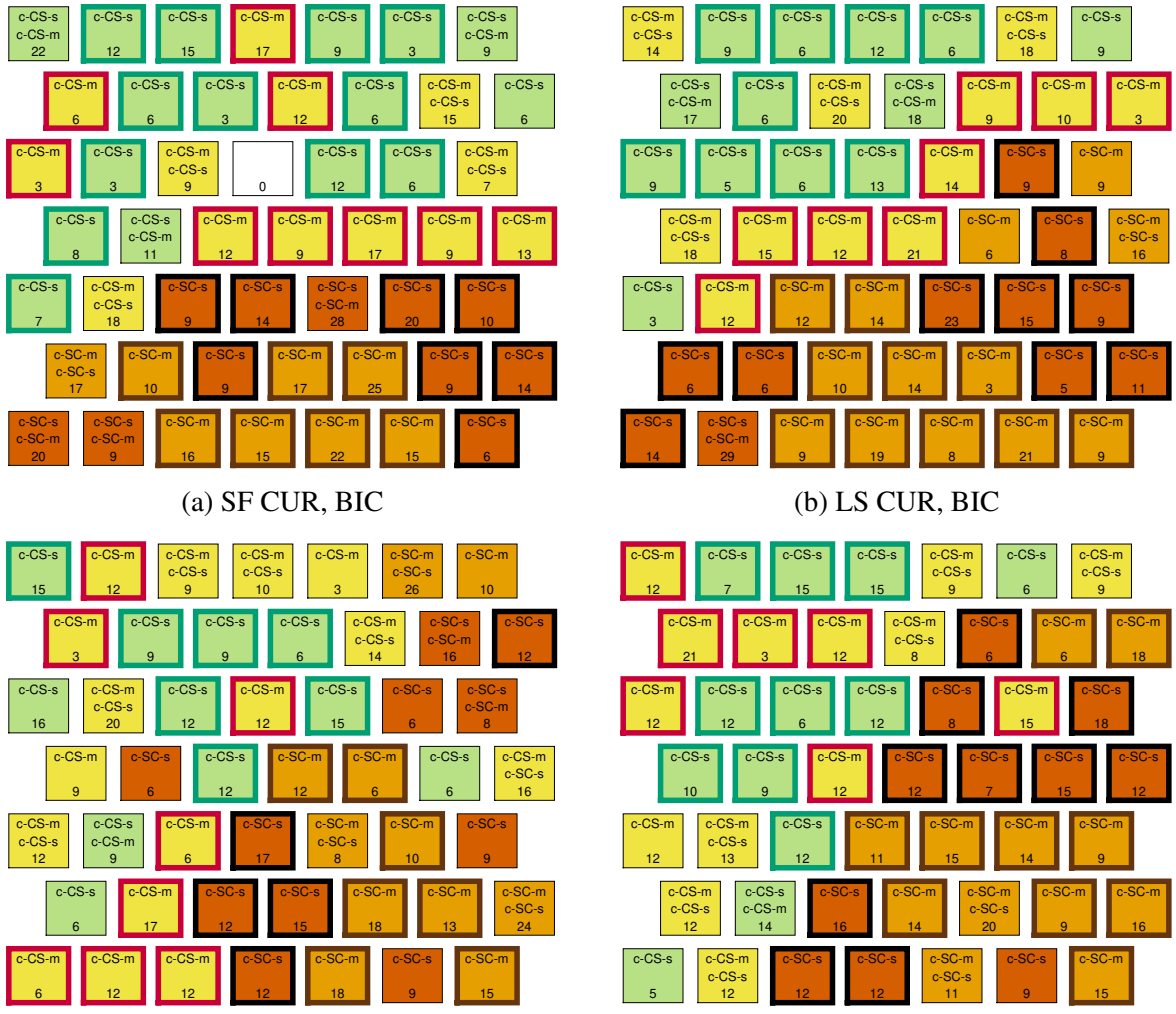
The best performing feature selection methods are the DEIM CUR - AIC and DEIM CUR - BIC, each resulting in two mixed-SC-class neurons containing 42 observations. The Wilcoxon rank-sum test also performs well, resulting in two mixed-SC-class neurons containing 47 observations. While the SF CUR - AIC results in three mixed-SC-class neurons, it does achieve the minimum number of observations in mixed-SC-class neurons with 37; however, it selects 49 discriminating proteins which is at least 20 more than all other feature selection methods. The SF CUR - BIC, LS CUR - AIC, and LS CUR - BIC perform poorly compared to the other feature selection methods. In addition, these three methods and the QR CUR - BIC perform worse than the baseline of four mixed-SC-class neurons that contain 54 observations when there is no feature selection.

4.5 Conclusion

We explored the CUR approximation as a feature selection method in the application by Higuera et al. [15] to determine discriminant proteins when clustering protein expression data in an SOM. This is a novel application of CUR and to the best of our knowledge, this is the first use of CUR on protein expression data. We compared the performance of four CUR algorithms and the Wilcoxon rank-sum test (the feature selection method in [15]) as the feature selection method in two experiments of this application. While performance varied between CUR algorithms, CUR performance was generally poor in Experiment 1, with the exception of the QR CUR - AIC. However, multiple CUR algorithms performed well in Experiment 2: the DEIM CUR - AIC, DEIM CUR - BIC, SF CUR - AIC, and the QR CUR - AIC. In fact, the DEIM CUR - AIC and DEIM CUR - BIC were the best performing feature selection methods in Experiment

2. Hence, we have demonstrated that CUR can be used effectively for feature selection in this application.

4.6 Supplementary Material: Results Using CUR Algorithms and BIC



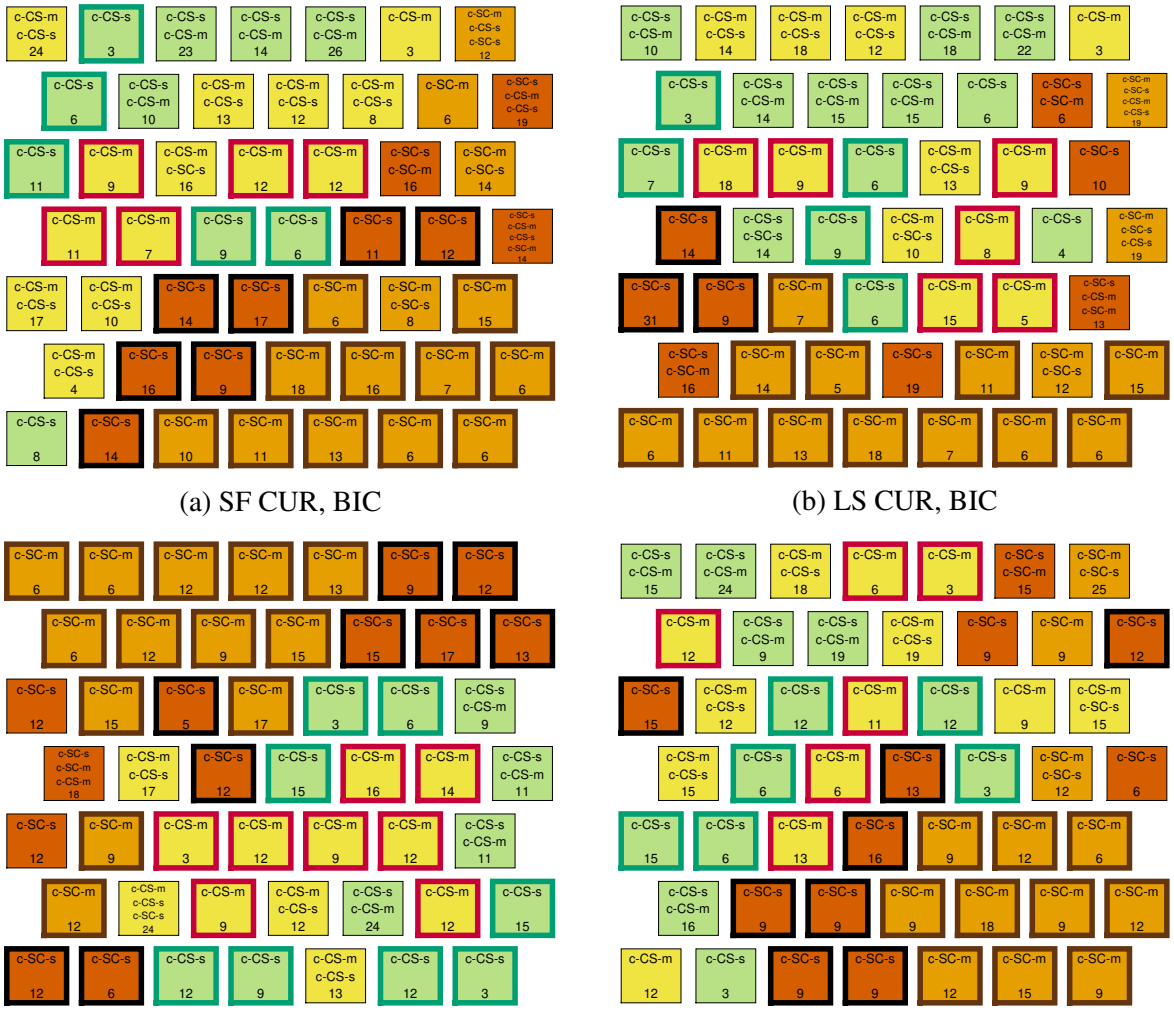
(a) SF CUR, BIC

(b) LS CUR, BIC

(c) DEIM CUR, BIC

(d) QR CUR, BIC

Figure 4.4: Experiment 1 discriminant protein SOMs for CUR algorithms using the BIC model selection criteria.



(a) SF CUR, BIC

(b) LS CUR, BIC

(c) DEIM CUR, BIC

(d) QR CUR, BIC

Figure 4.5: Experiment 2 discriminant protein SOMs for CUR algorithms using the BIC model selection criteria.

Chapter 5: Conclusion

In this thesis, we have presented 1) a neural network approximation of the proximal operator of the ℓ_∞ norm, 2) a novel CUR matrix approximation using convex optimization, and 3) a novel application of CUR to determine discriminant proteins in protein expression data. Item 1) is related to items 2) and 3) since our CUR implementation requires repeated use of the proximal operator of the ℓ_∞ norm.

In Chapter 2, we developed an $O(m)$ neural network approach to approximate the proximal operator of the ℓ_∞ norm, $\mathbf{prox}_{\alpha||\cdot||_\infty}(\mathbf{x})$, for $\mathbf{x} \in \mathbb{R}^m$. The network is novel since it can accept vectors of varying lengths due to a feature selection process based on moments of the input data. The feature selection process for the network was motivated by the exact proximal operator algorithm we developed in Chapter 2. We demonstrated the computational efficiency of the network approach as compared to the exact $O(m \log m)$ algorithm to compute the proximal operator, and computed the error in the approximate proximal operator and objective function as compared to the exact proximal operator and objective function, respectively. In terms of approximation error, we also showed that the performance of the network is dependent on the distribution of the input data.

In Chapter 3, we demonstrated a novel CUR formulation using convex optimization and a corresponding algorithm that uses the surrogate functional [14] to solve the convex optimization

problems that arise. To the best of our knowledge, this is the only CUR method using convex optimization that solves for $\mathbf{C}$ and $\mathbf{R}$ separately and allows the user to choose the number of columns and rows for inclusion in $\mathbf{C}$ and $\mathbf{R}$, respectively. In addition, we extended the theory of the surrogate functional technique to apply to our CUR formulation. We note that we experimented using the neural network approximation of the proximal operator in our CUR algorithm. It did not speed up the CUR calculation, since the convergence theory of the CUR algorithm no longer held, and more iterations of the surrogate functional were needed. Hence, we used the exact proximal operator calculation in the CUR algorithm.

In Chapter 3 we also numerically demonstrated the use of our CUR approximation on sparse and dense data. Specifically, we 1) calculated its reconstruction accuracy on a document-term dataset and a gene expression dataset, and 2) used it as a feature selection method on the gene expression dataset in order to classify patients as those with and without a tumor as done in [2]. We compared our CUR algorithm on these numerical tasks to the SVD and other deterministic CUR approximations that solve for $\mathbf{C}$ and $\mathbf{R}$ separately and allow the user to select the number of columns and rows for inclusion in $\mathbf{C}$ and $\mathbf{R}$, respectively. We found that while the SVD provides the optimum approximation to each matrix, two CUR methods (including ours) performed the best in selecting features to separate patient classes on the gene expression dataset.

In Chapter 4, we presented a novel application of CUR to determine discriminant proteins when clustering protein expression data in a Self-Organizing Map. These experiments were based on those in [15], with the exception that CUR was used as the feature selection method instead of the Wilcoxon rank-sum test. We compared the performance of our CUR algorithm with that of the Wilcoxon rank-sum test and other deterministic CUR approximations that solve for $\mathbf{C}$ and $\mathbf{R}$ separately and allow the user to select the number of columns and rows for inclusion

in $\mathbf{C}$ and $\mathbf{R}$, respectively. We performed two experiments and found that performance varied between CUR algorithms. While CUR performance was generally poor in Experiment 1, multiple CUR algorithms performed well in Experiment 2, two of which were the best performing feature selection methods in Experiment 2. In addition, to the best of our knowledge, this was also the first use of CUR on protein expression data.

The work contained in this thesis has also given rise to other research questions, which are potential areas for future work. In Chapter 2, we approximated τ using a neural network, but also mentioned that τ can be approximated using a polynomial in the moments of the input vector. One avenue of future work could be comparing the approximation of τ achieved by the neural network to that achieved using the polynomial with a few potential bisection steps. As mentioned in Chapter 3, we can generalize the problem to find $\mathbf{C}$ in our CUR algorithm as

$$\min_{\mathbf{W}\in\mathbb{R}^{n\times m}} ||\mathbf{X}-\mathbf{X}\mathbf{W}\mathbf{X}||_p^p + \lambda_C \sum_{i=1}^{n} ||\mathbf{W}(i,:)||_\infty,$$

where $p \in [1,\infty]$. We think that this problem can be solved using a mixture of ADMM and the surrogate functional; however, numerical experimentation and proving convergence of this method are potential areas for future work. Other possible areas for future work include investigating implementation speed ups for our CUR algorithm on sparse data, and investigating why certain CUR algorithms perform better than others in particular applications or on particular datasets.

Bibliography

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

[2] D. C. Sorenson and Mark Embree. A deim induced cur factorization. *SIAM Journal on Scientific Computing*, 38(3):A1454–A1482, 2016.

[3] Keaton Hamm and Longxiu Huang. Perspectives on cur decompositions. *Applied and Computational Harmonic Analysis*, 48(3):1088 – 1099, 2020.

[4] S.A. Goreinov, E.E. Tyrtyshnikov, and N.L. Zamarashkin. A theory of pseudoskeleton approximations. *Linear Algebra and its Applications*, 261(1):1–21, 1997.

[5] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. *SIAM Journal on Computing*, 36(1):184–206, 2006.

[6] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Relative-error cur matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30(2):844–881, 2008.

[7] G.W. Stewart. Four algorithms for the the efficient computation of truncated pivoted qr approximations to a sparse matrix. *Numerische Mathematik*, 83:313–323, 1999.

[8] Julien Mairal, Rodolphe Jenatton, Guillaume Obozinski, and Francis Bach. Convex and network flow optimization for structured sparsity. *J. Mach. Learn. Res.*, 12(null):2681–2720, nov 2011.

[9] Yang Liu and Jian Shao. High dimensionality reduction using cur matrix decomposition and auto-encoder for web image classification. In Guoping Qiu, Kin Man Lam, Hitoshi Kiya, Xiang-Yang Xue, C.-C. Jay Kuo, and Michael S. Lew, editors, *Advances in Multimedia Information Processing - PCM 2010*, pages 1–12, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[10] HanQin Cai, Keaton Hamm, Longxiu Huang, and Deanna Needell. Robust cur decomposition: Theory and imaging applications. *SIAM Journal on Imaging Sciences*, 14(4):1472–1503, 2021.

[11] Ashkan Esmaeili, Mohsen Joneidi, Mehrdad Salimitari, Umar Khalid, and Nazanin Rahnavard. Two-way spectrum pursuit for cur decomposition and its application in joint column/row subset selection. In *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2021.

[12] Laurent Condat. Fast projection onto the simplex and the l_1 ball. *Mathematical Programming*, 158(1):575 – 585, 2016.

[13] Jacob Bien, Ya Xu, and Michael W. Mahoney. Cur from a sparse optimization viewpoint. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10, page 217–225, Red Hook, NY, USA, 2010. Curran Associates Inc.

[14] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

[15] Clara Higuera, Katheleen J. Gardiner, and Krzysztof J. Cios. Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PLOS ONE*, 10(6):1–28, 06 2015.

[16] Neal Parikh and Stephen Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, jan 2014.

[17] Nicholas G. Polson, James G. Scott, and Brandon T. Willard. Proximal Algorithms in Statistics and Machine Learning. *Statistical Science*, 30(4):559 – 581, 2015.

[18] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 272–279, New York, NY, USA, 2008. Association for Computing Machinery.

[19] Michael Held, Philip Wolfe, and Harlan P. Crowder. Validation of subgradient optimization. *Math. Program.*, 6(1):62–88, dec 1974.

[20] Shai Shalev-Shwartz and Yoram Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, 7(58):1567–1599, 2006.

[21] Guillaume Perez, Michel Barlaud, Lionel Fillatre, and Jean-Charles Régin. A filtered bucket-clustering method for projection onto the simplex and the ℓ_1 ball. *Mathematical Programming*, 182:445–464, 2020.

[22] A. Wayne Roberts and Dale E. Varberg. *Convex Functions*. Academic Press, New York, NY, 1973.

[23] Radu Balan, Naveed Haghani, and Maneesh Singh. Permutation invariant representations with applications to graph deep learning. *arXiv preprint arXiv:2203.07546 [math.FA]*, 2022.

[24] Yijun Dong and Per-Gunnar Martinsson. Simpler is better: a comparative study of randomized pivoting algorithms for cur and interpolative decompositions. *Advances in Computational Mathematics*, 49, 2023.

[25] Yasutoshi Ida, Sekitoshi Kanai, Yasuhiro Fujiwara, Tomoharu Iwata, Koh Takeuchi, and Hisashi Kashima. Fast deterministic CUR matrix decomposition with accuracy assurance. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4594–4603. PMLR, 13–18 Jul 2020.

[26] Zhen Peng, Minnan Luo, Jundong Li, Huan Liu, and Qinghua Zheng. Anomalous: A joint modeling approach for anomaly detection on attributed networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3513–3519. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[27] Changsheng Li, Xiangfeng Wang, Weishan Dong, Junchi Yan, Qingshan Liu, and Hongyuan Zha. Joint active learning with feature selection via cur matrix decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(6):1382–1396, 2019.

[28] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[29] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.