Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Huan He Date

Acceleration Algorithms for Machine Learning Models

By

Huan He
Doctor of Philosophy

Computer Science and Informatics

Joyce C Ho, Ph.D.
Advisor

Yuanzhe Xi, Ph.D.
Co-Advisor

Liang Zhao, Ph.D.
Committee Member

Yousef Saad, Ph.D.
Committee Member

Accepted:

Kimberly Jacob Arriola, Ph.D, MPH
Dean of the James T. Laney School of Graduate Studies

Date

Acceleration Algorithms for Machine Learning Models

By

Huan He
B.A., Shanghai Finance University, Shanghai, 2014
M.Sc., Emory University, GA, 2018

Advisor: Joyce C Ho, Ph.D., Yuanzhe Xi, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2022

Abstract

Acceleration Algorithms for Machine Learning Models
By Huan He

Machine Learning has revolutionized the fields of computer vision, natural language understanding, speech recognition, information retrieval and more. However, with the progressive improvements in deep learning models, their number of parameters, latency, resources required to train, etc. have all have increased significantly. There are many recent examples that can illustrate the tremendous growth in scientific data generation in the literature. It is estimated that there are thousands of wireless sensors currently in place, which generates about a gigabyte of data per sensor per day. Consequently, it has become important to pay attention to these footprint metrics of a model as well, not just its quality. Nowadays, there is a greater need to develop efficient machine learning models to cope with future demands that are in line with similar energy-related initiatives. Either training or inference efficient algorithms are important for a number of data-intensive areas, as they affect many related industries. However, despite the fact that advanced and powerful machine learning models are proposed, there is a huge demand and space for such efficient and fast machine learning methods for large and complex data-intensive fields.

Acceleration Algorithms for Machine Learning Models

By

Huan He
B.A., Shanghai Finance University, Shanghai, 2014
M.Sc., Emory University, GA, 2018

Advisor: Joyce C Ho, Ph.D., Yuanzhe Xi, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2022

Acknowledgments

I would like to thank my esteemed supervisors – Dr. Ho and Dr. Xi for their invaluable supervision, support and tutelage during the course of my PhD degree. Additionally, I would like to express gratitude to Dr. Saad for his treasured support which was really influential in shaping my research methods and critiquing my results. I would also like to thank Dr.Liang Zhao for his impressive discussions on my work and inspiring suggestions for this dissertation.

Thanks my friends, lab mates, colleagues and research team for a cherished time spent together in the lab, and in social settings. My appreciation also goes out to my family and friends for their encouragement and support all through my studies.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Machine Learning has revolutionized the fields of computer vision, natural language understanding, speech recognition, information retrieval and more. However, with the progressive improvements in deep learning models, their number of parameters, latency, resources required to train, etc. have all have increased significantly. There are many recent examples that can illustrate the tremendous growth in scientific data generation in the literature [24, 31, 39]. Meanwhile, there has been rapid growth in the quantity and variety of data. Unfortunately, these large sets of data are usually high dimensional (*e.g.* patients, their diagnoses, and medications) and cannot be adequately represented as matrices. Thus, many existing algorithms can not analyze them. To accommodate these high dimensional data, tensor factorization [46] has attracted much attention and emerged as a promising solution. However, tensor factorization is a computationally expensive task, and existing methods developed to factor large tensors are not flexible enough for real-world situations, *e.g.* sparsity, simplex, and non-negativity constraint.

On the other hand, although the huge data volume has provided a great platform

for training machine learning models, the privacy concern is a common barrier for researchers to obtain access to real world data. As a result, this obstacle has hindered the research of effectiveness of machine learning models on real world. One approach that could overcome privacy issues is to use synthetic datasets that capture as many of the complexities of the original data set (distributions, non-linear relationships, and noise). There have been several distinguished efforts conducted in a variety of domains about synthetic EHR generation [10, 11, 20, 117, 134]. There are several noticeable drawbacks of these models, including mode-collapse for GANs or poor sample diversity and quality for autoencoders. A recent proposed denosing diffusion probalistic model [54] can avoid this issue and generate high fidelity samples, but it suffers from a major drawback that the sampling process is extremely slow.

As a result, there is a huge demand and space for either training or inference efficient machine learning algorithms for large and complex data-intensive fields despite the fact that advanced and powerful machine learning models are proposed. It is tempting to use parallel processing algorithms to simplify the problem by separating the machine learning tasks to different independent parts and running them in parallel. When the model and data can be stored on a single machine, we can use multi-core processing. Here, we assume that the whole model and the data can be fit into the memory of a single machine with multiple cores. For example, [90] propose to use multiple cores to perform SGD of multiple mini-batches in parallel where multiple cores share the same memory. However, modern machine learning models or data are so large that it is often impossible for the whole model and data to fit into one single machine. When it is not possible to store the whole data-set or a model on a single machine, it becomes necessary to store the data or model across multiple machines. [93] shows that distributed training can address this issue effectively by proposing both synchronous and asynchronous training frameworks.

However, there are three main limitations of distributed algorithms. 1) Distributed

algorithms means expensive equipment and maintenance costs which is a common barrier for most researchers. 2) Distributed algorithms often harms the model performance due to the overwriting during asynchronous updates, otherwise it is often as slow as sequential processing. 3) Such computational requirements make it difficult for practitioners to readily adopt. For example, in the healthcare setting, there is limited access to such high-performance systems and data cannot be transferred due to patient privacy concerns. As a result, it is necessary to design smart parallel algorithms that can fully utilize available resources.

Alternatively, numerical algorithms have attracted lots of attention due to its demonstrated performance and stability. For example, [63] proposed Adam which is a commonly used optimizer. It gains popularity because it converges much faster than the basic gradient descent. For another example, recent proposed algorithm, Regularized Nonlinear Extrapolation [109] utilizes the extrapolation algorithm to accelerate logistic regression. However, there is still much space to investigate its performance and theoretical analysis for other machine learning tasks. Furthermore, existing proposed numerical based algorithms [89, 109] can still exhibit very poor convergence properties. It is necessary to develop more advanced acceleration algorithm to address the limitations of existing numerical algorithms.

1.2 Research Contributions

Our contributions are summarized as follows:

1.2.1 SGranite:

We propose SGranite, a distributed tensor decomposition framework that can incorporate a variety of regularization terms to constrain the latent factors. In particular, we show that integrating three forms of regularization terms can achieve easier-to-

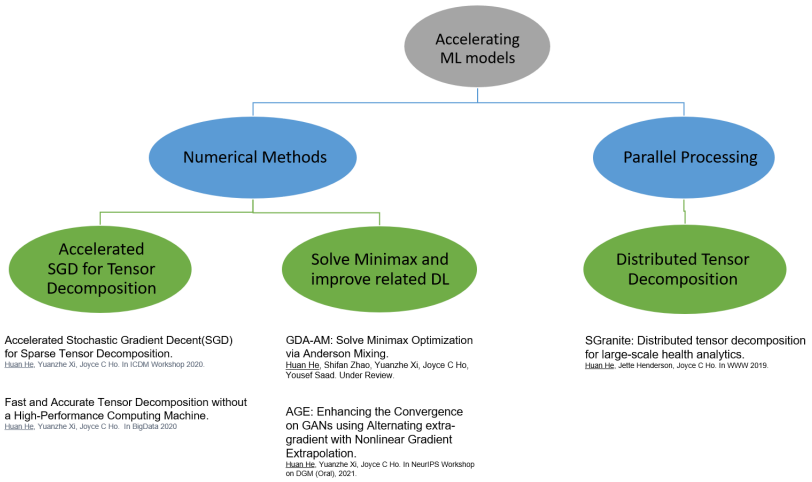


Figure 1.1: Dissertation Contributions

interpret factors, provide robustness in the presence of noise, and map to existing domain knowledge. Moreover, SGranite is very fast and scalable. Using a Spark-based implementation, we demonstrate the ability to decrease computation time by distributing both the data as well as the parameters without sacrificing accuracy. The contributions of our work can be summarized as follows:

1. **Flexibility:** Our framework supports a variety of meaningful constraints such as sparsity, diversity, and distinguish-ability.

2. **Scalability:** Our scalability analysis of SGranite on a large tensor constructed from healthcare data achieves near linearity speed-up as we scale to the number of machines. Moreover, our framework achieves at least a $4\times$ speed-up compared to an existing state-of-the-art distributed tensor factorization method.

3. **Accuracy:** Our empirical results in two health-related case studies show that incorporating the variety of constraints improves interpretability and robustness compared to the standard decomposition models.

1.2.2 Fast-CP:

We propose Fast-CP, an accelerated SGD-based CP decomposition model for large-scale tensors on a personal computer. Our model accelerates the convergence speed by mixing past iterates in a systematic fashion and decreases variance of the stochastic gradients. Fast-CP is up to 4 times faster than the state-of-the-art tensor SGD algorithm [67] across three different datasets, and can execute in a reasonable time for large datasets. Our contribution can be summarized as follows:

1. **Efficient extrapolation step:** We propose a computationally cheap technique that we call extrapolation to speed up the convergence of stochastic gradient updates for large-scale tensor CP decomposition with various loss functions.

2. **Improved convergence:** While SGD is memory efficient, it usually performs poorly in terms of convergence rate and quality. We illustrate how extrapolation of gradient sequences can fix this issue and yield better factor matrices.

3. **Robustness:** We show that Fast-CP improves the learning stability and lowers the variance of its base optimizer with negligible computation and memory cost.

4. **Generalizability:** We empirically demonstrate Fast-CP can significantly improve the performance of various types of tensors and different variants of SGD algorithms including the standard (vanilla) version and Adam.

1.2.3 GDA-AM:

We propose a different approach to solve minimax optimization. Our starting point is to cast the GDA dynamics as a fixed-point iteration. We then highlight that the fixed-point iteration can be solved effectively by using advanced non-linear extrapolation methods such as Anderson Mixing [4], which we name as GDA-AM. redAlthough first

mentioned in [6], to our best knowledge, this is still the first work to investigate and improve the GDA dynamics by tapping into advanced fixed-point algorithms.

We demonstrate that GDA dynamics can benefit from Anderson Mixing. In particular, we study bilinear games and give a systematic analysis of GDA-AM for both simultaneous and alternating versions of GDA. We theoretically show that GDA-AM can achieve global convergence guarantees under mild conditions.

We complement our theoretical results with numerical simulations across a variety of minimax problems. We show that for some convex-concave and non-convex-concave functions, GDA-AM an converge to the optimal point with little hyper-parameter tuning whereas existing first-order methods are prone to divergence and cycling behaviors.

We also provide empirical results for GAN training across two different datasets, CIFAR10 and CelebA. Given the limited computational overhead of our method, the results suggest that an extrapolation add-on to GDA can lead to significant performance gains. Moreover, the convergence behavior across a variety of problems and the ease-of-use demonstrate the potential of GDA-AM to become the minimax optimization workhorse.

1.2.4 MedDiff:

We introduce MedDiff, a novel denoising diffusion probabilistic model. MedDiff generates realistic synthetic patient records that build upon diffusion models to achieve high quality, robust samples while also being simple enough for practitioners to train. We further accelerate the generation process of MedDiff using Anderson acceleration [5], a numerical method that can improve convergence speed of fixed-point sequences. In summary, our contributions are as follows:

1. We investigate the effectiveness of diffusion based models on generating discrete Electronic Health Records (EHRs).

2. We proposed a novel method to accelerate the generation process, which is the main drawback of diffusion models.

3. We further adopt a novel conditioned sampling technique to generate discriminative synthetic EHR.

4. We show that MedDiff can generate realistic synthetic data that mimics the real data and provides similar predictive value according to our analysis and assessments.

1.3 Organization

I describe my several projects in this thesis. It is organized as follows. Chapter 2 introduces the basic background of acceleration algorithms. Chapter 3 introduces projects on accelerating tensor decomposition via parallel algorithms. Chapter 4 proposes the accelerated tensor decomposition framework without a high-performance computing machine using numerical algorithms. Chapter 5 proposes a novel algorithm to solve minimax optimization and its related deep learning models. Chapter 6 introduces the project on accelerating diffusion generative models using numerical algorithms. Chapter 7 concludes the thesis.

Chapter 2

Background

In this chapter, I will briefly introduce the background of parallel processing and numerical algorithms for developing efficient machine learning models.

2.1 Parallel Processing

Parallel Processing simply means algorithms are deployed across the multiple processors . A typical ML algorithm involves doing a lot of computation (work/tasks) on a lot of data set . Traditionally, machine learning has been executed in single processor environments, where algorithmic bottlenecks can lead to substantial delays in model processing, from training to classification to distance and error calculation and beyond. Training such models commonly can be held in a MapReduce Framework (sits across many machines).

2.1.1 The Map-Reduce algorithm

MapReduce is a programming framework that allows us to perform distributed and paral-

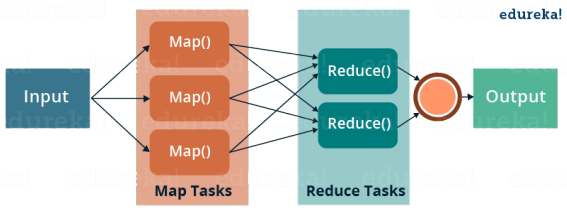


Figure 2.1: What's Map-Reduce?

lel processing on large data sets
in a distributed environment.
MapReduce consists of two distinct tasks, Map and Reduce.
As the name Map-Reduce suggests, the reducer phase takes place after the mapper phase has been completed. So, the first is the map job, where a block of data is read and processed to produce key-value pairs as intermediate outputs. The output of a Mapper or map job (key-value pairs) is input to the Reducer. The reducer receives the key-value pair from multiple map jobs. Then, the reducer aggregates those intermediate data tuples (intermediate key-value pair) into a smaller set of tuples or key-value pairs which is the final output.

2.2 Acceleration via mathematical methods

Optimization is at the heart of machine learning. Most of the machine learning problems are, in the end, optimization problems. Gradient descent (GD) has been one of the most commonly used first-order method due to its simplicity to implement and low computational cost per iteration. Although practical and effective, GD converges slowly in many applications. To accelerate its convergence, there has been a surge of interest in accelerated gradient methods, where "accelerated" means that the convergence rate can be improved without much stronger assumptions or significant additional computational burden. Nesterov has proposed several accelerated gradient descent (AGD) methods in his celebrated works which have provable faster convergence rates than the basic GD.

2.2.1 Stochastic Gradient Descent

The stochastic gradient descent method solves the above problem by repeatedly updating m. Given some initial value of m_0, each update is as follows:

$$m_k := m_{k-1} - \gamma \nabla f_{i_k}(m_{k-1}) \tag{2.1}$$

where γ is the learning rate and $i_k \in \{1, \ldots, n\}$ is some set of indices chosen at iteration k.

Since convergence of vanilla SGD can be slow, several algorithms have been introduced to speed up the process. Adam's method, one instance of momentum-based iterations, has been developed specifically for machine learning. 'ADAM' [63], which stands for ADAptive Moment estimation, uses an estimate of first moment (mean) and second moment (variance):

$$m_j = \beta_1 m_{j-1} + (1-\beta_1) g_j \ , \qquad \hat{m}_j = m_j/(1-\beta_1^j) \tag{2.2}$$

$$v_j = \beta_2 v_{j-1} + (1-\beta_2) g_j^2 \ , \qquad \hat{v}_j = v_j/(1-\beta_2^j) \tag{2.3}$$

$$w_j = w_{j-1} - \alpha \frac{\hat{m}_j}{(\hat{v}_j)^{1/2} + \epsilon}, \tag{2.4}$$

to accelerate the convergence of vanilla SGD. Commonly recommended parameters are $\beta_1 = 0.9,\ \beta_2 = 0.999,\ \epsilon = 10^{-8}$.

2.2.2 Nonlinear Acceleration Techniques

When a sequence of numbers, vectors, matrices or tensors converge slowly, or even diverge, extrapolation techniques can be used to transform the current sequence into a new sequence, which, under certain assumptions, converges faster. There exist many such sequence transformations which range across a wide range of disciplines with varying goals and various degree of success. For a review, see [13].

Classical acceleration techniques take a sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n$, e.g., vectors in R^d, and produce an accelerated sequence $\{\mathbf{t}_n^{(k)}\}$ of the form

$$\mathbf{t}_n^{(k)} = a_0\mathbf{x}_n + a_1\mathbf{x}_{n+1} + \cdots + a_k\mathbf{x}_{n+k}, \tag{2.5}$$

where the a_i's usually depend on k and n but satisfy the constraint $\sum_{i=0}^{k} a_i = 1$.

Most existing extrapolation techniques are based on the assumption that $\mathbf{x}_n$ satisfies a k-term kernel of the form:

$$a_0(\mathbf{x}_n - \mathbf{x}) + a_1(\mathbf{x}_{n+1} - \mathbf{x}) + \cdots + a_k(\mathbf{x}_{n+k} - \mathbf{x}) = \mathbf{0}, \forall n, \tag{2.6}$$

where $\mathbf{x}$ is the exact limit for the original sequence and the scalars $a_0, \ldots, a_k$ and $\mathbf{x}$ are unknowns with $a_0 a_k \neq 0$ and satisfy the constraint. (2.6) is called *Shanks kernel* in the literature.

Chapter 3

Accelerating Tensor Decomposition via Parallel Algorithms

In the past few decades, there has been rapid growth in the quantity and variety of data. Unfortunately, these large sets of data are usually high dimensional (*e.g.* patients, their diagnoses, and medications) and cannot be adequately represented as matrices. As a motivating example, search activities on diseases such as influenza can be used and correlated with actual influenza surveillance data. Estimation of influenza-like illness (ILI) rates is a well-studied task [70, 99], Google Flu Trends, while flawed, demonstrated a link between influenza related search queries and the Centers for Disease Control and Prevention's (CDC) [58]. Similarly, programs such as the National Institute of Health's All for Us, are looking to gather data and make it publicly available to researchers to enable precision medicine. Extracting influenza patterns or clinical characteristics from such high-dimensional data can pose challenges, even before considering whether the data has been appropriately labeled. Thus, many existing algorithms can not analyze them. To accommodate these high dimensional data, tensor factorization has attracted much attention and emerged as a promising solution. For example, our work on 'Phenotyping through Semi-Supervised

Tensor Factorization' successfully uncovered predictive and clinically interesting phenotypes for type-2 diabetes and resistant hypertension patients [50]. However, tensor factorization is a computationally expensive task, and existing methods developed to factor large tensors are not flexible enough for real-world situations, *e.g.* sparsity, simplex, and non-negativity constraint. Increasingly large amounts of health-related data are released on the Internet and have great potential for enabling better disease surveillance and disease management.

A vast majority of the algorithms for disease surveillance or disease prediction adopt a supervised learning approach, but the need for labels can limit the possible scope of the task. However, unsupervised learning methods such as tensor factorization have been successfully applied in many application domains including social network analysis [95, 96, 136] and health analytics [51, 52, 56, 125, 138].

3.1 Background and Notations

This section briefly introduces tensor decomposition and related work. The list of operations and symbols used in this thesis are listed in Table 3.1.

Tensor and Tensor Operations

Tensors are generalizations of matrices and vectors to higher dimensions. An N-way tensor is denoted as $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and each element of the tensor represents the interactions between N types of data. Each dimension of the tensor is referred to as a *mode*. Tensors can be unfolded or flattened as a matrix, which is called *matricization*. $\mathbf{X}_{(n)}$ denotes the matricization of $\mathcal{X}$ along mode-n.

Definition 1. *A rank-one N-way tensor is the outer product of N vectors:* $\mathcal{X} = a^{(1)} \circ a^{(2)} \circ \cdots \circ a^{(N)}$. *Each element of a rank-one tensor is the product of the corresponding vector elements (i.e.,* $x_{i_1 i_2 \cdots i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \cdots a^{(N)}_{i_N}$*).*

Table 3.1: Symbols and their associated definitions

Symbol	Definition
$\mathcal{X}$, $\mathbf{X}$, $\mathbf{x}$, x	Tensor, Matrix, Column Vector, Scalar
$\mathbb{1}$	All one matrix
$\mathbf{X}_{(n)}$	n-mode matricization of a tensor $\mathcal{X}$
$\mathbf{X}(\mathbf{r},:)$	rth row of $\mathbf{X}$
$\mathbf{X}(:,\mathbf{r})$	rth column of $\mathbf{X}$
$\mathbf{A}^{(n)}$	nth factor matrix
$\mathbf{x}_n$	nth element of vector $\mathbf{x}$
$\|\cdot\|_2, \|\cdot\|_F$	Matrix 2 norm, Frobenius norm
$*$	Hadamard (elementwise) product
$\oslash$	Hadamard (elementwise) division
$\circ$	outer product
$\otimes$	Kronecker product
$\odot$	Khatri-Rao product (column-wise $\otimes$)

Definition 2. *The Khatri-Rao product of two real-valued matrices* $\mathbf{A} \odot \mathbf{B}$ *of sizes* $I_{\mathbf{A}} \times R$ *and* $I_{\mathbf{B}} \times R$*, respectively, produces a matrix* $\mathbf{Z}$ *of size* $I_{\mathbf{A}} I_{\mathbf{B}} \times R$ *such that* $\mathbf{Z} = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \cdots & \mathbf{a}_R \otimes \mathbf{b}_R \end{bmatrix}$*, where* $\otimes$ *is the Kronecker product.*

The matrix $\mathbf{Z}_n$ represents the Khatri-Rao product of all the factor matrices except $\mathbf{A}^{(n)}$ such that $\mathbf{Z}_n = \mathbf{A}^{(1)} \odot \cdots \odot \mathbf{A}^{(n-1)} \odot \mathbf{A}^{(n+1)} \odot \cdots \odot \mathbf{A}^{(N)}$.

CP decomposition

The CANDECOMP / PARAFAC (CP) model [46] is one of the most popular and well-studied tensor decomposition methods. In CP decomposition, the observed tensor, $\mathcal{X}$, is approximated using a sum of rank-one tensors (or $\mathcal{M}$):

$$\mathcal{X} \approx \mathcal{M} = \sum_{r=1}^{R} \mathbf{A}^{(1)}(:,r) \circ \mathbf{A}^{(2)}(:,r) \circ \cdots \circ \mathbf{A}^{(N)}(:,r). \tag{3.1}$$

Figure 3.1 provides an example of the CP decomposition for a sparse tensor, where each rank-one tensor represents a latent factor. Fitting a CP decomposition involves minimizing an objective function between the tensor $\mathcal{X}$ and a model tensor $\mathcal{X}$. In

general, it takes the form of summation of element-wise loss functions over all entries, and is chosen based on assumptions about the underlying distribution of the data.

$$\text{minimize } F(\mathcal{X}, \mathcal{M}) \equiv \sum_{i_1 i_2 \cdots i_N} f\left(x_{i_1 \ldots i_N}, m_{i_1 \ldots i_N}\right) \tag{3.2}$$

For the numeric data, it is common to assume that the tensor elements follow a Gaussian distribution, which corresponds to the least squares approximation. The objective function f associated with the least squares approximation is defined as:

$$F(\mathcal{X}, \mathcal{M}) = \sum_{i_1 \cdots i_N} (x_{i_1 \cdots i_N} - m_{i_1 \cdots i_N})^2. \tag{3.3}$$

For count data, an appropriate assumption about the underlying distribution of the data is Poisson [18, 44] and the following KL-divergence fitting function f is used:

$$F(\mathcal{X}, \mathcal{M}) = \sum_{i_1 \cdots i_N} (m_{i_1 \cdots i_N} - x_{i_1 \cdots i_N} \log m_{i_1 \cdots i_N}). \tag{3.4}$$

The CP decomposition with the loss function (3.4) is often referred to as *CP-APR* [18, 44]. Using a Poisson model leads to a much better explanation for the zero observations encountered in sparse data, where these zeros correspond to events that are unlikely to be observed.

For binary data, it is natural to assume the tensor elements follow a Bernoulli distribution and the objective function f takes the following form:

$$F(\mathcal{X}, \mathcal{M}) = \sum_{i_1 \cdots i_N} (\log(1 + m_{i_1 \cdots i_N}) - x_{i_1 \cdots i_N} \log m_{i_1 \cdots i_N}). \tag{3.5}$$

Both (3.4) and (3.5) require a non-negative constraint.

The CP decomposition for different data types as shown in Equations (3.3), (3.4) and (3.5) is a finite sum problem. Stochastic gradient descent algorithm is one of the key tools for such an optimization problem. Consider a finite sum problem defined

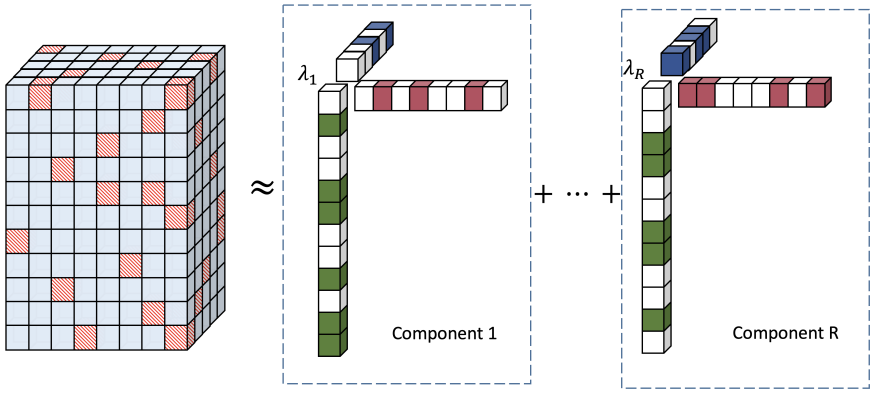


Figure 3.1: An illustration of CP decomposition for a sparse tensor. Shaded squares stand for nonzeros. The original tensor is approximated by the weighted sum of R rank-one tensors.

as follows:

$$\min_{m \in \mathbb{R}^d} \left\{ F(x, m) \stackrel{\text{def}}{=} \sum_{i=1}^{n} f_i(x, m) \right\} \tag{3.6}$$

where there are n given data points x, each f_i is an error function evaluated at the ith datapoint, and we want to obtain m that gives us the lowest error value that is summed over all datapoints.

SGD methods with this acceleration techniques have been shown effective for deep learning training [108, 110]. These methods compute an extrapolated set of parameters in neural networks by mixing the information from the past few iterates and can dramatically reduce the number of epochs. However, simple adoption of such techniques fails to speed up tensor CP decomposition.

3.2 SGranite

We propose `SGranite` [48], a distributed and flexible constrained CP model, to impose a variety of constraints on the latent factors. Our algorithm uses distributed stochastic gradient descent (DSGD) approaches to scale the CP decomposition on count data to huge datasets. SGranite has the following benefits:

- Simultaneously supports multiple constraints on the factor matrices.
- Learns patterns even when data cannot be stored on a single server.
- Maintains computational efficiency across a large number of workers.

A distributed framework for incorporating a variety of constraints in CP decomposition is appealing for several reasons including the ability to extract patterns from large datasets that cannot be readily stored on a centralized server, to encode prior knowledge, to improve interpretability, and to democratize high-dimensional learning by running on standard commodity servers.

We builds on several existing nonnegative CP decomposition algorithms to model sparse count data using the Poisson distribution [19, 52]. Let $\mathcal{X}$ denote an observed tensor constructed from count data with size $I_1 \times I_2 \times \cdots \times I_N$ and M represent a same-sized tensor of Poisson parameters for $\mathcal{X}$. In addition to KL divergence, we introduce generalized constraints on the factor matrices, $\mathcal{R}(\mathbf{A}^{(n)})$ to the objective function. Thus, the optimization problem is defined as:

$$\begin{aligned} \min f(\mathcal{M}) &= \sum_{\vec{i}} (m_{\vec{i}} - x_i \log m_{\vec{i}}) + \sum_k \underbrace{\beta_k \mathcal{R}_k \left(\mathbf{A}^{(n)}\right)}_{\text{regularization terms}} \\ \text{s.t. } \mathcal{M} &= [\![\boldsymbol{\lambda}; \mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(N)}]\!] \\ \lambda_r &\geq 0, ||\mathbf{a}_r^{(n)}||_1 = 1, \ \forall r \\ \mathbf{A}^{(n)} &\in [0,1]^{I_n \times R}, \ \forall n \end{aligned} \tag{3.7}$$

The Poisson parameters, $\mathbf{m}$, can be determined by minimizing the negative log-likelihood of the observed data $\mathbf{x}$. We also maintain the stochasticity (i.e., elements sum to 1) and non-negativity constraints (i.e., factor elements and weights, or λ, must be non-negative) that were introduced in the original CP-APR model [19].

3.2.1 Example of Useful Regularization Terms

Equation 3.7 supports a variety of regularization items, $\mathcal{R}(\mathbf{A}^{(n)})$. While we describe three forms of special regularizations that are useful for analyzing health data, SGranite was developed to handle any regularization that is either smooth and differentiable or has an easy-to-compute proximal operator [97].

Diversity on $\mathcal{A}^{(n)}$ For analyzing flu patterns or clinical characteristics of patient subgroups, it is preferable for the rank-one factor components to be distinct from each other. This allows domain experts to more easily interpret the patterns. While several mechanisms for encouraging diversity have been proposed [52, 62, 125], we adopt the angular penalty term in [52] that encourages diversity between rank-one tensors by penalizing overlapping elements. There are two benefits to this regularization. It does not require prior knowledge to construct a similarity matrix that is used in [62]. Similarly, it does not require the discovered patterns to be orthogonal to one another [125], which may be too restrictive. Under angular regularization, any element that has large values in multiple columns in the factor matrix are penalized. Thus, the angular penalty for the n^{th} factor matrix, $\mathbf{A}^{(n)}$, is formulated as follows:

$$\mathcal{R}_k\left(\mathbf{A}^{(n)}\right) = \sum_{r=1}^{R}\sum_{p=1}^{r} \max\left(0, \frac{(a_p^n)^T a_r^n}{\|a_p^n\|_2 \|a_r^n\|_2} - \theta_n\right)^2$$

Sparsity and Smoothness on $\mathcal{A}^{(n)}$ Sparsity and smoothness constraints have been introduced in a wide range of applications to improve interpretability and increase robustness to noise. Our framework supports a general class of ℓ_p penalties including simplex constraint term ($||\mathbf{a}_r||_1 = 1, a_{ir} \in [0, 1]$); ℓ_2 regularization on the weight and the first factor matrix, $\lambda\mathcal{A}^{(1)}$ to mitigate overfitting to large count data; and the ℓ_0-norm regularization which caps the number of non-zeros elements in the factor.

We first consider the simplex constraint term, which can yield sparse factors while providing a probabilistic interpretation. For the n^{th} factor matrix, $\mathbf{A}^{(n)}$, we restrict the elements to lie on the ℓ_1-ball of diameter s, where s is a user-specified parameter, such that:

$$\mathcal{R}_k\left(\mathbf{A}^{(n)}\right) = \sum_{r=1}^{R}(s - ||\mathbf{a}_r^{(n)}||_1)$$

When $s = 1$, this results in the projection of the factor onto the probabilistic (or canonical) simplex [26]. By decreasing s to be less than 1, the resulting factors will be sparser.

The ℓ_2-norm regularization was introduced in [52] to encourage terms in the factor matrix vectors to be similar-sized. Together with the simplex projection, the interaction of these two regularizations achieved further sparsity by driving specific elements to 0 more quickly in a similar manner to the elastic net regularization [145].

$$\mathcal{R}_k\left(\mathbf{A}^{(n)}\right) = \sum_{r=1}^{R} \|a_r^{(n)}\|_2$$

The ℓ_0-norm regularization, introduced in [3], is an alternative to the simplex projection that limits the number of non-zero elements. While its usage in Equation 3.7 results in a non-convex optimization problem, the hard thresholding properties can yield easy to interpret factors (top-k elements). To perform hard-thresholding on the n^{th} factor matrix, $\mathbf{A}^{(n)}$, the regularization term is:

$$\mathcal{R}_k\left(\mathbf{A}^{(n)}\right) = \sum_{r=1}^{R} \|a_r^{(n)}\|_0$$

Discriminative Factors In some scenarios, the discovered patterns should be discriminative of a certain outcome of interest. For example, we may want to use the clinical characteristics to predict things like mortality or whether or not the patient is likely to be readmitted in 30 days. [62] introduced a logistic regression regularization

that encouraged the derivation of latent factors that can distinguish in-hospital mortality outcomes. SGranite also adopts the regularization term to derive discriminative latent factors when such information exists. Without loss of generality, we assume that the first mode has labeled records. Then the discriminative regularization is of the form:

$$\mathcal{R}_k\left(\mathbf{A}^{(1)}\right) = \log P(\mathcal{A}^{(1)}, y|\theta) \tag{3.8}$$

The probability of a sample $\mathbf{a}(i,:)$ (i^{th} row in $\mathbf{A}^{(1)}$) having the outcome of interest, $P(\mathcal{A}^{(1)}, y|\theta)$, is obtained by training a logistic regression model on the factor matrix $\mathbf{A}^{(1)}$.

Sparse, Diverse, and Discriminative Patterns To demonstrate the flexibility of `SGranite`, we introduce all three forms of regularization into our final optimization problem. Thus, the final objective function is:

$$\begin{aligned} f(\mathcal{M}) = & \sum_{\vec{i}}(m_{\vec{i}} - x_i \log m_{\vec{i}}) + \\ & \beta_1 \sum_{n=1}^{N}\sum_{r=1}^{R}\sum_{p=1}^{r} \max\left(0, \frac{(a_p^n)^T a_r^n}{\|a_p^n\|_2 \|a_r^n\|_2} - \theta_n\right)^2 + \\ & \beta_2 \sum_{n=1}^{N}\sum_{r=1}^{R}(s - ||\mathbf{a}_r^{(n)}||_2) + \beta_3 \log P(\mathcal{A}^{(1)}, y|\theta) \end{aligned} \tag{3.9}$$

SGD Updates

This section provides details of how to solve our optimization problem efficiently (Equation 3.9). `SGranite` uses an alternating minimization approach, cycling through each mode while fixing all the other modes. For each mode, the resulting subproblem is solved using stochastic gradient descent (SGD). To derive the SGD updates, we first re-write the objective function as a scalar-valued function of the parameter vector y using the same approach as [1]. The parameter vector y represents the vectorization

of the factor matrices, with the weights $\boldsymbol{\lambda}$ absorbed into the first factor matrix.

$$y = \begin{bmatrix} vec(\boldsymbol{\lambda}\mathbf{A}^{(1)}) \\ vec(\mathbf{A}^{(2)}) \\ \vdots \\ vec(\mathbf{A}^{(n)}) \end{bmatrix}$$

As a result, the gradients of the objective function can be formed by vectorizing the partial derivatives with respect to each component of this parameter vector:

$$\nabla f(y) = \left[vec(\frac{\partial f}{\partial \mathbf{A}^{(1)}}) \cdots vec(\frac{\partial f}{\partial \mathbf{A}^{(n)}}) \right]$$

For notational convenience, we also represent the matricized form of the tensor decomposition as:

$$[\![\boldsymbol{\lambda}; \mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(N)}]\!]_{(n)} = \boldsymbol{\lambda}\mathbf{A}^{(n)}(\mathbf{A}^{(-n)})^T$$

where

$$\mathbf{A}^{(-n)} = \mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)}.$$

Thus, the partial derivatives of Equation 3.9 with respect to the factor matrix, $\mathbf{A}^{(n)}$ are the following:

$$\begin{aligned} \frac{\partial f}{\partial A_r^{(n)}} &= \left[1 - X_{(n)} \oslash Z_{(n)}\right] a_r^{(-n)} + \\ &\beta_1 \sum_{p \neq r} \max\left(0, g(a_r^{(n)}, a_p^{(n)})\right) \frac{\partial g(a_r^{(n)}, a_p^{(n))}}{\partial \mathbf{a}_r^{(n)}} + \\ &\beta_2 \mathbf{a}_r^{(n)} + \beta_3 y \frac{1}{1 + \exp(\mathbf{y}\mathbf{A}_r^{(1)})} \theta \end{aligned} \tag{3.10}$$

We refer the reader to [52, 62] for the detailed derivation of the gradients.

For large datasets, the calculation of the derivatives simultaneously for all modes is

computationally expensive. Thus, `SGranite` uses an SGD approach to avoid storing the entire tensor in memory. For faster convergence, we adopt a variant of SGD named Adaptive Moment Estimation (Adam) to adaptively update the learning rate [64]. Our preliminary experiments on a single machine showed that SGD with Adam converged faster and more accurately than using a fixed learning rate.

Algorithm 3.1 SGD updating process

for $l = 1, \ldots, L$ **do**
  Randomly select n samples.
  Calculate the gradients for samples using Equation 3.10.
  Compute the decaying averages of past and past squared gradients.
  Take a step using averaged gradients.
end

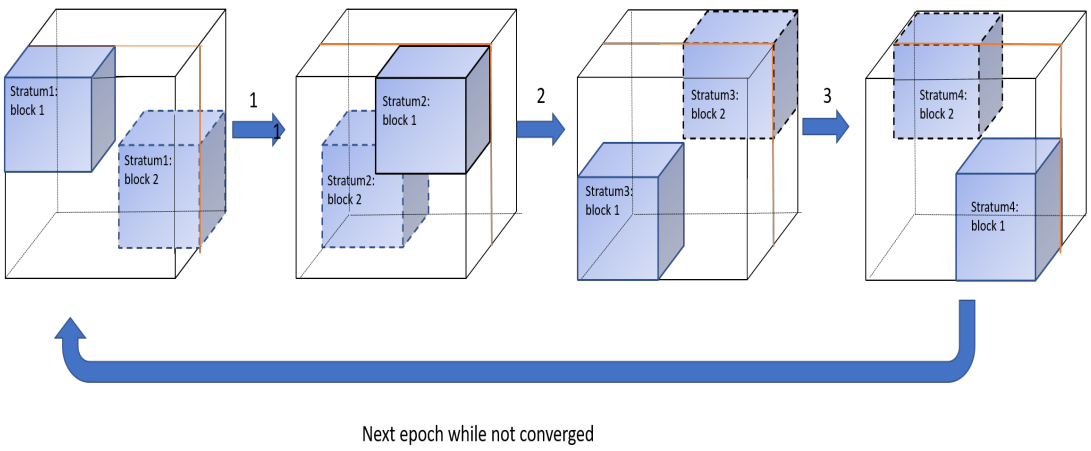


Figure 3.2: A graphical example of our method: Suppose there are 2 workers, we will have 8 blocks and 4 strata after partition. We run this process iteratively until convergence. In each epoch, start from strata one, each worker runs SGD for its own assigned block in parallel. Check the convergence until all strata are iterated. We repeat above algorithm again if the stopping criteria is not satisfied. All intermediate results are saved as Resilient Distributed Datasets (RDD) collections and cached in memory.

3.2.2 Parallel Algorithm using Spark

Although SGD scales well to sparse data, we would like to distribute the computation to achieve results faster. FlexiFact proposed distributing the computation by dividing

the tensor such that no two blocks share any elements (along with any dimension) [8]. Thus, the SGD algorithm can be run in parallel on each block without sacrificing accuracy. We refer the reader to [8, 33] for detailed proof of convergence. `SGranite` uses the same approach to distribute the non-zero elements of the count tensor using this tensor partition. However, we note several main differences between our framework and Flexifact: (1) support for sparse, count data by using an appropriate objective function (KL divergence), (2) flexibility to incorporate a variety of constraints beyond sparsity and non-negativity, and (3) distributed computation using Apache Spark.

Unlike `SGranite`, FlexiFact uses the Hadoop Map-Reduce platform to distribute the data collection across multiple nodes. Unfortunately, a Hadoop workflow spends an exorbitant amount of time on disk operations, as it needs to read and write intermediary results on the disk. On the other hand, Apache Spark [139] has been proposed as an alternative that eliminates unnecessary disk operations for iterative algorithms. By performing the data analytic operations in-memory and in near real-time, Spark can achieve lower computation times. Thus, the proposed method is developed using Spark to distribute the computation.

Tensor Partition First, we define a stratum as a set of independent blocks, and we denote the number of blocks in each stratum by d. Suppose we have d available workers, in order to iterate all regions of $\mathcal{X}$, we need d^3 blocks and thus d^2 strata. For a stratum s we have d blocks $Z_i^{(s)}$ for $i = 0, 1, \cdots, n-1$. A detailed partition

function for a size of $I \times J \times K$ tensor $\mathcal{X}$ is provided below:

$$\begin{aligned} b_i &= (i\lceil I/d \rceil, (i+1)\lceil I/d \rceil) \\ b_j &= (j\lceil J/d \rceil, (j+1)\lceil J/d \rceil) \\ b_k &= (k\lceil K/d \rceil, (k+1)\lceil K/d \rceil) \\ j_{s,i} &= (j+s) \\ k_{s,i} &= (j+s) \mod d \\ Z_i^{(s)} &= \mathcal{X}_{(b_i, b_{j_{s,i}}, b_{k_{s,i}})} \end{aligned} \tag{3.11}$$

Figure 3.2 provides an example of how to divide a count tensor for 2 available workers.

Block Parallelization Prior to introducing how `SGranite` iteratively solves the optimization problem in parallel, we introduce some definitions. A full epoch is defined as when the algorithm has seen all the d^3 blocks in the tensor. Since we need d^2 strata to cover all the blocks, we need to perform d^2 inner iterations. Therefore, we refer to each stratum training as a single inner iteration. Thus, in `SGranite`, the computation of each stratum is performed sequentially in each epoch. But for each stratum, we run SGD on the d^2 blocks in parallel. After each inner iteration, we update the factor matrices and use them as the initialization for the next stratum. Figure 3.3 provides an example of training using a single stratum. Upon the completion of an epoch (all strata have been run), the factor matrices are combined from all the workers, and then re-normalized for identifiability. The normalization can be performed for a user-specified mode, otherwise it defaults to the first mode. Convergence is checked between epochs by measuring the changes in the KL divergence to see if it is below a given tolerance. The details for the parallel-version of `SGranite` is described in Algorithm 3.2.

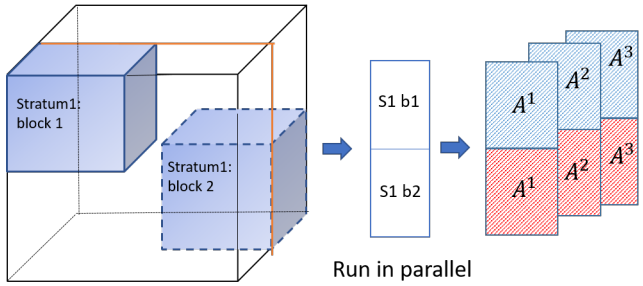


Figure 3.3: A graphical example of one stratum training: Given one stratum of training data and factor matrices $A^{(1)}, A^{(2)}, A^{(3)}$, we run SGD on each block in parallel. Then factor matrices $A^{(1)}, A^{(2)}, A^{(3)}$ are updated and used as the initialization for the next stratum training.

Algorithm 3.2 `SGranite`

Randomly initialize factors
Partition the tensor and construct d^2 strata
for m = *1,2,...* **do**
  for $l = 1 : d^2$ **do**
    Assign each block in lth strata to a worker
    Each worker runs Algorithm 1 in parallel
    Update the factors $[\![\boldsymbol{\lambda}; \mathbf{A}^{(1)}; \cdots ; \mathbf{A}^{(N)}]\!]$
  end
  Gather results from each worker
  Normalize factor matrices according to the specified mode
end
Return $[\![\boldsymbol{\lambda}; \mathbf{A}^{(1)}; \cdots ; \mathbf{A}^{(N)}]\!]$

Spark Implementation Details The non-zero elements of the count tensor are stored in a list using the coordinate format and loaded as Resilient Distributed Datasets (RDDs), and then it is shared throughout our cluster as a broadcast variable. A broadcast variable in Spark is immutable, meaning that it cannot be changed later on. This may seem inconvenient but it truly suits our case since we only need to read values from the tensor to calculate gradients in each iteration.

We do not broadcast factor matrices since we need to update them in each iteration. Due to our partition function, each worker has a chance to update factors matrices with different boundaries. The best way is to partition factor matrices using

Block ID. In this way, we can reduce the memory and communication cost. Specifically, we applied `map` and `aggregateByKey` functions to partition the factor matrices into blocks. The function `map` transforms each entry of the sparse tensor into an element in the RDD whose key is a block ID. Then `aggregateByKey` groups each block together and persists in memory. In each inner iteration, we use `groupWith` to build a stratum partitioned using `partitionBy` and then use `mapPartitions` to assign tasks to each node.

We found storing factor matrices RDDs and partitioned result in a significant acceleration, but not doing this will cause virtual memory issues in our experiments. Our experiments suggest such a design will enable us to obtain better speed-up and scalability.

3.3 Experimental Results

Datasets: We use the following two publicly available datasets:

- Influenza: Using Google Flu Trends historical data[1] from 2003 to 2015, we generated a tensor to uncover temporal influenza patterns that are unique and similar across multiple states. For each region in the United States, we collected the number of search queries related to influenza on a weekly basis over 11 years. The resulting tensor is 12 regions $\times$ 52 weeks $\times$ 11 years. Although the data quality has been shown to be low [92], this dataset is used to demonstrate the feasibility of `SGranite` on search data.

- MIMIC-III [61]: MIMIC-III is large database containing de-identified health data associated with approximately sixty thousand admissions of critical care unit patients from the Beth Israel Deaconess Medical Center collected between 2001 and 2012. For each patient, we extract medications and the International

[1] https://www.google.org/flutrends

Classification of Diseases (ICD-9) diagnosis codes. ICD-9 codes are aggregated using Clinical Classification Software (CCS) categories[2], a standard preprocessing step in healthcare analysis. Similarly, medications are grouped using the Anatomical Therapeutic Chemical (ATC) Classification via the RxNorm RESTful Web API, a web service developed by the National Library of Medicine[3]. The aggregation step results in a 38159 patient $\times$ 234 diagnosis $\times$ 511 medication tensor.

Baselines: We will compare `SGranite` to both centralized and distributed CP decomposition methods.

- CP-APR [19]: The first algorithm proposed for modeling sparse count data using a Poisson distribution. There is no support for constraints, and the updates are performed using multiplicative updates. The algorithm has been ported to Python by the authors of [52].
- Granite [52]: A centralized extension of CP-APR that incorporates the angular penalty, ℓ_2, and the simplex projection as regularization terms. The authors shared a python implementation of Granite fit using SGD.
- FlexiFact [8]: A distributed algorithm based on the DSGD approach that factorizes a coupled tensor and matrix using a similar partition method. However, it uses least squares as an objective and only supports non-negativity and ℓ_1 constraints. For a fair comparison, we implemented the algorithm in Spark according to the paper.

Scalability and speed-up First, we assess the quality of the approximation (measured by KL divergence) for `SGranite` and the other baseline methods. Figure 3.4

[2]The mapping from ICD-9 to CCS can be found at `https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp`.

[3]Details for the RxNorm RESTful Web API can be found at `https://mor.nlm.nih.gov/download/rxnav/RxNormAPIREST.html`.

and 3.5 show the KL divergence as the function of the number of epochs on both datasets. For the centralized algorithms (CP-APR and Granite), each epoch corresponds to a full iteration. The plots demonstrate that `SGranite` converges at least 4× faster than FlexiFact and also faster than the centralized algorithms. Moreover, the quality of the approximation is better than any of the existing methods. This suggests that SGD-based methods may help escape undesirable local minima (compared to CP-APR). The figure also highlights the importance of appropriately modeling the data distribution as opposed to using the least-squares loss (FlexiFact) may not yield the best approximation.

Next, we evaluate the scalability of our algorithm with respect to the number of workers. We calculate the speed-up as the ratio between the total execution time and the sequential execution time. Figure 3.6 demonstrates the speed-up of `SGranite` with respect to the number of workers. As can be seen in the figure, the speed up for the MIMIC tensor is very close to the ideal speed-up, as it is relatively large. However, there is a limited improvement on the Influenza tensor, a small dataset. This is due to the communication cost that is incurred in coordinating the different nodes. We note that because `SGranite` caches the updated factor matrices in memory to minimize disk accesses between consecutive iterations, it is able to scale to a large dataset and a large number of workers. Since this speed-up would not be possible on a system like Hadoop, we do not provide a comparison with FlexiFact.

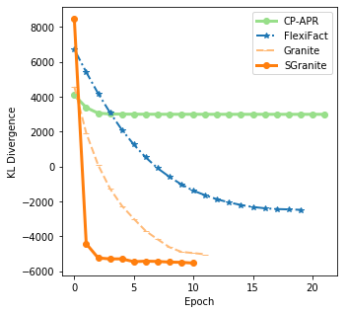


Figure 3.4: Influenza

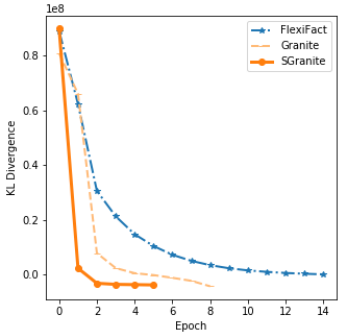


Figure 3.5: MIMIC

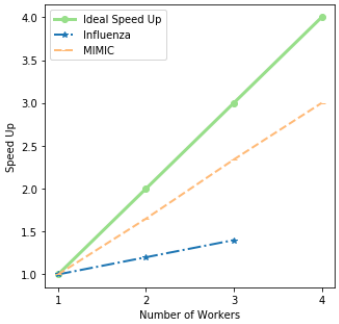


Figure 3.6: Speed-up

Qualitative and Quantitative Assessment of the Constraints To examine the impact of the constraints, we first compare the results from `SGranite` on the influenza dataset both with and without the angular and simplex regularization terms. Figure 3.7a shows the latent factors learned without the regularization terms, and Figure 3.7b shows the latent factors learned with the regularization terms. From these plots, we observe that the learned factors without regularization are highly correlated and can be difficult to distinguish from one another. In Figure 3.7a, it is hard to discern any noticeable pattern across the weeks and the different regions. In comparison, Figure 3.7b demonstrates the potential of incorporating both diversity and sparsity. We can observe that factor 2 predominantly captures the peak influenza season that occurs both towards the end of December and in mid-February in region 7, whereas factor 3 is slightly delayed and captures the influenza trend in regions 1 and 10. Furthermore, all three factors capture the peak in influenza season that occurs in late December and early February through March.

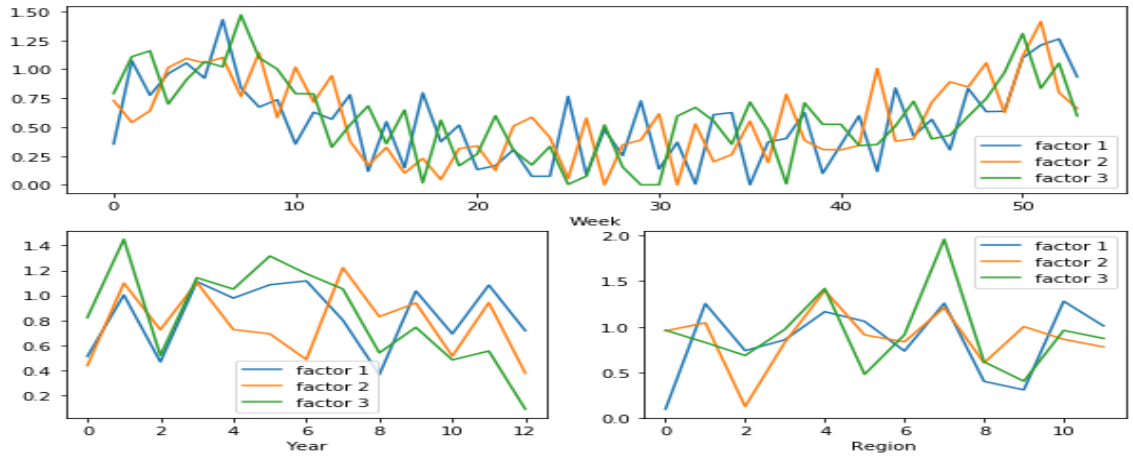
Next, we quantitatively assessed the impact of the logistic regression and angular penalty on the MIMIC-III dataset. To evaluate the discriminative power and distinctiveness of the learned factors, we used the in-hospital mortality cohort similar to that proposed in [62]. We used 37,000 patients, including all 5,014 patients who died during admission. We split our dataset into 80% training and 20% testing. We measured the discrimination on the test set using the area under the receiver operating characteristic curve (AUC). Distinctiveness is measured using the average overlap or the degree of overlapping between latent factors. It is defined as the average of cosine similarities between all latent factor pairs:

$$\text{Avg Overlap} = \frac{\sum_{r_1}^{R} \sum_{r_2 > r_1}^{R} \cos(a_{r_1}^{(2)}, a_{r_2}^{(2)}) + \cos(a_{r_1}^{(3)}, a_{r_2}^{(3)})}{R(R-1)} \tag{3.12}$$

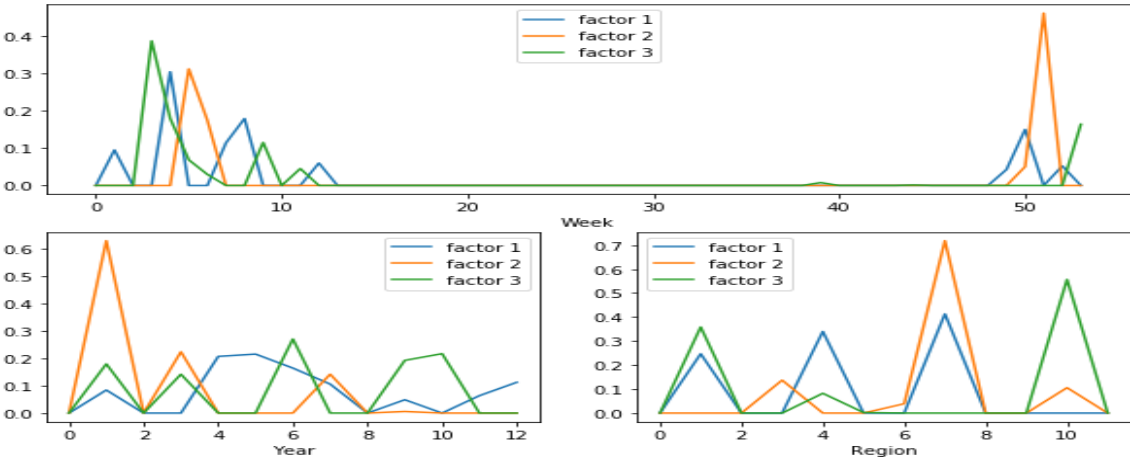
Table 3.2 summarizes the AUC, total computation time (or running time), and the average overlap. We observe that `SGranite` can not only accelerate the tensor decomposition but also provides better prediction than other baseline methods. Moreover, the average overlap is smaller than Granite even without the angular constraints. This suggests that the partition function may also have some beneficial impact in terms of reducing overlapping factors. Moreover, incorporating the angular constraints further helps the discriminative ability of the model. This suggests that adding diversity constraints to yield less correlated latent features may also help the resulting predictive model. Therefore, `SGranite` supports a variety of flexible constraints and yields improved predictive performance.

Model	**AUC**	**Time**	**Avg Overlap**
CP-APR [19]	0.63	> 1 hour	0.3
FlexiFact [8]	0.65	35 mins	0.37
Granite [52]	0.67	> 1 hour	0.3
SGranite ($\beta_1 = 0$)	0.68	**20 mins**	0.1
SGranite ($\beta_1 > 0$)	**0.71**	25 mins	**0.07**

Table 3.2: Table of AUC, running time, and average overlapping using different methods. The highest AUC value means extracted phenotypes have stronger discrimination. The lowest running time indicates our distributed method can significantly accelerate the computation time. Compared to CP-APR and FlexiFact, adding angular penalty improved the distinction significantly.



(a) No angular penalty and simplex projection ($\beta_1 = \beta_2 = 0$)



(b) Angular penalty and simplex projection constraints

Figure 3.7: A comparison of the learned latent factors with and without constraints using $R = 3$. Year from 2003 to 2015

Chapter 4

Acceleration Tensor Decomposition via numerical methods

Although parallel algorithms are powerful, limitations of such methods include expensive costs, vulnerability to data leakage and so on. Alternatively, numerical algorithms can often accelerate training dramatically. The most obvious benefits is that they do not require access to high-performance computing resources. Other benefits include that they usually can accelerate a wide range of problems and are immune to data-leakage and thus appropriate for sensitive data. We can view the training procedure as fixed-point iterations and use numerical methods to further speed up the convergence. Advanced numerical techniques like the minimal residual method have been widely studied and proven to be successful for well-defined problems. However, existing approaches focus solely on numeric data and may not yield desirable results for binary or count data. Therefore, we proposed `FAST-CP`[49], a novel algorithm to accelerate the convergence of the SGD-based tensor decomposition model via gradient extrapolation. First, we found naïve adoption of extrapolation did not improve training. We speculated this was due to the intrinsic noisy updates for SGD method and proposed to perform extrapolation periodically to meet the smoothness

assumption of extrapolation. Further, by incorporating different loss functions based on the inherent data distribution, our algorithm modeled various tensor data types, accelerated convergence in terms of speed and quality, and improved the learning stability of SGD. Our empirical results on three real-world datasets demonstrated that it decreases the total computation time while providing accurate results for downstream tasks without necessitating a high-performance computing platform or environment.

We begin by explaining how SGD works for variety of CP objective functions. Next we present our novel extrapolation technique that improves the convergence of SGD methods and obtains a more optimal convergence point. Then we provide some theoretical convergence analyses to support our method.

4.1 Generalized CP decomposition using SGD

Gradient descent is a common technique used to deal with CP decomposition associated with various loss functions f. If we define $\mathcal{Y} = \mathcal{X} - \mathcal{M}$, the partial derivative of F in Equation (3.3) with respect to $\mathbf{A}^{(n)}$ can be written as

$$\frac{\partial F}{\partial \mathbf{A}^{(n)}} = -2\mathbf{Y}_{(n)}\mathbf{Z}_n, \tag{4.1}$$

where $\mathbf{Y}_{(n)}$ is the matricization of $\mathcal{Y}$ along mode-n and $\mathbf{Z}_n$ is the Khatri-Rao product of all factor matrices except $\mathbf{A}^{(n)}$. The operation on the right hand size of (4.1) is called the *matricized tensor times Khatri-Rao product* (MTTKRP).

Similarly, the partial derivative of F in Equation (3.4) with respect to $\mathbf{A}^{(n)}$ can be written as

$$\frac{\partial F}{\partial \mathbf{A}^{(n)}} = -(\mathbb{1} - \mathbf{X}_n \oslash \mathbf{Y}_{(n)})\mathbf{Z}_n, \tag{4.2}$$

and the partial derivative of F in Equation (3.5) with respect to $\mathbf{A}^{(n)}$ can be written

as

$$\frac{\partial F}{\partial \mathbf{A}^{(n)}} = -(\mathbb{1} \oslash \mathbf{Y}_{(n)} - \mathbf{X}_n \oslash \mathbf{Y}_{(n)})\mathbf{Z}_n, \tag{4.3}$$

For Equations (4.1)–(4.3), even when $\mathcal{X}$ is sparse, $\mathcal{Y}$ is usually a fully dense tensor. If $S = \prod_{i=1}^{N} I_i$, then the calculation of gradients for CP decomposition involves an intermediate sequence of N matricized-tensor times Khatri-Rao products (MTTKRPs) with a dense tensor of size S. These operations cost $O(RS)$, even when $\mathcal{X}$ is sparse. Thus, for large-scale tensor problems, the computational and storage costs of computing the exact gradient may be infeasible.

In recent years, SGD algorithms have been proposed to alleviate the difficulty of applying standard gradient descent in tensor decomposition problems. To create a random sparse instance $\tilde{\mathbf{Y}}_{(n)}$ of $\mathbf{Y}_{(n)}$, one can sample K indices uniformly with replacement. This uniform sampling is one of the most common strategies used for fitting dense tensors. However it may not be appropriate for sparse tensors since nonzeros will rarely be sampled. A stratified sampling-based SGD algorithm has recently been proposed in [67] to fix this issue. Different from previously proposed tensor SGD algorithms[9, 111], this algorithm samples both zero and non-zero elements of $\mathcal{X}$ and constructs $\tilde{\mathcal{Y}}$ as an unbiased estimation of $\mathcal{Y}$.

While SGD-based algorithms are memory efficient, both uniform sampling and stratified sampling-based approaches often require too many iterations to converge in practice. Moreover, the SGD-based CP algorithm can oscillate around the minimal. Thus, we are interested in designing an algorithm that can further accelerate SGD algorithms and yield more accurate results.

4.2 Extrapolated Stochastic Gradient Descent

Classical extrapolation methods reviewed in Section 2.2.2 can be quite effective but they all rely heavily on some intrinsic *smoothness* characteristics of the sequence. This

smoothness is expressed by either the Shanks kernel or the differential of the function f. Preliminary experiments show that extrapolation on factor matrices did not yield any considerable speedup. This is because the factor matrices generated by SGD algorithms are both random and noisy, which violates the smoothness assumption required by classical extrapolation methods.

In the deep learning community, a similar idea called *smoothing* has been explored to avoid sharp minima and obtain better generalization performance [47, 130]. The basic idea is to uniformly average a sample of past gradients to obtain the so-called *extragradient*. In fact, these smoothing techniques correspond to a special case in tensor SGD algorithms when only the last gradient matrix is used to update the current factor matrix. Since the convergence of these smoothing methods depends on the magnitude of the extragradient and classical nonlinear acceleration techniques can produce a gradient with smaller norm than the extragradient, applying extrapolation techniques on gradient sequences can lead to faster convergence.

In this thesis, we adopt the Vector Epsilon Algorithm (VEA) [132] framework to implement our gradient sequence extrapolation. This framework uses the generalized matrix inverse to extend the scalar ϵ-algorithm to sequences of matrices. The algorithm is summarized in the following formula:

$$\begin{aligned} \epsilon_{-1}^{(i)} &= 0 \\ \epsilon_{0}^{(i)} &= \Delta\mathbf{A}_i^{(n)} \\ \epsilon_{k+1}^{(i)} &= \epsilon_{k-1}^{(i+1)} + \frac{(\epsilon_k^{(i+1)} - \epsilon_k^{(i)})}{\left\|\epsilon_k^{(i+1)} - \epsilon_k^{(i)}\right\|_F^2} \quad \text{for } k > 0. \end{aligned} \tag{4.4}$$

The final output of those sequences defined in ϵ-algorithm represent the Shanks transforms of the original sequence $\Delta\mathbf{A}_i^{(n)}$. As soon as a new iterate $\Delta\mathbf{A}_i^{(n)}$ becomes available we can immediately compute $\epsilon_1^{(i-1)}, \epsilon_2^{(i-2)}, \ldots$. The VEA implementation is detailed in Algorithm 4.1.

After we obtain the extrapolated gradient matrix, we use it to update the current

factor matrix with the same stepsize as the baseline optimization algorithm. Note that this is different from VEA where the original sequence is not interlaced with the extrapolated one. We call this extrapolated SGD method as `FAST-CP` and detail its major operations in Algorithm 4.2. We further extend the extrapolation idea of SGD to Adam and validate its effectiveness (compared with Adam) on CP decomposition in the experiments. When Adam is used as the baseline optimizer, we still pass the stochastic gradient sequence in Algorithm 4.1.

Algorithm 4.1 Extrapolation of gradient sequence for mode n

For the first time: Initialize a table $\mathbf{T}^{(n)}$ with $2k+1$ columns. Set a window size k.
Input: The latest gradient matrix $\Delta\mathbf{A}_i^{(n)}$ and factor matrix $\mathbf{A}_{i+1}^{(n)}$, current iteration number i, table $\mathbf{T}^{(n)}$
Output: Updated factor matrix $\mathbf{A}_{i+1}^{(n)}$, table $\mathbf{T}^{(n)}$
$j = 1, \mathbf{z} = 0$ and an empty array $\mathbf{Y}$
Set $\mathbf{Y}(:,1) = \Delta\mathbf{A}_i^{(n)}$
while $j < 2k+1$ and $j < i$ **do**
    Compute $\triangle\varepsilon = \mathbf{Y}(:,j) - \mathbf{T}^{(n)}(:,j)$
    Calculate $\mathbf{z} = \mathbf{z} + \triangle\varepsilon / \|\triangle\varepsilon\|_F^2$
    $\mathbf{Y}(:,j+1) = \mathbf{z}$
    Reset $\mathbf{z} = \mathbf{T}^{(n)}(:,j)$
    $j = j + 1$
end while
Set $\mathbf{T}^{(n)} = \mathbf{Y}$ and $\mathbf{E}^{(n)} = \mathbf{T}^{(n)}(:, 2k+1)$
if $i \leq 2k$ and $j \bmod 2 == 1$ **then** {when i is too small}
    $\mathbf{E}^{(n)} = \mathbf{E}^{(n)} / \left\|\mathbf{E}^{(n)}\right\|_F^2$
end if
Update $\mathbf{A}_{i+1}^{(n)} := \mathbf{A}_{i+1}^{(n)} - \gamma\mathbf{E}^{(n)}$
return $\mathbf{A}_{i+1}^{(n)}$, $\mathbf{T}^{(n)}$ =0

4.3 Theoretical Analysis

In this section, we provide some theoretical justifications for the proposed method shown in Algorithm 4.1.

Following most convergence analyses for SGD methods [74, 144], we make the

Algorithm 4.2 FAST-CP

Input: N-way tensor $\mathcal{X}$, batch size K, learning rate γ, max epoch number M, number of SGD updates per epoch I, gradient sequence length k
Output: CP decomposition $\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(N)}$
Initialize factors $\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(N)}$
while Not converged or max epoch number not reached **do**
  Shuffle indices of tensor $\mathcal{X}$
  for $i = 1 : I$ **do**
    Get samples either using uniform sampling or stratified sampling
    for all modes **do** {optimize all at once}
      Calculate gradients $\Delta\mathbf{A}^{(n)}$ according to (4.1)–(4.3)
      Update $\mathbf{A}_{i+1}^{(n)} = \mathbf{A}_i^{(n)} - \gamma\Delta\mathbf{A}_i^{(n)}$
      Update $\mathbf{A}_{i+1}^{(n)}$ and $\mathbf{T}^{(n)}$ according to Algorithm 4.1
    end for
  end for
  Check convergence
end while
return $\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(N)} = 0$

following three assumptions for the tensor gradient sequence:

Assumption 1 The objective function F is continuously differentiable and the gradient of F is Lipschitz continuous with Lipschitz constant $L > 0$.

Assumption 2 The stochastic gradient computed by Equations (4.1)–(4.3) is an unbiased estimator of the true gradient.

Assumption 3 The variance of the stochastic gradients is bounded. That is there exists a constant $\sigma^2 > 0$ such that

$$\mathbb{E}\left[\|\frac{\partial F}{\partial \mathbf{A}^{(n)}} - \Delta\mathbf{A}^{(n)}\|^2\right] \leq \sigma^2$$

for all modes n.

We first review the known convergence result for mini-batch SGD on non-convex functions from [34].

Theorem 4.3.1. *Under the assumptions 1-3, after T mini-batch gradient updates,*

each with K samples, the mini-batch SGD returns an iterate $\mathbf{x}$ which satisfies

$$\mathbb{E}\left[\|\nabla F(\mathbf{x})\|^2\right] \leq \mathcal{O}\left(\frac{L(F(\mathbf{x}_0)-F^*)}{T} + \frac{\sigma\sqrt{L(F(\mathbf{x}_0)-F^*)}}{\sqrt{KT}}\right),$$

where $\mathbf{x}_0$ is the initial guess and F^* is a lower bound on the values of F.

Theorem 4.3.1 indicates that when the mini-batch size is small, the converge rate is dominated by the term $\frac{\sigma\sqrt{L(F(\mathbf{x}_0)-F^*)}}{\sqrt{KT}}$ which has possible speedup as K increases. On the other hand, when the mini-batch size becomes large, further increasing the mini-batch size has little effect on the convergence as the convergence rate will be dominated by the first term $\frac{L(F(\mathbf{x}_0)-F^*)}{T}$. In the numerical experiments, we carefully tune this hyperparameter and choose the optimal one for baseline methods.

In the next theorem, we review the asymptotic convergence for SGD with extra-gradients when only one worker and no momentum term are used.

Theorem 4.3.2 (Theorem 4.4 [74])**.** *Under the same assumptions as in Theorem 4.3.1, when the stepsize $\gamma \leq \frac{1}{L}$, the sequence $\mathbf{x}_t$ generated by extragradients satisfies*

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla F(\mathbf{x}_t)\|^2\right] \tag{4.5}$$

$$\leq \frac{2}{\gamma T}\mathbb{E}\left[F(\mathbf{x}_0)-F^*\right] + \left(\frac{4\gamma^2L^2}{K} + \frac{\gamma L}{K}\right)\sigma^2, \tag{4.6}$$

where $\mathbf{x}_0$ is the initial guess and F^* is a lower bound on the values of F.

Although Theorem 4.3.2 cannot show that SGD with extragradients achieves a speedup over mini-batch SGD, its superior performance in terms of faster convergence and better generalization has been demonstrated in various deep learning tasks [23, 35, 74].

4.4 Experimental Results

All experiments were run using Python on a Dual Socket Intel E5-2683v3 2.00GHz CPU with 64 GB memory. [1]

Datasets

We use the following three publicly available tensors that are from small to large and consist of continuous, count and binary data respectively.

- Human Connectome Project (HCP): Human Connectome Project collects measurements of structural and functional neural connections in vivo within and across individuals[2]. We use the constructed tensor from [124]. It is a $68 \times 68 \times 212$ binary tensor consisting of structural connectivity patterns among brain regions for 212 individuals. Each entry encodes the presence or absence of fiber connections between the brain regions.

- MNIST: A dataset of handwritten numbers from 250 different people, available from the National Institute of Standards and Technology (NIST) [71]. The MNIST dataset contains 60,000 images in the training set, each of size 28×28 pixels with 256 gray levels. We normalize each element by using the global mean (0.1307) and standard deviation (0.3081). After the normalization process, a $60000 \times 28 \times 28$ numeric tensor is obtained.

- Yelp: A dataset that contains 4M ratings from 1M users in Yelp across 149 months[3]. The tensor modes correspond to 1,029,432 Yelp users, 144,072 businesses and 149 months. Each entry represents the user rating (integer from 1 to 5) for the business. We use this dataset to demonstrate the scalability of .

[1] Our implementation is available at https://github.com/hehuannb/fast-cp.
[2] `http://www.humanconnectomeproject.org/data/hcp-project/`
[3] `http://www.yelp.com/datasetchallenge`

Baseline Methods

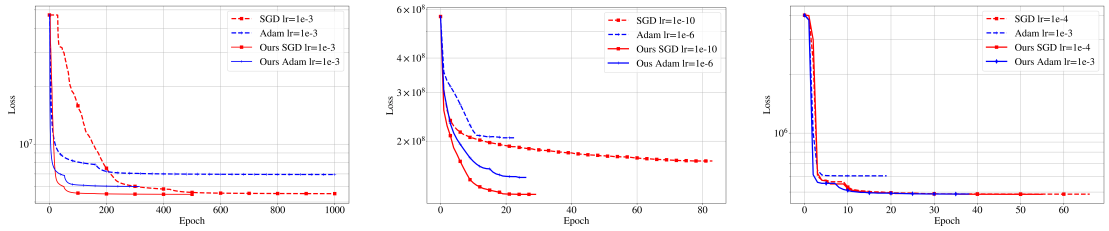
In this experiment, four baselines have been selected to evaluate the performance. The baseline methods contain alternating minimization including CP-ALS[45] and CP-APR[18, 44] and SGD-based methods including GCP-SGD and Adam[67].

- CP-ALS[45]: The standard method for fitting the CP model to numeric data. The algorithm alternates among the modes, fixing every factor matrix but $\mathbf{A}^{(i)}$. CP-ALS has a closed-form solution for each mode but requires significant memory.

- CP-APR [18, 44]: An algorithm proposed for modeling sparse count data using a Poisson distribution. This algorithm employs an alternating optimization scheme that sequentially optimizes one factor matrix while holding the others fixed. We use the state-of-the-art CP-APR[44] as a baseline model which uses limited-memory quasi-Newton approximations.

- GCP-SGD [67]: An algorithm for fitting the generalized CP decomposition using SGD. It adopts both uniform sampling and stratified sampling strategies. When the tensor is dense, we use the uniform sampling. When the tensor is sparse, we choose stratified sampling since it is more efficient and converges faster. It uses SGD as the base optimizer.

- GCP-Adam [67]: The only difference is that GCP-Adam uses Adam [63] as the base optimizer, which usually converges faster than SGD in deep learning applications.



A: MNIST B: Yelp C: HCP

Figure 4.1: Convergence plots for the SGD-based methods for the 3 datasets. In A, $R = 60$, batch size=2000, each epoch contains 3000 iterations. In B, $R = 100$, batch size=5000, each epoch contains 100 iterations. In C, $R = 10$, batch size=500, each epoch contains 300 iterations.

Chapter 5

Accelerating general minimax optimization via Anderson Acceleration

Since the operator of many machine learning training loops is not a simple linear operator, it is natural to ask the question of whether numerical techniques can help other machine learning tasks? In this section, we answer this question affirmatively by introducing `GDA-AM` that solves minimax optimization efficiently and accurately.

5.1 Minimax Optimization

Minimax optimization has received a surge of interest due to its wide range of applications in modern machine learning, such as generative adversarial networks (GAN), adversarial training and multi-agent reinforcement learning [38, 73, 78]. Formally, given a bivariate function $f(\boldsymbol{x}, \boldsymbol{y})$, the objective is to find a stable solution where the players cannot improve their objective, i.e., to find the Nash equilibrium of the

underlying game [121]:

$$\arg\min_{x\in\mathcal{X}} \arg\max_{y\in\mathcal{Y}} f(\boldsymbol{x},\boldsymbol{y}). \tag{5.1}$$

Definition 3. *Point* $(\mathbf{x}^*, \mathbf{y}^*)$ *is a* ***local Nash equilibrium*** *of* f *if there exists* $\delta > 0$ *such that for any* $(\mathbf{x}, \mathbf{y})$ *satisfying* $\|\mathbf{x} - \mathbf{x}^*\| \leq \delta$ *and* $\|\mathbf{y} - \mathbf{y}^*\| \leq \delta$ *we have:* $f(\mathbf{x}^*, \mathbf{y}) \leq f(\mathbf{x}^*, \mathbf{y}^*) \leq f(\mathbf{x}, \mathbf{y}^*)$.

To find the Nash equilibria, common algorithms including GDA, EG and OG, can be formulated as follows. For the two variants of GDA, simultaneous GDA (SimGDA) and alternating GDA (AltGDA), the updates have the following forms:

$$\begin{aligned} \text{Simultaneous}: &\quad \mathbf{x}_{t+1} = \mathbf{x}_t - \eta\nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t), \quad \mathbf{y}_{t+1} = \mathbf{y}_t + \eta\nabla_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y}_t) \\ \text{Alternating}: &\quad \mathbf{x}_{t+1} = \mathbf{x}_t - \eta\nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t), \quad \mathbf{y}_{t+1} = \mathbf{y}_t + \eta\nabla_{\mathbf{y}} f(\mathbf{x}_{t+1}, \mathbf{y}_t). \end{aligned} \tag{5.2}$$

The EG update has the following form:

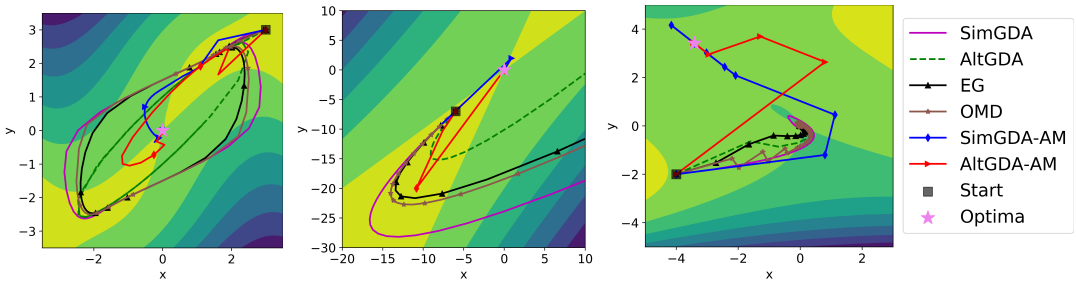
$$\begin{aligned} \mathbf{x}_{t+\frac{1}{2}} &= \mathbf{x}_t - \eta\nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t), & \mathbf{y}_{t+\frac{1}{2}} &= \mathbf{y}_t + \eta\nabla_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y}_t) \\ \mathbf{x}_{t+1} &= \mathbf{x}_t - \eta\nabla_{\mathbf{x}} f(\mathbf{x}_{t+\frac{1}{2}}, \mathbf{y}_{t+\frac{1}{2}}), & \mathbf{y}_{t+1} &= \mathbf{y}_t + \eta\nabla_{\mathbf{y}} f(\mathbf{x}_{t+\frac{1}{2}}, \mathbf{y}_{t+\frac{1}{2}}). \end{aligned} \tag{5.3}$$

The OG update has the following form:

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{x}_t - \eta\nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t) + \frac{\eta}{2}\nabla_{\mathbf{x}} f(\mathbf{x}_{t-1}, \mathbf{y}_{t-1}), \\ \mathbf{y}_{t+1} &= \mathbf{y}_t + \eta\nabla_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y}_t) - \frac{\eta}{2}\nabla_{\mathbf{y}} f(\mathbf{x}_{t-1}, \mathbf{y}_{t-1}). \end{aligned} \tag{5.4}$$

It is commonplace to use simple algorithms such as gradient descent ascent (GDA) to solve such problems, where both players take a gradient update simultaneously or alternatively. Despite its simplicity, GDA is known to suffer from a generic issue for minimax optimization: it may cycle around a stable point, exhibit divergent behavior, or converge very slowly since it requires very small learning rates [36, 82]. Given the widespread usage of gradient-based methods for solving machine learning

problems, first-order optimization algorithms to solve minimax problems have gained considerable popularity in the last few years. Algorithms such as optimistic Gradient Descent Ascent (OG) [22, 82] and extra-gradient (EG) [36] can alleviate the issue of GDA for some problems. Yet, it has been shown that these methods can still diverge or cycle around a stable point [2, 80, 98]. For example, these algorithms even fail to find a local minimax (the set of local minimax is a superset of local Nash [59, 126]) as shown in Figure 5.1.



(a) Cycling Behavoir (b) Diverging Behavoir (c) Converging to a non-optima

Figure 5.1: **Left**:$f(x,y) = (4x^2 - (y - 3x + 0.05x^3)^2 - 0.1y^4)e^{-0.01(x^2+y^2)}$. **Middle**: $-3x^2 - y^2 + 4xy$. **Right**: $f(x,y) = 2x^2 + y^2 + 4xy + \frac{4}{3}y^3 - \frac{1}{4}y^4$. We can observe that baseline methods fail to converge to a local minimax, whereas `GDA-AM` with table size $p = 3$ always exhibits desirable behaviors.

5.2 Our Method: `GDA-AM`

Our contributions: In this thesis, we propose a different approach to solve minimax optimization. Our starting point is to cast the GDA dynamics as a fixed-point iteration. We then highlight that the fixed-point iteration can be solved effectively by using advanced non-linear extrapolation methods such as Anderson Mixing [4], which we name as `GDA-AM`. redAlthough first mentioned in [6], to our best knowledge, this is still the first work to investigate and improve the GDA dynamics by tapping into advanced fixed-point algorithms.

We demonstrate that GDA dynamics can benefit from Anderson Mixing. In particular, we study bilinear games and give a systematic analysis of `GDA-AM` for both

simultaneous and alternating versions of GDA. We theoretically show that `GDA-AM` can achieve global convergence guarantees under mild conditions.

We complement our theoretical results with numerical simulations across a variety of minimax problems. We show that for some convex-concave and non-convex-concave functions, `GDA-AM` can converge to the optimal point with little hyper-parameter tuning whereas existing first-order methods are prone to divergence and cycling behaviors.

We also provide empirical results for GAN training across two different datasets, CIFAR10 and CelebA. Given the limited computational overhead of our method, the results suggest that an extrapolation add-on to GDA can lead to significant performance gains. Moreover, the convergence behavior across a variety of problems and the ease-of-use demonstrate the potential of `GDA-AM` to become the minimax optimization workhorse.

5.2.1 Fixed-Point Iteration and Anderson Mixing (AM)

Definition 4. $\mathbf{w}^\star$ *is a fixed point of the mapping* g *if* $\mathbf{w}^\star = g(\mathbf{w}^\star)$.

Consider the simple fixed-point iteration $w_{t+1} = g(w_t)$ which produces a sequence of iterates $\{w_0, w_1, \cdots, w_N\}$. In most cases, this converges to the fixed-point, $w^* = g(w^*)$. Take gradient descent as an example, it can be viewed as iteratively applying the operation: $w_{t+1} = g(w_t) \triangleq w_t - \alpha_t \nabla f(w_t)$, where the limit is the fixed-point $\mathbf{w}^\star = g(\mathbf{w}^\star)$ (i.e.$\nabla f(w_t = 0)$. SimGDA updates can be defined as the repeated application of a nonlinear operator:

$$\mathbf{w}_{t+1} = G_\eta^{(\mathrm{sim})}(\mathbf{w}_t) \triangleq \mathbf{w}_t - \eta V(\mathbf{w}_t) \text{ with } \mathbf{w} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}, V(\mathbf{w}) = \begin{bmatrix} \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) \\ -\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \end{bmatrix}$$

Similarly, we can write AltGDA updates as $\mathbf{w}_{t+1} = G_\eta^{(\mathrm{alt})}(\mathbf{w}_t)$. An issue with fixed-

point iteration is that it does not always converge, and even in the cases where it does converge, it might do so very slowly. GDA is one example that it could result in the possibility of the operator converging to a limit cycle instead of a single point for the GDA dynamic. A way of dealing with these problems is to use acceleration methods, which can potentially speed up the convergence process and in some cases even decrease the likelihood for divergence.

There are many different acceleration methods, but we will put our focus on an algorithm which we refer to as Anderson Mixing (or Anderson Acceleration). In short, Anderson Mixing (AM) shares the same idea as Nesterov's acceleration. Given a fixed-point iteration $w_t = g(w_{t-1})$, Anderson Mixing argues that a good approximation to the final solution w^* can be obtained as a linear combination of the previous p iterates $w_{t+1} = \sum_{i=0}^{p} \beta_i g(w_{t-p_t+i})$. Since obtaining the proper coefficients β_i is a nonlinear procedure, Anderson Mixing is also known as a nonlinear extrapolation method. The general form of Anderson Mixing is shown in Algorithm 5.1. For efficiency, we prefer a 'restarted' version with a small table size p that cleans up the table F every p iterations because it avoids solving a linear system of increasing size.

Algorithm 5.1 Anderson Mixing Prototype (truncated version)

Input: Initial point w_0, Anderson restart dimension p, fixed-point mapping $g : \mathbf{R}^n \to \mathbf{R}^n$.

Output: w_{t+1}

for *t = 0, 1, ...* **do**

Set $p_t = \min\{t, p\}$.

Set $F_t = [f_{t-p_t}, \ldots, f_t]$, where $f_i = g(w_i) - w_i$ for each $i \in [t - p_t \,..\, t]$.

Determine weights $\beta = (\beta_0, \ldots, \beta_{p_t})^T$ that solves $\min_\beta \|F_t\beta\|_2$, s. t. $\sum_{i=0}^{p_t} \beta_i = 1$.

Set $w_{t+1} = \sum_{i=0}^{p_t} \beta_i g(w_{t-p_t+i})$.

end

5.2.2 AM and Generalized Minimal Residual (GMRES)

Developed by Saad and Schultz [103], Generalized Minimal Residual method (GMRES) is a Krylov subspace method for solving linear system equations. The method approximates the solution by the vector in a Krylov subspace with minimal residual, which is described below.

Definition 5. *Assume we have the linear system of equations* $\mathbf{Ax} = \mathbf{b}$ *with* $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{b} \in \mathbb{R}^n$ *and an initial guess* $\mathbf{x}_0$*. Then we denote the initial residual by* $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$ *and define the* t*th Krylov subspace as* $\mathcal{K}_t = span\{\mathbf{r}_0, \mathbf{Ar}_0, \cdots, \mathbf{A}^{t-1}\mathbf{r}_0\}$*.*

The tth iterate $\mathbf{x}_t$ of GMRES minimizes the norm of the residual $\mathbf{r}_t = \mathbf{b} - \mathbf{Ax}_t$ in $\mathcal{K}_t$, that is, $\mathbf{x}_t$ solves

$$\min_{\mathbf{x}_t \in \mathbf{x}_0 + \mathcal{K}_t} \|\mathbf{b} - \mathbf{Ax}_t\|_2 .$$

The following formulation is equivalent to GMRES minimization problem and more convenient for implementation. It computes $\widehat{\mathbf{x}}_t$ such that

$$\widehat{\mathbf{x}}_t = \underset{\widehat{\mathbf{x}}_t \in \mathcal{K}_t}{\arg\min} \|\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \widehat{\mathbf{x}}_t)\|_2 = \underset{\widehat{\mathbf{x}}_t \in \mathcal{K}_t}{\arg\min} \|\mathbf{r}_0 - \mathbf{A}\widehat{\mathbf{x}}_t\|_2 .$$

Using a larger Krylov dimension will improve the convergence of the method, but will require more memory. For this reason, a smaller Krylov subspace dimension t and 'restarted' versions of the method are used in practice [104].

The convergence of GMRES can be studied through the magnitude of the residual polynomial.

Theorem 5.2.1 (Lemma 6.31 of [104])**.** *Let* $\widehat{\mathbf{x}}_t$ *be the approximate solution obtained at the t-th iteration of GMRES being applied to solve* $\mathbf{Ax} = \mathbf{b}$*, and denote the residual as* $\mathbf{r}_t = \mathbf{b} - \mathbf{A}\widehat{\mathbf{x}}_t$*. Then,* $\mathbf{r}_t$ *is of the form*

$$\mathbf{r}_t = f_t(\mathbf{A})\mathbf{r}_0, \tag{5.5}$$

where

$$\|\mathbf{r}_t\|_2 = \|f_t(\mathbf{A})\mathbf{r}_0\|_2 = \min_{f_t \in \mathcal{P}_t} \|f_t(\mathbf{A})\mathbf{r}_0\|_2, \tag{5.6}$$

where $\mathcal{P}_p$ is the family of polynomials with degree p such that $f_p(0) = 1, \forall f_p \in \mathcal{P}_p$, which are usually called residual polynomials.

Although GMRES is applied to a system of linear equations not a fixed-point problem, there is a strong connection between Anderson Mixing and GMRES. In AM we are looking for a fixed-point $\mathbf{x}$ such that $\mathbf{G}\mathbf{x} - \mathbf{b} - \mathbf{x} = 0$ and by rearranging this equation we get

$$\mathbf{b} + (\mathbf{G} - \mathbf{I})\mathbf{x} = 0 \Leftrightarrow (\mathbf{I} - \mathbf{G})\mathbf{x} = \mathbf{b}.$$

Theorem 5.2.2 shows that if GMRES is applied to the system $(\mathbf{I}-\mathbf{G})\mathbf{x} = \mathbf{b}$ and AM is applied to $g(\mathbf{x}) = \mathbf{G}\mathbf{x} + \mathbf{b}$ with the same initial guess and $\mathbf{I} - \mathbf{G}$ is non-singular, then these are equivalent in the sense that the iterates of each algorithm can be obtained directly from the iterates of the other algorithm.

Theorem 5.2.2 (Equivalence between AM with restart and GMRES [122])**.** *Consider the fixed point iteration $\mathbf{x} = g(\mathbf{x})$ where $g(\mathbf{x}) = \mathbf{G}\mathbf{x} + \mathbf{b}$ for $\mathbf{G} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$. If $\mathbf{I} - \mathbf{G}$ is non-singular, Algorithm 5.1 produces exactly the same iterates as GMRES being applied to solve $(\mathbf{I} - \mathbf{G})\mathbf{x} = \mathbf{b}$ when both algorithms start with the same initial guess.*

Theorem 5.2.2 can also be generalized to the restart version of AM an GMRES as well.

5.2.3 `GDA-AM`

We propose a novel minimax optimizer, called `GDA-AM`, that is inspired by recent advances in parameter (or weight) averaging [131, 137]. We argue that a nonlinear

adaptive average (combination) is a more appropriate choice for minimax optimization.

We propose to exploit the dynamic information present in the GDA iterates to "smartly" combine the past iterates. This is in contrast to the classical averaging methods (moving averaging and exponential moving averaging) [135] that "blindly" combine past iterates. A naïve adoption of Anderson Mixing using the past p GDA iterates for both simGDA and altGDA has the following form:

$$\text{Anderson mixing}: \quad \mathbf{x}_{t+1} = \sum_{i=0}^{p} \beta_i \mathbf{x}_{t-p+i}, \mathbf{y}_{t+1} = \sum_{i=0}^{p} \beta_i \mathbf{y}_{t-p+i}. \tag{5.7}$$

Since [37, 141] show the AltGDA is superior to SimGDA in many aspects, we briefly summarized both Simultaneous and Alternating `GDA-AM` in Algorithms 5.2 and 5.3 with the truncated Anderson Mixing Algorithm 5.1 using a table size p.

Algorithm 5.2 Simultaneous `GDA-AM`

Input: $\mathbf{x}_0, \mathbf{y}_0$, stepsize η, Anderson table size p

Output: $\mathbf{x}_t, \mathbf{y}_t$

Set $\mathbf{w}_0 = [\mathbf{x}_0, \mathbf{y}_0], sx = length(x_0)$

for *t = 0, 1, ...* **do**

$\mathbf{x}_t, \mathbf{y}_t = \mathbf{w}_t[0 : sx - 1], \mathbf{w}_t[sx : end]$

$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t)$

$\mathbf{y}_{t+1} = \mathbf{y}_t - \eta \nabla_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y}_t)$

$\mathbf{w}_{t+1} = \begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_{t+1} \end{bmatrix}$

Use Anderson Mixing with table size p to extrapolate $\mathbf{w}_{t+1}$

end

$\mathbf{x}_t, \mathbf{y}_t = \mathbf{w}_{t+1}[0 : sx-1], \mathbf{w}_{t+1}[sx : end]$

return $\mathbf{x}_t, \mathbf{y}_t$

Algorithm 5.3 Alternating `GDA-AM`

Input: $\mathbf{x}_0, \mathbf{y}_0$, stepsize η, Anderson table size p

Output: $\mathbf{x}_t, \mathbf{y}_t$

Set $\mathbf{w}_0 = [\mathbf{x}_0, \mathbf{y}_0], sx = length(x_0)$

for *t = 0, 1, ...* **do**

$\mathbf{x}_t, \mathbf{y}_t = \mathbf{w}_t[0 : sx - 1], \mathbf{w}_t[sx : end]$

$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t)$

$\mathbf{y}_{t+1} = \mathbf{y}_t - \eta \nabla_{\mathbf{y}} f(\mathbf{x}_{t+1}, \mathbf{y}_t)$

$\mathbf{w}_{t+1} = \begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_{t+1} \end{bmatrix}$

Use Anderson Mixing with table size p to extrapolate $\mathbf{w}_{t+1}$

end

$\mathbf{x}_t, \mathbf{y}_t = \mathbf{w}_{t+1}[0 : sx-1], \mathbf{w}_{t+1}[sx : end]$

return $\mathbf{x}_t, \mathbf{y}_t$

Implementation details It is important to note that the Anderson Mixing form shown in Algorithm 5.1 is for illustrative purpose and not computationally efficient. For example, only one column of F_t needs to be updated at each iteration. In addition, the solution of the least-square problem in Algorithm 5.1 can also be solved by a quick QR update scheme which costs $(2n + 1)p^2$ [122]. Thus, from Algorithms 5.2 and 5.3, we can see that the major cost of `GDA-AM` arises from solving the additional linear least squares problem compared to regular GDA at each iteration.

Fast Implementation Details

In this section, we discuss the efficient implementation of Anderson Mixing. We start with generic Anderson Mixing prototype (Algorithm 6.3) and then present the idea

of Quick QR-update Anderson Mixing implementation as described in Walker and Ni [123], which is commonly used in practice. For each iteration $t \geq 0$, AM prototype solves a least squares problem with a normalization constraint. The intuition is to minimize the norm of the weighted residuals of the previous m iterates.

Algorithm 5.4 Anderson Mixing Prototype (truncated version)

Input: Initial point w_0, Anderson restart dimension p, fixed-point mapping $g : \mathbf{R}^n \rightarrow \mathbf{R}^n$.

Output: w_{t+1}

for *t = 0, 1, ...* **do**

Set $p_t = \min\{t, p\}$.

Set $F_t = [f_{t-p_t}, \ldots, f_t]$, where $f_i = g(w_i) - w_i$ for each $i \in [t - p_t \,..\, t]$.

Determine weights $\beta = (\beta_0, \ldots, \beta_{p_t})^T$ that solves $\min_\beta \|F_t\beta\|_2$, s. t. $\sum_{i=0}^{p_t} \beta_i = 1$.

Set $w_{t+1} = \sum_{i=0}^{p_t} \beta_i g\left(w_{t-p_t+i}\right)$.

end

The constrained linear least-squares problem in Algorithm AA can be solved in a number of ways. Our preference is to recast it in an unconstrained form suggested in Fang and Saad [30], Walker and Ni [123] that is straightforward to solve and convenient for implementing efficient updating of QR.

Define $f_i = g(w_i) - w_i$, $\triangle f_i = f_{i+1} - f_i$ for each i and set $F_t = [f_{t-p_t}, \ldots, f_t]$, $\mathcal{F}_t = [\triangle f_{t-p_t}, \ldots, \triangle f_t]$. Then solving the least-squares problem ($\min_\beta \|F_t\beta\|_2$, s. t. $\sum_{i=0}^{p_t} \beta_i = 1$) is equivalent to

$$\min_{\gamma=\left(\gamma_0,\ldots,\gamma_{p_t-1}\right)^T} \|f_t - \mathcal{F}_t\gamma\|_2 \tag{5.8}$$

where α and γ are related by $\alpha_0 = \gamma_0, \alpha_i = \gamma_i - \gamma_{i-1}$ for $1 \leq i \leq p_t - 1$, and $\alpha_{p_t} = 1 - \gamma_{p_t-1}$.

Now the inner minimization subproblem can be efficiently solved as an uncon-

strained least squares problem by a simple variable elimination. This unconstrained least-squares problem leads to a modified form of Anderson Mixing

$$w_{t+1} = g\left(w_t\right) - \sum_{i=0}^{p_t-1} \gamma_i^{(t)} \left[g\left(w_{t-p_t+i+1}\right) - g\left(w_{t-p_t+i}\right)\right] = g\left(w_t\right) - \mathcal{G}_t \gamma^{(t)}$$

where $\mathcal{G}_t = [\triangle g_{t-p_t}, \ldots, \triangle g_{t-1}]$ with $\triangle g_i = g(w_{i+1}) - g(w_i)$ for each i.

To obtain $\gamma^{(t)} = \left(\gamma_0^{(t)}, \ldots, \gamma_{p_t-1}^{(t)}\right)^T$ by solving (5.8) efficiently, we show how the successive least-squares problems can be solved efficiently by updating the factors in the QR decomposition $\mathcal{F}_t = Q_t R_t$ as the algorithm proceeds. We assume a think QR decomposition, for which the solution of the least-squares problem is obtained by solving the $p_t \times p_t$ linear system $R\gamma = Q' * f_t$. Each $\mathcal{F}_t$ is $n \times p_t$ and is obtained from $\mathcal{F}_{t-1}$ by adding a column on the right and, if the resulting number of columns is greater than p, also cleaning up (re-initialize) the table. That is,we never need to delete the left column because cleaning up the table stands for a restarted version of AM. As a result, we only need to handle two cases; 1 the table is empty(cleaned). 2 the table is not full. When the table is empty, we initialize $\mathcal{F}_1 = Q_1 R_1$ with $Q_1 = \triangle f_0 / \left\|\triangle f_0\right\|_2$ and $R = \left\|\triangle f_0\right\|_2$. If the table size is smaller than p, we add a column on the right of $\mathcal{F}_{t-1}$. Have $\mathcal{F}_{t-1} = QR$, we update Q and R so that $\mathcal{F}_t = [\mathcal{F}_{t-1}, \Delta f_{t-1}] = QR$. It is a single modified Gram–Schmidt sweep that is described as follows:

Algorithm 5.5 QR-updating procedures

for $i = 1, \ldots, p_{t-1}$ **do**

Set $R(i, p_t) = Q(:, i)' * \triangle f_{t-1}$.

Update $\triangle f_{t-1} \leftarrow \Delta f_{t-1} - R\left(i, p_t\right) * Q(:, i)$

end

Set $Q\left(:, p_t\right) = \triangle f_{t-1} / \left\|\triangle f_{t-1}\right\|_2$ and $R\left(p_t, p_t\right) = \left\|\Delta f_{t-1}\right\|_2$

Note that we do not explicitly conduct QR decomposition in each iteration, instead we update the factors ($O(p^2 n)$) and then solve a linear system using back substitution

which has a complexity of $O(p^2)$. Based on this complexity analysis, we can find Anderson Mixing with QR-updating scheme has limited computational overhead than GDA (or OG). This explains why `GDA-AM` is faster than EG but slower than OG in terms of running time of each iteration.

5.3 Convergence results for `GDA-AM`

In this section, we show that both simultaneous and alternating version `GDA-AM` converge to the equilibrium for bilinear problems. First, we do not require the learning rate to be sufficiently small. Second, we explicitly provide a linear convergence rate that is faster than EG and OG. More importantly, we derive nonasymptotic rates from the spectrum analysis perspective because existing theoretical results can not help us derive a convergent rate.

Bilinear Games

Bilinear games are often regarded as an important simple example for theoretically analyzing and understanding new algorithms and techniques for solving general minimax problems [36, 82, 107]. In this section, we analyze the convergence property of simultaneous `GDA-AM` and alternating `GDA-AM` schemes on the following zero-sum bilinear games:

$$\min_{\mathbf{x}\in\mathbb{R}^n}\max_{\mathbf{y}\in\mathbb{R}^n} f(\mathbf{x},\mathbf{y}) = \mathbf{x}^T\mathbf{A}\mathbf{y} + \mathbf{b}^T\mathbf{x} + \mathbf{c}^T\mathbf{y}, \quad \mathbf{A} \text{ is full rank.} \tag{5.9}$$

The Nash equilibrium to the above problem is given by $(\mathbf{x}^*, \mathbf{y}^*) = (-\mathbf{A}^{-T}\mathbf{c}, -\mathbf{A}^{-1}\mathbf{b})$.

We also investigate bilinear-quadratic games from a spectrum analysis perspective. In addition, we show that analysis based on the numerical range [12] can be also extended to such games, although it can not help derive a convergent bound for (5.9). Detailed discussion can be found in Appendix 5.6 and 5.6.

Simultaneous `GDA-AM`

Suppose $\mathbf{x}_0$ and $\mathbf{y}_0$ are the initial guesses for $\mathbf{x}^*$ and $\mathbf{y}^*$, respectively. Then each iteration of simultaneous GDA can be written in the following matrix form:

$$\begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I} & -\eta\mathbf{A} \\ \eta\mathbf{A}^T & \mathbf{I} \end{bmatrix}}_{\mathbf{G}^{(Sim)}} \underbrace{\begin{bmatrix} \mathbf{x}_t \\ \mathbf{y}_t \end{bmatrix}}_{\mathbf{w}_t^{(Sim)}} - \eta \underbrace{\begin{bmatrix} \mathbf{b} \\ \mathbf{c} \end{bmatrix}}_{\mathbf{b}^{(Sim)}}. \tag{5.10}$$

It has been shown that the iteration in (5.10) often cycles and fails to converge for the bilinear problem due to the poor spectrum/numerical range of the fixed point operator $\mathbf{G}^{(Sim)}$ [6, 36, 86]. Next we show that the convergence can be improved with Algorithm 5.2.

Theorem 5.3.1. *[Global convergence for simultaneous* `GDA-AM` *on bilinear problem] Denote the distance between the stationary point* $\mathbf{w}^*$ *and current iterate* $\mathbf{w}_{(k+1)p}$ *of Algorithm 5.2 with table size* p *as* $N_{(k+1)p} = \|\mathbf{w}^* - \mathbf{w}_{(k+1)p}\|$. *Then we have the following bound for* N_t

$$N_{(k+1)p}^2 \leq \rho(A) N_{kp}^2 \tag{5.11}$$

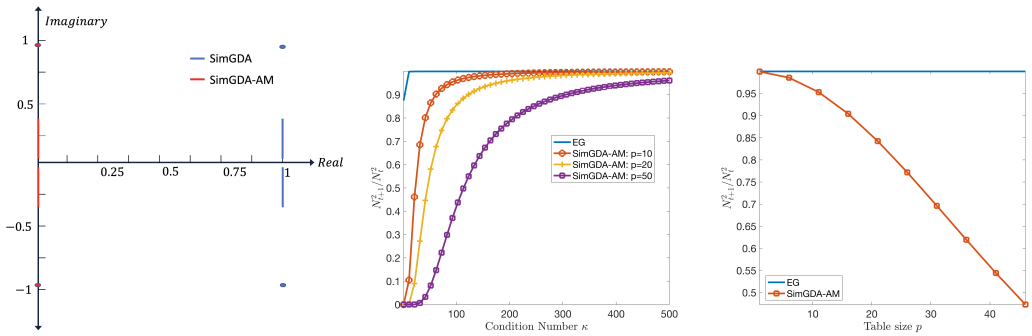
where $\rho(A) = (\frac{1}{T_p(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1})})^2$. *Here,* T_p *is the Chebyshev polynomial of first kind of degree* p *and* $\frac{1}{T_p(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1})} < 1$ *since* $1 + \frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1} > 1$.

It is worthy emphasizing that the convergence rate of Algorithm 5.2 is independent of learning rate η while the convergence results of other methods like EG and OG depend on the learning rate.

Remark 5.3.1.1. *Both EG and OG have the following form of convergence rate [86] for bilinear problem*

$$N_{t+1}^2 \leq (1 - \frac{c}{\kappa(\mathbf{A}^T\mathbf{A})}) N_t^2,$$

where c *is a positive constant independent of the problem parameters.*



(a) Eigenvalues of iteration matrix of SimGDA and GDA-AM

(b) Different condition number

(c) Different table size, condition number $\kappa = 100$

Figure 5.2: **Figure 5.2a**: The blue line is the spectrum of matrix $\mathbf{G}^{(Sim)}$ while the red line is spectrum of matrix $\mathbf{I} - \mathbf{G}^{(Sim)}$. Our method transforms the divergent problem to a convergent problem due to the transformed spectrum. **Figure 5.2b**: Convergence rate comparison between SimGDA-AM and EG for different condition numbers of $\mathbf{A}$ and fixed table size $p = 10, 20, 50$. **Figure 5.2c**: Convergence rate comparison between SimGDA-AM and EG for increasing table size on a matrix $\mathbf{A}$ with condition number 100.

Alternating GDA-AM

The underlying fixed point iteration in Algorithm 5.3 can be written in the following matrix form:

$$\begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I} & -\eta\mathbf{A} \\ \eta\mathbf{A}^T & \mathbf{I} - \eta^2\mathbf{A}^T\mathbf{A} \end{bmatrix}}_{\mathbf{G}^{(Alt)}} \underbrace{\begin{bmatrix} \mathbf{x}_t \\ \mathbf{y}_t \end{bmatrix}}_{\mathbf{w}_t^{(Alt)}} - \eta \underbrace{\begin{bmatrix} \mathbf{b} \\ \mathbf{c} \end{bmatrix}}_{\mathbf{b}^{(Alt)}}.$$

According to the equivalence between truncated Anderson acceleration and GMRES with restart, we can analyze the convergence of Algorithm 5.3 through the convergence analysis of applying GMRES to solve linear systems associated with $\mathbf{G} = \mathbf{I} - \mathbf{G}^{(Alt)}$:

$$\mathbf{G} = \begin{bmatrix} \mathbf{0} & \eta\mathbf{A} \\ -\eta\mathbf{A}^T & \eta^2\mathbf{A}^T\mathbf{A} \end{bmatrix}.$$

Theorem 5.3.2. *[Global convergence for alternating GDA-AM on bilinear problem] Denote the distance between the stationary point* $\mathbf{w}^*$ *and current iterate* $\mathbf{w}_{(k+1)p}$ *of Algorithm 5.3 with table size* p *as* $N_{(k+1)p} = \|\mathbf{w}^* - \mathbf{w}_{(k+1)p}\|$. *Assume* $\mathbf{A}$ *is normalized such that its largest singular value is equal to* 1. *Then when the learning rate* η *is less than* 2, *we have the following bound for* N_t

$$N^2_{(k+1)p} \leq \sqrt{1 + \frac{2\eta}{2-\eta}}(\frac{r}{c})^p N^2_{kp}$$

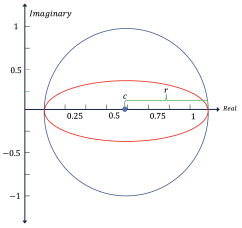


Figure 5.3: An illustration of the spectrum of $\mathbf{G}$ (red) and the closing circle (blue) in Theorem 5.3.2.

where c *and* r *are the center and radius of a disk* $D(c, r)$ *which includes all the eigenvalues of* $\mathbf{G}$. *Especially,* $\frac{r}{c} < 1$.

Theorem 5.3.2 shows that when $p > \frac{\log \sqrt{\frac{2-\eta}{2+\eta}}}{\log \frac{r}{c}}$, alternating GDA-AM will converge globally.

Discussion of obtained rates We would like to first explain on why taking Chebyshev polynomial of degree p at the point $1+\frac{2}{\kappa-1}$. We evaluate the Chebyshev polynomial at this specific point because the reciprocal of this value gives the minimal value of infinite norm of the all polynomials of degree p defined on the interval $\tilde{I} = [\eta^2\sigma^2_{min}(\mathbf{A}), \quad \eta^2\sigma^2_{max}(\mathbf{A})]$ based on Theorem 6.25 (page 209) [104]. In other words, taking the function value at this point leads to the tight bound.

When comparing between existing bounds, we would like to point our our derived bounds are hard to compare directly. Alternatively, we can derive another bound for comparison with existing bounds for simultaneous GDA-AM. If we use the inequality that $T_p(t) \geq \frac{1}{2}((t + \sqrt{t^2-1})^p)$, we can obtain the bound $\rho(A) = 4(\frac{\sqrt{\kappa(A^TA)}-1}{\sqrt{\kappa(A^TA)}+1})^2 = 4(1 - O(\frac{1}{\sqrt{\kappa(A^TA)}}))$, which is in a form that is comparable with EG and can compete with EG + positive momentum. The numerical experiments in figure 2b numerically verify that our bound is smaller than EG. We wanted to numerically compare our

rate with EG with positive momentum. However the bound of EG with positive momentum is asymptotic. Moreover, it does not specify the constants so we can not numerically compare them. We do provide empirical comparison between GDA-AM and EG with positive momentum for bilinear problems in Appendix **??**. It shows GDA-AM outperforms EG with positive momentum. Regarding alternating `GDA-AM` , we would like to note that the bound in Theorem 5.3.2 depends on the eigenvalue distribution of the matrix $\mathbf{G}$. Condition number is not directly related to the distribution of eigenvalues of a nonsymmetric matrix $\mathbf{G}$. Thus, the condition number is not a precise metric to characterize the convergence. If these eigenvalues are clustered, then our bound can be small. On the other hand, if these eigenvalues are evenly distributed in the complex plane, then the bound can very close to 1.

More importantly, we would like to stress several technical contributions.

1 : Our obtained Theorem 5.3.1 and 5.3.2 provide nonasymptotic guarantees, while most other work are asymptotic. For example, EG with positive momentum can achieve a asymptotic rate of $1 - O(1/\sqrt{\kappa})$ under strong assumptions [6].

2 : Our contribution is not just about fix the convergence issue of GDA by applying Anderson Mixing; another contribution is that we arrive at a convergent and tight bound on the original work and not just adopting existing analyses. We developed Theorem 5.3.1 and 5.3.2 from a new perspective because applying existing theoretical results fail to give us neither convergent nor tight bounds.

3 : Theorem 5.3.1 and 5.3.2 only requires mild conditions and reflects how the table size p controls the convergence rate. Theorem 5.3.1 is independent of the learning rate η. However, the convergence results of other methods like EG and OG depend on the learning rate, which may yield less than desirable results for ill-specified learning rates.

5.4 Related Work

There is a rich literature on different strategies to alleviate the issue of minimax optimization. A useful add-on technique, Momentum, has been shown to be effective for bilinear games and strongly-convex-strongly-concave settings [6, 37, 140]. Several second-order methods [2, 80, 83, 98] show that their stable fixed points are exactly either Nash equilibria or local minimax by incorporating second-order information. However, such methods are computationally expensive and thus unsuitable for large applications such as image generation. Focusing on variants of GDA, EG and OG are two widely studied algorithms on improving the GDA dynamics. EG proposed to apply extra-gradient to overcome the cycling behaviour of GDA. OG, originally proposed in [101] and rediscovered in [22, 82], is more efficient by storing and re-using the extrapolated gradient for the extrapolation step. Without projection, OG is equivalent to extrapolation from past. [87] shows that both of these algorithms can be interpreted as approximations of the classical proximal point method and did a unified analysis for bilinear games. These approaches mentioned the GDA dynamics can be viewed as a fixed-point iteration, but none of them further provides a solution to improve it. In this work, we fill this gap by proposing the application of the extrapolation method directly on the entire GDA dynamics. Unlike OG, EG and their variants [57, 72, 116, 135], which regard minimax problems as variational inequality problems [14, 89], our work is from a new perspective and thus orthogonal to these previous approaches.

In addition, several recent works consider nonconvex-concave minimax problems. [143] introduced a "smoothing" scheme combined with GDA to stabilize the dynamic of GDA. [77] proposed a method called Stochastic Recursive gradiEnt Descent Ascent (SREDA) for stochastic nonconvex-strongly-concave minimax problems, by estimating gradients recursively and reducing its variance. [75] showed that the two-timescale GDA can find a stationary point of nonconvex-concave minimax problems effectively.

[94] proposed a variant of Nesterov's accelerated algorithm to find ϵ -first-order Nash equilibrium that is a stronger criterion than the commonly used proximal gradient norm. [91] proposed a iterative method that finds ϵ -first-order Nash equilibrium in $O(\epsilon^{-2})$ iterations under Polyak-Lojasiewicz (PL) condition. Focusing on nonconvex minimax problems, they studied an interesting and difficult problem. Since our work cast insight on the effectiveness of solving minimax optimization via Anderson Mixing, we expect the extension of this algorithm to general nonconvex problems can be further investigated in the future.

5.5 Experiments

In this section, we conduct experiments to see whether `GDA-AM` improves GDA for minimax optimization from simple to practical problems. We first investigate performance of `GDA-AM` on bilinear games. In addition, we evaluate the efficacy of our approach on GANs.

Bilinear Problems

In this section, we answer following questions: **Q1:** How is `GDA-AM` perform in terms of iteration number and running time? **Q2:** How is the scalability of `GDA-AM` ? **Q3:** How is the performance of `GDA-AM` using different table size p? **Q4:** Does `GDA-AM` converge for large step size η?

We compare the performance with SimGDA, AltGDA, EG, and OG, and EG with Negative Momentum([6]) on bilinear minimax games shown in (5.9) without any constraint.

$\mathbf{A}, \mathbf{b}, \mathbf{c}$, and initial points are generated using normally distributed random number. We set the maximum iteration number as 1×10^6, stopping criteria 1×10^{-5} and depict convergence by use of the norm of distance to optima, which is defined as

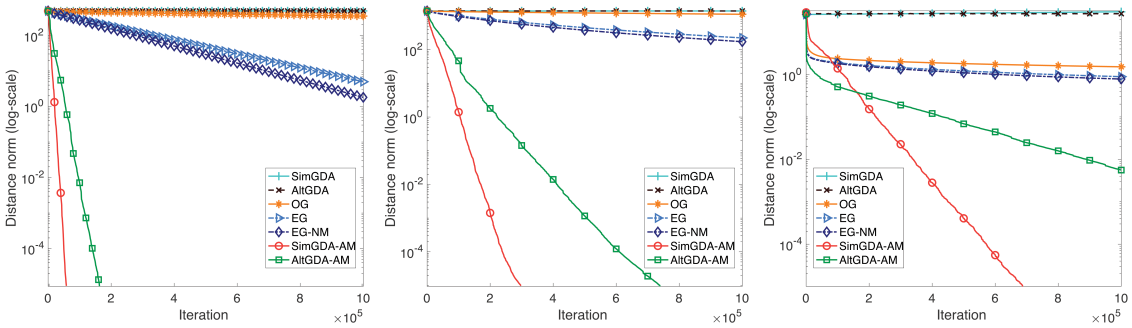
$\|\mathbf{w}^* - \mathbf{w}_t\|$. Similar to Azizian et al. [6], Wei et al. [127], the step size is set as 1 after rescaling $\mathbf{A}$ to have 2-norm 1. We present results of different settings in Figures 5.4, 5.5, and 5.6.

We first generate different problem size ($n = 100, 1000, 5000$) and present results of convergence in terms of iteration number in Figure 5.4. It can be observed that `GDA-AM` converges in much fewer iterations for different problem sizes. Note that EG, EG-NM, and OG converge in the end but requires many iterations, thus we plot only a portion for illustrative purposes. Figure 5.5 depicts the convergence for all methods in terms of time. It can be observed that the running time of `GDA-AM` is faster than EG. Although slower than OG, we can observe `GDA-AM` converges in much less time for all problems. Figure 5.4 and Figure 5.5 answer Q1 and Q2; although there is additional computation for `GDA-AM` , it does not hinder the benefits of adopting Anderson Mixing. Even for a large problem size, `GDA-AM` still converges in much less time than the baselines.

Next, we run `GDA-AM` using different table size p and show the results in Figure 5.6a and Figure 5.6b. Figure 5.6a indicates an increasing of table size results in faster convergence in terms of iteration number, which also verifies our claim in Theorem 5.3.1. However, we also observe an increased running time when using a larger table size in Figure 5.6b. Further, we can see that $p = 50$ converges in a comparable time and iterations to $p = 100$. Similar results are found in repeated experiments as well. As a result, our answer to Q3 is that although a larger p means less iterations, a medium p is sufficient and a small p still outperforms the baselines. The optimal choice of p is related to the condition number and step size, which is another interesting topic in the Anderson Mixing community.

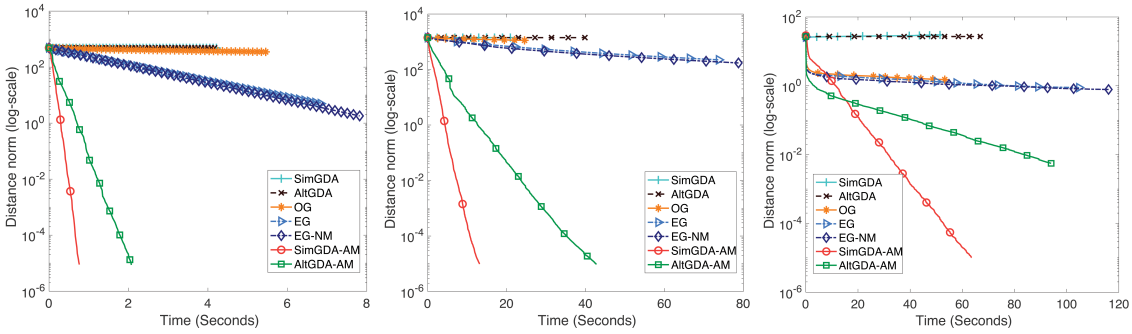
Next, we answer Q4 on convergence under different step sizes. Although `GDA-AM` usually converges with suitable step size, our theorem suggests it requires a larger table size when combined with a extremely aggressive step size. Figure 5.6c shows the

convergence under such circumstance. We can observe that although a very large step size goes the wrong way in the beginning, Anderson Mixing can still make it back on track except when $\eta > 1$. It answers the question and confirms our claim that `GDA-AM` can achieve global convergence for bilinear problems for a large step size $\eta > 0$.



(a) $n = 100$ (b) $n = 500$ (c) $n = 1000$

Figure 5.4: Comparison in terms of iteration: $\min_{\mathbf{x}} \max_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{A}\mathbf{y} + \mathbf{b}^T\mathbf{x} + \mathbf{c}^T\mathbf{y}$. We use different problem size and fix $p = 10, \eta = 1$ for all experiments.

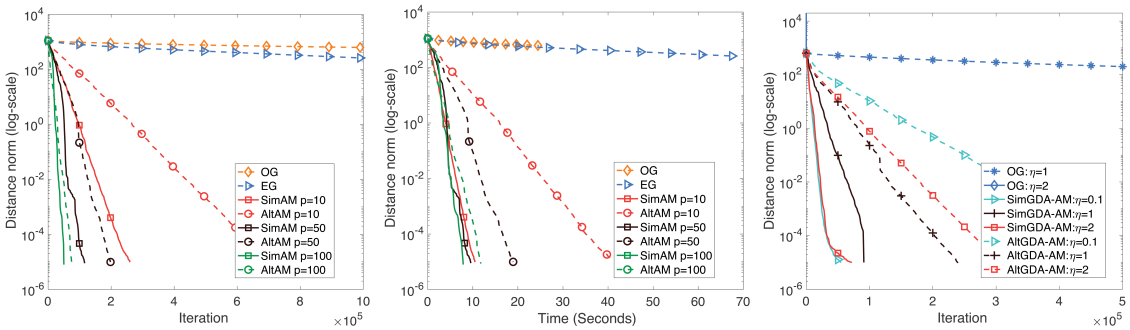


(a) Time comparison for Figure 5.4a (b) Time comparison for Figure 5.4b (c) Time comparison for Figure 5.4c

Figure 5.5: Comparison between methods in terms of time.

Robust Neural Network Training In this section, we test the effectiveness of `GDA-AM` by training a robust neural network on MNIST data set against adversarial attacks [40, 69, 79] . The optimization formulation is

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \max_{\delta_i,\ \text{s.t.}\ |\delta_i|_\infty \leq \varepsilon} \ell\left(f\left(x_i + \delta_i; \mathbf{w}\right), y_i\right) \tag{5.12}$$



(a) Effects of p in terms of iteration (b) Time compasion for Figure 5.6a (c) Effect of step size η, $p = 10$

Figure 5.6: Effects of table size p and step size η, $n = 500$

where w is the parameter of the neural network, the pair (x_i, y_i) denotes the i-th data point, and δ_i is the perturbation added to data point i. The accuracy of our formulation against popular attacks, FGSM [40] and PGD [69], are summarized in Table 5.1.. Since solving such problem is computationally challenging, [91] proposed an approximation of the above optimization problem with a new objective function as the following nonconvex-concave problem:

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \max_{\mathbf{t}\in\mathcal{T}} \sum_{j=0}^{9} t_j \ell\left(f\left(x_{ij}^K; \mathbf{w}\right), y_i\right), \mathcal{T} = \left\{(t_1, \cdots, t_m) \mid \sum_{i=1}^{m} t_i = 1, t_i \geq 0\right\} \quad (5.13)$$

where K is a parameter in the approximation, and x_{ij}^K is an approximated attack on sample x_i by changing the output of the network to label j. We use the public available implementation [91] [1]. We apply our algorithm on top of [91] and compare our results ($p = 50$) with [79, 91, 142, 143]. Results are summarized in table 5.1. We can observe that `GDA-AM` leads to a comparable or slightly better performance to the other methods. In addition, `GDA-AM` does not exhibit a significant drop in accuracy when ϵ is larger and this suggests the learned model is more robust.

[1] https://github.com/optimization-for-data-driven-science/Robust-NN-Training

	Natural	FGSM L_∞			PGD^{40} L_∞		
		$\varepsilon = 0.2$	$\varepsilon = 0.3$	$\varepsilon = 0.4$	$\varepsilon = 0.2$	$\varepsilon = 0.3$	$\varepsilon = 0.4$
[79]	98.58%	96.09%	94.82%	89.84%	94.64%	91.41%	78.67%
Trade: $\varepsilon = 0.35$	97.37%	95.47%	94.86%	79.04%	94.41%	92.69%	85.74%
Trade: $\varepsilon = 0.40$	97.21%	96.19%	96.17%	96.14%	95.01%	94.36%	94.11%
[91]	98.20%	97.04%	96.66%	96.23%	96.00%	95.17%	94.22%
[143]	**98.89**%	**97.87**%	97.23%	95.81%	**96.71**%	95.62%	94.51%
`GDA-AM`	98.61%	97.75%	**97.74**%	**97.75**%	96.47%	**95.91**%	**95.41**%

Table 5.1: Test accuracies under FGSM and PGD attack. Trade refers to [142].

GAN Experiments: Image Generation We apply our method to the CIFAR10 dataset [68] and use the ResNet architecture with WGAN-GP [43] and SNGAN [85] objective. We also compared the performance of `GDA-AM` using cropped CelebA (64×64) [76] on WGAN-GP. We compare with Adam and extra-gradient with Adam (EG) as it offers significant improvement over OG. Models are evaluated using the inception score (IS) [106] and FID [53] computed on 50,000 samples. For fair comparison, we fixed the same hyperparamters of Adam for all methods after an extensive search. Experiments were run with 5 random seeds. We show results in Table 5.2. Table 5.2 reports the best IS and FID (averaged over 5 runs) achieved on these datasets by each method. We see that `GDA-AM` yields improvements over the baselines in terms of generation quality.

Table 5.2: Best inception scores and FID for Cifar10 and FID for CelebA (IS is a less informative metric for celebA).

	WGAN-GP(ResNet)			SNGAN(ResNet)	
	CIFAR10		CelebA	CIFAR10	
Method	IS ↑	FID ↓	FID	IS	FID
Adam	7.76 ±.11	22.45 ±.65	8.43 ±.05	8.21 ±.05	20.81 ±.16
EG	7.83 ±.08	20.73 ±.22	8.15 ±.06	8.15 ±.07	21.12 ±.19
Ours (GDA-AM)	**8.05** ±.06	**19.32** ±.16	**7.82** ±.06	**8.38** ±.04	**18.84** ±.13

5.6 Theoretical Results

Difficulty of analysis on GDA with Anderson Mixing

In the analysis, we study the inherent structures of the dynamics of the fixed point iteration and provide the convergence analysis for both simultaneous and alternating schemes. We want to emphasize that the direct application of existing convergence results of GMRES can not lead to convergent results. A recent paper [12] study the convergence acceleration schemes for multi-step optimization algorithms using Regularized Nonlinear Acceleration. We also want to point out that a naïve application of Crouzeix's bound to the minimax optimization problem can not be used to derive the convergent result.

Theorem 5.6.1 ([32]). *Let $n \geq 5$ be an integer, $r > 1$, and $c \in \mathbb{R}$. Consider the following constrained polynomial minmax problem*

$$\min_{p \in \mathbb{P}_n : p(c)=1} \max_{z \in E_r} |p(z)| \tag{5.14}$$

where

$$E_r := \left\{ z \in \mathbb{C} \;\middle|\; |z-1| + |z+1| \leq r + \frac{1}{r} \right\} \tag{5.15}$$

and $c \in \mathbb{C} \setminus E_r$. Then this problem can be solved uniquely by

$$t_n(z; c) := \frac{T_n(z)}{T_n(c)}, \tag{5.16}$$

where

$$T_n(z) = \frac{1}{2}\left(v^n + \frac{1}{v^n}\right), \quad z = \frac{1}{2}\left(v + \frac{1}{v}\right) \tag{5.17}$$

if

(a) $|c| \geq \frac{1}{2}\left(r^{\sqrt{2}} + r^{-\sqrt{2}}\right)$ *or*

(b) $|c| \geq (1/2a_r)\left(2a_r^2 - 1 + \sqrt{2a_r^4 - a_r^2 + 1}\right)$, *where* $a_r := \frac{1}{2}\left(r + \frac{1}{r}\right)$.

This is because the point 0 where all the residual polynomials take the fixed value of 1 is included in the numerical range of the iteration matrix, which violates the assumption of Theorem 5.6.1. As a result, it can not be used to prove that the residual norm is decreasing based on this approach. Instead, we show that although the coefficient matrix is non-normal, it is diagonalizable. We then give the convergence results based on the eigenvalues instead of the numerical range. More specifically, Anderson mixing is equivalent to GMRES being applied to solve the following linear system:

$$(\mathbf{I} - \mathbf{G}^{(Alt)})\mathbf{w} = \mathbf{b}^{(Alt)}, \quad \text{with } \mathbf{w}_0 = \mathbf{w}_0^{(Alt)}. \tag{5.18}$$

Writing this linear system in the block form:

$$\begin{bmatrix} \mathbf{0} & \eta\mathbf{A} \\ -\eta\mathbf{A}^T & \eta^2\mathbf{A}^T\mathbf{A} \end{bmatrix} \mathbf{w} = \mathbf{b}^{(Alt)}. \tag{5.19}$$

The residual norm bound for GMRES reads:

$$\|\mathbf{r}_t\|_2 = \min_{p\in\mathbb{P}_t^1} \|p(\mathbf{I} - \mathbf{G}^{(Alt)})\mathbf{r}_0\|_2. \tag{5.20}$$

Notice that the matrix $(\mathbf{I} - \mathbf{G}^{(Alt)})$ is non-normal. If we apply Crouzeix's bound in [21] to our problem as [12] did, then we have the following bound

$$\frac{\|\mathbf{r}_t\|_2}{\|\mathbf{r}_0\|_2} \leq \min_{p\in\mathbb{P}_t^1} \|p(\mathbf{I} - \mathbf{G}^{(Alt)})\| \leq (1+\sqrt{2}) \min_{p\in\mathbb{P}_t^1} \sup_{z\in W(\mathbf{I}-\mathbf{G}^{(Alt)})} \|p(z)\| \tag{5.21}$$

where $W(\mathbf{I} - \mathbf{G}^{(Alt)}) = \{\mathbf{z}^*(\mathbf{I} - \mathbf{G}^{(Alt)})\mathbf{z}, \forall z \in \mathbb{C}^{2n} \setminus \{\mathbf{0}\}, \|z\| = 1\}$ is the numerical range for $\mathbf{I} - \mathbf{G}^{(Alt)}$. In order to simplify the upper bound in the previous theorem, we study the numerical range of $\mathbf{I} - \mathbf{G}^{(Alt)}$ similar to [12]. Writing $\mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix}$ and

computing the numerical range of $\mathbf{I} - \mathbf{G}^{(Alt)}$ explicitly yields:

$$\begin{bmatrix} \mathbf{z}_1^*, \mathbf{z}_2^* \end{bmatrix} \begin{bmatrix} \mathbf{0} & \eta\mathbf{A} \\ -\eta\mathbf{A}^T & \eta^2\mathbf{A}^T\mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} = \eta^2\mathbf{z}_2^*\mathbf{A}^T\mathbf{A}\mathbf{z}_2 + \eta\mathbf{z}_1^*\mathbf{A}\mathbf{z}_2 - \eta\mathbf{z}_2^*\mathbf{A}^T\mathbf{z}_1. \tag{5.22}$$

For a general matrix A, there is no special structure about the numerical range of $\mathbf{I} - \mathbf{G}^{(Alt)}$. However, when $\mathbf{A}$ is symmetric, we can decompose $\mathbf{A}$ as $\mathbf{A} = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^T$ where $\{\lambda_i\}_{i=1}^n$ are eigenvalues of $\mathbf{A}$ in decreasing order and $\{\mathbf{v}_i\}_{i=1}^n$ are associated eigenvectors, and write $\mathbf{A}^T\mathbf{A} = \sum_{i=1}^{n} \lambda_i^2 \mathbf{v}_i \mathbf{v}_i^T$. Then we can compute the numerical range of $\mathbf{G}^{(Alt)}$ as follows:

$$\sum_i^n \begin{bmatrix} \mathbf{z}_1^*, \mathbf{z}_2^* \end{bmatrix} \begin{bmatrix} \mathbf{0} & \eta\lambda_i\mathbf{v}_i\mathbf{v}_i^T \\ -\eta\lambda_i\mathbf{v}_i\mathbf{v}_i^T & \eta^2\lambda_i^2\mathbf{v}_i\mathbf{v}_i^T \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} = \sum_i^n \begin{bmatrix} \mathbf{z}_1^*\mathbf{v}_i, \mathbf{z}_2^*\mathbf{v}_i \end{bmatrix} \begin{bmatrix} 0 & \eta\lambda_i \\ -\eta\lambda_i & \eta^2\lambda_i^2 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{v}_i^T\mathbf{z}_1 \\ \mathbf{v}_i^T\mathbf{z}_2 \end{bmatrix} \tag{5.23}$$

Following the techniques proposed in [12] to analyze the numerical range of general 2×2 matrices, we can show that the numerical range of $\mathbf{I} - \mathbf{G}^{(Alt)}$ is equal to the convex hull of the union of the numerical range of

$$\mathbf{G}_i = \begin{bmatrix} 0 & \eta\lambda_i \\ -\eta\lambda_i & \eta^2\lambda_i^2 \end{bmatrix} \quad , i = 1, \ldots, n. \tag{5.24}$$

And the boundary of numerical range of $\mathbf{G}_i$ is an ellipse whose axes are the line segments joining the points x to y and w to z, respectively, with

$$x = 0, \quad y = \eta^2\lambda_i^2, \quad , w = \frac{\eta^2\lambda_i^2}{2} - \sqrt{-1}\eta|\lambda_i|, \quad z = \frac{\eta^2\lambda_i^2}{2} + \sqrt{-1}\eta|\lambda_i|. \tag{5.25}$$

Thus, the numerical range of $\mathbf{I} - \mathbf{G}^{(Alt)}$ can be spanned by convex hull of the union of the numerical range of a set of 2-by-2 matrices and the numerical range of each such a 2-by-2 matrix is an ellipse. We can compute the center o and focal distance

d of the ellipse generated by numerical range of $\mathbf{I} - \mathbf{G}^{(Alt)}$ explicitly. Then a linear transformation enables us to use Theorem 5.6.1 to show that the near-best polynomial for the minimax problem on the numerical range of $\mathbf{I} - \mathbf{G}^{(Alt)}$ is given by $t_n(z; c) := \frac{T_n(\frac{z-o}{d})}{T_n(\frac{c-o}{d})}$ if 0 is excluded from the numerical range of $\mathbf{I} - \mathbf{G}^{(Alt)}$. However, according to equation 5.25 the numerical range includes the point 0 where the residual polynomial takes value 1, thus the analysis based on numerical range can not help derive the convergent result as the upper bound is not guaranteed to be less than 1.

Proofs of theorem

We first provide proof of Theorem. 5.3.1.

Theorem 5.6.2 (Global convergence for simultaneous `GDA-AM` on bilinear problem)**.** *Denote the distance between the stationary point* $\mathbf{w}^*$ *and current iterate* $\mathbf{w}_{(k+1)p}$ *of Algorithm 5.2 with Anderson restart dimension* p *as* $N_{(k+1)p} = dist(\mathbf{w}^*, \mathbf{w}_{(k+1)p})$*. Then we have the following bound for* N_t *Algorithm 5.2 is unconditionally convergent*

$$N_{(k+1)p} \leq \frac{1}{T_p(1 + \frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1})} N_{kp} \tag{5.26}$$

where T_p *is the Chebyshev polynomial of first kind of degree* p *and* $\frac{1}{T_p(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1})} < 1$ *since* $1 + \frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1} > 1$*.*

Proof of Theorem 5.3.1. Note that $\mathbf{I} - \mathbf{G}^{(Sim)}$ is a normal matrix which will be denoted as $\mathbf{G}$ for notational simplicity. Thus it admits the following eigendecomposition:

$$\mathbf{G} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T, \quad \mathbf{U}\mathbf{U}^T = \mathbf{I}, \quad \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_{2n}). \tag{5.27}$$

Based on the equivalence between GMRES and Anderson Mixing, we know that the convergence rate of simultaneous `GDA-AM` can be estimated by the spectrum of $\mathbf{G}$.

Especially, it holds that

$$\mathbf{r}_{(k+1)p} = \mathbf{U} f_p(\mathbf{\Lambda}) \mathbf{U}^T \mathbf{r}_{kp}. \quad f_p \in \mathcal{P}_p \tag{5.28}$$

where $\mathcal{P}_p$ is the family of residual polynomials with degree p such that $f_p(0) = 1, \forall f_p \in \mathcal{P}_p$. According to Lemma 5.2.1, we have the following estimation

$$\|\mathbf{r}_{(k+1)p}\|_2 = \min_{f_p \in \mathcal{P}_p} \|f_p(\mathbf{G})\mathbf{r}_{kp}\|_2 \leq \min_{f_p \in \mathcal{P}_p} \max_i |f_p(\lambda_i)| \|\mathbf{r}_{kp}\|_2. \tag{5.29}$$

Due to the block structure of $\mathbf{G}$, the eigenvalues of $\mathbf{G}$ can be computed explicitly as

$$\pm \eta \sigma_i \sqrt{-1}, \ i = 1, \ldots, n, \tag{5.30}$$

where σ_i is the ith largest singular value of matrix $\mathbf{A}$. This shows that the eigenvalues of $\mathbf{G}$ are n pairs of purely imaginary numbers excluding 0 since $\mathbf{A}$ has full rank.

Since the eigenvalues of $\mathbf{G}$ are distributed in two intervals excluding the origin

$$I = [-\eta \sigma_{max}(\mathbf{A})\sqrt{-1}, -\eta \sigma_{min}(\mathbf{A})\sqrt{-1}] \cup [\eta \sigma_{min}(\mathbf{A})\sqrt{-1}, \eta \sigma_{max}(\mathbf{A})\sqrt{-1}],$$

it can be shown that the following p-th degree polynomial with value 1 at the origin that has the minimal maximum deviation from 0 on I is given by:

$$f_p(z) = \frac{T_l(q(\sqrt{-1}z))}{T_l(q(0))}, \quad q(\sqrt{-1}z) = 1 - \frac{2(\sqrt{-1}z - \eta\sigma_{min})(\sqrt{-1}z + \eta\sigma_{min})}{(\eta\sigma_{max}(\mathbf{A}))^2 - (\eta\sigma_{min}(\mathbf{A}))^2} \tag{5.31}$$

where $l = [\frac{p}{2}]$ and T_l is the Chebyshev polynomial of first kind of degree l. The function $q(\sqrt{-1}z)$ maps I to $[-1, 1]$. Thus the numerator of the polynomial f_p is bounded by 1 on I. The size of denominator can be determined by the method discussed in Chapter 3 of [41]. Assume $q(0) = \frac{1}{2}(y + y^{-1})$, then $T_l(q(0)) = \frac{1}{2}(y^l + y^{-l})$. Then y can be

determined by solving

$$q(0) = \frac{(\eta\sigma_{max}(\mathbf{A}))^2 + (\eta\sigma_{min}(\mathbf{A}))^2}{(\eta\sigma_{max}(\mathbf{A}))^2 - (\eta\sigma_{min}(\mathbf{A}))^2}. \tag{5.32}$$

The solutions to this equation are

$$y_1 = \frac{\eta\sigma_{max}(\mathbf{A}) + \eta\sigma_{min}(\mathbf{A})}{\eta\sigma_{max}(\mathbf{A}) - \eta\sigma_{min}(\mathbf{A})} \quad \text{or} \quad y_2 = \frac{\eta\sigma_{max}(\mathbf{A}) - \eta\sigma_{min}(\mathbf{A})}{\eta\sigma_{max}(\mathbf{A}) + \eta\sigma_{min}(\mathbf{A})}. \tag{5.33}$$

Then plugging the value of $q(0)$ into the polynomial f_p yields

$$\begin{aligned} \frac{\|\mathbf{r}_{(k+1)p}\|}{\|\mathbf{r}_{kp}\|} &\leq 2\Big(\frac{\sqrt{\eta^2\sigma_{max}^2(\mathbf{A})} - \sqrt{\eta^2\sigma_{min}^2(\mathbf{A})}}{\sqrt{\eta^2\sigma_{max}^2(\mathbf{A})} + \sqrt{\eta^2\sigma_{min}^2(\mathbf{A})}}\Big)^l \\ &= 2\Big(\frac{\sigma_{max}(\mathbf{A}) - \sigma_{min}(\mathbf{A})}{\sigma_{max}(\mathbf{A}) + \sigma_{min}(\mathbf{A})}\Big)^l = 2\Big(\frac{\kappa(\mathbf{A}) - 1}{\kappa(\mathbf{A}) + 1}\Big)^l \end{aligned} \tag{5.34}$$

Note that N_t and $\mathbf{r}_t$ is related through $\mathbf{G}(\mathbf{w}_t - \mathbf{w}^*) = \mathbf{r}_t$. Therefore,

$$\begin{aligned} N_{(k+1)p} = \|\mathbf{w}_{(k+1)p} - \mathbf{w}^*\|_2 = \|\mathbf{G}^{-1}\mathbf{r}_{(k+1)p}\|_2 = \min_{f_p \in \mathcal{P}_p} \|\mathbf{G}^{-1} f_p(\mathbf{G})\mathbf{G}(\mathbf{w}_{kp} - \mathbf{w}^*)\|_2 \\ \leq \min_{f_p \in \mathcal{P}_p} \max_i |f_p(\lambda_i)| \|\mathbf{w}_{kp} - \mathbf{w}^*\|_2 \leq 2\Big(1 - \frac{2}{\kappa(\mathbf{A}) + 1}\Big)^{\frac{p}{2}} N_{kp}. \end{aligned} \tag{5.35}$$

Actually a tighter bound can be proved after noting that the problem is essentially equivalent to polynomial minmax problem on the interval:

$$\tilde{I} = [\eta^2\sigma_{min}^2(\mathbf{A}), \quad \eta^2\sigma_{max}^2(\mathbf{A})],$$

Then it is well known that,

$$N_{(k+1)p} \leq \min_{f_p \in \mathcal{P}_p} \max_{\lambda_i \in [\eta^2\sigma^2_{min}(\mathbf{A}),\ \eta^2\sigma^2_{max}(\mathbf{A})]} |f_p(\lambda_i)| \|\mathbf{w}_{kp} - \mathbf{w}^*\|_2 \leq \frac{1}{T_p(1 + 2\frac{\sigma^2_{min}}{\sigma^2_{max} - \sigma^2_{min}})} N_{kp}$$
$$\leq \frac{1}{T_p(1 + \frac{2}{\kappa(\mathbf{A}^T\mathbf{A}) - 1})} N_{kp} \tag{5.36}$$

where T_p Chebyshev polynomial of degree p of the first kind and $\frac{1}{T_p(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1})} < 1$. Explicitly,

$$T_p(1 + \frac{2}{\kappa(\mathbf{A}^T\mathbf{A}) - 1}) = \frac{1}{2}\Big[\Big(1 + \frac{2}{\kappa(\mathbf{A}^T\mathbf{A}) - 1} + \sqrt{(1 + \frac{2}{\kappa(\mathbf{A}^T\mathbf{A}) - 1})^2 - 1}\Big)^p$$
$$+ \Big(1 + \frac{2}{\kappa(\mathbf{A}^T\mathbf{A}) - 1} + \sqrt{(1 + \frac{2}{\kappa(\mathbf{A}^T\mathbf{A}) - 1})^2 - 1}\Big)^{-p}\Big]$$

□

Next, we give the proof of Theorem 5.3.2.

Theorem 5.6.3 (Global convergence for alternating `GDA-AM` on bilinear problem)**.** *Denote the distance between the stationary point* $\mathbf{w}^*$ *and current iterate* $\mathbf{w}_{(k+1)p}$ *of Algorithm 5.3 with Anderson restart dimension* p *as* $N_{(k+1)p} = dist(\mathbf{w}^*, \mathbf{w}_{(k+1)p})$*. Assume* $\mathbf{A}$ *is normalized such that its largest singular value is equal to* 1*. Then when the learning rate* η *is less than* 2*, we have the following bound for* N_t

$$N^2_{(k+1)p} \leq \sqrt{1 + \frac{2\eta}{2 - \eta}} (\frac{r}{c})^p N^2_{kp}$$

where c *and* r *are the center and radius of a disk* $D(c, r)$ *which includes all the eigenvalues of* $\mathbf{G}$ *in* (5.3)*. Especially,* $\frac{r}{c} < 1$*.*

Proof. Since the residual $\mathbf{r}_p$ of AA at p-th iteration has the form of

$$\mathbf{r}_p = (\mathbf{I} - \sum_{i=1}^{p} \mathbf{G}^i)\mathbf{r}_0,$$

and AA minimizes the residual, we have

$$\|\mathbf{r}_{(k+1)p}\|_2^2 \leq \min_{\beta} \|\mathbf{r}_{kp} - \beta \mathbf{G}^i \mathbf{r}_{kp}\|_2^2 \leq \min_{f_p \in \mathcal{P}_p} \|f_p(\mathbf{G})\mathbf{r}_{kp}\|_2^2,$$

where $\mathcal{P}_p$ is the family of polynomials with degree p such that $f_p(0) = 1, \forall f_p \in \mathcal{P}_p$. It's easy to see that $\mathbf{G}$ is unitarily similar to a block diagonal matrix Λ with 2×2 blocks as follows:

$$\begin{bmatrix} 0 & \eta\sigma_i \\ -\eta\sigma_i & (\eta\sigma_i)^2 \end{bmatrix} \quad \forall\ i \in [n].$$

Thus the eigenvalues of G can be easily identified as

$$\lambda_{\pm i} = \frac{(\eta\sigma_i(\eta\sigma_i \pm \sqrt{(\eta\sigma_i)^2 - 4}))}{2}, \quad i \in [n].$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ are the singular values of $\mathbf{A}$. Furthermore, the eigenvector and eigenvalue associated with each 2×2 diagonal block are

$$\begin{bmatrix} 0 & \eta\sigma_i \\ -\eta\sigma_i & (\eta\sigma_i)^2 \end{bmatrix} \begin{bmatrix} 1 \\ \frac{\lambda_{\pm i}}{\eta\sigma_i} \end{bmatrix} = \lambda_{\pm i} \begin{bmatrix} 1 \\ \frac{\lambda_{\pm i}}{\eta\sigma_i} \end{bmatrix}$$

Thus $\mathbf{G}$ is diagonalizable and denote the matrix with the columns of eigenvectors of $\mathbf{G}$ by X. The real part of the eigenvalues of $\mathbf{G}$ are at least

$$\mathcal{R}(\lambda_{\pm i}) \geq \frac{(\eta\sigma_i)^2}{2}, \quad i \in [n]. \tag{5.37}$$

And since $|\eta\sigma_i| \geq |\sqrt{(\eta\sigma_i)^2 - 4})|$, all the eigenvalues will be included in a disk $D(c, r)$

which is included in the right half plane. Moreover, both c and r being greater than zero indicates that $\frac{r}{c} < 1$. Start from the following inequality:

$$\begin{aligned} N_{(k+1)p} &= \left\|\mathbf{w}_{(k+1)p} - \mathbf{w}^*\right\|_2 = \left\|\mathbf{G}^{-1}\mathbf{r}_{(k+1)p}\right\|_2 \leq \min_{f_p \in \mathcal{P}_p} \left\|\mathbf{G}^{-1} f_p(\mathbf{G})\mathbf{r}_{kp}\right\|_2 \\ &= \min_{f_p \in \mathcal{P}_p} \left\|\mathbf{G}^{-1} f_p(\mathbf{G})\mathbf{G}\left(\mathbf{w}_{kp} - \mathbf{w}^*\right)\right\|_2 = \min_{f_p \in \mathcal{P}_p} \left\|\mathbf{G}_p^{-1}(\mathbf{G})\left(\mathbf{w}_{kp} - \mathbf{w}^*\right)\right\|_2 \quad (5.38) \\ &= \min_{f_p \in \mathcal{P}_p} \left\|f_p(\mathbf{G})\left(\mathbf{w}_{kp} - \mathbf{w}^*\right)\right\|_2 \end{aligned}$$

We will use the eigendeomposition of G and the special polynomial $(\frac{c-t}{c})^p$ to derive the inequality in Theorem 5.3. Now we know $\frac{r}{c} < 1$. If we choose $g_p(t) = (\frac{c-t}{c})^p$, we can obtain

$$\min_{f_p \in \mathcal{P}_p} \|f_p(\mathbf{G})(\mathbf{w}_{kp} - \mathbf{w}^*)\|_2 \ \leq \|g_p(\mathbf{G})(\mathbf{w}_{kp} - \mathbf{w}^*)\|_2$$

which implies

$$\min_{f_p \in \mathcal{P}_p} \|f_p(\mathbf{G})(\mathbf{w}_{kp} - \mathbf{w}^*)\|_2 \leq \|g_p(\mathbf{X}\Lambda\mathbf{X}^{-1})\|\|(\mathbf{w}_{kp} - \mathbf{w}^*)\|_2$$

Since G is diagonalizable (which has been shown above), we assume the eigendecomposition of $\mathbf{G}$ is $\mathbf{G} = \mathbf{X}\Lambda\mathbf{X}^{-1}$. Then

$$\begin{aligned} \min_{f_p \in \mathcal{P}_p} \|g_p(\mathbf{G})(\mathbf{w}_{kp} - \mathbf{w}^*)\|_2 &\leq \|\mathbf{X}\|\|\mathbf{X}^{-1}\| \max_{\{\lambda_i\}_{i=1}^{2n}} \|g_p(\Lambda)\|\|(\mathbf{w}_{kp} - \mathbf{w}^*)\|_2 \\ &\leq \kappa_{\mathbf{G}}(\frac{r}{c})^p \|\mathbf{w}_{kp} - \mathbf{w}^*\|_2 \end{aligned}$$

(5.39)

where $\kappa_{\mathbf{G}}$ is the condition number of X. The last inequality comes from Lemma 6.26 and Proposition 6.32 in [104].. Since $\mathbf{G}$ and Λ are unitarily similar, $\kappa_{\mathbf{G}}$ is equal to the condition number of the eigenvector matrix of Λ. The eigenvector matrix of Λ is a

block diagonal matrix with the ith block as $\begin{bmatrix} 1 & 1 \\ \frac{\lambda_{+i}}{\eta\sigma_i} & \frac{\lambda_{-i}}{\eta\sigma_i} \end{bmatrix}$. Thus the singluar values of the eigenvector matrix of Λ is equal to the union of the singular values of these 2-by-2 blocks. Under the assumption that the largest singular value of $\mathbf{A}$ are equal to 1 and the learning rate is less than 2, it is easy to find the singular values of the eigenvector matrix of Λ are $\sqrt{2 \pm \eta\sigma_i}$. Thus, $\kappa_{\mathbf{G}} = \frac{\sqrt{2+\eta\sigma_{\max}}}{\sqrt{2-\eta\sigma_{\max}}} = \frac{\sqrt{2+\eta}}{\sqrt{2-\eta}} = \sqrt{1 + \frac{2\eta}{2-\eta}}$. □

5.6.1 Discussion of obtained rates

We would like to first explain on why taking Chebyshev polynomial of degree p at the point $1 + \frac{2}{\kappa-1}$. We evaluate the Chebyshev polynomial at this specific point because the reciprocal of this value gives the minimal value of infinite norm of the all polynomials of degree p defined on the interval $\tilde{I} = [\eta^2\sigma^2_{min}(\mathbf{A}), \quad \eta^2\sigma^2_{max}(\mathbf{A})]$ based on Theorem 6.25 (page 209) [104]. In other words, taking the function value at this point leads to the tight bound.

When comparing between existing bounds, we would like to point our our derived bounds are hard to compare directly. Alternatively, we can derive another bound for comparison with existing bounds for simultaneous `GDA-AM`. If we use the inequality that $T_p(t) \geq \frac{1}{2}((t + \sqrt{t^2-1})^p)$, we can obtain the bound $\rho(A) = 4(\frac{\sqrt{\kappa(A^TA)}-1}{\sqrt{\kappa(A^TA)}+1})^2 = 4(1 - O(\frac{1}{\sqrt{\kappa(A^TA)}}))$, which is in a form that is comparable with EG and can compete with EG + positive momentum. The numerical experiments in figure 2b numerically verify that our bound is smaller than EG. We wanted to numerically compare our rate with EG with positive momentum. However the bound of EG with positive momentum is asymptotic. Moreover, it does not specify the constants so we can not numerically compare them. We do provide empirical comparison between GDA-AM and EG with positive momentum for bilinear problems. It shows GDA-AM outperforms EG with positive momentum. Regarding alternating `GDA-AM` , we would like to note that the bound in Theorem 5.3.2 depends on the eigenvalue distribution

of the matrix $\mathbf{G}$. Condition number is not directly related to the distribution of eigenvalues of a nonsymmetric matrix $\mathbf{G}$. Thus, the condition number is not a precise metric to characterize the convergence. If these eigenvalues are clustered, then our bound can be small. On the other hand, if these eigenvalues are evenly distributed in the complex plane, then the bound can very close to 1.

More importantly, we would like to stress several technical contributions.

1. Our obtained Theorem 5.3.1 and 5.3.2 provide nonasymptotic guarantees, while most other work are asymptotic. For example, EG with positive momentum can achieve a asymptotic rate of $1 - O(1/\sqrt{\kappa})$ under strong assumptions [6].

2. Our contribution is not just about fix the convergence issue of GDA by applying Anderson Mixing; another contribution is that we arrive at a convergent and tight bound on the original work and not just adopting existing analyses. We developed Theorem 5.3.1 and 5.3.2 from a new perspective because applying existing theoretical results fail to give us neither convergent nor tight bounds.

3. Theorem 5.3.1 and 5.3.2 only requires mild conditions and reflects how the table size p controls the convergence rate. Theorem 5.3.1 is independent of the learning rate η. However, the convergence results of other methods like EG and OG depend on the learning rate, which may yield less than desirable results for ill-specified learning rates.

Convex-concave and general case

Given the widespread usage of minimax problems in applications of machine learning, it is natural to ask about its properties when being applied to general nonconvex-nonconcave settings. If f is a nonconvex-nonconcave function, the problem of finding global Nash equilibrium is NP-hard in general. Recently, [59] show that local or global Nash equilibrium may not exist in nonconvex-nonconcave settings and propose a new

notation *local minimax* as defined below:

Definition 6. *A point* $(\mathbf{x}^\star, \mathbf{y}^\star)$ *is said to be a* ***local minimax*** *point of* f*, if there exists* $\delta_0 > 0$ *and a function* h *satisfying* $h(\delta) \to 0$ *as* $\delta \to 0$*, such that for any* $\delta \in (0, \delta_0]$*, and any* $(\mathbf{x}, \mathbf{y})$ *satisfying* $\|\mathbf{x} - \mathbf{x}^\star\| \le \delta$ *and* $\|\mathbf{y} - \mathbf{y}^\star\| \le \delta$*, we have*

$$f(\mathbf{x}^\star, \mathbf{y}) \le f(\mathbf{x}^\star, \mathbf{y}^\star) \le \max_{\mathbf{y}' : \|\mathbf{y}' - \mathbf{y}^\star\| \le h(\delta)} f(\mathbf{x}, \mathbf{y}').$$

[59] also establishes the following first- and second-order conditions to characterize local minimax:

Proposition 1 (First-order Condition)**.** *Any local minimax point* $(\mathbf{x}^*, \mathbf{y}^*)$ *satisfies* $\nabla f(\mathbf{x}^*, \mathbf{y}^*) = \mathbf{0}$.

Proposition 2 (Second-order Necessary Condition)**.** *Any local minimax point* $(\mathbf{x}^*, \mathbf{y}^*)$ *satisfies*

$$\nabla_{\mathbf{yy}} f(\mathbf{x}^*, \mathbf{y}^*) \preccurlyeq \mathbf{0} and \nabla_{\mathbf{xx}} f(\mathbf{x}^*, \mathbf{y}^*) - \nabla_{\mathbf{xy}} f(\mathbf{x}^*, \mathbf{y}^*)(\nabla_{\mathbf{yy}} f(\mathbf{x}^*, \mathbf{y}^*))^{-1} \nabla_{\mathbf{yx}} f(\mathbf{x}^*, \mathbf{y}^*) \succcurlyeq \mathbf{0} \tag{5.40}$$

Proposition 3 (Second-order Sufficient Condition)**.** *Any stationary point* $(\mathbf{x}^*, \mathbf{y}^*)$ *satisfies* $\nabla_{\mathbf{yy}} f(\mathbf{x}^*, \mathbf{y}^*) \prec \mathbf{0}$ *and* $\nabla_{\mathbf{xx}} f(\mathbf{x}^*, \mathbf{y}^*) -$
$\nabla_{\mathbf{xy}} f(\mathbf{x}^*, \mathbf{y}^*)(\nabla_{\mathbf{yy}} f(\mathbf{x}^*, \mathbf{y}^*))^{-1} \nabla_{\mathbf{yx}} f(\mathbf{x}^*, \mathbf{y}^*) \succ \mathbf{0}$ *is a local minimax point.*

Given the second-order conditions of local minimax, it turns out that above question is extremely challenging—`GDA-AM` is a first-order method. But we can prove the following result for `GDA-AM`:

Theorem 5.6.4 (Local minimax as subset of limiting points of `GDA-AM`)**.** *Consider a general objective function* $f(\mathbf{x}, \mathbf{y})$*. The set of limiting points of* `GDA-AM` *for minimax problem*

$$\min_{\mathbf{x} \in \mathbb{R}^n} \max_{\mathbf{y} \in \mathbb{R}^n} f(\mathbf{x}, \mathbf{y})$$

includes the local minimax points of this function.

The definition of local minimax is stronger than that of first order ϵ point. The convergence analysis for complexity of finding ϵ stationary point is included in the next section. The proof of Theorem 5.6.4 needs the result from the following theorem.

Theorem 5.6.5 ([15]). *Let δ satisfy $0 < \delta \leq \delta_0$ for some constant $\delta_0 > 0$ (refer to [15] for details), and let $b^\delta \in \mathcal{X}$ satisfy $\left\|b - b^\delta\right\| \leq \delta$. Let $k \leq \ell$ and let x_k^δ denote the kth iterate determined by the GMRES method applied to equation $Ax = b^\delta$, with initial guess $x_0^\delta = 0$. Similarly, let x_k denote the kth iterate determined by the GMRES method applied to equation $Ax = b$ with initial guess $x_0 = 0$. Then, there are constants σ_k independent of δ, such that*

$$\left\|x_k - x_k^\delta\right\| \leq \sigma_k \delta, \quad 1 \leq k \leq \ell$$

Then, we give the proof of Theorem 5.6.4.

Proof of Theorem 5.6.4. For notational simplicity, we will denote $\nabla_{\mathbf{xx}} f(\mathbf{x}^*, \mathbf{y}^*)$, $\nabla_{\mathbf{xy}} f(\mathbf{x}^*, \mathbf{y}^*)$ and $\nabla_{\mathbf{yy}} f(\mathbf{x}^*, \mathbf{y}^*)$ by $\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*}$, $\mathbf{H}_{\mathbf{x}^*\mathbf{y}^*}$ and $\mathbf{H}_{\mathbf{y}^*\mathbf{y}^*}$, respectively. Simultaneous GDA can be written as

$$\mathbf{w}_{t+1} = \begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_t - \eta \nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t) \\ \mathbf{y}_t + \eta \nabla_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y}_t) \end{bmatrix}.$$

Since the function is differentiable, Taylor expansion holds for $\nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t)$
and $\nabla_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y}_t)$ at a local minimx point $\mathbf{w}^* = (\mathbf{x}^*, \mathbf{y}^*)$,

$$\nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t) = \nabla_{\mathbf{x}} f(\mathbf{x}^*, \mathbf{y}^*) + \mathbf{H}_{\mathbf{x}^*\mathbf{x}^*}(\mathbf{x}_t - \mathbf{x}^*) + \mathbf{H}_{\mathbf{x}^*\mathbf{y}^*}(\mathbf{y}_t - \mathbf{y}^*) + o(\|\mathbf{w}_t - \mathbf{w}^*\|_2)$$

$$\nabla_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y}_t) = \nabla_{\mathbf{y}} f(\mathbf{x}^*, \mathbf{y}^*) + \mathbf{H}_{\mathbf{y}^*\mathbf{y}^*}(\mathbf{y}_t - \mathbf{y}^*) + \mathbf{H}_{\mathbf{y}^*\mathbf{x}^*}(\mathbf{x}_t - \mathbf{x}^*) + o(\|\mathbf{w}_t - \mathbf{w}^*\|_2).$$

Use the fact that $\nabla f(\mathbf{x}^*, \mathbf{y}^*) = \mathbf{0}$ to simplify the above equations and obtain

$$\nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t) = \mathbf{H}_{\mathbf{x}^*\mathbf{x}^*}(\mathbf{x}_t - \mathbf{x}^*) + \mathbf{H}_{\mathbf{x}^*\mathbf{y}^*}(\mathbf{y}_t - \mathbf{y}^*) + o(\|\mathbf{w}_t - \mathbf{w}^*\|_2)$$
$$\nabla_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y}_t) = \mathbf{H}_{\mathbf{y}^*\mathbf{y}^*}(\mathbf{y}_t - \mathbf{y}^*) + \mathbf{H}_{\mathbf{y}^*\mathbf{x}^*}(\mathbf{x}_t - \mathbf{x}^*) + o(\|\mathbf{w}_t - \mathbf{w}^*\|_2).$$

Inserting the above formulas into the iteration scheme, it yields

$$\mathbf{w}_{t+1} = \begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{I} - \eta\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*} & -\eta\mathbf{H}_{\mathbf{x}^*\mathbf{y}^*} \\ \eta\mathbf{H}_{\mathbf{y}^*\mathbf{x}^*} & \mathbf{I} + \eta\mathbf{H}_{\mathbf{y}^*\mathbf{y}^*} \end{bmatrix} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{y}_t \end{bmatrix} + \begin{bmatrix} \eta\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*}\mathbf{x}^* + \eta\mathbf{H}_{\mathbf{x}^*\mathbf{y}^*}\mathbf{y}^* + \epsilon \\ -\eta\mathbf{H}_{\mathbf{y}^*\mathbf{y}^*}\mathbf{y}^* - \eta\mathbf{H}_{\mathbf{x}^*\mathbf{y}^*}\mathbf{x}^* + \epsilon \end{bmatrix}$$

where ϵ denotes the higher order error $o(\|\mathbf{w}_t - \mathbf{w}^*\|_2)$. According to Theorem 5.2.2, we know that simultaneous `GDA-AM` is equivalent to applying GMRES to solve the following linear system (I - $\begin{bmatrix} (1-\alpha)\mathbf{I} - \eta\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*} & -\eta\mathbf{H}_{\mathbf{x}^*\mathbf{y}^*} \\ \eta\mathbf{H}_{\mathbf{y}^*\mathbf{x}^*} & (1-\alpha)\mathbf{I} + \eta\mathbf{H}_{\mathbf{y}^*\mathbf{y}^*} \end{bmatrix}$) w= $\begin{bmatrix} \alpha\mathbf{I} + \eta\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*} & \eta\mathbf{H}_{\mathbf{x}^*\mathbf{y}^*} \\ -\eta\mathbf{H}_{\mathbf{y}^*\mathbf{x}^*} & \alpha\mathbf{I} - \eta\mathbf{H}_{\mathbf{y}^*\mathbf{y}^*} \end{bmatrix}$w= b+ ϵ where $\mathbf{b} = \begin{bmatrix} \eta\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*}\mathbf{x}^* + \eta\mathbf{H}_{\mathbf{x}^*\mathbf{y}^*}\mathbf{y}^* \\ -\eta\mathbf{H}_{\mathbf{y}^*\mathbf{y}^*}\mathbf{y}^* - \eta\mathbf{H}_{\mathbf{x}^*\mathbf{y}^*}\mathbf{x}^* \end{bmatrix}$. We now know that `GDA-AM` is equivalent to GMRES being applied to solve the following linear system

$$\begin{bmatrix} \alpha\mathbf{I} + \eta\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*} & \eta\mathbf{H}_{\mathbf{x}^*\mathbf{y}^*} \\ -\eta\mathbf{H}_{\mathbf{y}^*\mathbf{x}^*} & \alpha\mathbf{I} - \eta\mathbf{H}_{\mathbf{y}^*\mathbf{y}^*} \end{bmatrix} \tilde{\mathbf{w}} = \mathbf{b}$$

The symmetric part of the coefficient matrix of the above linear system is

$$\begin{bmatrix} \alpha\mathbf{I} + \eta\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*} & \mathbf{0} \\ \mathbf{0} & \alpha\mathbf{I} - \eta\mathbf{H}_{\mathbf{y}^*\mathbf{y}^*} \end{bmatrix}.$$

According to Proposition 2, $\alpha\mathbf{I} - \eta\mathbf{H}_{\mathbf{y}^*\mathbf{y}^*}$ is positive definite since $\mathbf{H}_{\mathbf{y}^*\mathbf{y}^*} \preccurlyeq \mathbf{0}$. If $\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*}$ is positive semidefinite, then $\alpha\mathbf{I} + \eta\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*}$ is positive definite and we're done. Otherwise, assume $\lambda_{\min}(\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*}) < 0$. Then for fixed α, when $\eta < -\frac{\alpha}{\lambda_{\min}(\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*})}$, $\alpha\mathbf{I} + \eta\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*}$ will be positive definite. Then according to Theorem 5.2.2, we know `GDA-AM` indeed

converges. Let's create a new companion linear system as follows

$$\begin{bmatrix} \alpha\mathbf{I} + \eta\mathbf{H}_{\mathbf{x}^*\mathbf{x}^*} & \eta\mathbf{H}_{\mathbf{x}^*\mathbf{y}^*} \\ -\eta\mathbf{H}_{\mathbf{y}^*\mathbf{x}^*} & \alpha\mathbf{I} - \eta\mathbf{H}_{\mathbf{y}^*\mathbf{y}^*} \end{bmatrix} \hat{\mathbf{w}} = \mathbf{b} + \alpha\mathbf{w}^*$$

Note that $\hat{\mathbf{w}} = \mathbf{w}^*$ and GMRES on this companion linear system is convergent under suitable choice of learning rate η. Let the iterates of GMRES for $\tilde{\mathbf{w}}, \hat{\mathbf{w}}, \mathbf{w}$ be denoted by $\tilde{\mathbf{w}}_t, \hat{\mathbf{w}}_t, \mathbf{w}_t$. Then $\|\tilde{\mathbf{w}}_t - \hat{\mathbf{w}}_t\| \leq \|\tilde{\mathbf{w}}_t - \mathbf{w}_t\| + \|\hat{\mathbf{w}}_t - \mathbf{w}_t\|$. According to Theorem 5.6.5, we also have $\|\tilde{\mathbf{w}}_t - \mathbf{w}_t\| \leq \sigma_k\epsilon, 1 \leq k \leq t$. Further more, again according to Theorem 5.6.5, we know $\|\hat{\mathbf{w}}_t - \hat{\mathbf{w}}_t\| \leq \sigma_k(\alpha\mathbf{w}^* + \epsilon)$. Starting from an initial point very close to $\mathbf{w}^*$ and let $t \to \infty$ and $\alpha, \epsilon \to 0$, $\hat{\mathbf{w}}_t$ will converge to $\mathbf{w}^* = (\mathbf{x}^*, \mathbf{y}^*)$, which means the local minimax $\mathbf{w}^* = (\mathbf{x}^*, \mathbf{y}^*)$ is a limiting point of GDA-RAM. □

Theorem 5.6.6. *For strongly-convex-strongly-concave function $f(\mathbf{x}, \mathbf{y})$, `GDA-AM` will converge to the Nash equilibrium of this function.*

Proof: Since strongly-convex-strongly-concave function $f(\mathbf{x}, \mathbf{y})$ has unique Nash equilibrium which is also the unique minimax point, this minimax point must be the limiting point of `GDA-AM` according to Theorem 5.6.4. □

Bilinear-quadratic games

Moreover, we can further show that the `GDA-AM` converges on bilinear-quadratic games. Consider a quadratic problem as follows,

$$\min_{\mathbf{x}\in\mathbb{R}^n} \max_{\mathbf{y}\in\mathbb{R}^n} f(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{A}\mathbf{y} + \mathbf{x}^T\mathbf{B}\mathbf{x} - \mathbf{y}^T\mathbf{C}\mathbf{y} + \mathbf{b}^T\mathbf{x} + \mathbf{c}^T\mathbf{y}, \tag{5.41}$$

where $\mathbf{A}$ is full rank, $\mathbf{B}$ and $\mathbf{C}$ are both positive definite.

Theorem 5.6.7. *[Global convergence for simultaneous `GDA-AM` on bilinear-quadratic problem] Let $\mathbf{r}_t^{(Sim)}$ be the residual of Algorithm 5.2 being applied to problem* (5.41).

For some constant $\rho < 1$,

$$\|\mathbf{r}_t^{(Sim)}\|_2 \leq \underbrace{\left(1 - \frac{(\lambda_{\min}(\mathbf{J}^T + \mathbf{J}))^2}{4\lambda_{\max}(\mathbf{J}^T\mathbf{J})}\right)^{t/2}}_{\rho^{t/2}} \|\mathbf{r}_0\|_2, \tag{5.42}$$

where $\mathbf{J} = \begin{bmatrix} \eta\mathbf{B} & \eta\mathbf{A} \\ -\eta\mathbf{A}^T & \eta\mathbf{C} \end{bmatrix}$ *and* $\lambda_{\min}$ *and* $\lambda_{\max}$ *denote the smallest and largest eigenvalue, respectively.*

The convergence property of GMRES has been studied in the next theorem. We use this theorem to show the convergence rate of `GDA-AM` for bilinear-quadratic games.

Theorem 5.6.8 ([28])**.** *Consider solving a linear system* $\mathbf{E}\mathbf{x} = \mathbf{b}$ *using GMRES. Let* $\mathbf{r}_t = \mathbf{b} - \mathbf{E}\mathbf{x}_t$ *be the residual at* t*th iteration. If the Hermitian part of* $\mathbf{E}$ *is positive definite, then for some positive constant* $\rho < 1$*, it holds that*

$$\|\mathbf{r}_t\|_2 \leq \underbrace{\left(1 - \frac{(\lambda_{\min}(\mathbf{E}^H + \mathbf{E}))^2}{4\lambda_{\max}(\mathbf{E}^H\mathbf{E})}\right)^{t/2}}_{\rho^{t/2}} \|\mathbf{r}_0\|_2. \tag{5.43}$$

Proof of Theorem 5.6.7. Applying simultaneous `GDA-AM` to solve the above problem is equivalent to applying Anderson Mixing on the following fixed point iteration:

$$\begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I} - \eta\mathbf{B} & -\eta\mathbf{A} \\ \eta\mathbf{A}^T & I - \eta\mathbf{C} \end{bmatrix}}_{\mathbf{G}^{(Quad-sim)}} \underbrace{\begin{bmatrix} \mathbf{x}_t \\ \mathbf{y}_t \end{bmatrix}}_{\mathbf{w}_t^{(Quad-sim)}} + \underbrace{\begin{bmatrix} -\eta\mathbf{b} \\ -\eta\mathbf{c} \end{bmatrix}}_{\mathbf{b}^{(Quad-sim)}}. \tag{5.44}$$

We know that we need to study the convergence properties of GMRES for solving

the following linear system

$$\begin{bmatrix} \eta\mathbf{B} & \eta\mathbf{A} \\ -\eta\mathbf{A}^T & \eta\mathbf{C} \end{bmatrix} \mathbf{w} = \mathbf{b}. \tag{5.45}$$

For notational simplicity, the superscripts has been dropped. Denote the coefficient matrix $\begin{bmatrix} \eta\mathbf{B} & \eta\mathbf{A} \\ -\eta\mathbf{A}^T & \eta\mathbf{C} \end{bmatrix}$ by $\mathbf{J}$. The symmetric part of $\mathbf{J}$ is

$$\frac{\mathbf{J}+\mathbf{J}^T}{2} = \begin{bmatrix} \frac{\eta}{2}(\mathbf{B}+\mathbf{B}^T) & \mathbf{0} \\ \mathbf{0} & \frac{\eta}{2}(\mathbf{C}+\mathbf{C}^T) \end{bmatrix}$$

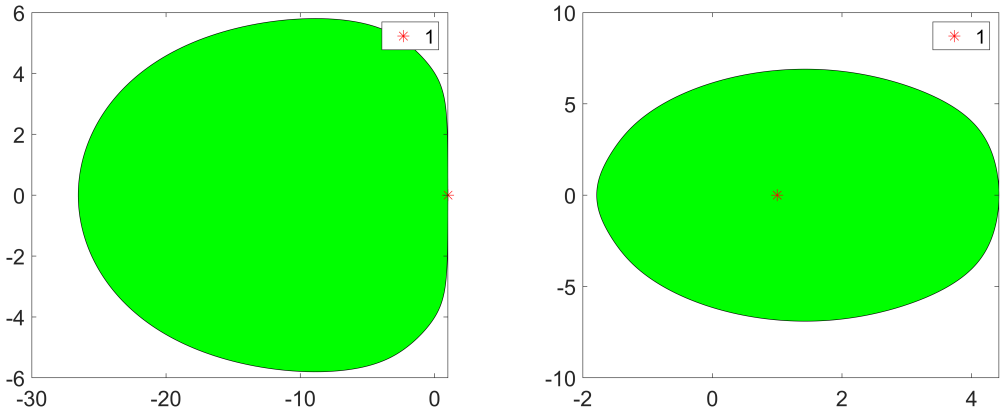
which is positive definite. Then immediately by Theorem 5.6.8, the following convergence rate holds For some constant $0 < \rho < 1$,

$$\begin{aligned} \|\mathbf{r}_t\|_2 = \min_{p\in\mathbb{P}_t^1} \|p(\mathbf{J})\mathbf{r}_0\|_2 &\leq \underbrace{\left(1 - \frac{(\lambda_{\min}(\mathbf{J}+\mathbf{J}^T)^2}{(4\lambda_{\max}(\mathbf{J}^T\mathbf{J}))}\right)^{t/2}}_{\rho^{t/2}} \|\mathbf{r}_0\|_2 \\ &= \rho^{t/2}\|\mathbf{r}_0\|_2 \end{aligned} \tag{5.46}$$

□

Note that the convergence of `GDA-AM` for bilinear-quadratic games can also be analyzed by numerical range as shown in [12]. Although we previously show that analysis based on the numerical range can not help us derive a convergent bound for bilinear games, we show analysis in [12] can be extended to bilinear-quadratic games. When $\mathbf{B}$ and $\mathbf{C}$ are positive definite, 1 is outside of the numerical range of matrix $\mathbf{G}^{(Quad-sim)}$ as shown in 5.7a. When $\mathbf{B}$ or $\mathbf{C}$ is not positive definite, 1 can be included in the numerical range of matrix $\mathbf{G}^{(Quad-sim)}$ as shown in 5.7b. That is saying analysis based on the numerical range [12, 21] to the bilinear-quadratic problem can lead to

a convergent result when $\mathbf{B}$ and $\mathbf{C}$ are positive definite. And analysis based on the numerical range can not help us derive convergent results when $\mathbf{B}$ or $\mathbf{C}$ is not positive definite.



(a) Positive definite $\mathbf{B}$ and $\mathbf{C}$ (b) Random generated $\mathbf{B}$ and $\mathbf{C}$

Figure 5.7: Numerical range of fixed-point operator (Simultaneous `GDA-AM`) $\mathbf{G} = \begin{bmatrix} \mathbf{I} - \eta\mathbf{B} & -\eta\mathbf{A} \\ \eta\mathbf{A}^T & I - \eta\mathbf{C} \end{bmatrix}$ for bilinear-quadratic games.

Stochastic convex-nonconvace case

In this section, we study the convergence of `GDA-AM` for convex-noncovace problem in the stochastic setting with the same assumptions in [128, 133]. The recent work [128] proves the convergence of the stochastic gradient descent with Anderson Mixing for min optimization. The convergence of `GDA-AM` for minimax optimization builds on top of it with several modifications. The minimax problem is equivalent to minimizing a function $\Phi(\cdot) = \max_{\mathbf{y}\in\mathcal{Y}} f(\cdot, \mathbf{y})$ [75]. And we are interested in complexity of a pair of ϵ-stationary point (x, y) instead of analysis of a point x.

Definition 7. *[75] A pair of points* $(\mathbf{x}, \mathbf{y})$ *is an* ϵ*-stationary point* $(\epsilon \geq 0)$ *of a differentiable function* Φ *if*

$$\|\nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y})\| \leq \epsilon$$

$$\|\mathcal{P}_{\mathcal{Y}}(\mathbf{y} + (1/\ell)\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})) - \mathbf{y}\| \leq \epsilon/\ell$$

Assumption 1. $f : \mathbb{R}^d \mapsto \mathbb{R}$ *is continuously differentiable.* $f(x) \geq f^{low} > -\infty$ *for any* $x \in \mathbb{R}^d$. ∇f *is globally L-Lipschitz continuous; namely* $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2$ *for any* $x, y \in \mathbb{R}^d$.

Assumption 2. *For any iteration* k*, the stochastic gradient* $\nabla f_{\xi_k}(x_k)$ *satisfies*

$$\mathbb{E}_{\xi_k}\left[\nabla f_{\xi_k}(x_k)\right] = \nabla f(x_k), \mathbb{E}_{\xi_k}\left[\|\nabla f_{\xi_k}(x_k) - \nabla f(x_k)\|_2^2\right] \leq \sigma^2 \tag{5.47}$$

, where $\sigma > 0$*, and* $\xi_k, k = 0, 1, \ldots,$ *are independent samples that are independent of* $\{x_i\}^k$

Theorem 5.6.9. *For a general convex-nonconcave function* f*, suppose that Assumptions 1 and 2 hold. Batch size* $n_t = n$ *for* $t = 0, \ldots, N-1$. $C > 0$ *is a constant.* $\beta_t = \frac{\mu}{4L(1+C^{-1})}.\delta_t \geq C\beta_t^{-2}, 0 \leq \alpha_t \leq \min\left\{1, \beta_t^{\frac{1}{2}}\right\}$ *and* α_t *is chosen to make sure the positive definiteness of* H_t*. Let* R *be a random variable following* $P_R(t) \overset{def}{=} \text{Prob}\{R = t\} = 1/N$*, and* $\bar{N}$ *be the total number of stochastic GDA-AM calls needed to calculate stochastic gradients* $\tilde{\nabla} f_{S_t}(\mathbf{w}_t)$ *in our algorithm. To ensure* $\mathbb{E}\left[\left\|\tilde{\nabla} f(\mathbf{w}_R)\right\|_2\right] \leq \epsilon$*, total number of stochastic GDA-AM calls needed to calculate stochastic gradients* $\tilde{\nabla} f_{S_t}(\mathbf{w}_t)$ *is* $O(\epsilon^{-4})$.

Recall that we can recast GDA scheme as the following fixed point iteration.

$$\mathbf{w}_{t+1} = G_{\eta}^{(\mathrm{sim})}(\mathbf{w}_t) \triangleq \mathbf{w}_t + \eta V(\mathbf{w}_t) \text{ with } \mathbf{w} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}, V(\mathbf{w}) = \begin{bmatrix} -\nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) \\ \nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \end{bmatrix}$$

Ignoring the stepsize η and let $\mathbf{W}_t$ and $\mathbf{R}_t$ record the first and second order diffrence of recent m iterates:

$$\mathbf{W}_t = [\Delta \mathbf{w}_{t-m}, \Delta \mathbf{w}_{t-m+1}, \cdots, \Delta \mathbf{w}_{t-1}], \mathbf{R}_t = [\Delta V_{t-m}, \Delta V_{t-m+1}, \cdots, \Delta V_{t-1}]$$

Similarly as [128],the Anderson mixing can be decoupled into

$$\begin{aligned} \bar{\mathbf{w}}_{t+1} &= \mathbf{w}_t - \mathbf{W}_t \Gamma_t, \quad \text{(Projection step)} \\ \bar{\mathbf{w}}_{t+1} &= \mathbf{w}_t + \beta_t \bar{V}_t, \quad \text{(Mixing step)} \end{aligned}$$

where β_t is the mixing parameter, and $\bar{V}_t = V_t - \mathbf{W}_t \Gamma_t$ and Γ_t is solved by

$$\Gamma_t = \underset{\Gamma \in \mathbb{R}^m}{\arg\min} \|V_t - \mathbf{R_t} \Gamma\|_2 + \delta_t \|\Gamma\|_2$$

We want to argue that similar arguments in [128] can be applied to the problem here. To see why Anderson mixing works for minimax optimization, we assume function f is smooth. Then the hessian matrix for $G_{\eta}^{(\mathrm{sim})}$ is

$$H = \begin{pmatrix} -\nabla_{\mathbf{xx}}^2 f & -\nabla_{\mathbf{xy}}^2 f \\ \nabla_{\mathbf{yx}}^2 f & \nabla_{\mathbf{yy}}^2 f \end{pmatrix}$$

Notice that in a small neighborhood of $\mathbf{w}_{t+1}$, we have

$$\mathbf{R_t} = -H\mathbf{W}_t = \begin{pmatrix} \nabla_{\mathbf{xx}}^2 f & \nabla_{\mathbf{xy}}^2 f \\ -\nabla_{\mathbf{yx}}^2 f & -\nabla_{\mathbf{yy}}^2 f \end{pmatrix} \mathbf{W}_t$$

Thus $\|V_t - \mathbf{R_t}\Gamma\|_2 \approx \|V_t + H\mathbf{W}_t\Gamma\|_2$, which is equivalent to solving for a vector p_t such that $Hp_k = V_t$. This is exactly the second order method for the fixed point iteration problem. Also at each step the AM is minimizing the residual, the reason that AM is equivalent to GMRES for linear problem is that this quadratic approximation is exact. Finally, we rewrite AM as the quasi-newton framework as [128] did. $\mathbf{w}_{t+1} = \mathbf{w}_t + H_t V_t$ where

$$\min_{H_t} \|H_t - \beta_t I\|_F \text{ subject to } H_t \mathbf{R}_t = -X_t$$

Finally, with damping parameter, Anderson mixing has the following form

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \beta_t V_t - \alpha_t \left(\mathbf{W}_t + \beta_t \mathbf{R_t}\right) \Gamma_t \tag{5.48}$$

we can also apply the very similar arguments to prove key results in lemma 1, lemma 2 in [128]. There is also a key difference with [128]. Here we are considering minimax optimization problem. Thus our gradient is actually $V(\mathbf{w}) = \begin{bmatrix} -\nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) \\ \nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \end{bmatrix}$ rather than $\nabla f(\mathbf{w}) = \begin{bmatrix} \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) \\ \nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \end{bmatrix}$ This will introduce some difficulty to the dynamics of the fixed pointe iteration. However, noticing that $\|V\| = \|\nabla f(\mathbf{w})\|$ and

$$\begin{aligned} f\left(\mathbf{w}_{t+1}\right) &\leq f\left(\mathbf{w}_t\right) + \nabla f\left(\mathbf{w}_t\right)^{\mathrm{T}} \left(\mathbf{w}_{t+1} - \mathbf{w}_t\right) + \frac{L}{2} \left\|\mathbf{w}_{t+1} - \mathbf{w}_t\right\|_2^2 \\ &\leq f\left(\mathbf{w}_t\right) + \tilde{\nabla} f\left(\mathbf{w}_t\right)^{\mathrm{T}} \left(\mathbf{w}_{t+1} - \mathbf{w}_t\right) + \frac{L}{2} \left\|\mathbf{w}_{t+1} - \mathbf{w}_t\right\|_2^2 \\ &= f\left(\mathbf{w}_t\right) + \tilde{\nabla} f\left(\mathbf{w}_t\right)^{\mathrm{T}} H_t V_t + \frac{L}{2} \left\|H_t V_t\right\|_2^2 \end{aligned} \tag{5.49}$$

where

$$\tilde{\nabla} f\left(\mathbf{w}_t\right) = \begin{bmatrix} -\nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) \\ \nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \end{bmatrix} \tag{5.50}$$

we call this the ascent-descent gradient (ADG) which is the gradient for minimax

optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^d}\max_{\mathbf{y}\in\mathbb{R}^d} f(\mathbf{x},\mathbf{y}).$$

To see why $\nabla f\left(\mathbf{w}_t\right)^{\mathrm{T}}\left(\mathbf{w}_{t+1}-\mathbf{w}_t\right) \leq \tilde{\nabla} f\left(\mathbf{w}_t\right)^{\mathrm{T}}\left(\mathbf{w}_{t+1}-\mathbf{w}_t\right)$, we consider their difference

$$(\tilde{\nabla}-\nabla) f\left(\mathbf{w}_t\right)^{\mathrm{T}}\left(\mathbf{w}_{t+1}-\mathbf{w}_t\right) = -2\nabla_{\mathbf{x}} f(\mathbf{x}_t,\mathbf{y}_t)^T(\mathbf{x}_{t+1}-\mathbf{x}_t).$$

For fixed $\mathbf{y}_t$, $f(\mathbf{x}_t,\mathbf{y}_{t+1})$ has the Talyor expansion:

$$\begin{aligned} f(\mathbf{x}_{t+1},\mathbf{y}_t) = f(\mathbf{x}_t,\mathbf{y}_t) + \nabla_{\mathbf{x}} f(\mathbf{x}_t,\mathbf{y}_t)^T(\mathbf{x}_{t+1}-\mathbf{x}_t) \\ + (\mathbf{x}_{t+1}-\mathbf{x}_t)^T\nabla_{\mathbf{xx}} f(\mathbf{x}_t+\theta(\mathbf{x}_{t+1}-\mathbf{x}_t),\mathbf{y}_t)(\mathbf{x}_{t+1}-\mathbf{x}_t) \quad (5.51) \end{aligned}$$

Assuming f is convex w.r.t $\mathbf{x}$ and apply safeguard to ensure $f(\mathbf{x}_{t+1},\mathbf{y}_t) \leq f(\mathbf{x}_t,\mathbf{y}_t)$ can guarantee $(\tilde{\nabla}-\nabla) f\left(\mathbf{w}_t\right)^{\mathrm{T}}\left(\mathbf{w}_{t+1}-\mathbf{w}_t\right) \geq 0$. Now applying lemmas in [128], we can derive the convergence of our method for general convex-nonconcave function similarly.

Details on the experiments

For our experiments, we used the PyTorch [2] deep learning framework. Experiments were run one NVIDIA V100 GPU. The residual network architecture for generator and discriminator are summarized in Table 5.3 and 5.4. We use a WGAN-GP loss, with gradient penalty $\lambda = 10$. When using the gradient penalty (WGAN-GP), we remove the batch normalization layers in the discriminator. When using SNGAN, we replace the batch normalization layers with spectral normalization. Hyperparamters of Adam are selected after grid search. We use a learning rate of 2×10^{-4} and batch size of 64. For table size of `GDA-AM` , we set it as 120 for CIFAR10 and 150 for CelebA. We set $\beta_1 = 0.0$ and $\beta_2 = 0.9$ as we find it gives us better models than default settings.

[2] https://pytorch.org/

Table 5.3: ResNet architecture used for our CIFAR-10 experiments.

Generator
Input: $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
Linear $128 \rightarrow 256 \times 4 \times 4$
ResBlock $128 \rightarrow 128$
ResBlock $256 \rightarrow 256$
ResBlock $256 \rightarrow 256$
Batch Normalization
ReLu
transposed conv. (256, kernel:3×3, stride:1, pad: 1
$\tanh(\cdot)$

Discriminator
Input: $x \in \mathbb{R}^{3 \times 32 \times 32}$
Linear $128 \rightarrow 128 \times 4 \times 4$
ResBlock $128 \rightarrow 128$
ResBlock $128 \rightarrow 128$
ResBlock $128 \rightarrow 128$
Linear $128 \rightarrow 1$

Table 5.4: ResNet architecture used for our CelebA (64×64) experiments.

Generator
Input: $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
Linear $128 \rightarrow 512 \times 8 \times 8$
ResBlock $512 \rightarrow 256$
ResBlock $256 \rightarrow 128$
ResBlock $128 \rightarrow 64$
Batch Normalization
ReLu
transposed conv. (64, kernel:3×3, stride:1, pad: 1
$\tanh(\cdot)$

Discriminator
Input: $x \in \mathbb{R}^{3 \times 64 \times 64}$
Linear $128 \rightarrow 128 \times 4 \times 4$
ResBlock $128 \rightarrow 128$
ResBlock $128 \rightarrow 256$
ResBlock $256 \rightarrow 512$
Linear $512 \rightarrow 1$

Chapter 6

Accelerating Sampling Procedure for diffusion based generative models

Previously, many work have focused on acceleration in terms of training time. In real world, fast and accurate inference is also very necessary. For example, giving a weather forecast that takes long time is useless. In this section, we show how to improve the inference efficiency with a focus on diffusion based generative models in particular.

6.1 Motivation of Synthetic EHRs

Recent digitisation of health records has provided a great platform for training deep learning models for precision medicine, personalised prediction of risks and health trajectories [29, 84]. However, there are many issues concerning patient privacy that need to be accounted for in order to aggregate more data to train more robust models. Due to this constraint, it is still hard for researchers to obtain access to real electronic health record (EHR) data. One approach is to mitigate privacy risks through the prac-

tice of de-identification such as perturbation and randomization [27, 81]. However, this approach is vulnerable to re-identification. [88] demonstrates that under very mild assumptions about the distribution from which the records are drawn, the adversary with a small amount of background knowledge about an individual can use it to identify, with high probability, this individual's record in the anonymized dataset. Another approach that could overcome privacy issues is to use synthetic datasets that capture as many of the complexities of the original data set (distributions, non-linear relationships, and noise). Ideally, synthetic EHRs can offer a potential solution as they yield a database that is beyond de-identification and hence are immune to re-identification, while preserving temporal patterns in real longitudinal EHRs. Thus, generating realistic, but not real data is a key element to advance AI for the healthcare community. However, realizing this goal in practice has been challenging because the resulting synthetic data are often not sufficiently realistic for machine learning tasks. There have been several distinguished efforts conducted in a variety of domains about synthetic EHR generation [10, 11, 20, 117, 134]. The vast majority of the existing proposed algorithms for synthetic EHR adopt a variant of Generative Adversarial Network (GAN) [39], auto-encoder [120] or a combination of them. While GANs and autoencoders are natural and widely used candidates for generation, there are several noticeable drawbacks of these models, including mode-collapse for GANs or poor sample diversity and quality for autoencoders. In recent years, diffusion (score) based models [55, 112, 114] have emerged as a family of powerful generative models that can yield state-of-the-art performance across a range of domains, including image and speech synthesis [16, 119]. Diffusion models rely on a long Markov chain to generate samples by gradually reversing a noising process. Diffusion models have been shown to achieve high-quality, diverse samples that are superior to their GAN-based counterparts. In addition to not being limited to high-fidelity samples, other key advantages of diffusion models include ease of training and tractability, in contrast to generative

adversarial networks, and speed of generation, in contrast to autoregressive models [42]. This leads to the natural question: Is a diffusion model promising for Electronic Health Records (EHRs) generation as well? We answer this in the affirmative, by introducing `MedDiff`, a novel denoising diffusion probabilistic model. `MedDiff`, shown in Figure 6.1, generates realistic synthetic patient records that build upon diffusion models to achieve high quality, robust samples while also being simple enough for practitioners to train. We further accelerate the generation process of `MedDiff` using Anderson acceleration [5], a numerical method that can improve convergence speed of fixed-point sequences. In summary, our contributions are as follows:

1. We investigate the effectiveness of diffusion based models on generating discrete Electronic Health Records (EHRs).

2. We proposed a novel method to accelerate the generation process, which is the main drawback of diffusion models.

3. We further adopt a novel conditioned sampling technique to generate discriminative synthetic EHR.

4. We show that `MedDiff` can generate realistic synthetic data that mimics the real data and provides similar predictive value according to our analysis and assessments.

6.2 Background in Diffusion Models

First proposed in [112], diffusion models are latent variable generative models characterized by a forward and a reverse Markov process. The forward process gradually adds noise to the original data sample, whereas the reverse process reverses the gradual noising process. For the reverse process, the sampling starts with the Tth noise level, x_T, and each timestep produces less-noisy samples, $x_{T-1}, x_{T-2}, ...$ until the final

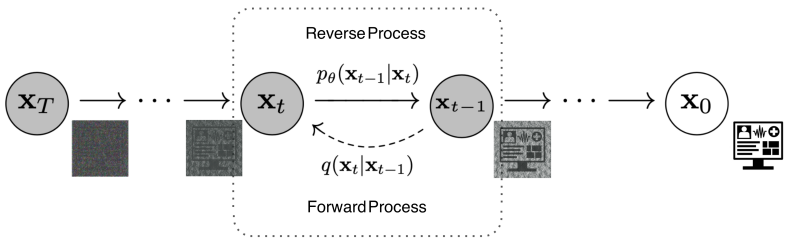


Figure 6.1: A graphical model summarizing the idea of diffusion model for generating synthetic electronic health records. This illustrates the Markov chain of the reverse (and forward) process of generating a sample by slowly removing noise. As can be shown, the forward process adds noise to the original patient record.

sample x_0 is produced. In essence, the diffusion model learns the "denoised" version from x_{t-1} to x_t.

Ho et al. [55] proposed a specific parameterization of the diffusion model to simplify the training process using a score matching-like loss that minimizes the mean-squared error between the true noise and the predicted noise. They also note that the sampling process can be interpreted as equivalent to Langevian dynamics, which allows them to relate the proposed denoising diffusion probabilistic models (DDPM) to score-based methods in [114]. Here, we introduce the forward process, the reverse process, and training of DDPM.

Forward Process

Given a data distribution $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, DDPM will gradually add noise to the original data distribution until it loses all the original information and becomes an entirely noisy distribution (as shown by the x_T sample in Figure 6.1). To be more specific, we will convolve $q(\mathbf{x}_0)$ with an isotropic Gaussian noise $\mathcal{N}(0, \sigma_i^2\mathbf{I})$ in T steps to produce a noise corrupted sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$. $\mathbf{x}_T$ will converge to an isotropic Gaussian distribution as $T \to \infty$. Under the variance schedule that produces $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$,

the distribution is derived as

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{6.1}$$

Using the reparameterization trick introduced in [66], the following nice property holds according to [55]

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}), \quad \bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i \text{ where } \alpha_t = 1 - \beta_t$$

Here $\bar{\alpha}$ controls the scale of noise added at each time step. At the beginning, noise should be small so that it is possible for the model to learn well, e.g., $\beta_1 < \beta_2 < \cdots < \beta_T$ and $\bar{\alpha}_1 > \bar{\alpha}_2 \cdots > \bar{\alpha}_T$.

Reverse Process

Since the forward process defined above is a Markov chain, the true sample can be recreated from Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ by reversing the forward process:

$$\begin{aligned} p_\theta(\mathbf{x}_{0:T}) &:= p(\mathbf{x}_T) \prod_{i=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \\ p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) &= \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \mathbf{\Sigma}_\theta)(\mathbf{x}_t, t)) \end{aligned} \tag{6.2}$$

Ho et al. [55] noticed that using Bayes's rule and conditioning on $\mathbf{x}_0$, the reverse process has a closed form expression:

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I}), \\ \text{where} \quad \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_t)}{1-\bar{\alpha}_t}\mathbf{x}_t \\ \text{and} \quad \tilde{\beta}_t &= \frac{(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\beta_t \end{aligned} \tag{6.3}$$

Training

Based on the reverse process, the training process is based on optimizing the variational lower bound on the negative log likelihood:

$$\begin{aligned}\mathbb{E}[-\log p_\theta(\mathbf{x}_0)] &\leq \mathbb{E}_q\Big[-\log\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big] \\ &= \mathbb{E}_q\Big[-\log p(\mathbf{x}_T) - \sum_{t\geq 1}\log\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\Big] =: L\end{aligned} \tag{6.4}$$

For variance reduction purpose, the loss can be rewritten as:

$$\begin{aligned}\mathbb{E}_q\Big[\underbrace{D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{L_T} + \sum_{t\geq 1}\underbrace{D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \\ - \underbrace{\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\Big]\end{aligned} \tag{6.5}$$

Based on (6.2), (6.3) and (6.5), L_{t-1} can be computed as:

$$L_{t-1} = \mathbb{E}_q\Big[\frac{1}{2\sigma_t^2}\|\tilde{\mu}_t(\mathbf{x}_t,\mathbf{x}_0) - \mu_\theta(\mathbf{x}_t,t)\|^2\Big] + C \tag{6.6}$$

where C is a constant independent of θ. Notice that $\mathbf{x}_t(\mathbf{x}_0,\epsilon) = \sqrt{\bar{\alpha}t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})$. Thus if we apply (6.3), the loss can be reformulated as:

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0,\epsilon}\Big[\frac{1}{2\sigma_t^2}\|\tilde{\mu}_t\Big(\mathbf{x}_t(\mathbf{x}_0,\epsilon), \frac{\beta_t}{\sqrt{1-\bar{\alpha}}}\epsilon\Big) - \mu_\theta(\mathbf{x}_t,t)\|^2\Big] \tag{6.7}$$

Since $\mathbf{x}_t$ is accessible during training, DDPM chooses the following parametrization:

$$\mu_\theta(\mathbf{x}_t,t) = \frac{1}{\sqrt{\bar{\alpha}}}\Big(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}}}\epsilon_\theta(\mathbf{x}_t,t)\Big) \tag{6.8}$$

Finally, the training is performed based on the following loss function:

$$\mathbb{E}_{\mathbf{x}_0,\epsilon}\Big[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha})}\|\epsilon-\epsilon_\theta(\sqrt{\bar{\alpha}}\mathbf{x}_0+\sqrt{1-\bar{\alpha}}\epsilon,t)\|\Big] \tag{6.9}$$

Algorithm

Based on above sections, we can summarize the two key algorithms of DDPM. Algorithm 6.1 displays the complete training procedure with the simplified objective introduced in [55]. Algorithm 6.2 illustrates the reverse process sampling procedure, where the goal is to predict the error, ϵ.

Algorithm 6.1 DDPM Training

while $L_{\dot{c}}$ *tol* **do**
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t_1, t_2, \ldots, t_k \sim Uniform([T])$
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Perform gradient descent on the mini batch: $\sum_{i=1}^{k} \nabla_\theta \|\epsilon-\epsilon_\theta(\sqrt{\bar{\alpha}}\mathbf{x}_0+\sqrt{1-\bar{\alpha}}\epsilon,t)\|^2$
end

Algorithm 6.2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = 0$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\bar{\alpha}}}\Big(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}}}\epsilon_\theta(\mathbf{x}_t, t)\Big) + \sigma_t\mathbf{z}$
end
return $\mathbf{x}_0$

6.3 Related Work

To our best knowledge, `MedDiff` is the first work to leverage the idea of diffusion based modeling to generate EHRs. Since our work casts insight on the effectiveness of generating EHR via diffusion based models, we expect the extension of this algorithm to more complex EHR can be left for further investigation. Here, we discuss the related works on synthetic EHRs generation and related diffusion based models.

Synthetic EHR generation

Closely related to this work are recent efforts that leverage deep generative models for synthesizing EHRs [7, 20, 117]. MedGAN [20] and CorGAN [117] were introduced to generate patient feature matrices. However, these works rely heavily on the performance of a pre-trained autoencoder model to reduce the dimensionality of the latent variable. Without the pre-trained autoencoder, these GAN-based models fail to generate high-quality samples which highlight the difficulty of using these models for generalizing to multiple institutions (e.g., smaller clinics or different patient distributions). `MedDiff` extends this line of work but is orthogonal to these methods by leveraging a novel and powerful family of generative model, that do not rely on the quality of a pre-trained encoder.

Diffusion Models

Diffusion models are a family of likelihood-based generative models that generate samples by gradually introducing (or diffusing) noise and then reversing the noising process from the data. Several pioneer diffusion-based or score-based generative models have been proposed using similar ideas including diffusion probabilistic models, noise-conditioned score network (NCSN) [114] and denoising diffusion probabilistic models (DDPM). At its core, diffusion models rely on a long Markov chain to generate samples by gradually reversing a noising process.

Diffusion models have several advantages over existing generative modeling families. They do not rely on adversarial training which can be susceptible to mode collapse and are difficult to train. They also offer better diversity coverage, can accommodate flexible model architectures to learn any arbitrary complex data distributions. With respect to image and speech synthesis, diffusion-based models can achieve high-quality, diverse samples that are superior to their GAN counterparts.

However, the process of producing less-noisy samples can be quite expensive in

terms of computing time. There have been several recent works to accelerate the computational speed of the generation process. Denoising Diffusion Implicit Models (DDIMs) [113] accelerate sampling from pre-trained DDPMs by relying on a family of non-Markovian processes. They accelerate the generative process through taking multiple steps in the diffusion process. On the other hand, several other works have been proposed and applied to different domains including.

6.4 Proposed Method: `MedDiff`

In this section, we introduce the three components of `MedDiff`. We first introduce the base architecture. Next, we propose a numerical method to accelerate the generation process and explain the motivation of using it. We then equip `MedDiff` with the ability to conduct conditioned sampling.

DDIM

`MedDiff` uses the recently proposed denoising diffusion implicit models (DDIM) [113] as the backbone as it offers a more flexible framework. Using the same notation described in the background section, DDIM considers the following distributions:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := q(\mathbf{x}_T|\mathbf{x}_0)\prod_{t=2}^{T} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \tag{6.10}$$

where $q(\mathbf{x}_T|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}}\mathbf{x}_0, (1-\bar{\alpha}_T)\mathbf{I})$ and for all $t > 1$,

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma^2}\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma^2\mathbf{I}) \tag{6.11}$$

The reason why the distribution is parameterized in this way is to guarantee the marginal density is equivalent to DDPM. However, the key difference between DDPM and DDIM is that the forward process of DDIM is no longer a Markov process, which

allows different reverse samplers by changing the variance of the reverse noise. It means DDIM could be compatible with other samplers that will be explained later. Thus, the reverse process with a prior $p_\theta(\mathbf{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ can be computed as

$$\begin{aligned} p_\theta(\mathbf{x}_0|\mathbf{x}_1) &= \mathcal{N}\Big(\frac{\mathbf{x}_1 - \sqrt{1-\bar{\alpha}_1}\epsilon_\theta(\mathbf{x}_1, 1)}{\sqrt{\bar{\alpha}_1}}, \sigma_1^2\mathbf{I}\Big) \\ p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) &= q(\mathbf{x}_{t-1}|\mathbf{x}_t, \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}}), \quad t > 1 \end{aligned} \tag{6.12}$$

The model is trained by minimizing the variation lower bound as well. After training, sampling can be done using the following equation:

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\Big(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}}\Big) + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\epsilon_\theta(\mathbf{x}_t, t) + \sigma_t\mathbf{z}_t, \tag{6.13}$$

where $\mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. If we denote $\frac{\sqrt{1-\bar{\alpha}_s}}{\sqrt{\bar{\alpha}_s}}$ by λ_s, then the updating rule for continuous time is

$$\mathbf{x}_s = \frac{\lambda_s}{\lambda_t}[\mathbf{x}_t - \bar{\alpha}_t\epsilon_\theta(\mathbf{x}_t)] + \bar{\alpha}_s\epsilon_\theta(\mathbf{x}_t) \tag{6.14}$$

In a recent paper [105], it is showed that DDIM is an integrator of the probability flow ODE defined in [115]:

$$d\mathbf{x} = [f(\mathbf{x}, t) - \frac{1}{2}g^2(t)\nabla_\mathbf{x}\log p_t(\mathbf{x})]dt \tag{6.15}$$

where $d\mathbf{x} = f(\mathbf{x}, t)dt + g(t)dW$ is a SDE and W is Brownian motion. In practice $\nabla_\mathbf{x}\log p_t(\mathbf{x}) = \frac{\bar{\alpha}_t\epsilon_\theta(\mathbf{x}_t) - \mathbf{x}_t}{\lambda_t^2}$ so $\mathbf{x}_s$ can be regarded as the integrator of $\frac{1}{2}[\bar{\alpha}_t\epsilon_\theta(\mathbf{x}_t) - \bar{\alpha}_t^2\mathbf{x}_t]$.

Model Architecture

In this section, we provide a general architecture of `MedDiff` . First of all, we found a simple fully connected diffusion model is enough for low-dimensional data. However, we notice that directly applying DDPM or score based methods for images and audio can not yield satisfying result. As a result, we build on a modified U-Net architecture

[55, 102] with some modifications including larger model depth/width, positional embeddings, residual blocks for up/downsampling and residual connection rescale. In addition, as we are dealing with 1d data, we choose the 1d convolutional as a particular form of the U-Net instead of 2d convolutional. Preliminary results show that this approach enables us to better capture the neighboring feature correlations.

Accelerated Generation Process

Although the above architecture can provide satisfying results, the speed of generation process can still be a bottleneck. The dilemma is that a small T usually performs worse than a larger T, but a larger T indicates a longer generation process and time. To alleviate this issue, we propose an acceleration algorithm from the perspective of iterative methods. In short, we propose to utilize Anderson Acceleration [5] to run the generation process. Anderson acceleration (AA) is a method to accelerate the convergence of fixed-point iterations. The idea of Anderson acceleration is to approximate the final solution using a linear combination of the previous k iterates. Since solving the proper combination of iterates is a nonlinear procedure, Anderson acceleration is also known as a nonlinear extrapolation method. We start with the generic Anderson acceleration prototype Algorithm 6.3 and explain its implementation. For each iteration $t \geq 0$, AA solves a least squares problem with a normalization constraint. The intuition is to minimize the norm of the weighted residuals of the previous k iterates. The constrained linear least-squares problem in Algorithm 6.3 can be solved in a number of ways. Our preference is to recast it in an unconstrained form suggested in [30, 123] that is straightforward to solve and convenient for implementing efficient updating of QR. The idea of Quick QR-update Anderson acceleration implementation is described in [123] which is commonly used in practice.

Now we give our theoretical motivation and results. There is a long time history of applying Anderson acceleration to Picard iteration for solving differential equations

Algorithm 6.3 Anderson Acceleration Prototype (truncated version)

Input: Initial point w_0, Anderson restart dimension k, fixed-point mapping $g : \mathbf{R}^n \to \mathbf{R}^n$.

Output: w_{t+1}

for *t = 0, 1, ...* **do**

  Set $p_t = \min\{t, k\}$.

  Set $F_t = [f_{t-p_t}, \ldots, f_t]$, where $f_i = g(w_i) - w_i$ for each $i \in [t - p_t, t]$.

  Determine weights $\beta = (\beta_0, \ldots, \beta_{p_t})^T$ that solves $\min_\beta \|F_t\beta\|_2$, s. t. $\sum_{i=0}^{p_t} \beta_i = 1$.

  Set $w_{t+1} = \sum_{i=0}^{p_t} \beta_i g(w_{t-p_t+i})$.

end

[100]. We know we can regard the DDIM as a integrator of an ODE and the updating rule of DDIM can be regarded as the integrator of (6.15) which is approximated by $\frac{1}{2}[\bar{\alpha}_t \epsilon_\theta(\mathbf{x}_t) - \bar{\alpha}_t^2 \mathbf{x}_t]$. Thus, (6.14) can be written as the Picard iteration

$$\mathbf{x}_s = \mathbf{x}_T + \int_T^s \underbrace{\frac{1}{2}[\bar{\alpha}_t \epsilon_\theta(\mathbf{x}_t) - \bar{\alpha}_t^2 \mathbf{x}_t]dt}_{F(\mathbf{x}_t, t)} \tag{6.16}$$

and we can then apply Anderson acceleration to this sequence $\{\mathbf{x}_t\}$.

Assuming $F(\mathbf{x}_t, t)$ is uniformly Lipschitz continuous in $\mathbf{x}$ which is a common assumption made in neural ODE literature [17] and following the results from Theorem 2.3 in [118], we can derive the following acceleration guarantees for applying Anderson acceleration to iterates (intermediate samples) of DDIM.

Theorem 6.4.1. *Assume operator F has a fixed point $\mathbf{x}^*$, and satisfies the following two conditions*

1. *F is Lipschitz continuously differentiable in a ball $B(r) = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}^*\| < r\}$ for some $r > 0$.*

2. *F is locally L-Lipschitz on $B(r)$ where $L < 1$.*

Then if $\|\gamma^t\|_{l_1}$ is uniformly bounded by C_γ for all $t > 0$, `GDA-AM` converges to $\mathbf{x}^*$ locally with contraction ratio $L < \hat{L} < 1$.

Proof. Here we just give a sketch of proof due to space limit. For more details, we refer readers to [118]. Set $G(\mathbf{x}) = F(\mathbf{x}, t) - \mathbf{x}$ and $\delta = \mathbf{x} - \mathbf{x}^*$. For $\hat{r} < r$ sufficiently small and $\mathbf{x} \in B(\hat{r})$, it can be shown that $\|G(\mathbf{x}) - G^{'}(\mathbf{x}^*)\delta\| \leq \frac{\gamma}{2}\|\delta\|^2$ and $\|\delta\|(1-L) \leq \|G(\mathbf{x})\| \leq (1+L)\|\delta\|$ where γ is the Lipschitz constant of $G^{'}(\mathbf{x})$ on $B(\hat{r})$. Then we have $\|G(\mathbf{x}_t)\| \leq \hat{L}^k\|G(\mathbf{x}_0)\|$ for all $0 \leq t \leq T$, which obviously holds for $T = 0$. Leveraging the two inequalities above, it can be shown that

$$\|G(\mathbf{x}_{T+1})\| \leq \underbrace{\frac{\frac{L}{\hat{L}} + \left(\frac{C_\alpha \gamma \hat{r}}{2(1-L)}\right)\hat{L}^{-k-1}}{1 - \frac{\gamma r}{2(1-L)}}}_{C \leq 1} \hat{L}^{T+1}\|G(\mathbf{x}_0)\|$$

To ensure $C \leq 1$, just reduce $\hat{r}$ until it satisfies $\hat{r} < \frac{2(1-L)}{\gamma}$. □

Theorem 6.4.1 provides the theoretical justification for leveraging Anderson acceleration to predict the next denoised intermediate sample. As this combination requires much fewer iterations and reduced computational costs to generate realistic samples, it can dramatically save inference time.

Conditioned Sampling

It is not sufficient for the synthesizer model to produce realistic-looking data, it is equally important that the generated examples also match the label. If a particular class label is passed to a generative model, it should produce a health record that matches the distribution of that label. To equip `MedDiff` with the ability to conduct conditioned sampling which MedGAN and CorGAN do not have, we incorporate the idea of a classifier-guided sampling process [25]. From a conceptual level, the estimated noise $\epsilon_\theta(\mathbf{x}_t, t)$ in each step is deducted by $\sqrt{1-\bar{\alpha}_t}\nabla_{\mathbf{x}_t} \log f_\phi(y \mid \mathbf{x}_t)$, where $f_\phi(y \mid \mathbf{x}_t)$ is a trained classifier on the noisy $\mathbf{x}_t$. Algorithm 6.4 summaries the corresponding accelerated and conditioned sampling algorithm.

Algorithm 6.4 Accelerated Conditioned Sampling

Choose a small integer k as Anderson restart dimension
Given a random noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and label y
for $t = T, \ldots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = 0$
  $\epsilon = \epsilon_\theta(\mathbf{x}_t) - \sqrt{1 - \bar{\alpha}_t} \nabla_{\mathbf{x}_t} \log p_\phi(y \mid \mathbf{x}_t)$

  Update $\mathbf{x}_{t-1}$ according to equation (6.14)
  $\mathbf{x}_{t-1} = Anderson(\mathbf{x}_t, \mathbf{x}_{t-1}, k)$
end
return $\mathbf{x}_0$

6.5 Experiments

In this section, we answer following questions: **Q1:** Can `MedDiff` generate high quality synthetic EHRs compared to existing methods? **Q2:** Can `MedDiff` accurately generate conditioned samples that matches the distribution of the given label? **Q3:** Can our proposed acceleration technique alleviate the issue of slow generation process of a diffusion model? How is its performance under different settings? We first provide descriptions for two health datasets. Next, we then give an overview of baseline methods and implementation details. We then evaluate the effectiveness of the proposed acceleration technique with an ablation study.

Datases

We use the following two publicly available datasets:

1. MIMIC-III [60]: MIMIC-III is large database containing de-identified health data associated with approximately sixty thousand admissions of critical care unit patients from the Beth Israel Deaconess Medical Center collected between 2001 and 2012. For each patient, we extract the International Classification of Diseases (ICD-9) diagnosis codes. We represent a patient record as a fixed-size vector with 1071 entries for each patient record. The preprocess datasets is a 46520×1071 binary matrix. This dataset is used for experiments with binary

discrete variables and evaluation of proposed acceleration technique.

2. Patient Treatment Classification: The dataset is Electronic Health Record Predicting collected from a private Hospital in Indonesia. It contains the 8 different laboratory test results of 3309 patient used to determine next patient treatment whether in care or out care. We use this datasets to perform continuous synthetic EHR generation and investigate the effectiveness of conditional generation of `MedDiff` . It is available at Kaggle[1].

Baselines and Implementation

We compare `MedDiff` with following methods.

1. MedGAN [20] [2]: It generates the representations of patient records with a GAN. It then decodes them to simulated patient records with an autoencoder.

2. CorGAN [117] [3]: Similar to MedGAN, it is a framework that combines Convolutional Generative Adversarial Networks and Convolutional Autoencoders.

3. Noise Conditional Score Network (NCSN) [114][4]: Instead of learning directly the probability of the data $\log p(x)$, this method rather aims to learn the gradients of $\log p(x)$ with respect to x. This can be understood as learning the direction of highest probability at each point in the input space. Therefore, when the model is trained, we can improve a sample by moving it along the directions of highest probability. After training, the sampling process is achieved by applying Langevin dynamics.

4. DDPM [55][5] : A class of latent variable models inspired by considerations from

[1] `https://www.kaggle.com/manishkc06/patient-treatment-classification`
[2] `https://github.com/astorfi/cor-gan`
[3] `https://github.com/astorfi/cor-gan`
[4] We adopt the codes from `https://github.com/acids-ircam/diffusion_models`
[5] We adopt the codes from `https://github.com/lucidrains/denoising-diffusion-pytorch`

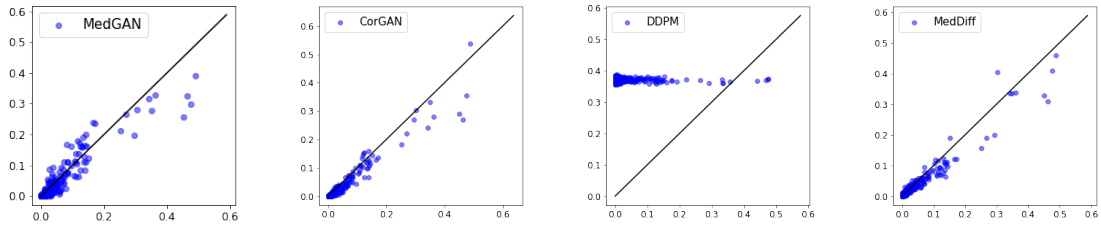
nonequilibrium thermodynamics [112]. In our experiments, we found a direct application of both DDPM and NCSN can not yield high quality samples.

We implemented `MedDiff` with Pytorch. For training models, we used Adam [65] with the learning rate set to 0.001, and a mini-batch of 128 for MIMIC-III and 64 for the patient treatment dataset on a machine equipped with one Nvidia GeForce RTX 3090 and CUDA 11.2. Hyperparamters of `MedDiff` are selected after grid search without too much effort. We use a timestep T of 200, a noise scheduling β from 1×10^{-4} to 1×10^{-2} and a table size $k = 3$. The code will be public available upon publication.

Quantitative Evaluation

We first evaluate the effectiveness of `MedDiff` on MIMIC-III (a binary dataset). Following previous works CorGAN and MedGAN, we use the dimension-wise probability as a basic evaluation metric to see if `MedDiff` can learn the distribution of the real data (for each dimension). This measurement refers to the Bernoulli success probability of each dimension (each dimension is a unique ICD-9 code). We report the dimension-wise probability in Figure 6.2. We also use the correlation coefficient ρ and sum of absolute errors (SAE) as our quantitative metrics. From Figure 6.2, we can observe `MedDiff` shows the best performance.

Next, we answer the question that if `MedDiff` allows the conditioned sampling of health records with a given label. We use the Gaussian Kernel Density Estimation to depict the underlying probability density function of the original dataset and generated samples. We show the results of `MedDiff` for unconditional sampling in Figure 6.3a and conditional sampling in Figures 6.3b and 6.3c. We can observe that the distribution of in-care patients and out-care patients are different in terms of shape, local modes and range. It means the generated samples would be more informative and useful for data sharing if it comes with a fake label. However all baselines do not



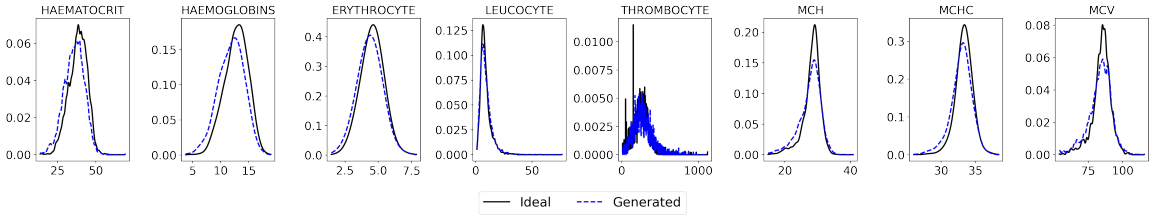
(a) $\rho = 0.94, SAE = 5.68$ (b) $\rho = 0.97, SAE = 4.49$ (c) $\rho = -0.06, SAE = 366$ (d) $\rho = 0.98, SAE = 4.16$

Figure 6.2: The scatter plots of dimension-wise probability. Each point depicts one unique diagnosis code. The x-axis and y-axis represent the Bernoulli success probability for real and synthetic datasets, respectively. The diagonal line shows the ideal case.

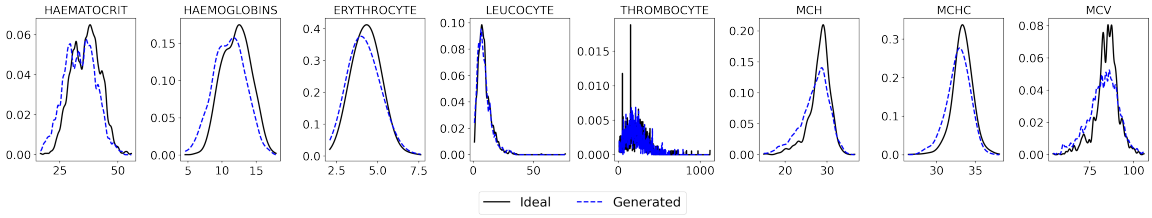
support conditioned sampling for EHRs, this experiment demonstrates the flexibility and advantage of `MedDiff` .

Noising and Denoising Visualization

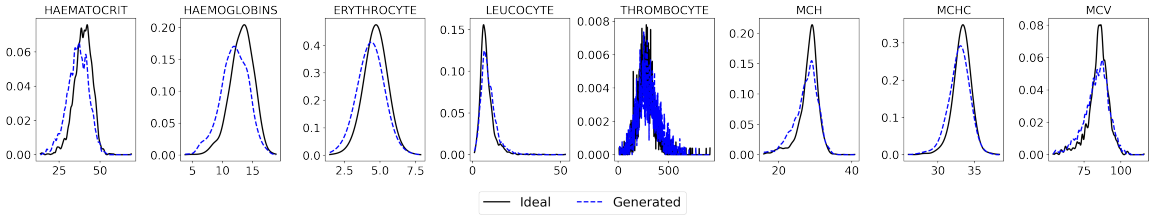
Figure 6.4a shows that the forward process is to destroy the input by adding scaled random Gaussian noise step by step. If we can reverse the above process and sample from $q(x_{t-1}|x_t)$, we will be able to recreate the true sample from a Gaussian noise input. We cannot easily estimate $q(x_{t-1}|x_t)$, so we learn a model p_θ using neural networks to approximate these conditional probabilities in order to run the reverse diffusion process. Figure 6.4b shows that given a random noise, `MedDiff` is able to generate a new sample. Using one sample for illustration, Figure 6.4 depicts the forward and reverse process of a diffusion model. In addition, we can observe that `MedDiff` is able to reconstruct inputs if it uses a non-perturbed sampling procedure (set the posterior variance σ_t as 0) and the noise input for the reverse process is exactly the last noisy input from a forward process. By setting the posterior variance σ_t as a nonzero number and not sharing the destroyed inputs after the forward process, it is guaranteed to achieve sample diversity with a bare risk of data leakage.



(a) *Black:* KDE for all 3309 patient records. *Blue:* KDE for 3309 Unconditioned synthetic records.

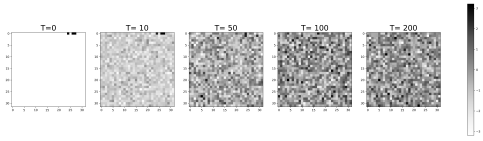


(b) *Black:* KDE for 1317 in-care patient records. *Blue:* KDE for 1317 conditioned synthetic in-care patient records.

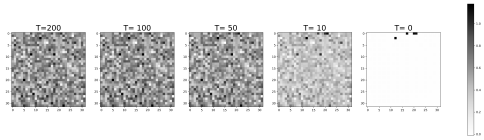


(c) *Black:* KDE for 1992 out-care patient records. *Blue:* KDE for 1992 conditioned synthetic out-care patient records.

Figure 6.3: Kernel density estimation for each feature. *Black*: true density. *Blue*: KDE wit generated samples. The benchmark (true) estimated distribution is obtained from the original dataset using a Gaussian Kernel. We can observe that with the classifier guided sampling, `MedDiff` is able to match local modes and thus generate realistic and useful conditioned samples.



(a) Forward Process: Adding random Gaussian noise



(b) Reverse Process: : Intermediate samples

Figure 6.4: Visualization of the forward and backward process. It is worth to mention that we are able to reconstruct the noised input X_T by by fixing the posterior variance σ_t as 0 and running the denosing step.

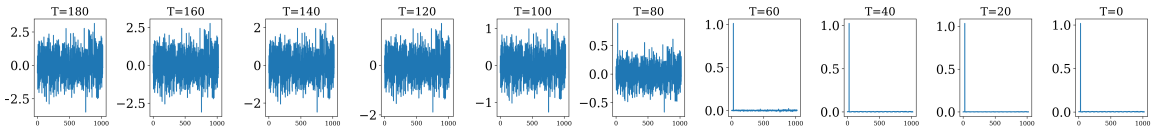
Effects of Accelerated Sampling

Next, we evaluate the usefulness of our proposed accelerated sampling algorithm. For $T = 200$, we present the evolution process of generating one sample by use of the regular procedure in Figure 6.5a and `MedDiff` in Figure 6.5b. Similarly, Figures 6.5c and 6.5d depict the evolution process of generating one sample for $T = 100$. Since $T = 200$ means the start time step which means the current iteration number of reverse process is 0, the iteration number can be defined as $T - t$. When $T = 200$, we can observe `MedDiff` converges in around 80 iterations and the regular sampling process takes around 160 iterations, which implies a $2\times$ speed up in terms of iteration number. A similar speed-up in terms of computational time is also observed, which is shown in the next experiment. In addition, it can be observed from Figures 6.5d and 6.5c that `MedDiff` can generate a high-quality sample when the regular sampling process fails to converge.

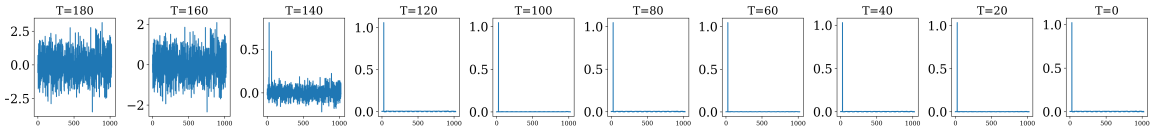
We also perform an additional ablation quantitatively study of `MedDiff` to investigate whether the `MedDiff` has actually accelerated the generation process with different hyperparameters. We use the root mean squared error of the Bernoulli success probability of each dimension for the generated samples and original dataset, which is defined as follows,

$$RMSE = \sqrt{\frac{\sum_{d=1}^{D} (p_d - \hat{p}_d)^2}{D}}, \tag{6.17}$$

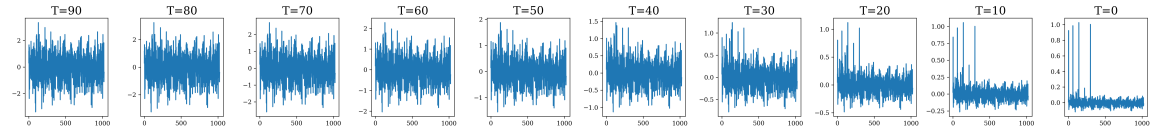
where D is the feature size. We run `MedDiff` using different T and Anderson acceleration restart dimension (or table size) k and show the results in Figure 6.6. Figures 6.6a and 6.6c indicate an increasing of table size results in faster convergence in terms of iteration number. Figures 6.6b and 6.6d show that although there is additional computation, it does not hinder the benefits of adopting the accelerated algorithm with a small k. In addition, Figures 6.6a and 6.5d verify that adopting the accelerated



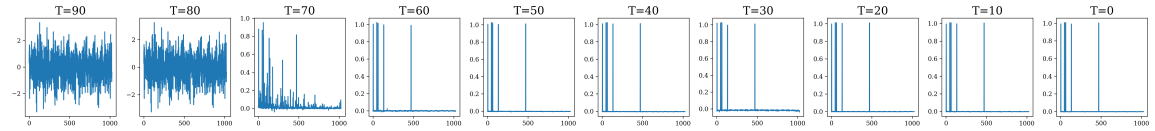
(a) Regular Reverse Process for $T = 200$: it takes about 160 iterations to converge.



(b) Accelerated Reverse Process for $T = 200$: it takes about 80 iterations to converge, which implies a $2\times$ speed up.

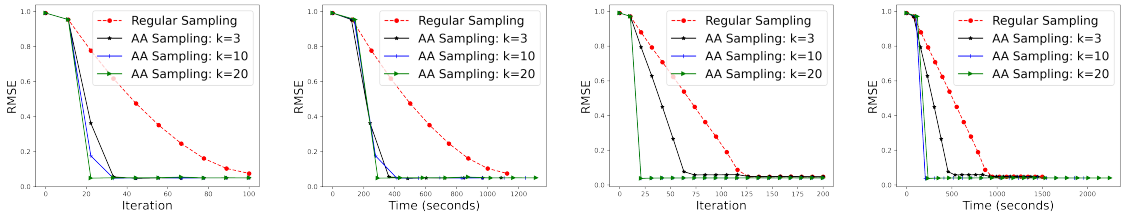


(c) Regular Reverse Process for $T = 100$: the generated sample is still very noisy after 100 iterations.



(d) Accelerated Reverse Process for $T = 100$: After 40 iteration, `MedDiff` is able to generate a high-quality sample.

Figure 6.5: Evolution plot of regular generation process (6.5a, 6.5c) and `MedDiff` (6.5b, 6.5d).



(a) Comparison in terms of iterations, $T = 100$

(b) Comparison in terms of time, $T = 100$

(c) Comparison in terms of iterations, $T = 200$

(d) Comparison in terms of time, $T = 200$

Figure 6.6: Results of accelerated sampling versus regular sampling with different T and k.

algorithm can also help to generate high-quality samples if the model is trained with fewer time step T.

Case Study: Discriminative Analysis

Classification models are often developed on EHR data to determine next patient treatment whether in care or out care. To evaluate the utility of our synthetic EHRs, we test how well such prediction tasks are supported by the synthetic conditioned EHR data. We use an 80/20 training/test split on the patient treatment dataset. Next, we use the training set (2647 patients) to train the generative model and generate conditioned in care and out care synthetic data. Then, we report the performance of the real test set (662 patients) using a classier trained on synthetic data only. A logistic regression classier trained on this data produced an AUC score of 0.7419, which is comparable to the real data AUC score of 0.7565.

We also investigate whether the synthetic EHRs generated by `MedDiff` is beneficial for data augmenting, we augment the training set with varying amounts of synthetically generated data and repeated the predictive experiments. The results are shown in Figure 6.7. It can be observed that augmenting the training set with additional 2.5K synthetic records can outperforms the real data. In addition, training with 2.5K synthetic data can have comparable prediction ability to real data, which

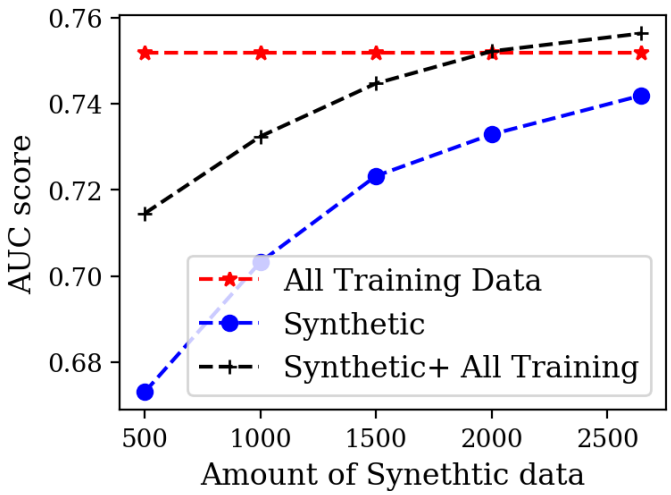


Figure 6.7: Data augmentation performance of `GDA-AM` as a function of the number of synthetic records

demonstrates the effectiveness of `MedDiff` in generating conditioned records.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we presented our works on developing efficient algorithms for machine learning models. We first show the effectiveness of parallel algorithms on accelerating tensor decomposition by proposing SGranite. In SGranite, we propose to incorporate a generalized form of regularization and special partitioned parallel algorithm. Using two real world datasets, we demonstrate that SGranites enables a tensor decomposition to efficiently extract more sparse, distinct and discriminative latent factors. We then present our works on numerical algorithms for accelerating machine learning models. We proposed Fast-CP for tensor decomposition, GDA-AM for minimax optimization and MedDiff for synthetic EHR generation. All these works share a key idea that most machine learning tasks can be viewed as fixed-point iteration problems and be solved efficiently by extrapolation algorithms. Our experimental result show that such extrapolation technique accelerates training by using less iterates and time and thus verify our claim.

7.2 Future Work

This dissertation can be extended in the following direction.

- Short-Term Anderson Acceleration: In machine learning, we often encounter practical situations where the number of parameters is quite large and for this reason, it is not practical to use a extreme large number of vectors in a scheme like Anderson acceleration. A recent work [129] proposed a heuristic correction to achieve fast computation. However, we noted that Anderson Mixing has a strong connection to second order methods and we are investigating whether or not the structure of the Hessian can be exploited to the reduce memory cost. As a result, it is reasonable to improve the algorithm by utilizing the symmetry in the Hessian. Ideally, this memory efficient method can solve problems that many traditional methods fail, including nonlinear mappings and indefinite linear systems. In addition to the short-term recurrence Anderson Acceleration, another contribution is to explore advanced preconditioning methods in the context of acceleration for machine learning problems. For example, in the usual setting of a linear system $Ax = b$, quasi-Newton methods play the role of an accelerator and the preconditioning is applied to transform the system into one like $MAx = Mb$ which is easier to solve. However, preconditioners developed so far in the literature are rather simple and not efficient enough. We can incorporate advanced preconditioning techniques into extrapolation algorithms because we find that approximating inverse preconditioning methods can indeed be more efficient by taking advantage of sparsity for sparse problems.

- Higher-Order MedDiff: MedDiff discussed above can only generate 2 dimensional health records. However, modern health records are usuaaly high-dimensional, meaning that the effectiveness of generated samples by MedDiff are still limited. To address this issue and generate more realistic high-dimensional EHRS,

it would be important to investigate several principles involved in denoising diffusion probabilistic models, which include but are not limited to the Markov chains process, sampling procedure, noise scales progression, and controllable generation for inverse problem solving.

- Efficient Uncertainty Quantification: The other perspective is to improve decision making by incorporating efficient uncertainty quantification (UQ) that requires less memory and computational cost. Given the large volume of high-dimensional data, it is of importance to develop efficient UQ methods. We can utilize lottery ticket pruning so that it can reduce the parameter counts of networks and decrease storage requirements. In the meantime, we can avoid the additional computational cost (multiple runs) of sampling based methods by training multiple subnetworks independently in one run using snapshot learning rate scheduling. Preliminary results have shown that this framework can bring a significant improvement in negative log-likelihood, accuracy, and calibration error for image classification problems.

Bibliography

[1] Evrim Acar, Daniel M Dunlavy, and Tamara G Kolda. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics*, 25(2):67–86, February 2011.

[2] Leonard Adolphs, Hadi Daneshmand, Aurelien Lucchi, and Thomas Hofmann. Local saddle point optimization: A curvature exploitation approach. Proceedings of Machine Learning Research. PMLR.

[3] Ardavan Afshar, Ioakeim Perros, Evangelos E Papalexakis, Elizabeth Searles, Joyce C Ho, and Jimeng Sun. COPA: Constrained PARAFAC2 for sparse & large datasets. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 793–802, 2018.

[4] Donald G. Anderson. Iterative procedures for nonlinear integral equations. 1965.

[5] Donald G Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560, 1965.

[6] Waïss Azizian, Damien Scieur, Ioannis Mitliagkas, Simon Lacoste-Julien, and Gauthier Gidel. Accelerating smooth games by manipulating spectral shapes. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 1705–1715. PMLR, 2020.

[7] Brett K. Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P. Bhavnani, James Brian Byrd, and Casey S. Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes*.

[8] Alex Beutel, Partha Pratim Talukdar, Abhimanu Kumar, Christos Faloutsos, Evangelos E Papalexakis, and Eric P Xing. FlexiFaCT - Scalable Flexible Factorization of Coupled Tensors on Hadoop. *SDM*, 2014.

[9] Alex Beutel, Partha Pratim Talukdar, Abhimanu Kumar, Christos Faloutsos, Evangelos E Papalexakis, and Eric P Xing. FlexiFaCT - Scalable Flexible Factorization of Coupled Tensors on Hadoop. *SDM*, pages 109–117, 2014.

[10] Simon Bing, Andrea Dittadi, Stefan Bauer, and Patrick Schwab. Conditional generation of medical time series for extrapolation to underrepresented populations, 2022.

[11] Siddharth Biswal, Soumya Ghosh, Jon Duke, Bradley A. Malin, Walter F. Stewart, and Jimeng Sun. EVA: generating longitudinal electronic health records using conditional variational autoencoders. *CoRR*, 2020.

[12] Raghu Bollapragada, Damien Scieur, and Alexandre d'Aspremont. Nonlinear acceleration of momentum and primal-dual algorithms. *arXiv preprint arXiv:1810.04539*, 2018.

[13] Claude. Brezinski, Michela. Redivo-Zaglia, and Yousef. Saad. Shanks sequence transformations and anderson acceleration. *SIAM Review*, 60(3):646–669, 2018.

[14] Ronald E. Bruck. On the weak convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in hilbert space. *Journal of Mathematical Analysis and Applications*, 1977.

[15] Daniela Calvetti, Bryan Lewis, and Lothar Reichel. On the regularizing properties of the gmres method. *Numerische Mathematik*, 91(4):605–625, 2002.

[16] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2021.

[17] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[18] Eric C Chi and Tamara G Kolda. On tensors, sparsity, and nonnegative factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299, 2012.

[19] Eric C Chi and Tamara G Kolda. On tensors, sparsity, and nonnegative factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299, 2012.

[20] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. In *Proceedings of the 2nd Machine Learning for Healthcare Conference*, Proceedings of Machine Learning Research. PMLR, 2017.

[21] Michel Crouzeix and César Palencia. The numerical range is a (1+2)-spectral set. *SIAM Journal on Matrix Analysis and Applications*, 38(2):649–655, 2017.

[22] C. Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. *ArXiv*, abs/1711.00141, 2018.

[23] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL `https://arxiv.org/abs/1810.04805`.

[25] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[26] John C Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. *ICML*, 2008.

[27] Khaled El Emam, Sam Rodgers, and Bradley Malin. Anonymising and sharing individual patient data. 2015.

[28] Howard C Elman. *Iterative methods for large, sparse, nonsymmetric systems of linear equations*. PhD thesis, Yale University New Haven, Conn, 1982.

[29] Alvin Rajkomar et al. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1, 2018.

[30] Haw-ren Fang and Yousef Saad. Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications*, 16(3):197–221, 2009.

[31] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, mar

2019. doi: 10.1007/s10618-019-00619-1. URL https://doi.org/10.1007%2Fs10618-019-00619-1.

[32] Bernd Fischer and Roland Freund. Chebyshev polynomials are not always optimal. *Journal of Approximation Theory*, 65(3):261–272, 1991.

[33] Gemulla, Rainer, Nijkamp, Erik, Haas, Peter J, and Sismanis, Yannis. *Large-scale matrix factorization with distributed stochastic gradient descent*. ACM, New York, New York, USA, August 2011.

[34] Saeed Ghadimi and Guanghui Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Math. Program.*, 156(1-2):59–99, 2016. doi: 10.1007/s10107-015-0871-8. URL https://doi.org/10.1007/s10107-015-0871-8.

[35] Gauthier Gidel, Hugo Berard, Pascal Vincent, and Simon Lacoste-Julien. A variational inequality perspective on generative adversarial nets. *CoRR*, abs/1802.10551, 2018. URL http://arxiv.org/abs/1802.10551.

[36] Gauthier Gidel, Hugo Berard, Gaëtan Vignoud, Pascal Vincent, and Simon Lacoste-Julien. A variational inequality perspective on generative adversarial networks. In *7th International Conference on Learning Representations, ICLR*, 2019.

[37] Gauthier Gidel, Reyhane Askari Hemmat, Mohammad Pezeshki, Rémi Le Priol, Gabriel Huang, Simon Lacoste-Julien, and Ioannis Mitliagkas. Negative momentum for improved game dynamics. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research. PMLR, 2019.

[38] I. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-

Farley, S. Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[39] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. NIPS'14, 2014.

[40] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[41] Anne Greenbaum. *Iterative methods for solving linear systems*. SIAM, 1997.

[42] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2014.

[43] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2017.

[44] Samantha Hansen, Todd Plantenga, and Tamara G. Kolda. Newton-based optimization for kullback–leibler nonnegative tensor factorizations. *Optimization Methods and Software*, 30(5):1002–1029, 2015.

[45] R. Harshman. Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-model factor analysis. 1970.

[46] Richard A. Harshman, Peter Ladefoged, Heinrich Graf von Reichenbach, Robert I. Jennrich, Dale Terbeek, Lee Cooper, Andrew L. Comrey, Peter M.

Bentler, Jeanne Yamane, and Diane Vaughan. Foundations of the parafac procedure: Models and conditions for an "explanatory" multimodal factor analysis. 1970.

[47] Kosuke Haruki, Taiji Suzuki, Yohei Hamakawa, Takeshi Toda, Ryuji Sakai, Masahiro Ozawa, and Mitsuhiro Kimura. Gradient noise convolution (gnc): Smoothing loss function for distributed large-batch sgd. *ArXiv*, abs/1906.10822, 2019.

[48] Huan He, Jette Henderson, and Joyce C Ho. SGranite: Distributed Tensor Decomposition for Large Scale Health Analytics. *The Web Conference*, 2019.

[49] Huan He, Yuanzhe Xi, and Joyce C Ho. Fast and accurate tensor decomposition without a high performance computing machine. In *2020 IEEE International Conference on Big Data (Big Data)*, 2020.

[50] Jette Henderson, Huan He, Bradley Malin, Joshua Denny, Abel Kho, Joydeep Ghosh, and Joyce Ho. Phenotyping through semi-supervised tensor factorization (psst). *AMIA Annual Symposium proceedings. AMIA Symposium*, 2018.

[51] Jette Henderson, Ryan Bridges, Joyce C Ho, Byron C Wallace, and Joydeep Ghosh. PheKnow-Cloud: A Tool for Evaluating High-Throughput Phenotype Candidates using Online Medical Literature. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, 2017:149–157, 2017.

[52] Jette Henderson, Joyce C Ho, Abel N Kho, Joshua C Denny, Bradley A Malin, Jimeng Sun, and Joydeep Ghosh. Granite - Diversified, Sparse Tensor Factorization for Electronic Health Record-Based Phenotyping. *ICHI*, 2017.

[53] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a

local nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017.

[54] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020. URL `https://arxiv.org/abs/2006.11239`.

[55] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[56] Joyce C Ho, Joydeep Ghosh, and Jimeng Sun. Marble - high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. *KDD*, 2014.

[57] Yu-Guan Hsieh, F. Iutzeler, J. Malick, and P. Mertikopoulos. On the convergence of single-call stochastic extra-gradient methods. In *NeurIPS*, 2019.

[58] Ginsberg J, Mohebbi MH, Patel RS, Brammer L, Smolinski MS, and Brilliant L. Detecting influenza epidemics using search engine query data. *Nature*, 457 (7232), 2008.

[59] Chi Jin, Praneeth Netrapalli, and Michael Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 4880–4889. PMLR, 2020.

[60] Alistair Johnson, Tom Pollard, Lu Shen, Li-wei Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Celi, and Roger Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 2016.

[61] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[62] Yejin Kim, Robert El-Kareh, Jimeng Sun, Hwanjo Yu, and Xiaoqian Jiang. Discriminative and Distinct Phenotyping by Constrained Tensor Factorization. *Scientific Reports*, 7(1):1114, April 2017.

[63] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[64] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv.org*, page arXiv:1412.6980, December 2014.

[65] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[66] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[67] Tamara G. Kolda and David Hong. Stochastic gradients for large-scale tensor decomposition. arXiv, June 2019. submitted for publication.

[68] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[69] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *ArXiv*, abs/1611.01236, 2017.

[70] V Lampos, A C Miller, S Crossan, C Stefansen Scientific reports, and 2015. Advances in nowcasting influenza-like illness rates using search query logs. *nature.com*.

[71] Y. LeCun, L. Bottou, Yoshua Bengio, and P. Haffner. Gradient-based learning applied to document recognition. 1998.

[72] Qi Lei, Sai Ganesh Nagarajan, Ioannis Panageas, and Xiao Wang. Last iterate convergence in no-regret learning: constrained min-max optimization for convex-concave landscapes. In *AISTATS*, 2021.

[73] S. Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart J. Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *AAAI*, 2019.

[74] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Extrapolation for large-batch training in deep learning. *CoRR*, abs/2006.05720, 2020. URL `https://arxiv.org/abs/2006.05720`.

[75] Tianyi Lin, Chi Jin, and Michael I. Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In *ICML*, pages 6083–6093, 2020. URL `http://proceedings.mlr.press/v119/lin20a.html`.

[76] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[77] Luo Luo, Haishan Ye, Zhichao Huang, and Tong Zhang. Stochastic recursive gradient descent ascent for stochastic nonconvex-strongly-concave minimax problems. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, 2020.

[78] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018,*, 2018.

[79] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

[80] Eric V. Mazumdar, Michael I. Jordan, and S. Sastry. On finding local nash equilibria (and only local nash equilibria) in zero-sum games. *ArXiv*, abs/1901.00838, 2019.

[81] Scott McLachlan, Kudakwashe Dube, and Thomas Gallagher. Using the caremap with health incidents statistics for generating the realistic synthetic electronic healthcare record. *2016 IEEE International Conference on Healthcare Informatics (ICHI)*, 2016.

[82] Panayotis Mertikopoulos, Bruno Lecouat, Houssam Zenati, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras. Optimistic mirror descent in saddle-point problems: Going the extra gradient mile. In *7th International Conference on Learning Representations, ICLR*, 2019.

[83] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 2017.

[84] Riccardo Miotto, Li Li, Brian A. Kidd, and Joel T. Dudley. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, 6, 2016.

[85] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *ArXiv*, abs/1802.05957, 2018.

[86] Aryan Mokhtari, Asuman Ozdaglar, and Sarath Pattathil. A unified analysis of extra-gradient and optimistic gradient methods for saddle point problems:

Proximal point approach. In *International Conference on Artificial Intelligence and Statistics*, pages 1497–1507. PMLR, 2020.

[87] Aryan Mokhtari, Asuman Ozdaglar, and Sarath Pattathil. A unified analysis of extra-gradient and optimistic gradient methods for saddle point problems: Proximal point approach. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research. PMLR, 2020.

[88] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, 2008.

[89] A. Nemirovski. Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM J. Optim.*, 2004.

[90] Feng Niu, Benjamin Recht, Christopher Re, and Stephen J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent, 2011. URL `https://arxiv.org/abs/1106.5730`.

[91] Maher Nouiehed, Maziar Sanjabi, Tianjian Huang, Jason D. Lee, and Meisam Razaviyayn. *Solving a Class of Non-Convex Min-Max Games Using Iterative First Order Methods*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[92] Olson, Donald R, Konty, Kevin J, Paladini, Marc, Viboud, Cécile, and Simonsen, Lone. Reassessing Google Flu Trends Data for Detection of Seasonal and Pandemic Influenza - A Comparative Epidemiological Study at Three Geographic Scales. *PLoS Computational Biology*, 9:e1003256–, 2013.

[93] Beng Chin Ooi, Kian-Lee Tan, Sheng Wang, Wei Wang, Qingchao Cai, Gang Chen, Jinyang Gao, Zhaojing Luo, Anthony K.H. Tung, Yuan Wang, Zhongle

Xie, Meihui Zhang, and Kaiping Zheng. Singa: A distributed deep learning platform. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, page 685–688, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334594. doi: 10.1145/2733373.2807410. URL https://doi.org/10.1145/2733373.2807410.

[94] Dmitrii M. Ostrovskii, Andrew Lowy, and Meisam Razaviyayn. Efficient search of first-order nash equilibria in nonconvex-concave smooth min-max problems, 2021.

[95] Evangelos E Papalexakis. Automatic Unsupervised Tensor Mining with Quality Assessment. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 711–719, Philadelphia, PA, August 2016. Society for Industrial and Applied Mathematics.

[96] Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. ParCube - Sparse Parallelizable CANDECOMP-PARAFAC Tensor Decomposition. *TKDD*, 10(1):1–25, 2015.

[97] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[98] Jack Parker-Holder, Luke Metz, Cinjon Resnick, Hengyuan Hu, Adam Lerer, Alistair Letcher, Alexander Peysakhovich, Aldo Pacchiano, and Jakob Foerster. Ridge rider: Finding diverse solutions by following eigenvectors of the hessian. In *Advances in Neural Information Processing Systems*, 2020.

[99] Philip M Polgreen, Yiling Chen, David M Pennock, and Forrest D Nelson. Using Internet Searches for Influenza Surveillance. *Clinical Infectious Diseases*, 47(11):1443–1448, 2008.

[100] Sara Pollock, Leo G Rebholz, and Mengying Xiao. Anderson-accelerated convergence of picard iterations for incompressible navier–stokes equations. *SIAM Journal on Numerical Analysis*, 57(2):615–637, 2019.

[101] L. Popov. A modification of the arrow-hurwicz method for search of saddle points. *Mathematical notes of the Academy of Sciences of the USSR*, 1980.

[102] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[103] Youcef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 1986.

[104] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[105] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[106] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016.

[107] Florian Schaefer and Anima Anandkumar. Competitive gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2019.

[108] Damien Scieur, Alexandre d'Aspremont, and Francis Bach. Regularized nonlinear acceleration. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 712–720, USA, 2016. Curran Associates Inc.

[109] Damien Scieur, Alexandre d'Aspremont, and Francis Bach. Regularized nonlinear acceleration. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 712–720, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[110] Damien Scieur, Edouard Oyallon, Alexandre d'Aspremont, and Francis Bach. Nonlinear acceleration of deep neural networks. *arXiv preprint arXiv:1805.09639*, 2018.

[111] Shaden Smith, Jongsoo Park, and George Karypis. HPC formulations of optimization algorithms for tensor completion. *Parallel Computing*, 74(Complex Syst. 3 4 1989):99–117, 2018. ISSN 0167-8191.

[112] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15. JMLR.org, 2015.

[113] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[114] Yang Song and Stefano Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution*. 2019.

[115] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[116] Kiran K Thekumparampil, Prateek Jain, Praneeth Netrapalli, and Sewoong Oh. Efficient algorithms for smooth minimax optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2019.

[117] Amirsina Torfi and Edward A. Fox. Corgan: Correlation-capturing convolutional generative adversarial networks for generating synthetic healthcare records. In *FLAIRS Conference*, 2020.

[118] Alex Toth and CT Kelley. Convergence analysis for anderson acceleration. *SIAM Journal on Numerical Analysis*, 53(2):805–819, 2015.

[119] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[120] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

[121] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[122] Homer F Walker and Peng Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

[123] Homer F. Walker and Peng Ni. Anderson acceleration for fixed-point iterations. 2011.

[124] Lu Wang, Zhengwu Zhang, and David Dunson. Common and individual structure of brain networks. *The Annals of Applied Statistics*, 13:85–112, 03 2019. doi: 10.1214/18-AOAS1193.

[125] Yichen Wang, Robert Chen, Joydeep Ghosh, Joshua C Denny, Abel N Kho, You Chen, Bradley A Malin, and Jimeng Sun. Rubik - Knowledge Guided Tensor Factorization and Completion for Health Data Analytics. *KDD*, pages 1265–1274, 2015.

[126] Yuanhao Wang, Guodong Zhang, and Jimmy Ba. On solving minimax optimization locally: A follow-the-ridge approach. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[127] Chen-Yu Wei, Chung-Wei Lee, Mengxiao Zhang, and Haipeng Luo. Linear last-iterate convergence in constrained saddle-point optimization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=dx11_7vm5_r`.

[128] Fuchao Wei, Chenglong Bao, and Yang Liu. Stochastic anderson mixing for nonconvex stochastic optimization. *arXiv preprint arXiv:2110.01543*, 2021.

[129] Fuchao Wei, Chenglong Bao, and Yang Liu. A class of short-term recurrence anderson mixing methods and their applications. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=_X90SIKbHa`.

[130] Wei Wen, Yandan Wang, Feng Yan, C. Xu, Chunpeng Wu, Yiran Chen, and Hongbing Li. Smoothout: Smoothing out sharp minima to improve generalization in deep learning. *arXiv: Learning*, 2018.

[131] Yue Wu, Pan Zhou, A. Wilson, E. Xing, and Zhiting Hu. Improving gan training with probability ratio clipping and sample reweighting. *ArXiv*, abs/2006.06900, 2020.

[132] P. Wynn. Acceleration techniques for iterated vector and matrix. 16:301–322, 1962.

[133] Zi Xu, Huiling Zhang, Yang Xu, and Guanghui Lan. A unified single-loop alternating gradient projection algorithm for nonconvex-concave and convex-nonconcave minimax problems, 2021.

[134] Chao Yan, Ziqi Zhang, Steve Nyemba, and Bradley A. Malin. Generating electronic health records with multiple data types and constraints, 2020.

[135] Minghan Yang, A. Milzarek, Z. Wen, and T. Zhang. A stochastic extra-step quasi-newton method for nonsmooth nonconvex optimization. *arXiv: Optimization and Control*, 2019.

[136] Shihao Yang, Mauricio Santillana, John S Brownstein, Josh Gray, Stewart Richardson, and S C Kou. Using electronic health records and Internet search information for accurate influenza forecasting. *BMC Infectious Diseases*, 17(1): 89, May 2017.

[137] Yasin Yazici, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in GAN training. In *7th International Conference on Learning Representations, ICLR , 2019*, 2019.

[138] Hwanjo Yu Xiaoqian Jiang Yejin Kim, Jimeng Sun. Federated Tensor Factorization for Computational Phenotyping. *KDD*, 2017:887 895, 2017. ISSN 2154-817X.

[139] M Zaharia, R S Xin, P Wendell, T Das Communications of the, and 2016. Apache spark: a unified engine for big data processing. *dl.acm.org*.

[140] Guodong Zhang and Yuanhao Wang. On the suboptimality of negative momentum for minimax optimization. In *AISTATS*, 2021.

[141] Guodong Zhang, Yuanhao Wang, Laurent Lessard, and Roger B. Grosse. Don't fix what ain't broke: Near-optimal local convergence of alternating gradient descent-ascent for minimax optimization. *CoRR*, 2021.

[142] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. *CoRR*, 2019. URL `http://arxiv.org/abs/1901.08573`.

[143] Jiawei Zhang, Peijun Xiao, Ruoyu Sun, and Zhiquan Luo. A single-loop smoothed gradient descent-ascent algorithm for nonconvex-concave min-max problems. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, 2020.

[144] Fan Zhou and Guojing Cong. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 3219–3227. ijcai.org, 2018.

[145] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Series B Stat. Methodol.*, 67(2):301–320, 2005.