Mean Imputation and Stochastic Coordinate Descent for Linear Systems with Missing Data

Meha Patel*, Samuel Rath†, and Chupeng Zheng‡

Projet Advisor: Anna Ma§

Abstract. As big data problems become more prevalent, the need to accurately approximate solutions to large-scale linear systems increases. Many real-world big data problems are also accompanied by the risk of missing or incomplete data, further complicating the linear models assigned to them. Current methods to address missing data involve deletion or zero imputation, which introduces bias to the model. We propose a model that adapts Stochastic Coordinate Descent (SCD) to handle missing data in linear systems and utilizes μ-imputation to retrieve a better approximation of the original data. We prove that in expectation, our proposed algorithm, μ-imputation mSCD utilizes an unbiased estimator of the gradient of the least-squares objective function when using mean imputation in the absence of data. Furthermore, we compare our algorithm's performance on synthetic data to closely related algorithms: zero-imputation mSCD and SCD. Finally, we apply μ-imputation mSCD on real-world data to demonstrate the usefulness and viability of our proposed algorithm.

1. Introduction.

Large-scale data is crucial in training algorithms that are being implemented and utilized today. However, the data collection process to train such algorithms is often imperfect and can lead to noisy and incomplete data. For example, malfunctions in physical measurement devices can cause data to become corrupt or, in extreme instances, unavailable. As another example, a person may skip questions on a questionnaire to save time resulting in incomplete survey data. While a straightforward approach for dealing with missing data is to impute the missing data with zeros or ignore missing data altogether (throwing out any and all incomplete data points), this quickly becomes impractical and inefficient.

Other methods for solving missing data problems consider imputation methods in which a new data set is created using information from the available data set. For example, in [6], Mukhopadhyay and Mukherjee propose an algorithm for imputing incomplete streaming data using a constant factor times the possibly imputed data at the previous time instance. While this method eliminates any inefficiency of solving a matrix with missing entries and provides a more accurate estimation of the full matrix, it is not independent of data being missing at random. Mukhopadhyay and Mukherjee assume that there is a previous entry from which we can draw information to determine the following entry. When we face the problem of incomplete data sets, we often find that the data is missing independently at random. This work focuses on missing data and how it arises in linear systems. In particular, we consider the following linear system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is the measurement matrix that is not entirely known, i.e. only some of its entries are known, and we consider the over-determined case in which $m \geq n$, $\boldsymbol{x} \in \mathbb{R}^n$ is the unknown signal we wish to find, and $\boldsymbol{y} \in \mathbb{R}^m$ are the measurements. In most general cases, this system can be solved as $\boldsymbol{x} = \boldsymbol{A}^\dagger \boldsymbol{y}$,

*California State University Long Beach, Long Beach, CA (meha.patel01@student.csulb.edu)

†San Diego State University, San Diego, CA (sarath@sdsu.edu)

‡University of Chicago, Chicago, IL (chupenz@uchicago.edu)

§University of California, Irvine, Irvine, CA (anna.ma@uci.edu)

where $\boldsymbol{A}^\dagger$ denotes the Moore-Penrose pseudo-inverse of $\boldsymbol{A}$. However, this method can become computationally expensive for large-scale linear systems, i.e., when m and n become very large. In this regime, one may not even be able to load the entire matrix $\boldsymbol{A}$ into working memory, let alone compute its pseudo-inverse to approximate the solution to the linear system.

Handling large-scale data sets may be impossible if the computer memory is limited, so stochastic iterative methods with low memory footprint have become a popular choice for solving large-scale linear systems. Randomized Kaczmarz (RK) and Stochastic Gradient Descent (SGD) are examples of stochastic iterative methods that can be employed to solve linear systems. By adapting these methods for specialized settings, improved performance results can be empirically and theoretically verified. For example, in [3], Ma and Needell propose missing Stochastic Gradient Descent (mSGD), a stochastic iterative projection method used for solving large-scale data that is missing at random. Another example is in [7], where Needell shows that RK can be used to solve a linear system corrupted by noise.

Another example of a stochastic iterative method that works well on large-scale data is the Stochastic Coordinate Descent (SCD) method, also known as the Randomized Gauss-Seidel (RGS) method. When applied to linear systems, SCD requires processing one column of the matrix at a time, and thus is useful when we can only access the column-wise information. SCD minimizes a given objective function by moving the approximate solution along one coordinate direction at each iteration. In the linear system case, since only one column is required in every iteration, SCD does not consume much working memory. Other works that have examined the use of SCD to solve linear systems include [2], where Leventhal and Lewis show the expected linear convergence rate of a randomized coordinate descent algorithm, and [8], where Nesterov compares the convergence rates for both constrained and unconstrained versions of the randomized coordinate descent method and their efficiency estimates. It should also be noted that SCD can be applied to solve systems in the under-constrained case, when $m < n$. For example, in [4], Ma, Needell, and Ramdas, extend SCD to allow for convergence to the least squares norm solution. Much like SGD or RK, SCD can also be adapted to specialized settings, such as when data are missing at random.

In this work, we adapt SCD to solve the least squares problem by utilizing mean imputation when missing-ness in data can be modeled as independent and identically distributed (i.i.d.) Bernoulli random variables. We show that the iterates of the proposed adaptation to SCD, named μ-imputed mSCD, move in the direction of the least squares gradient in expectation. To demonstrate the efficacy of our algorithm, we perform numerical experiments on synthetic and real-world data.

2. Background.

We examine two stochastic iterative methods that can be used to solve linear systems. Both methods are advantageous to use in the large-scale setting as they only require rows or columns of the matrix in each iteration.

Before we begin, we briefly discuss the notation adopted in this manuscript. We will denote matrices by bold capital letters and vectors by bold lowercase letters. We let $m \in \mathbb{N}$ and $n \in \mathbb{N}$ be the number of rows and columns for the data matrix $\boldsymbol{A}$ respectively. We let $\mathbb{1}_{m\times n}$ be an $m \times n$ ones matrix. The missing data matrix will be denoted as $\tilde{\boldsymbol{A}}$. We let $\boldsymbol{A}_{:j}$ denote the j^{th} column and $\boldsymbol{A}_{i:}$ denote the i^{th} row of matrix $\boldsymbol{A}$. Let $||\cdot||_F$ denote the Frobenius norm and $diag(\boldsymbol{B}) \in \mathbb{R}^{n\times n}$ denote the diagonal matrix containing the diagonal of $\boldsymbol{B}$.

2.1. Stochastic Gradient Descent (SGD). The SGD algorithm is a popular method for solving optimization problems. In its most general form, the algorithm minimizes an objective function $F(\boldsymbol{x}) = \sum_i f_i(\boldsymbol{x})$ via the iterative procedure

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f_i(\boldsymbol{x}_k),$$

where α_k is the step size at the k^{th} iteration and $\nabla f_i(\boldsymbol{x}_k)$ is the gradient of $f_i(\boldsymbol{x}_k)$. Intuitively, SGD uses $f_i(\boldsymbol{x}_k)$ is an unbiased estimator of the gradient of the objective $\nabla F(\boldsymbol{x})$ and so on average moves iterates in the direction of descent towards the minimizer. When solving large-scale linear systems, SGD can be used to minimize the least-squares objective function

$$F(\boldsymbol{x}) = \frac{1}{2m}||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||^2 = \frac{1}{m}\sum_{i=1}^{m} f_i(\boldsymbol{x}),$$

where $f_i(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{A}_{i:}\boldsymbol{x} - y_i)^2$.Thus, at every iteration, this method selects the i^{th} row of $\boldsymbol{A}$ at random and computes $f_i(\boldsymbol{x})$ such that $\nabla f_i(\boldsymbol{x}_k) = \boldsymbol{A}^*_{i:}(\boldsymbol{A}_{i:}\boldsymbol{x}_k - y_i)$.This is ideal when $\boldsymbol{A}$ is extremely large and only the rows of the matrix can be accessed.

2.2. Stochastic Coordinate Descent (SCD). The Stochastic Coordinate Descent algorithm is another iterative method for solving optimization problems. In general, the SCD algorithm randomly selects a coordinate j and approximately minimizes an objective function $L(\boldsymbol{x})$ in that coordinate direction via the iterates:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \ell_j(\boldsymbol{x}_k), \tag{2.1}$$

where α_k is the step size at the k^{th} iteration. When solving linear systems using SCD, we wish to minimize the least squares objective

$$L(\boldsymbol{x}) = \frac{1}{2n}||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||^2,$$

whose gradient is given by

$$\nabla L(\boldsymbol{x}) = \sum_{j=1}^{n} \frac{1}{n}\ell_j(\boldsymbol{x})$$

where $\ell_j(\boldsymbol{x}) = (\boldsymbol{A}_{:j}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}))\boldsymbol{e}_j$, and $\boldsymbol{A}_{:j}$ is the j^{th} column of our matrix $\boldsymbol{A}$. Standard implementations of SCD use $\ell_j(\boldsymbol{x})$ as an unbiased estimator for the gradient of the objective function, i.e., $\mathbb{E}[\ell_j(\boldsymbol{x}_k)] = \nabla L(\boldsymbol{x})$. Because of the coordinate-wise updates on $\boldsymbol{x}_{k+1}$, one need not recompute the residual vector $\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}$ at every iteration. Instead, the residual vector can be tracked in each iteration and its update would only require the chosen column of $\boldsymbol{A}$. Thus, this algorithm will only require a single column of the matrix of A in each iteration.

3. Main Results. This section introduces the missing data model adopted in this work and the motivation for mean imputation. We also discuss a variation of the previously studied mSGD algorithm [3], which we identify in this paper as a zero imputation approach. In Section 3.1, we introduce our model and motivate it in Section 3.2. Section 3.3 outlines our proposed method and the theoretical justification for our method is provided in Section 3.4.

3.1. Missing Data Model. In this work, we assume that data are missing completely at random. This is the same assumption that was adopted in [3]. Such model assumptions can be satisfied by design: in a survey the surveyor can randomly select which questions to reveal to a participant, causing survey answers to be missing completely at random. In another

example, in an extremely large-scale setting where even entire rows or columns of a matrix cannot be loaded into memory, one can consider taking a random sample of entries in the row or column such that the sampling satisfies our model. For simulation purposes, we choose p to represent the probability of existing data in our sample set. However, in real applications, we assume that we have some knowledge regarding much data is already missing from our dataset. Using this probabilistic assumption of missing entries, we are able to calculate the least squares solution without any additional information regarding our data matrix $\boldsymbol{A}$. More concretely, we assume that the entries of $\boldsymbol{A}$ are missing with probability $1-p$, as done in [3]. We also similarly define a *binary mask* as:

Definition 3.1 (Binary Mask). *A binary mask is a matrix* $\boldsymbol{D} \in \{0,1\}^{m\times n}$ *such that*

$$\boldsymbol{D}_{ij} \overset{\text{iid}}{\sim} Bern(p),$$

where $p \in [0,1]$.

The binary mask in Definition 3.1 determines whether entries of $\boldsymbol{A}$ are missing. If $\boldsymbol{D}_{ij} = 1$ then $\boldsymbol{A}_{\boldsymbol{ij}}$ is not missing and if $\boldsymbol{D}_{ij} = 0$ then $\boldsymbol{A}_{ij}$ is missing. In other words, entries of $\boldsymbol{A}$ are missing with probability $1-p$ independently, and the matrix $\tilde{\boldsymbol{A}}$ where zeros are imputed to replace missing values, can be written as $\tilde{\boldsymbol{A}} = \boldsymbol{A} \odot \boldsymbol{D}$ where $\odot$ denotes the Hadamard (i.e., element wise) product.

3.2. Mean Imputation. The missing data model we consider here is the same model adopted in [3] where $\tilde{\boldsymbol{A}}$ is a matrix with zeros imputed where data is missing and data is assumed to be missing completely at random. If the original matrix has mean zero entries then this is a reasonable imputation, though that is not always the case. In addition, in some applications zero values can be meaningful. For example, in single-cell RNA sequences, zeros represent a lack of or low gene expression and thus imputing zeros into a matrix with missing data creates a misrepresentation of the data itself [1].

The choice of proper imputation value heavily depends on the application at hand. Without any additional information, a common choice for imputation is the mean value of the matrix (or feature). In fact, even when zero values are not meaningful in a data matrix, imputing empirical mean values from the given data provides a better approximation of the original matrix when compared to zero mean imputation, as shown in Lemma 3.5.

To allow for variable imputation, we define the μ-imputed matrix as follows:

Definition 3.2 (μ-Imputed Matrix). *Let* $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ *and* $\mu \in \mathbb{R}$. *Given binary mask* $\boldsymbol{D}$, *the* μ*-imputed matrix is defined as:*

$$\tilde{\boldsymbol{A}} = \boldsymbol{A} \odot \boldsymbol{D} + \mu(\mathbb{1}_{m\times n} - \boldsymbol{D}),$$

where $\odot$ *denotes the Hadamard product.*

Definition 3.2 presents the μ-imputed matrix $\tilde{\boldsymbol{A}}$ where the value μ is being imputed in all entries that are determined to be missing by $\boldsymbol{D}$. While Definition 3.2 imputes only a single value μ into the matrix $\tilde{\boldsymbol{A}}$, the definition can be generalized to allow for varying mean values across rows and/or columns. This can be accomplished by defining a mean matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ where $M_{ij} = \mu_{ij}$, and μ_{ij} is the appropriate imputation value of entry (i,j). Then, one can

define

$$\tilde{\boldsymbol{A}} = \boldsymbol{A} \odot \boldsymbol{D} + \boldsymbol{M} \odot (\mathbb{1}_{m\times n} - \boldsymbol{D}).$$

For the remainder of this work, we consider the setting in which the same value μ is imputed in all entries for simplicity, but this work easily extends to the more general case in which different values are imputed into the missing data matrix A.

3.3. μ-imputed mSCD Algorithm. In this work, we propose a variant of SCD for missing data such that the iterates utilize an unbiased estimator for the least squares gradient. Before we discuss our method, we introduce some additional notation. Let $\mathbb{1}_{m\times n}$ be an $m \times n$ ones matrix. Let $\mu \in \mathbb{R}$ be a fixed scalar, $\boldsymbol{\mu}$ be a vector containing only values of μ whose dimension will be specified when context is unclear, and $\boldsymbol{M} = \mu\mathbb{1}_{m\times n}$.

Suppose we apply SCD to the least squares objective using the $\mu-$imputed matrix $\tilde{\boldsymbol{A}}$ directly, and we let $\boldsymbol{\mu} = \mu\mathbb{1}_{m\times 1}$. In this case, we denote the objective function:

$$\tilde{L}(\boldsymbol{x}) = \frac{1}{2n}||\tilde{\boldsymbol{A}}\boldsymbol{x} - \boldsymbol{y}||^2,$$

and the SCD iterate (2.1) is

$$\tilde{\ell}_j(\boldsymbol{x}) = (\tilde{\boldsymbol{A}}_{:j}^T(\tilde{\boldsymbol{A}}\boldsymbol{x} - \boldsymbol{y}))\boldsymbol{e}_j.$$

If we take the expectation of $\tilde{\ell}_j(\boldsymbol{x})$ with respect to the randomness from the binary mask and random choice of coordinate direction j, we find that $\tilde{\ell}_j(\boldsymbol{x})$ is no longer an unbiased estimator of the gradient of the objective. In particular, assuming that coordinate choice and binary mask are independent, we have:

$$\begin{aligned}
\mathbb{E}\left[\tilde{\ell}_j(\boldsymbol{x})\right] &= \mathbb{E}_j\mathbb{E}_\delta\left[\left(\tilde{\boldsymbol{A}}_{:j}^T\left(\tilde{\boldsymbol{A}}\boldsymbol{x} - \boldsymbol{y}\right)\right)\boldsymbol{e}_j\right] \\
&= \left(\frac{p}{n}\sum_j \left(\boldsymbol{A}_{:j}^T\left(p\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\right)\right)\boldsymbol{e}_j\right) + \left(\frac{p-p^2}{n}\sum_j\left(\boldsymbol{A}_{:j}^T\boldsymbol{A}_{:j}x_j\right)\boldsymbol{e}_j\right) \qquad (3.1)\\
&+ \left(\frac{1-p}{n}\sum_j\left(p\left(\boldsymbol{A}_{:j}^T\boldsymbol{M} + \boldsymbol{\mu}^T\boldsymbol{A}\right)\boldsymbol{x} + (1-p)\boldsymbol{\mu}^T\boldsymbol{M}\boldsymbol{x} - \boldsymbol{\mu}^T\boldsymbol{y}\right)\boldsymbol{e}_j\right) \qquad (3.2)\\
&+ \left(\frac{p-p^2}{n}\sum_j\left(\boldsymbol{\mu}^T\boldsymbol{\mu} - \boldsymbol{\mu}^T\boldsymbol{A}_{:j} - \boldsymbol{A}_{:j}^T\boldsymbol{\mu}x_j\right)\boldsymbol{e}_j\right) \qquad (3.3)\\
&\neq \frac{1}{n}\sum_j\left(\boldsymbol{A}_{:j}^T\left(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\right)\right)\boldsymbol{e}_j.
\end{aligned}$$

Here, we see that $\mathbb{E}\left[\tilde{\ell}_j(\boldsymbol{x})\right] \neq \frac{1}{n}\sum_j\left(\boldsymbol{A}_{:j}^T\left(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\right)\right)\boldsymbol{e}_j$. To impose this, we propose a proxy function $s_j(\boldsymbol{x})$ that rescales the first term of (3.1), removes the second term of (3.1) in expectation, and removes the terms (3.2) and (3.3) in expectation. The proposed function is:

$$s_j(\boldsymbol{x}) := (c_j(\boldsymbol{x}) - d_j(\boldsymbol{x}))\,\boldsymbol{e}_j, \qquad (3.4)$$

where

$$c_j(\boldsymbol{x}) := \frac{1}{p^2}\tilde{\boldsymbol{A}}_{:j}^T\left(\tilde{\boldsymbol{A}}\boldsymbol{x} - p\boldsymbol{y}\right) - \frac{1-p}{p^2}\tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}_{:j}x_j, \qquad (3.5)$$

and

$$d_j(\boldsymbol{x}) := \frac{1-p}{p^2}\left[\left(\tilde{\boldsymbol{A}}_{:j}^T\boldsymbol{M} + \boldsymbol{\mu}^T\tilde{\boldsymbol{A}} - (1-p)\boldsymbol{\mu}^T\boldsymbol{M}\right)\boldsymbol{x}\right.$$
$$\left. -p\boldsymbol{\mu}^T\boldsymbol{y} - \left(\tilde{\boldsymbol{A}}_{:j}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\tilde{\boldsymbol{A}}_{:j} - \boldsymbol{\mu}^T\boldsymbol{\mu}\right)x_j\right]. \tag{3.6}$$

Here, the term $c_j(\boldsymbol{x})$ plays the role of rescaling the first term and removing the second term in (3.1). The term $d_j(\boldsymbol{x})$ plays the role of removing (3.2) and (3.3) in expectation.

Algorithm 3.1 presents pseudocode for our proposed algorithm, which we refer to as the μ-imputed missing Stochastic Coordinate Descent Algorithm or μ-imputed mSCD. In Section 3.4 we provide a theoretical justification for our method.

Algorithm 3.1 μ-imputed mSCD

Input $(\tilde{\boldsymbol{A}}, \boldsymbol{y}, p, T, \mu, \alpha)$
Initialize $\boldsymbol{x}_0 = 0_{n\times 1}, \boldsymbol{M} = \mu\mathbb{1}_{m\times n}$
for $k = 0, 1, 2, ..., T$ **do**
  Pick $j \sim \{1, ..., n\}$ uniformly at random
  Compute $c_j(\boldsymbol{x}_k)$ as defined in (3.5)
  Compute $d_j(\boldsymbol{x}_k)$ as defined in (3.6)
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha(c_j(\boldsymbol{x}_k) - d_j(\boldsymbol{x}_k))\boldsymbol{e}_j$
end for
return $\boldsymbol{x}_{k+1}$ =0

3.4. Theoretical Justification. In this section, we provide a theoretical justification for the use of our proposed methods. Theorem 3.3 shows that the iterates utilized in Algorithm 3.1 are unbiased approximations of the gradient of the least squares objective function. In Theorem 3.4, we demonstrate that our approach can also be applied to derive a mean-imputed mSGD algorithm or μ-imputed missing Stochastic Gradient Descent (μ-mSGD) and show that along the same vain as Theorem 3.3, our proposed algorithm utilizes an unbiased estimate of the gradient of the least squares objective. Remark 1 considers the zero imputation case and illuminates the role of $d_j(\boldsymbol{x})$. Remark 2 considers the zero imputation case for μ-mSGD to demonstrate that the proposed method is a generalization of the mSGD algorithm. Lastly, Lemma 3.5 provides a justification for using the empirical mean value to impute into a matrix with missing data.

Detailed proofs of all theorems and remarks are provided in the Appendix to maintain the flow of the main text and allow interested readers to delve into the rigorous mathematical foundations of our methods.

Theorem 3.3 (μ-imputed mSCD in Expectation). *Let $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, $\mu \in \mathbb{R}$ be fixed , and $\boldsymbol{\mu} = \mu\mathbb{1}_{m\times 1}$. Furthermore, let $\tilde{\boldsymbol{A}}$ be a random matrix such that*

$$\tilde{a}_{ij} = \begin{cases} a_{ij} & w.p \quad p \\ \mu & w.p \quad 1-p. \end{cases}$$

Where "w.p." stands for "with probability". Define for each $j \in [n]$, where $[n]$ denotes the set $1, 2, \ldots, n$, Then,

$$s_j(\boldsymbol{x}) = (c_j(\boldsymbol{x}) - d_j(\boldsymbol{x}))\, \boldsymbol{e}_j$$

where

$$c_j(\boldsymbol{x}) = \frac{1}{p^2} \tilde{\boldsymbol{A}}_{:j}^T \left(\tilde{\boldsymbol{A}} \boldsymbol{x} - p \boldsymbol{y} \right) - \frac{1-p}{p^2} \|\tilde{\boldsymbol{A}}_{:j}\|^2 x_j,$$

and

$$\begin{aligned} d_j(\boldsymbol{x}) = & \frac{1-p}{p^2} \left[\left(\tilde{\boldsymbol{A}}_{:j}^T \boldsymbol{M} + \boldsymbol{\mu}^T \tilde{\boldsymbol{A}} - (1-p) \boldsymbol{\mu}^T \boldsymbol{M} \right) \boldsymbol{x} \right. \\ & \left. - p \boldsymbol{\mu}^T \boldsymbol{y} - \left(\tilde{\boldsymbol{A}}_{:j}^T \boldsymbol{\mu} + \boldsymbol{\mu}^T \tilde{\boldsymbol{A}}_{:j} - \boldsymbol{\mu}^T \boldsymbol{\mu} \right) x_j \right]. \end{aligned}$$

Then picking $j \in [n]$ uniformly at random, we have

$$\mathbb{E}\left[s_j(\boldsymbol{x}) \right] = \nabla L(\boldsymbol{x}),$$

where $L(\boldsymbol{x}) = \frac{1}{2n} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2$ and the expectation is taken over randomness in j and in $\tilde{\boldsymbol{A}}$.

It should be noted that Theorem 3.3 holds for all μ and thus for $\mu = 0$, we can conclude a similar result for a linear system with i.i.d. Bernoulli missing column entries of a matrix in which values of 0 are imputed for missing entries. Remark 1 shows that, in expectation, the iterates of our proposed algorithm utilize unbiased estimators of the gradient of the least squares objective function.

Remark 1 (mSCD with 0 imputation). *When $\mu = 0$, (3.6) is an all zeros vector and thus $s_j(\boldsymbol{x})$ in Theorem 3.3 simplifies to*

$$s_j(\boldsymbol{x}) = c_j(\boldsymbol{x}) \boldsymbol{e}_j = \left(\frac{1}{p^2} \tilde{\boldsymbol{A}}_{:j}^T \left(\tilde{\boldsymbol{A}} \boldsymbol{x} - p \boldsymbol{y} \right) - \frac{1-p}{p^2} \tilde{\boldsymbol{A}}_{:j}^T \tilde{\boldsymbol{A}}_{:j} x_j \right) \boldsymbol{e}_j.$$

Thus, we can interpret the function $s_j(\boldsymbol{x})$ as a combination of two terms: the term $c_j(\boldsymbol{x})$ is a biased estimator of $\nabla L(\boldsymbol{x})$ when $\mu \neq 0$ and the term $d_j(\boldsymbol{x})$ corrects for the bias introduced when utilizing a non-zero μ imputation.

In addition to proposing an algorithm for solving missing data linear systems that use columns of a matrix, we also utilize our techniques to derive an unbiased estimator that utilizes rows of the matrix, i.e., for a μ-imputed SGD algorithm. Theorem 3.4 presents an estimator for $\nabla L(\boldsymbol{x})$ where the estimates use only rows of the matrix $\tilde{\boldsymbol{A}}$, instead of columns of the matrix. Such an estimate can be incorporated into the SGD algorithm, as has been done for the $\mu = 0$ case in [3] (see Remark 2). Similar to Theorem 3.3, the estimate $t_i(\boldsymbol{x})$ shown in (3.7) is the sum of two components: one which approximates the gradient using a single row of $\tilde{\boldsymbol{A}}$ and the other which debiases the estimate when $\mu \neq 0$ (see Remark 1).

Theorem 3.4 (mSGD with μ imputation). *Let $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, $\mu \in \mathbb{R}$ be fixed, and $\boldsymbol{\mu} = \mu \mathbb{1}_{1 \times n}$. Let $\tilde{\boldsymbol{A}}$ be a random matrix such that*

$$\tilde{a}_{ij} = \begin{cases} a_{ij} & w.p \quad p \\ \mu & w.p \quad 1-p \end{cases}$$

For $i \in [m]$, define

$$t_i(\boldsymbol{x}) := g_i(\boldsymbol{x}) - h_i(\boldsymbol{x}), \tag{3.7}$$

where

$$g_i(\boldsymbol{x}) = \frac{1}{p^2}\tilde{\boldsymbol{A}}_{i:}^T\left(\tilde{\boldsymbol{A}}_{i:}\boldsymbol{x} - py_i\right) - \frac{1-p}{p^2}\mathrm{diag}\left(\tilde{\boldsymbol{A}}_{i:}^T\tilde{\boldsymbol{A}}_{i:}\right)\boldsymbol{x},$$

and

$$\begin{aligned} h_i(\boldsymbol{x}) =& \frac{1-p}{p^2}\left[\left(\tilde{\boldsymbol{A}}_{i:}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\tilde{\boldsymbol{A}}_{i:} - (1-p)\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x}\right. \\ & \left. -p\boldsymbol{\mu}^T y_i - \mathrm{diag}\left(\tilde{\boldsymbol{A}}_{i:}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\tilde{\boldsymbol{A}}_{i:} - \boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x}\right]. \end{aligned}$$

Then, selecting $i \in [m]$ uniformly at random,

$$\mathbb{E}\left[t_i(\boldsymbol{x})\right] = \nabla F(\boldsymbol{x}),$$

where the expectation is taken with respect to the choice of i and randomness in $\tilde{\boldsymbol{A}}$.

Remark 2. *When $\mu = 0$, the update function $t_i(\boldsymbol{x})$ in Theorem 3.4 reduces to*

$$g_i(\boldsymbol{x}) = \frac{1}{p^2}\tilde{\boldsymbol{A}}_{i:}^T\left(\tilde{\boldsymbol{A}}_{i:}\boldsymbol{x} - py_i\right) - \frac{1-p}{p^2}\mathrm{diag}\left(\tilde{\boldsymbol{A}}_{i:}^T\tilde{\boldsymbol{A}}_{i:}\right)\boldsymbol{x},$$

which is exactly the iterate update proposed for mSGD [3].

Although Theorem 3.3 and Theorem 3.4 hold for any μ, the choice of μ impacts how well the imputed matrix approximates the original matrix $\boldsymbol{A}$ and thus the approximation error. We argue that a natural choice for μ is the mean of the given entries the matrix $\tilde{\boldsymbol{A}}$. In particular, when the empirical average is non-zero, Lemma 3.5 shows that performing a μ-imputation produces a better approximation of the original matrix $\boldsymbol{A}$ than performing a 0 imputation. Practically, one can use a priori information about the mean of $\boldsymbol{A}$ or the empirical mean of the given entries in $\tilde{\boldsymbol{A}}$ to approximate μ.

Theorem 3.5 (Expectation of Frobenius Norm). *Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ be a random matrix with i.i.d entries such that $\mu = \frac{1}{mn}\sum_{ij} a_{ij}$. Let $\boldsymbol{D}$ be a binary mask with parameter p as defined in Definition 3.1. Denoting $\tilde{\boldsymbol{A}}_\mu = \boldsymbol{A}\odot\boldsymbol{D} + \mu(\mathbb{1}_{m\times n} - \boldsymbol{D})$ and $\tilde{\boldsymbol{A}}_0 = \boldsymbol{A}\odot\boldsymbol{D}$, we have that*

$$\mathbb{E}_\delta\left[\|\boldsymbol{A} - \tilde{\boldsymbol{A}}_0\|_F^2\right] \geq \mathbb{E}_\delta\left[\|\boldsymbol{A} - \tilde{\boldsymbol{A}}_\mu\|_F^2\right], \tag{3.8}$$

where $\mathbb{E}_\delta$ denotes the expectation with respect to randomness in $\boldsymbol{D}$.

4. Experiments. In this section, we conduct various numerical experiments to show the efficacy of our algorithm. In the first experiment, we compare SCD, mSCD and μ-imputed mSCD if the mean value of matrix $\boldsymbol{A}$ is nonzero. Specifically, vanilla SCD solves the matrix without adaptive terms, and the missing entries of the matrix are imputed by 0; mean-imputed SCD solves the matrix without adaptive terms, but the matrix is imputed by its mean value; mSCD has adaptive terms, but the matrix is imputed by 0; lastly, our μ-imputed mSCD algorithm considers both adaptive terms and the non-zero imputation for the matrix. In our second and third experiments, we demonstrate μ-imputed mSCD on matrix $\boldsymbol{A}$ with different mean values and different data missing-ness probabilities. Finally in our third experiment, we apply μ-imputed mSCD on real world data.

In Figure 1, we compare the performance of three versions of SCD: vanilla SCD, mSCD, and our main contribution μ-imputed mSCD. We show that μ-imputed mSCD outperforms mSCD when the mean value of matrix $\boldsymbol{A}$ is no longer 0. First, we randomly generated a 1000×200 matrix with mean value $\mu = 10$, missing probability $p = 0.9$, and fixed learning

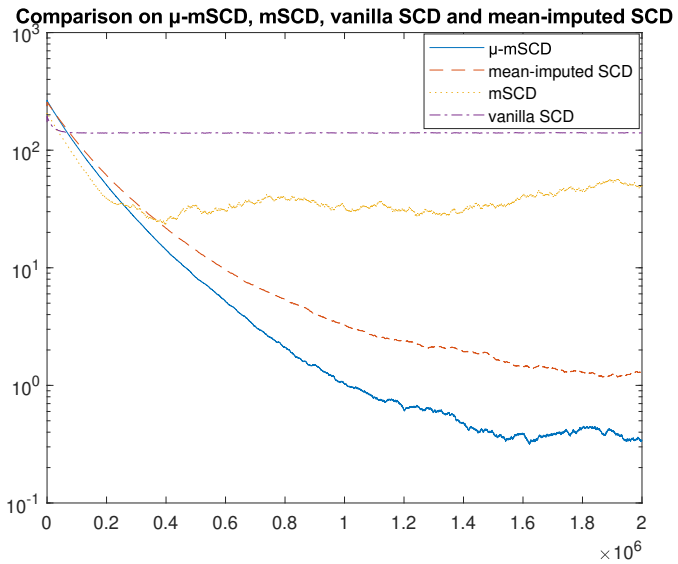

Figure 1. *Vanilla SCD, mean-imputed SCD, mSCD and μ-imputed mSCD on* 1000×200 *matrix, with* $\mu = 10$, $p = 0.9$, $\alpha = 10^{-6}$.

rate $\alpha = 10^{-6}$. For our tests, we ran SCD and mSCD on a zero-imputed matrix, and then SCD and μ-imputed mSCD on a mean-imputed matrix. We found that μ-imputed mSCD reaches a lower convergence horizon than mSCD, consistent with Lemma 3.5. Our μ-imputed mSCD also performs better than the simple mean-imputed SCD.

In Figure 2, we demonstrate that μ-imputed mSCD converges. By applying μ-imputed mSCD on 1000×200 matrix with mean value $\mu = 0, 5, 10$ separately, each of them successfully converges under learning rate $\alpha = 10^{-6}$ and missing probability $p = 0.9$.

Furthermore, in Figure 3, we ensure that μ-imputed mSCD converges under different missing data probabilities. Specifically, we generate a 1000×200 matrix with mean value $\mu = 40$, and then apply μ-imputed mSCD with fixed learning rate $\alpha = 10^{-7}$ and missing probability $p = 0.6, 0.8, 0.9$ respectively. Here we notice that the convergence rate is associated with missing probability. That is, if more data is missing, the convergence horizon will be reached sooner.

In addition to synthetic experiments, we also include an experiment on real world data. This data set was obtained from the UCI Machine Learning Repository [9]. This data set contains features relating to a garment factor's productivity including worker idle time, target productivity for the day, and the number of workers. We employ a subset of all possible features to train a linear model to predict worker productivity. In this experiment, we have $m = 1197$ rows and $n = 6$ columns where each row contains data pertaining to a specific day. The vector $\boldsymbol{y}$ is chosen to be the actual productivity of the workers for that day, as given in the data set.

It should be noted that the columns in the data set have significantly different mean values. The smallest mean value being 0.1504 and the largest being 15.0622. Due to the

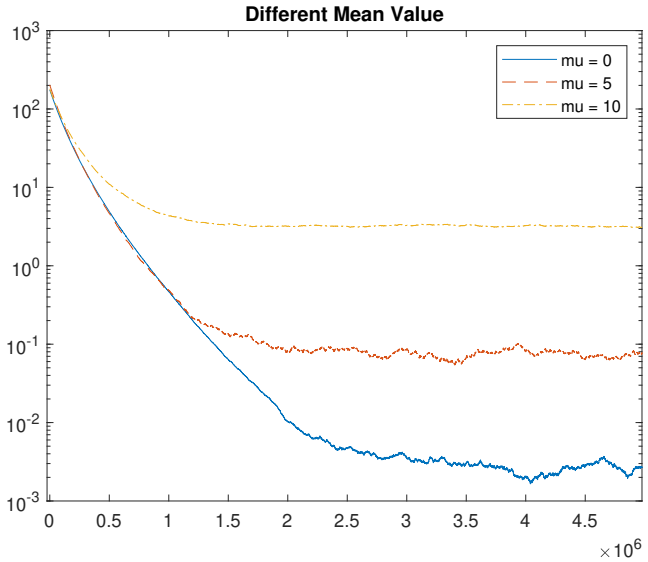

Figure 2. *μ-mSCD on 1000×200 matrix, with $\mu = 0, 5, 10$, $p = 0.9$, $\alpha = 10^{-6}$.*

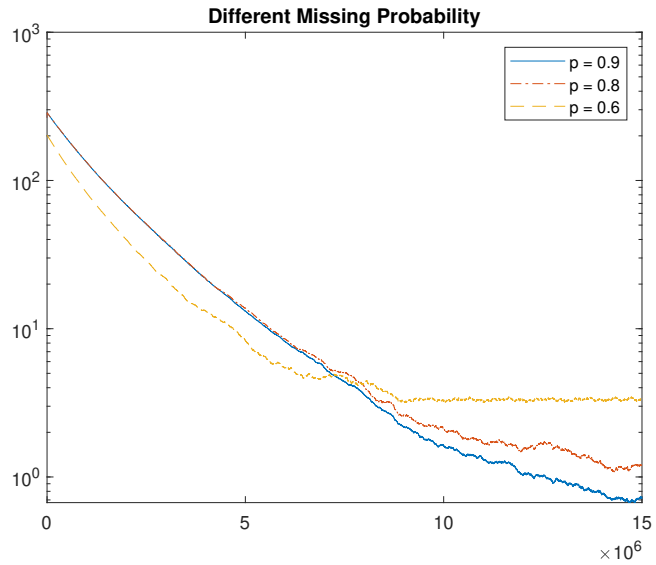

Figure 3. *μ-imputed mSCD on 1000×200 matrix, with $\mu = 40$, $p = 0.6, 0.8, 0.9$, $\alpha = 10^{-7}$.*

variation in means per feature, we employ the more general version of the proposed method which, instead of imputing a fixed mean μ, imputes the empirical column mean for each column. Practically, we would assume that this information is known a priori. The missingness is simulated in this data set by drawing a binary mask (3.1) at every iteration. Figure 4 presents the performance of our algorithm compared to a zero imputation averaged over 20

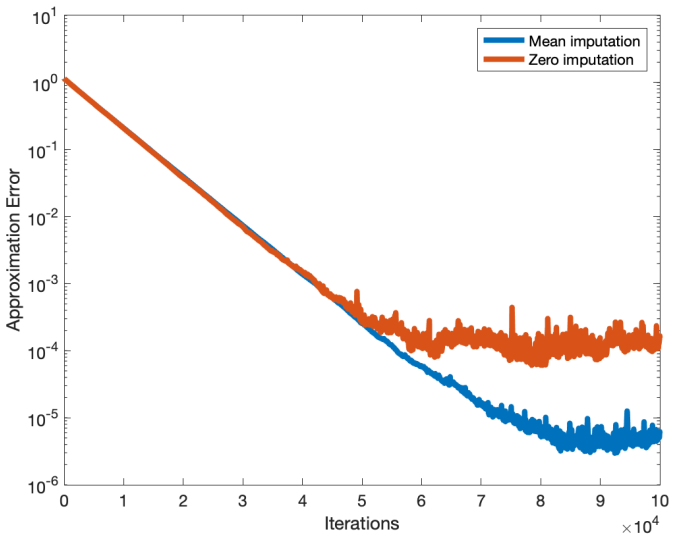

Figure 4. *Performance of μ-imputed mSCD on Garment Productivity data set from the UCI Machine Learning Repo.*

trials. We report the approximation error to the least squares solution for this data set using $\alpha = 5 \times 10^{-5}$ and $p = 0.85$. We observe that the approximation error decays linearly until the algorithm reaches a convergence horizon on the order of 10^{-5} for mean imputation and 10^{-4} for zero imputation, highlighting the benefit of using mean imputation.

5. Conclusion. We propose μ-imputed mSCD as a solution to big data problems in which missing or corrupt data issues arise. Our method utilizes an unbiased estimator of the gradient of the least-squares objective function as well as mean imputation to eliminate the issue of bias that may be introduced in the absence of data. The experimental results and theoretical proofs show that our method has less bias than current methods and, subsequently, performs better than these methods when solving large-scale linear systems with missing data. While we are able to show our algorithm's convergence capabilities through theoretical and practical experiments, we hope to analyze convergence guarantees in future work. Furthermore, for future work we believe our method can be expanded to other types of imputation methods or patterns of missing data. For example, we believe our idea of using mean imputation can be utilized to improve the cumulative information method used in [5] to create a cumulative information mean-imputed algorithm.

Acknowledgments. We thank Dr. Anna Ma from the University of California, Irvine for advising us and this work. We also thank Chelsea Huynh and Michael Strand from the University of California, Irvine for their contributions in facilitating the conversations around our research topic.

Appendix. In this section we prove the major theorem of the paper, namely Theorem 3.3. To do this, we first introduce some convenient notation, π and Π, to simplify the calculation.

This notation will also be used for the proof of Theorem 3.4, and in fact one of our goals is that it conveys the essential parallels between these proofs. Finally, we also prove Theorem 3.5.

By the assumption of our model, we treat the entries of a given matrix $\tilde{\boldsymbol{A}}$ as i.i.d. Bernoulli random variables such that

$$\tilde{a}_{ij} = \begin{cases} a_{ij} & w.p. \quad p \\ \mu & w.p. \quad 1-p \end{cases},$$

for which the expected value is simply

$$\mathbb{E}_\delta\left[\tilde{a}_{ij}\right] = pa_{ij} + (1-p)\mu. \tag{5.1}$$

However, when computing expectations we often run into products of 2 entries. For these, we need to consider 2 cases:

$$\tilde{a}_{ij}\tilde{a}_{ik} = \begin{cases} a_{ij}a_{ik} & w.p. \quad p^2 \\ \mu a_{ij} & w.p. \quad p(1-p) \\ \mu a_{ik} & w.p. \quad p(1-p) \\ \mu^2 & w.p. \quad (1-p)^2 \end{cases},$$

when $j \neq k$, and

$$\tilde{a}_{ij}\tilde{a}_{ik} = \begin{cases} a_{ij}a_{ik} & w.p. \quad p \\ \mu^2 & w.p. \quad 1-p \end{cases},$$

when $j = k$. These come from the fact that when an entry is multiplied by itself, it still corresponds to one coin toss with 2 outcomes. On the other hand, when the entries are distinct, we now have 2 independent coin tosses and 4 outcomes total. Thus, we get the following expectation:

$$\mathbb{E}_\delta\left[\tilde{a}_{ij}\tilde{a}_{ik}\right] = \begin{cases} p^2 a_{ij}a_{ik} + p(1-p)\left(a_{ij}\mu + \mu a_{ik}\right) + (1-p)^2\mu^2 & j \neq k \\ pa_{ij}a_{ik} + (1-p)\mu^2 & j = k \end{cases}. \tag{5.2}$$

The 2 cases in (5.2) occur so often in our proofs that we give them labels to condense the notation. These labels will also serve to indicate how the expected value operator $\mathbb{E}_\delta[\cdot]$ *acts* on different components in our calculations, depending on whether there are products of repeated or distinct entries. Let us define the functions $\Pi : \mathbb{R}^{m\times d} \times \mathbb{R}^{d\times n} \to \mathbb{R}^{m\times n}$ and $\pi : \mathbb{R}^{m\times d} \times \mathbb{R}^{d\times n} \to \mathbb{R}^{m\times n}$ such that

$$\Pi\left[\boldsymbol{A}, \boldsymbol{B}\right] := p^2\boldsymbol{AB} + p(1-p)\left(\boldsymbol{AM}_B + \boldsymbol{M}_A\boldsymbol{B}\right) + (1-p)^2\boldsymbol{M}_A\boldsymbol{M}_B \tag{5.3}$$

$$\pi\left[\boldsymbol{A}, \boldsymbol{B}\right] := p\boldsymbol{AB} + (1-p)\boldsymbol{M}_A\boldsymbol{M}_B, \tag{5.4}$$

where $\boldsymbol{M}_A = \mu\mathbb{1}_{m\times d}$ and $\boldsymbol{M}_B = \mu\mathbb{1}_{d\times n}$, and whose dimensions are that of $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively. Hence, using (5.3) and (5.4), we can now express (5.2) as

$$\mathbb{E}_\delta\left[\tilde{a}_{ij}\tilde{a}_{ik}\right] = \begin{cases} \Pi\left[a_{ij}, a_{ik}\right] & j \neq k \\ \pi\left[a_{ij}, a_{ik}\right] & j = k \end{cases}. \tag{5.5}$$

In addition to condensing the expectation, the functions $\Pi[\cdot,\cdot]$ and $\pi[\cdot,\cdot]$ have the added benefit of extending to higher dimensional expectations. For instance, suppose we need to take an expectation of the inner product of columns:

$$\begin{aligned}\mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}_{:k}\right] &= \mathbb{E}_\delta\left[\tilde{a}_{1j}\tilde{a}_{1k}+\tilde{a}_{2j}\tilde{a}_{2k}+\ldots+\tilde{a}_{mj}\tilde{a}_{mk}\right]\\ &= \mathbb{E}_\delta\left[\tilde{a}_{1j}\tilde{a}_{1k}\right]+\mathbb{E}_\delta\left[\tilde{a}_{2j}\tilde{a}_{2k}\right]+\ldots+\mathbb{E}_\delta\left[\tilde{a}_{mj}\tilde{a}_{mk}\right]\\ &= \sum_{i=1}^m \mathbb{E}_\delta\left[\tilde{a}_{ij}\tilde{a}_{ik}\right]\end{aligned}$$

which by (5.2) is

$$\begin{aligned}&= \sum_{i=1}^m\left[p^2a_{ij}a_{ik}+p(1-p)\left(a_{ij}\mu+\mu a_{ik}\right)+(1-p)^2\mu^2\right]\\ &= p^2\sum_{i=1}^m a_{ij}a_{ik}+p(1-p)\left(\sum_{i=1}^m a_{ij}\mu+\sum_{i=1}^m\mu a_{ik}\right)+(1-p)^2\sum_{i=1}^m\mu^2\\ &= p^2\boldsymbol{A}_{:j}^T\boldsymbol{A}_{:k}+p(1-p)\left(\boldsymbol{A}_{:j}^T\boldsymbol{\mu}+\boldsymbol{\mu}^T\boldsymbol{A}_{:k}\right)+(1-p)^2\boldsymbol{\mu}^T\boldsymbol{\mu}\end{aligned}$$

when $j\neq k$, and

$$\begin{aligned}&= \sum_{i=1}^m\left[pa_{ij}a_{ik}+(1-p)\mu^2\right]\\ &= p\sum_{i=1}^m a_{ij}a_{ik}+(1-p)\sum_{i=1}^m\mu^2\\ &= p\boldsymbol{A}_{:j}^T\boldsymbol{A}_{:k}+(1-p)\boldsymbol{\mu}^T\boldsymbol{\mu}\end{aligned}$$

when $j=k$. But these are exactly $\Pi\left[\boldsymbol{A}_{:j}^T,\boldsymbol{A}_{:k}\right]$ and $\pi\left[\boldsymbol{A}_{:j}^T,\boldsymbol{A}_{:k}\right]$, respectively.

5.1. Proof of Theorem 3.3. Here we seek to show that $s_j(\boldsymbol{x})$ is an unbiased estimator for the gradient of our loss function, i.e. $\mathbb{E}[s_j(\boldsymbol{x})]=\nabla L(\boldsymbol{x})$. To do this we use the law of iterated expectation $\mathbb{E}[s_j(\boldsymbol{x})]=\mathbb{E}_j[\mathbb{E}_\delta[s_j(\boldsymbol{x})]]$, showing one at a time that for some function $\ell_j(\boldsymbol{x})$, $\mathbb{E}_\delta[s_j(\boldsymbol{x})]=\ell_j(\boldsymbol{x})$, while $\mathbb{E}_j[\ell_j(\boldsymbol{x})]=\nabla L(\boldsymbol{x})$.

Proof. Recall that $s_j(\boldsymbol{x})=\left(c_j(\boldsymbol{x})-d_j(\boldsymbol{x})\right)\boldsymbol{e}_j$ such that

$$c_j(\boldsymbol{x}):=\frac{1}{p^2}\tilde{\boldsymbol{A}}_{:j}^T\left(\tilde{\boldsymbol{A}}\boldsymbol{x}-p\boldsymbol{y}\right)-\frac{1-p}{p^2}\tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}_{:j}x_j,$$

and

$$\begin{aligned}d_j(\boldsymbol{x}):=&\frac{1-p}{p^2}\left[\left(\tilde{\boldsymbol{A}}_{:j}^T\boldsymbol{M}+\boldsymbol{\mu}^T\tilde{\boldsymbol{A}}-(1-p)\boldsymbol{\mu}^T\boldsymbol{M}\right)\boldsymbol{x}\right.\\ &\left.-p\boldsymbol{\mu}^T\boldsymbol{y}-\left(\tilde{\boldsymbol{A}}_{:j}^T\boldsymbol{\mu}+\boldsymbol{\mu}^T\tilde{\boldsymbol{A}}_{:j}-\boldsymbol{\mu}^T\boldsymbol{\mu}\right)x_j\right],\end{aligned}$$

where $c_j(\boldsymbol{x})$ addresses the missing entry without mean shift, while $d_j(\boldsymbol{x})$ accounts for mean

shift. Now, observe that

$$\mathbb{E}_\delta\left[s_j(\boldsymbol{x})\right] = \left(\mathbb{E}_\delta\left[c_j(\boldsymbol{x})\right] - \mathbb{E}_\delta\left[d_j(\boldsymbol{x})\right]\right)\boldsymbol{e}_j, \tag{5.6}$$

where the expectations of c_j and d_j will be functions of inner products of columns of $\boldsymbol{A}$. Using (5.3) and (5.4) that notation can be simplified. In particular, we can write

$$\begin{aligned}
\mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}\right] &= \mathbb{E}_\delta\left[\left(\tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}_{:1}, \tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}_{:2}, ..., \tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}_{:j}, ..., \tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}_{:n}\right)\right] \\
&= \left(\mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}_{:1}\right], \mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}_{:2}\right], ..., \mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}_{:j}\right], ..., \mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}_{:n}\right]\right) \\
&= \left(\Pi\left[\boldsymbol{A}_{:j}^T, \boldsymbol{A}_{:1}\right], \Pi\left[\boldsymbol{A}_{:j}^T, \boldsymbol{A}_{:2}\right], ..., \pi\left[\boldsymbol{A}_{:j}^T, \boldsymbol{A}_{:j}\right], ..., \Pi\left[\boldsymbol{A}_{:j}^T, \boldsymbol{A}_{:n}\right]\right) \\
&= \Pi\left[\boldsymbol{A}_{:j}^T, \boldsymbol{A}\right] - \Pi\left[\boldsymbol{A}_{:j}^T, \boldsymbol{A}_{:j}\right]\boldsymbol{e}_j^T + \pi\left[\boldsymbol{A}_{:j}^T, \boldsymbol{A}_{:j}\right]\boldsymbol{e}_j^T, \qquad (5.7)
\end{aligned}$$

Thus, using (5.7), we compute the expectation of c_j and d_j:

$$\begin{aligned}
\mathbb{E}_\delta\left[c_j(\boldsymbol{x})\right] &= \frac{1}{p^2}\mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}\right]\boldsymbol{x} - \frac{1}{p}\mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{:j}^T\right]\boldsymbol{y} - \frac{1-p}{p^2}\mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{:j}^T\tilde{\boldsymbol{A}}_{:j}\right]x_j \\
&= \frac{1}{p^2}\left(\Pi\left[\boldsymbol{A}_{:j}^T, \boldsymbol{A}\right]\boldsymbol{x} - \Pi\left[\boldsymbol{A}_{:j}^T, \boldsymbol{A}_{:j}\right]x_j + \pi\left[\boldsymbol{A}_{:j}^T, \boldsymbol{A}_{:j}\right]x_j\right) \\
&\quad - \frac{1}{p}\left(p\boldsymbol{A}_{:j}^T + (1-p)\boldsymbol{\mu}^T\right)\boldsymbol{y} - \frac{1-p}{p^2}\left(\pi\left[\boldsymbol{A}_{:j}^T, \boldsymbol{A}_{:j}\right]\right)x_j \\
&= \frac{1}{p^2}\left(p^2\boldsymbol{A}_{:j}^T\boldsymbol{A} + p(1-p)\left(\boldsymbol{A}_{:j}^T\boldsymbol{M} + \boldsymbol{\mu}^T\boldsymbol{A}\right) + (1-p)^2\boldsymbol{\mu}^T\boldsymbol{M}\right)\boldsymbol{x} \\
&\quad - \frac{1}{p^2}\left(p^2\boldsymbol{A}_{:j}^T\boldsymbol{A}_{:j} + p(1-p)\left(\boldsymbol{A}_{:j}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{A}_{:j}\right) + (1-p)^2\boldsymbol{\mu}^T\boldsymbol{\mu}\right)x_j \\
&\quad + \frac{1}{p^2}\left(p\boldsymbol{A}_{:j}^T\boldsymbol{A}_{:j} + (1-p)\boldsymbol{\mu}^T\boldsymbol{\mu}\right)x_j - \frac{1}{p}\left(p\boldsymbol{A}_{:j}^T + (1-p)\boldsymbol{\mu}^T\right)\boldsymbol{y} \\
&\quad - \frac{1-p}{p^2}\left(p\boldsymbol{A}_{:j}^T\boldsymbol{A}_{:j} + (1-p)\boldsymbol{\mu}^T\boldsymbol{\mu}\right)x_j \\
&= \left(\boldsymbol{A}_{:j}^T\boldsymbol{A} + \frac{1-p}{p}\left(\boldsymbol{A}_{:j}^T\boldsymbol{M} + \boldsymbol{\mu}^T\boldsymbol{A}\right) + \frac{(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{M}\right)\boldsymbol{x} \\
&\quad - \left(\boldsymbol{A}_{:j}^T\boldsymbol{A} + \frac{1-p}{p}\left(\boldsymbol{A}_{:j}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{A}_{:j}\right) + \frac{(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)x_j \\
&\quad + \left(\frac{1}{p}\boldsymbol{A}_{:j}^T\boldsymbol{A} + \frac{1-p}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)x_j - \left(\boldsymbol{A}_{:j}^T + \frac{1-p}{p}\boldsymbol{\mu}^T\right)\boldsymbol{y} \\
&\quad - \left(\frac{1-p}{p}\boldsymbol{A}_{:j}^T\boldsymbol{A} + \frac{(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)x_j \\
&= \boldsymbol{A}_{:j}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}_{:j}^T\boldsymbol{y} + \left(\frac{1-p}{p}\left(\boldsymbol{A}_{:j}^T\boldsymbol{M} + \boldsymbol{\mu}^T\boldsymbol{A}\right) + \frac{(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{M}\right)\boldsymbol{x} \qquad (5.8)
\end{aligned}$$

$$
\begin{aligned}
&\quad - \left(\frac{1-p}{p}\left(\boldsymbol{A}_{:j}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{A}_{:j}\right) + \frac{2(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right) x_j \\
&\quad + \left(\frac{1-p}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right) x_j - \frac{1-p}{p}\boldsymbol{\mu}^T\boldsymbol{y} \\
\mathbb{E}_\delta\left[d_j(\boldsymbol{x})\right] &= \frac{1-p}{p^2}\left[\left(\mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{:j}^T\boldsymbol{M}\right] + \mathbb{E}_\delta\left[\boldsymbol{\mu}^T\tilde{\boldsymbol{A}}\right] - (1-p)\boldsymbol{\mu}^T\boldsymbol{M}\right)\boldsymbol{x}\right. \\
&\quad \left. -p\boldsymbol{\mu}^T\boldsymbol{y} - \left(\mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{:j}^T\boldsymbol{\mu}\right] + \mathbb{E}_\delta\left[\boldsymbol{\mu}^T\tilde{\boldsymbol{A}}_{:j}\right] - \boldsymbol{\mu}^T\boldsymbol{\mu}\right) x_j\right] \\
&= \frac{1-p}{p^2}\left[\left(\pi\left[\boldsymbol{A}_{:j}^T, \boldsymbol{M}\right] + \pi\left[\boldsymbol{\mu}^T, \boldsymbol{A}\right] - (1-p)\boldsymbol{\mu}^T\boldsymbol{M}\right)\boldsymbol{x}\right. \\
&\quad \left. -p\boldsymbol{\mu}^T\boldsymbol{y} - \left(\pi\left[\boldsymbol{A}_{:j}^T, \boldsymbol{\mu}\right] + \pi\left[\boldsymbol{\mu}^T, \boldsymbol{A}_{:j}\right] - \boldsymbol{\mu}^T\boldsymbol{\mu}\right) x_j\right] \\
&= \frac{1-p}{p^2}\left[\left(p\boldsymbol{A}_{:j}^T\boldsymbol{M} + p\boldsymbol{\mu}^T\boldsymbol{A} + 2(1-p)\boldsymbol{\mu}^T\boldsymbol{M} - (1-p)\boldsymbol{\mu}^T\boldsymbol{M}\right)\boldsymbol{x}\right. \\
&\quad \left. -p\boldsymbol{\mu}^T\boldsymbol{y} - \left(p\boldsymbol{A}_{:j}^T\boldsymbol{\mu} + p\boldsymbol{\mu}^T\boldsymbol{A}_{:j} + 2(1-p)\boldsymbol{\mu}^T\boldsymbol{\mu} - \boldsymbol{\mu}^T\boldsymbol{\mu}\right) x_j\right] \\
&= \left(\frac{1-p}{p}\boldsymbol{A}_{:j}^T\boldsymbol{M} + \frac{1-p}{p}\boldsymbol{\mu}^T\boldsymbol{A} + \frac{(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{M}\right)\boldsymbol{x} - \frac{1-p}{p}\boldsymbol{\mu}^T\boldsymbol{y} \\
&\quad - \left(\frac{1-p}{p}\boldsymbol{A}_{:j}^T\boldsymbol{\mu} - \frac{1-p}{p}\boldsymbol{\mu}^T\boldsymbol{A}_{:j} - \frac{(1-p)(1-2p)}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right) x_j \\
&= \left(\frac{1-p}{p}\left(\boldsymbol{A}_{:j}^T\boldsymbol{M} + \boldsymbol{\mu}^T\boldsymbol{A}\right) + \frac{(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{M}\right)\boldsymbol{x} \qquad (5.9) \\
&\quad - \left(\frac{1-p}{p}\left(\boldsymbol{A}_{:j}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{A}_{:j}\right) + \frac{2(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right) x_j \\
&\quad + \left(\frac{1-p}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right) x_j - \frac{1-p}{p}\boldsymbol{\mu}^T\boldsymbol{y}
\end{aligned}
$$

Plugging (5.8) and (5.9) into (5.6), we get

$$
\mathbb{E}_\delta\left[s_j(\boldsymbol{x})\right] = \left(\mathbb{E}_\delta\left[c_j(\boldsymbol{x})\right] - \mathbb{E}_\delta\left[d_j(\boldsymbol{x})\right]\right)\boldsymbol{e}_j = \left(\boldsymbol{A}_{:j}^T\left(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\right)\right)\boldsymbol{e}_j, \tag{5.10}
$$

for which we refer to the right-hand side expression as "$\ell_j(\boldsymbol{x})$". Recalling our loss function $L(\boldsymbol{x}) = \frac{1}{2n}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2$, it can be shown after some expansions that

$$
\frac{\partial L(\boldsymbol{x})}{\partial x_j} = \frac{1}{n}\boldsymbol{A}_{:j}^T\left(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\right).
$$

Thus, when we go to compute the column-wise expectation of $\ell_j(\boldsymbol{x})$ (assuming columns of $\boldsymbol{A}$ are selected uniformly at random) we get

$$
\mathbb{E}_j[\ell_j(\boldsymbol{x})] = \sum_{j=1}^{n}\frac{1}{n}\ell_j(\boldsymbol{x}) = \sum_{\mathrm{j}=1}^{n}\frac{1}{n}\left(\boldsymbol{A}_{:j}^T\left(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\right)\right)\boldsymbol{e}_j = \sum_{\mathrm{j}=1}^{n}\frac{\partial L(\boldsymbol{x})}{\partial x_j}\boldsymbol{e}_j. \tag{5.11}
$$

Finally, combining (5.10) and (5.11) gives us the desired gradient,

$$
\mathbb{E}\left[s_j(\boldsymbol{x})\right] = \mathbb{E}_j\left[\mathbb{E}_\delta\left[s_j(\boldsymbol{x})\right]\right] = \mathbb{E}_j\left[\ell_j(\boldsymbol{x})\right] = \nabla L(\boldsymbol{x}).
$$

■

5.2. Proof of Theorem 3.4. Similar to the last proof, here we show that $t_i(\boldsymbol{x})$ is an unbiased estimator for the gradient of the loss function, $\mathbb{E}\left[t_i(\boldsymbol{x})\right] = \nabla F(\boldsymbol{x})$. Again, we use an iterated expectation $\mathbb{E}\left[t_i(\boldsymbol{x})\right] = \mathbb{E}_i\left[\mathbb{E}_\delta\left[t_i(\boldsymbol{x})\right]\right]$, showing first that for the function $f_i(\boldsymbol{x})$, $\mathbb{E}_\delta\left[t_i(\boldsymbol{x})\right] = \nabla f_i(\boldsymbol{x})$ holds, then finally that $\mathbb{E}_i\left[\nabla f_i(\boldsymbol{x})\right] = \nabla F(\boldsymbol{x})$.

Proof. Recall that $t_i(\boldsymbol{x}) = g_i(\boldsymbol{x}) - h_i(\boldsymbol{x})$ such that

$$g_i(\boldsymbol{x}) := \frac{1}{p^2}\tilde{\boldsymbol{A}}_{i:}^T\left(\tilde{\boldsymbol{A}}_{i:}^T\boldsymbol{x} - py_i\right) - \frac{1-p}{p^2}\mathrm{diag}\left(\tilde{\boldsymbol{A}}_{i:}^T\tilde{\boldsymbol{A}}_{i:}\right)\boldsymbol{x},$$

and

$$\begin{aligned} h_i(\boldsymbol{x}) := \frac{1-p}{p^2}\Big[&\left(\tilde{\boldsymbol{A}}_{i:}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\tilde{\boldsymbol{A}}_{i:} - (1-p)\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} \\ &-p\boldsymbol{\mu}^T y_i - \mathrm{diag}\left(\tilde{\boldsymbol{A}}_{i:}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\tilde{\boldsymbol{A}}_{i:} - \boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x}\Big], \end{aligned}$$

where $g_i(\boldsymbol{x})$ plays the analogous role to $c_j(\boldsymbol{x})$ in the previous proof (missing entry without mean shift), and $h_i(\boldsymbol{x})$ the role of $d_j(\boldsymbol{x})$ (mean shift). Now, observe that

$$\mathbb{E}_\delta\left[t_i(\boldsymbol{x})\right] = \mathbb{E}_\delta\left[g_i(\boldsymbol{x})\right] - \mathbb{E}_\delta\left[h_i(\boldsymbol{x})\right], \tag{5.12}$$

where the expectations of g_i and h_i will be functions of inner products of columns of $\boldsymbol{A}$. Using (5.3) and (5.4) that notation can be simplified. In particular, we can write

$$\begin{aligned} \mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{i:}^T\tilde{\boldsymbol{A}}_{i:}\right] &= \mathbb{E}_\delta\left[\begin{pmatrix} \tilde{a}_{i1}\tilde{a}_{i1}, & \tilde{a}_{i1}\tilde{a}_{i2}, & \dots & \tilde{a}_{i1}\tilde{a}_{in} \\ \tilde{a}_{i2}\tilde{a}_{i1}, & \tilde{a}_{i2}\tilde{a}_{i2}, & \dots & \tilde{a}_{i2}\tilde{a}_{in} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{a}_{in}\tilde{a}_{i1}, & \tilde{a}_{in}\tilde{a}_{i2}, & \dots & \tilde{a}_{in}\tilde{a}_{in} \end{pmatrix}\right] \\ &= \begin{pmatrix} \mathbb{E}_\delta\left[\tilde{a}_{i1}\tilde{a}_{i1}\right], & \mathbb{E}_\delta\left[\tilde{a}_{i1}\tilde{a}_{i2}\right], & \dots & \mathbb{E}_\delta\left[\tilde{a}_{i1}\tilde{a}_{in}\right] \\ \mathbb{E}_\delta\left[\tilde{a}_{i2}\tilde{a}_{i1}\right], & \mathbb{E}_\delta\left[\tilde{a}_{i2}\tilde{a}_{i2}\right], & \dots & \mathbb{E}_\delta\left[\tilde{a}_{i2}\tilde{a}_{in}\right] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}_\delta\left[\tilde{a}_{in}\tilde{a}_{i1}\right], & \mathbb{E}_\delta\left[\tilde{a}_{in}\tilde{a}_{i2}\right], & \dots & \mathbb{E}_\delta\left[\tilde{a}_{in}\tilde{a}_{in}\right] \end{pmatrix} \\ &= \begin{pmatrix} \pi\left[a_{i1}, a_{i1}\right], & \Pi\left[a_{i1}, a_{i2}\right], & \dots & \Pi\left[a_{i1}, a_{in}\right] \\ \Pi\left[a_{i2}, a_{i1}\right], & \pi\left[a_{i2}, a_{i2}\right], & \dots & \Pi\left[a_{i2}, a_{in}\right] \\ \vdots & \vdots & \ddots & \vdots \\ \Pi\left[a_{in}, a_{i1}\right], & \Pi\left[a_{in}, a_{i2}\right], & \dots & \pi\left[a_{in}, a_{in}\right] \end{pmatrix} \\ &= \Pi\left[\boldsymbol{A}_{i:}^T, \boldsymbol{A}_{i:}\right] - \mathrm{diag}\left(\Pi\left[\boldsymbol{A}_{i:}^T, \boldsymbol{A}_{i:}\right]\right) + \mathrm{diag}\left(\pi\left[\boldsymbol{A}_{i:}^T, \boldsymbol{A}_{i:}\right]\right). \end{aligned} \tag{5.13}$$

Thus, taking each expectation separately and using (5.13), we get:

$$
\begin{aligned}
\mathbb{E}_\delta\left[g_i(\boldsymbol{x})\right] &= \frac{1}{p^2}\mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{i:}^T\tilde{\boldsymbol{A}}_{i:}\right]\boldsymbol{x} - \frac{1}{p}\mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{i:}^T\right]y_i - \frac{1-p}{p^2}\mathrm{diag}\left(\mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{i:}^T\tilde{\boldsymbol{A}}_{i:}\right]\right)\boldsymbol{x} \\
&= \frac{1}{p^2}\left(\Pi\left[\boldsymbol{A}_{i:}^T,\boldsymbol{A}_{i:}\right] - \mathrm{diag}\left(\Pi\left[\boldsymbol{A}_{i:}^T,\boldsymbol{A}_{i:}\right]\right) + \mathrm{diag}\left(\pi\left[\boldsymbol{A}_{i:}^T,\boldsymbol{A}_{i:}\right]\right)\right)\boldsymbol{x} \\
&\quad - \frac{1}{p}\left(p\boldsymbol{A}_{i:}^T + (1-p)\boldsymbol{\mu}^T\right)y_i - \frac{1-p}{p^2}\left(\mathrm{diag}\left(\pi\left[\boldsymbol{A}_{i:}^T,\boldsymbol{A}_{i:}\right]\right)\right)\boldsymbol{x} \\
&= \frac{1}{p^2}\left(p^2\boldsymbol{A}_{i:}^T\boldsymbol{A}_{i:} + p(1-p)\left(\boldsymbol{A}_{i:}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{A}_{i:}\right) + (1-p)^2\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} \\
&\quad - \frac{1}{p^2}\mathrm{diag}\left(p^2\boldsymbol{A}_{i:}^T\boldsymbol{A}_{i:} + p(1-p)\left(\boldsymbol{A}_{i:}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{A}_{i:}\right) + (1-p)^2\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} \\
&\quad + \frac{1}{p^2}\mathrm{diag}\left(p\boldsymbol{A}_{i:}^T\boldsymbol{A}_{i:} + (1-p)\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} - \frac{1}{p}\left(p\boldsymbol{A}_{i:}^T + (1-p)\boldsymbol{\mu}^T\right)y_i \\
&\quad - \frac{1-p}{p^2}\mathrm{diag}\left(p\boldsymbol{A}_{i:}^T\boldsymbol{A}_{i:} + (1-p)\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} \\
&= \left(\boldsymbol{A}_{i:}^T\boldsymbol{A}_{i:} + \frac{1-p}{p}\left(\boldsymbol{A}_{i:}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{A}_{i:}\right) + \frac{(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} \\
&\quad - \mathrm{diag}\left(\boldsymbol{A}_{i:}^T\boldsymbol{A}_{i:} + \frac{1-p}{p}\left(\boldsymbol{A}_{i:}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{A}_{i:}\right) + \frac{(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} \\
&\quad + \mathrm{diag}\left(\frac{1}{p}\boldsymbol{A}_{i:}^T\boldsymbol{A}_{i:} + \frac{1-p}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} - \left(\boldsymbol{A}_{i:}^T + \frac{1-p}{p}\boldsymbol{\mu}^T\right)y_i \\
&\quad - \mathrm{diag}\left(\frac{1-p}{p}\boldsymbol{A}_{i:}^T\boldsymbol{A}_{i:} + \frac{(1-p)^2}{p^2}\ \boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} \\
&= \boldsymbol{A}_{i:}^T\boldsymbol{A}_{i:}\boldsymbol{x} - \boldsymbol{A}_{i:}^Ty_i + \left(\frac{1-p}{p}\left(\boldsymbol{A}_{i:}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{A}_{i:}\right) + \frac{(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} \\
&\quad - \mathrm{diag}\left(\frac{1-p}{p}\left(\boldsymbol{A}_{i:}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{A}_{i:}\right) + \frac{2(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} \\
&\quad + \mathrm{diag}\left(\frac{1-p}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} - \frac{1-p}{p}\boldsymbol{\mu}^Ty_i,
\end{aligned}
\tag{5.14}
$$

and

$$
\begin{aligned}
\mathbb{E}_\delta\left[h_i(\boldsymbol{x})\right] &= \frac{1-p}{p^2}\left[\left(\mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{i:}^T\boldsymbol{\mu}\right] + \mathbb{E}_\delta\left[\boldsymbol{\mu}^T\tilde{\boldsymbol{A}}_{i:}\right] - (1-p)\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x}\right. \\
&\qquad \left. -p\boldsymbol{\mu}^Ty_i - \mathrm{diag}\left(\mathbb{E}_\delta\left[\tilde{\boldsymbol{A}}_{i:}^T\boldsymbol{\mu}\right] + \mathbb{E}_\delta\left[\boldsymbol{\mu}^T\tilde{\boldsymbol{A}}_{i:}\right] - \boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x}\right] \\
&= \frac{1-p}{p^2}\left[\left(\pi\left[\tilde{\boldsymbol{A}}_{i:}^T\boldsymbol{\mu}\right] + \pi\left[\boldsymbol{\mu}^T\tilde{\boldsymbol{A}}_{i:}\right] - (1-p)\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x}\right. \\
&\qquad \left. -p\boldsymbol{\mu}^Ty_i - \mathrm{diag}\left(\pi\left[\tilde{\boldsymbol{A}}_{i:}^T\boldsymbol{\mu}\right] + \pi\left[\boldsymbol{\mu}^T\tilde{\boldsymbol{A}}_{i:}\right] - \boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x}\right] \\
&= \frac{1-p}{p^2}\left[\left(p\boldsymbol{A}_{i:}^T\boldsymbol{\mu} + p\boldsymbol{\mu}^T\boldsymbol{A}_{i:} + 2(1-p)\boldsymbol{\mu}^T\boldsymbol{\mu} - (1-p)\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x}\right.
\end{aligned}
$$

$$
\begin{aligned}
&\quad -p\boldsymbol{\mu}^T y_i - \operatorname{diag}\left(p\boldsymbol{A}_{i:}^T\boldsymbol{\mu} + p\boldsymbol{\mu}^T\boldsymbol{A}_{i:} + 2(1-p)\boldsymbol{\mu}^T\boldsymbol{\mu} - \boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x}\Big] \\
&= \left(\frac{1-p}{p}\boldsymbol{A}_{i:}^T\boldsymbol{\mu} + \frac{1-p}{p}\boldsymbol{\mu}^T\boldsymbol{A}_{i:} + \frac{(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} - \frac{1-p}{p}\boldsymbol{\mu}^T y_i \\
&\quad - \operatorname{diag}\left(\frac{1-p}{p}\boldsymbol{A}_{i:}^T\boldsymbol{\mu} - \frac{1-p}{p}\boldsymbol{\mu}^T\boldsymbol{A}_{i:} - \frac{(1-p)(1-2p)}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} \\
&= \left(\frac{1-p}{p}\left(\boldsymbol{A}_{i:}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{A}_{i:}\right) + \frac{(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} \\
&\quad - \operatorname{diag}\left(\frac{1-p}{p}\left(\boldsymbol{A}_{i:}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{A}_{i:}\right) + \frac{2(1-p)^2}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} \\
&\quad + \operatorname{diag}\left(\frac{1-p}{p^2}\boldsymbol{\mu}^T\boldsymbol{\mu}\right)\boldsymbol{x} - \frac{1-p}{p}\boldsymbol{\mu}^T y_i.
\end{aligned} \tag{5.15}
$$

Plugging (5.14) and (5.15) into (5.12), we get

$$
\mathbb{E}_\delta\left[t_i(\boldsymbol{x})\right] = \mathbb{E}_\delta\left[g_i(\boldsymbol{x})\right] - \mathbb{E}_\delta\left[h_i(\boldsymbol{x})\right] = \boldsymbol{A}_{i:}^T\left(\boldsymbol{A}_{i:}\boldsymbol{x} - y_i\right). \tag{5.16}
$$

Now, recall our objective function $F(\boldsymbol{x}) = \frac{1}{2m}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2$, which can be rewritten as $F(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^m f_i(\boldsymbol{x})$, where $f_i(\boldsymbol{x}) = \frac{1}{2}\left(\boldsymbol{A}_{i:}\boldsymbol{x} - y_i\right)^2$. After some expansions, it is not too difficult to show that

$$
\nabla f_i(\boldsymbol{x}) = \boldsymbol{A}_{i:}^T\left(\boldsymbol{A}_{i:}\boldsymbol{x} - y_i\right), \tag{5.17}
$$

and notice this is exactly the right-hand side of (5.16). Thus, assuming rows i are uniformly selected from $[m]$, we take the row-wise expectation of $\nabla f_i(\boldsymbol{x})$ to get

$$
\mathbb{E}_i\left[\nabla f_i(\boldsymbol{x})\right] = \sum_{i=1}^m \frac{1}{m}\nabla f_i(\boldsymbol{x}) = \nabla\left[\frac{1}{m}\sum_{i=1}^m f_i(\boldsymbol{x})\right] = \nabla F(\boldsymbol{x}). \tag{5.18}
$$

Finally, combining (5.16), (5.17), and (5.18), we get

$$
\mathbb{E}\left[t_i(\boldsymbol{x})\right] = \mathbb{E}_i\left[\mathbb{E}_\delta\left[t_i(\boldsymbol{x})\right]\right] = \mathbb{E}_i\left[\nabla f_i(\boldsymbol{x})\right] = \nabla F(\boldsymbol{x}).
$$

the desired gradient. ∎

5.3. Proof of Theorem 3.5. In this proof, we first define the μ, then we consider the different cases, such as 0-imputation and μ-imputation for the matrix A. By calculating the expectation for both cases, $\mathbb{E}_\delta\|\boldsymbol{A} - \tilde{\boldsymbol{A}}_0\|_F^2 \geq \mathbb{E}_\delta\|\boldsymbol{A} - \tilde{\boldsymbol{A}}_\mu\|_F^2$ holds.

Proof. Given $\tilde{\boldsymbol{A}}_0$ is matrix $\boldsymbol{A}$ with missing entries $\tilde{a}_{0,ij}$ in which 0 is imputed and $\tilde{\boldsymbol{A}}_\mu$ is matrix $\boldsymbol{A}$ with entries $\tilde{a}_{\mu,ij}$ in which the mean value of $\boldsymbol{A}$, a fixed $\mu = \frac{1}{mn}\sum_{ij} a_{ij}$, is imputed, we want to find $\mathbb{E}_\delta\left[\|\boldsymbol{A} - \tilde{\boldsymbol{A}}_0\|_F^2\right]$ and $\mathbb{E}_\delta\left[\|\boldsymbol{A} - \tilde{\boldsymbol{A}}_\mu\|_F^2\right]$ and compare them.

We know,

$$
\tilde{a}_{0,ij} = \begin{cases} 0 & w.p \quad 1-p \\ a_{ij} & w.p \quad p \end{cases} \quad \Rightarrow \quad \tilde{a}_{0,ij}^2 = \begin{cases} 0 & w.p \quad 1-p \\ a_{ij}^2 & w.p \quad p \end{cases}
$$

$$\tilde{a}_{\mu,ij} = \begin{cases} \mu & w.p \quad 1-p \\ a_{ij} & w.p \quad p \end{cases} \quad \Rightarrow \quad \tilde{a}^2_{\mu,ij} = \begin{cases} \mu^2 & w.p \quad 1-p \\ a^2_{ij} & w.p \quad p \end{cases}$$

Therefore,

$$\begin{aligned}
\mathbb{E}_\delta \left[\|\boldsymbol{A} - \tilde{\boldsymbol{A}}_0\|_F^2 \right] &= \mathbb{E}_\delta \left[\sum_{i=1}^{m} \sum_{j=1}^{n} (a_{ij} - \tilde{a}_{0,ij})^2 \right] \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{E}_\delta \left[\left(a_{ij}^2 - 2a_{ij}\tilde{a}_{0,ij} + \tilde{a}_{0,ij}^2 \right) \right] \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{E}_\delta \left[a_{ij}^2 \right] + \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{E}_{\boldsymbol{A},\delta} \left[\tilde{a}_{0,ij}^2 \right] - \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{E}_\delta \left[2a_{ij}\tilde{a}_{0,ij} \right] \\
&= \|\boldsymbol{A}\|_F^2 + \sum_{i=1}^{m} \sum_{j=1}^{n} p a_{ij}^2 - \sum_{i=1}^{m} \sum_{j=1}^{n} 2p a_{ij}^2 \\
&= \|\boldsymbol{A}\|_F^2 - \sum_{i=1}^{m} \sum_{j=1}^{n} p a_{ij}^2 \\
&= \|\boldsymbol{A}\|_F^2 - p\|\boldsymbol{A}\|_F^2 = (1-p)\|\boldsymbol{A}\|_F^2
\end{aligned}$$

Similarly,

$$\begin{aligned}
\mathbb{E}_\delta \left[\|\boldsymbol{A} - \tilde{\boldsymbol{A}}_\mu\|_F^2 \right] &= \mathbb{E}_\delta \left[\sum_{i=1}^{m} \sum_{j=1}^{n} (a_{ij} - \tilde{a}_{\mu,ij})^2 \right] \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{E}_\delta \left[\left(a_{ij}^2 - 2a_{ij}\tilde{a}_{\mu,ij} + \tilde{a}_{\mu,ij}^2 \right) \right] \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{E}_\delta \left[a_{ij}^2 \right] + \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{E}_\delta \left[\tilde{a}_{\mu,ij}^2 \right] - \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{E}_\delta \left[2a_{ij}\tilde{a}_{\mu,ij} \right] \\
&= \|\boldsymbol{A}\|_F^2 + \sum_{i=1}^{m} \sum_{j=1}^{n} \left(p a_{ij}^2 + \mu^2(1-p) \right) - \sum_{i=1}^{m} \sum_{j=1}^{n} 2a_{ij} \left(p a_{ij} + \mu(1-p) \right) \\
&= \|\boldsymbol{A}\|_F^2 + \sum_{i=1}^{m} \sum_{j=1}^{n} p a_{ij}^2 + \sum_{i=1}^{m} \sum_{j=1}^{n} \mu^2(1-p) - 2\sum_{i=1}^{m} \sum_{j=1}^{n} p a_{ij}^2 - 2\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}\mu(1-p) \\
&= \|\boldsymbol{A}\|_F^2 - \sum_{i=1}^{m} \sum_{j=1}^{n} p a_{ij}^2 + \sum_{i=1}^{m} \sum_{j=1}^{n} \mu^2(1-p) - 2\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}\mu(1-p) \\
&= \|\boldsymbol{A}\|_F^2 - \sum_{i=1}^{m} \sum_{j=1}^{n} p a_{ij}^2 + \sum_{i=1}^{m} \sum_{j=1}^{n} \mu(1-p) \left(\mu - 2a_{ij} \right) \\
&= (1-p)\|\boldsymbol{A}\|_F^2 + \sum_{i=1}^{m} \sum_{j=1}^{n} \mu(1-p) \left(\mu - 2a_{ij} \right)
\end{aligned}$$

For the last term we have,

$$\begin{aligned}\sum_{i=1}^{m}\sum_{j=1}^{n}\mu(1-p)\left(\mu-2a_{ij}\right) &= \mu(1-p)\sum_{i=1}^{m}\sum_{j=1}^{n}\left(\mu-2a_{ij}\right)\\ &= \mu(1-p)\left(mn\mu-\sum_{i=1}^{m}\sum_{j=1}^{n}2a_{ij}\right)\\ &= \mu(1-p)\left(mn\mu-2mn\mu\right)\\ &= -mn\mu^2(1-p),\end{aligned}$$

which is non-positive. Thus, $\mathbb{E}_\delta[\|\boldsymbol{A}-\tilde{\boldsymbol{A}}_0\|_F^2] \geq \mathbb{E}_\delta[\|\boldsymbol{A}-\tilde{\boldsymbol{A}}_\mu\|_F^2]$

REFERENCES

[1] R. Jiang, T. Sun, D. Song, and J. J. Li, *Statistics or biology: the zero-inflation controversy about scrna-seq data*, Genome biology, 23 (2022), pp. 1–24.

[2] D. Leventhal and A. S. Lewis, *Randomized methods for linear constraints: Convergence rates and conditioning*, 2008.

[3] A. Ma and D. Needell, *Stochastic gradient descent for linear systems with missing data*.

[4] A. Ma, D. Needell, and A. Ramdas, *Convergence properties of the randomized extended gauss–seidel and kaczmarz methods*, SIAM Journal on Matrix Analysis and Applications, 36 (2015), pp. 1590–1604.

[5] S. Mukhopadhyay, *Stochastic gradient descent for linear systems with sequential matrix entry accumulation*, Signal Processing, 171 (2020), p. 107494.

[6] S. Mukhopadhyay and A. Mukherjee, *Imdlms: An imputation based lms algorithm for linear system identification with missing input data*, IEEE Transactions on Signal Processing, 68 (2020), pp. 2370–2385.

[7] D. Needell, *Randomized kaczmarz solver for noisy linear systems*, BIT Numerical Mathematics, 50 (2010), pp. 395–403.

[8] Y. Nesterov, *Efficiency of coordinate descent methods on huge-scale optimization problems*, SIAM Journal on Optimization, 22 (2012), pp. 341–362.

[9] M. S. Rahim, A. A. Imran, and T. Ahmed, *Mining the productivity data of garment industry*, International Journal of Business Intelligence and Data Mining, 1 (2021), p. 1.