ABSTRACT

SAANCHI, YENG. Impact of the Choice of Hyper-parameters on Statistical Inference using SGD Estimates and Optimal Experimental Designs for Precision Medicine with Multi-component Treatments. (Under the direction of Chair Dr. Leonard Stefanski and Chair Dr. Eric Laber).

This dissertation consists of two distinct projects. The first is a review of Stochastic Gradient Descent (SGD) and its applications in Statistics, as well as the introduction of a method for tuning the hyper-parameters of SGD.

SGD and its variants belong to a class of optimization methods known as Stochastic Approximation(SA) algorithms. SGD is the building block of most machine learning algorithms today. SGD methods have become increasingly popular because they use an inexpensive estimator of the gradient to find the optimum of an objective function. Their memory and computational efficiency have made them highly desirable, particularly when large amounts of data or streaming data need to be processed. SGD and its variants are widely used in many fields, including engineering, computer science, applied math, and statistics. Though they are extensively used and with great success in many cases, SGD methods are can be very erratic and are very sensitive to the choice of hyper-parameters. As a result, they require a significant amount of hyper-parameter tuning to achieve the desired results. We propose a method for tuning the hyper-parameters of SGD that employs the double bootstrap method in conjunction with the Simultaneous Perturbation Stochastic Approximation (SPSA) method.

The second project is concerned with optimal experimental designs for precision medicine with multi-component treatments. In recent years, many medical and behavioral interventions have evolved to combine multiple therapeutic options. The use of interventions that consist of multiple components arises in areas such as personalized messaging for colorectal cancer screening. Considering several therapeutic options in a trial can result in a potentially large number of unique treatment combinations at each decision point. Ignoring the overlap among components and viewing each combination as unique in evaluating the treatment combinations is inefficient. It is, therefore, necessary to understand how to allocate these combination treatments optimally to improve patient outcomes. We refer to treatments that consist of a combination of therapeutic options as multi-component treatments. We propose a method for evaluating multi-component treatments that exploits

a classical optimal design strategy to maximize statistical power for analyses that involve comparing multi-component treatments against each other and against a control. The goal is to allocate treatments to maximize power for a specified primary analysis while ensuring enough information is retained to conduct secondary analyses.

The organization of the rest of this manuscript is as follows. The first chapter provides a brief description of the two projects. The second chapter consists of a literature review interspersed with illustrative examples, that consolidates various SGD methods dispersed across the different literatures to better understand them and put them under common notation to make them more accessible to people who haven't seen them before. The third chapter considers the applications of SGD methods in various fields of Statistics. In the fourth chapter, we introduce a method for tuning hyper-parameters of SGD that uses the double bootstrap in conjunction with the Simultaneous Perturbation Stochastic Approximation (SPSA) method. The fifth chapter focuses on optimal experimental designs for precision medicine with multi-component treatments. The final chapter consists of a summary of our findings and a discussion about future work.

Impact of the Choice of Hyper-parameters on Statistical Inference using SGD Estimates and Optimal Experimental Designs for Precision Medicine with Multi-component Treatments

by
Yeng Saanchi

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Statistics

Raleigh, North Carolina
2023

APPROVED BY:

Chair Dr. Leonard Stefanski Co-chair of Advisory Committee	Chair Dr. Eric Laber Co-chair of Advisory Committee
Dr. Marie Davidian	Dr. Dennis Boos

Dr. Srinath Ekkad

DEDICATION

To my God and King, "for He who is mighty has done great things for me; And holy is His name." (Luke 1:49).

BIOGRAPHY

The author was born in Accra, Ghana. She obtained her Bachelor's degree in Economics and Statistics at the University of Ghana and her Masters degree in Applied Statistics at the University of Michigan, Ann Arbor. She will graduate with her Ph.D. in Statistics under the direction of Dr. Eric Laber and Dr. Leonard Stefanski.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisors, Dr. Leonard Stefanski and Dr. Eric Laber, for taking on the task of advising me, and for all their support and encouragement throughout my time as a graduate student.

My sincere thanks to Dr. Marie Davidian, Dr. Dennis Boos, and Dr. Srinath Ekkad for their willingness to serve on my committee and for their constructive feedback.

I would also like to thank Dr. Jacqueline Hughes-Oliver and Dr. Donald Martin for their assistance and advice.

Many thanks to Lanakila Alexander and Alison McCoy for all their help with administrative requirements.

I would also like to thank profoundly the people at JMP LLC., in particular, the members of the DOE and Statistical Development teams, for their encouragement and support.

My heartfelt thanks to my family and friends for their unflinching support and prayers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

The first part of this manuscript, which covers Chapters 2, 3, and 4, is concerned with Stochastic Gradient Descent (SGD) and its variants, some applications in Statistics, and hyper-parameter tuning.

Many modern statistical estimators can be cast as minimizers of a data-dependent objective function. The automated collection and curation of large data sets is now commonplace. Researchers seeking to maximize value from such data are increasingly looking to highly-flexible and data-hungry models in an attempt to identify subtle and complex patterns. Thus, there is a growing need for estimation algorithms that handle data streams of high dimension and high volume. Stochastic Gradient Descent (SGD) is a general class of optimization algorithms that scale efficiently to such data and is therefore proving to be a valuable tool for modern statistical applications. While there is already an extensive literature on SGD spread across the statistics, applied mathematics, computer science, and engineering literatures, an up-to-date introductory review for statistical applications is lacking. By cataloging many of the commonly used SGD methods into a single resource and by focusing on their statistical application, we hope that this review will serve as a gateway to the broader study and application of these methods. We use the logistic regression frame-

work as a running example because we wanted to pick an example that commonly arises in statistics but is sufficiently complicated to illustrate how SGD is used in a non-linear but well-understood problem. We also consider the impact of the choice of hyper-parameters of the SGD method on the resulting parameter estimates and present a way of tuning them.

The second part of this dissertation, found in the fifth chapter, consists of optimal experimental designs for precision medicine with multi-component treatments. Treatment decisions for complex diseases and disorders often involve a combination of multiple therapeutic options. Cancer treatment is one area in which the use of such treatments is increasingly becoming common (Garcia-Aguilar et al. 2016; Sambi et al. 2019; Philippou et al. 2020). For example, a cancer treatment could consist of a combination of chemotherapy, radiation, and drugs. In light of this, there is a need to understand better how to optimally combine treatment components to promote the patient's long-term well-being. We refer to treatments that consist of a combination of therapeutic options as multi-component treatments. We propose a strategy that leverages classical optimal design methods to assign combination treatments to maximize power for primary and secondary analyses of interest. Our proposed method minimizes a composite objective that depends on the c-optimality and D-optimality criteria to optimally allocate treatment combinations to patients given their covariate information.

CHAPTER

2

STOCHASTIC GRADIENT DESCENT AND SOME VARIANTS

2.1 Introduction

Gradient descent (GD), which approximates the minimizer of a differentiable function by following the negative gradient 'downhill' toward a solution, is a foundational idea in optimization. It is conceptually simple and can be highly effective in problems where computing or approximating a gradient can be done efficiently. However, in settings where computing the gradient with high precision is not feasible, e.g., because the data are too large, alternative approaches may be needed. Stochastic Gradient Descent (SGD) uses an inexpensive estimator of the gradient in each iteration of the GD algorithm and is, therefore often suitable for large problems. Furthermore, the use of a noisy approximation to the gradient can, in some cases, have a regularizing effect that leads to better numerical stability. Speed and numerical stability are two of the primary reasons that SGD is widely used to fit deep neural networks (Bottou 1991; Bottou and Cun 2004; Newton et al. 2018). This

popularity, along with promising empirical and theoretical evaluations of its computational efficiency, makes the statistical properties of SGD worth studying. SGD methods, also referred to as Incremental Gradient (IG) methods, belong to the broader family of optimization methods known as Stochastic Approximation (SA) algorithms. To introduce SGD, both from a historical and conceptual point of view, we begin with a brief review of SA. The two algorithms we review, Robbins-Monro (RM) and Keifer-Wolfowitz (KW), form the foundation of the SGD methods reviewed here.

The first SA algorithm was proposed by Robbins and Monro (1951) as a means of estimating the root of an unknown function when only noisy function values were available. Let $f : \mathbb{R} \to \mathbb{R}$ be a continuous but unknown function. For a given value m, suppose that we wish to find the value θ^* such that $f(\theta^*) = m$. Further, suppose that the function f is not directly available, but for any value $\theta \in \mathbb{R}$ we can generate draws of a random variable $Y(\theta)$ such that $\mathbb{E}\{Y(\theta)\} = f(\theta)$. For any given value of θ, one can generate a large number of draws of $Y(\theta)$ to obtain a precise approximation to $f(\theta)$; however, a precise estimator of $f(\theta)$ for values of θ far from θ^* may be of little value. Thus, an efficient algorithm must balance averaging to reduce error with exploring the input space trying to find the root. This balance can be seen in the Robbins and Monro algorithm which starts with an initial guess $\hat{\theta}_0$ and then performs the following iterative updates

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \epsilon_t\{Y_t(\hat{\theta}_t) - m\}, \quad t = 0, 1, 2, ...,$$

where Y_t is a noisy estimate of the function value $f(\theta_t)$ and ϵ_t is a sequence of step sizes satisfying $\epsilon_t > 0$, $\epsilon_t \to 0$, $\sum_t \epsilon_t = \infty$, $\sum_t \epsilon_t^2 < \infty$.

The preceding conditions on $\{\epsilon_t\}_{t\geq 1}$ are ubiquitous in SA. The diminishing step sizes indicate that while larger step sizes are needed at the initial stages to accelerate convergence, as the iterates approach the optimum, smaller steps are required to reduce overshooting. The conditions on the sequence $\{\epsilon_t\}_{t\geq 1}$ provide an implicit averaging of the observations that improves the rate of convergence (Kushner and Yin 2003).

The preceding sufficient conditions give only crude guidance about how one should choose their step sizes in practice. Unfortunately, in many settings, the performance of SA can be highly sensitive to this choice. The construction of general-purpose methods for tuning these step sizes remains an important open question. To provide some intuition for how these sufficient conditions arise in proving convergence of the RM algorithm, we consider a simple illustrative example in which one uses RM-type iterative updates

to estimate the mean of a random variable from a sequence of i.i.d. samples. We do this because, as far as we know, the literature contains convergence proofs that rely on measure-theoretic concepts which can be challenging to get through, and so we thought it would be helpful to use basic statistical concepts and simple bounds to illustrate where the conditions are used to achieve convergence of the procedure.

Suppose we observe a stream of data, $X_1, X_2, \ldots$ that are independent and identically distributed from some distribution $G(\mu, \sigma^2)$ with $\mathbb{E}(X) = \mu$ and $\text{Var}(X) = \sigma^2$. Upon observing the nth data point, the SA update for estimating μ is

$$\hat{\mu}_n = \hat{\mu}_{n-1} + \epsilon_n (X_n - \hat{\mu}_{n-1}).$$

It can be seen that choosing $\epsilon_n = 1/n$ leads to the sample mean and that, in this case, the key conditions on the step size sequence are satisfied ($1/n \to 0$, $\sum 1/n \to \infty$, and $\sum 1/n^2 < \infty$).

More generally, if we consider a non-negative sequence $0 < \epsilon_t \leq 1$, then an RM iterative estimator of μ with step sizes $\{\epsilon_t\}_{t \geq 1}$ is given by

$$\begin{aligned}
\hat{\mu}_1 &= \epsilon_1 X_1 \\
\hat{\mu}_2 &= \hat{\mu}_1 + \epsilon_2 (X_2 - \hat{\mu}_1) = (1 - \epsilon_2)\hat{\mu}_1 + \epsilon_2 X_2 \\
&\vdots \\
\hat{\mu}_n &= (1 - \epsilon_n)\hat{\mu}_{n-1} + \epsilon_n X_n.
\end{aligned}$$

It can be seen that if $\widehat{\mu}_n$ is converging in probability to a constant, then ϵ_n must be converging to zero. Furthermore, it can be seen that

$$\hat{\mu}_n = \epsilon_n X_n + \sum_{t=1}^{n-1} \Big[\epsilon_t \prod_{j=t+1}^{n} (1 - \epsilon_j) \Big] X_t.$$

Thus, the expectation of $\hat{\mu}_n$ is

$$\begin{aligned}
\mathbb{E}(\hat{\mu}_n) &= \epsilon_n \mathbb{E}(X_n) + \sum_{t=1}^{n-1}\Big[\epsilon_t \prod_{j=t+1}^{n}(1-\epsilon_j)\Big]\mathbb{E}(X_t) \\
&= \epsilon_n \mu + \mu \sum_{t=1}^{n-1}\Big[\epsilon_t \prod_{j=t+1}^{n}(1-\epsilon_j)\Big] \\
&= \epsilon_n \mu + \mu \sum_{t=1}^{n-1}\Big[\prod_{j=t+1}^{n}(1-\epsilon_j)\Big] - \mu \sum_{t=1}^{n-1}\Big[\prod_{j=t}^{n}(1-\epsilon_j)\Big] \\
&= \epsilon_n \mu + \mu\Big[(1-\epsilon_n) - \prod_{t=1}^{n}(1-\epsilon_t)\Big] \\
&= \mu\Big[1 - \prod_{t=1}^{n}(1-\epsilon_t)\Big].
\end{aligned}$$

Applying the inequality $e^{-\frac{u}{1-u}} < 1-u < e^{-u}$ yields

$$\mu\Big[1-e^{-\sum_{t=1}^{n}\frac{\epsilon_t}{1-\epsilon_t}}\Big] \leq \mu\Big[1-\prod_{t=1}^{n}(1-\epsilon_t)\Big] \leq \mu\Big[1-e^{-\sum_{t=1}^{n}\epsilon_t}\Big].$$

Note that, if $\sum_{t=1}^{n}\epsilon_t \to \infty$, then $\sum_{t=1}^{n}\frac{\epsilon_t}{1-\epsilon_t} \to \infty$ as $n \to \infty$ for $\epsilon_t \in [0,1]$ and thus, $\mathbb{E}(\hat{\mu}_n) \to \mu$.

We now consider the variance of $\hat{\mu}_n$. We will show that $\sum_{t=1}^{n}\epsilon_t \to \infty$ and $\sum_{t=1}^{n}\epsilon_t^2 < \infty$ as $n \to \infty$ are sufficient for $\mathrm{Var}(\hat{\mu}_n)$ to go to zero. The variance is

$$\begin{aligned}
\mathrm{Var}(\hat{\mu}_n) &= \sum_{t=1}^{n}\epsilon_t^2 \prod_{j=t+1}^{n}(1-\epsilon_j)^2 \mathrm{Var}(X_t) \\
&= \sigma^2 \sum_{t=1}^{n}\epsilon_t^2 \prod_{j=t+1}^{n}(1-\epsilon_j)^2.
\end{aligned}$$

Suppose $\sum_{t=1}^{\infty}\epsilon_t^2 < \infty$, $\forall\, N \leq n$, it follows that

$$\begin{aligned}
\mathrm{Var}(\hat{\mu}_n) &= \sigma^2\left\{\sum_{t=1}^{N-1}\epsilon_t^2 \prod_{j=t+1}^{n}(1-\epsilon_j)^2 + \sum_{t=N}^{n}\epsilon_t^2 \prod_{j=t+1}^{n}(1-\epsilon_j)^2\right\} \\
&\leq \sigma^2\left\{\sum_{t=1}^{N-1}\epsilon_t^2 e^{-2\sum_{j=t+1}^{n}\epsilon_j} + \sum_{t=N}^{n}\epsilon_t^2 \prod_{j=t+1}^{n}(1-\epsilon_j)^2\right\},
\end{aligned}$$

where we have used $1-u \leq e^{-u}$. Taking the limit as $n \to \infty$ of the above expression, we have

$$\lim_{n\to\infty} \text{Var}(\hat{\mu}_n) \leq \sigma^2 \left\{ \sum_{t=1}^{N-1} \epsilon_t^2 e^{-2\sum_{j=t+1}^{\infty} \epsilon_j} + \sum_{t=N}^{\infty} \epsilon_t^2 \prod_{j=t+1}^{\infty} (1-\epsilon_j)^2 \right\}.$$

Since $\sum_{j=t+1}^{\infty} \epsilon_j = \infty$ for each $t \in \{1,..,N-1\}$, each summand in the first term is 0, and so we have

$$\lim_{n\to\infty} \text{Var}(\hat{\mu}_n) \leq \sigma^2 \sum_{t=N}^{\infty} \epsilon_t^2,$$

which can be made arbitrarily small by taking N large. Thus, it follows that $\hat{\mu}_n$ converges in probability to μ. As this toy example illustrates, the requirement that the step sizes converge to zero and sum to infinity allows the iterates to move to any point in the parameter space and thus ensure convergence in mean, whereas the additional requirement that the step sizes be square summable is used to ensure the iterates converge.

A natural application of the RM root-finding algorithm is to identify extrema of a continuously differentiable function by finding roots of its gradient. However, if the function is unknown, so too is its gradient. Kiefer and Wolfowitz (1952) developed an SA method for estimating the maximum of a function, where the gradient is approximated using finite differences. Their algorithm, now known as the Kiefer-Wolfowitz (KW) algorithm, is as follows. Suppose we want to optimize a function f parameterized by $\theta \in \mathbb{R}^p$, whose form is unknown. However, for any value of θ we can generate draws $Y(\theta)$ such that $\mathbb{E}\{Y(\theta)\} = f(\theta)$.

Setting $\hat{\theta}_0$ to an initial guess, the algorithm computes iterates using the update

$$\hat{\theta}_{t+1,j} = \hat{\theta}_{t,j} - \epsilon_t \frac{Y_{t,1,j}(\hat{\theta}_t + c_t e_j) - Y_{t,2,j}(\theta_t - c_t e_j)}{2c_t},$$

where c_t is a positive non-random sequence, e_j is the standard unit vector in the direction of the jth coordinate, and $Y_{t,1,j}$ and $Y_{t,2,j}$ are independent noisy estimates of the function obtained at perturbed parameter values $\hat{\theta}_t + c_t e_j$ and $\hat{\theta}_t - c_t e_j$ respectively. The sequences ϵ_t and c_t satisfy the following conditions:
$\epsilon_t \to 0$, $c_t \to 0$, $\sum_t \epsilon_t = \infty$, $\sum_t \epsilon_t c_t < \infty$ and $\sum_t \epsilon_t^2 c_t^{-2} < \infty$.

Methods that approximate the gradient are useful even in settings where the gradient could, in theory, be computed analytically but is either computationally expensive or unstable. In some settings, using an approximate gradient can be seen as bias-variance trade-off in that using a noisy surrogate for the gradient can have a regularizing effect, thereby increasing bias but reducing variance (Bishop 1995; White et al. 2017). Such regularization is particularly appealing when iterative methods are applied with machine learning models. When applied in p-dimensional parameter settings, the computational cost of the KW algorithm may become prohibitive when p is large. To mitigate this cost, variants of the KW algorithm which require fewer noisy function values are available in the form of Spall's method (Spall 1998) and one-sided versions of the KW algorithm (Chen et al. 1999), which require $p+1$ function measurements instead of the standard $2p$.

SGD methods are widely used in both online and offline learning. In machine learning, online learning is a method where data are observed in real-time and processed sequentially. For example, data might comprise real-time measurements collected by sensors in transport vehicles (Chowdhury et al. 2017). Under this mode of learning, the data are typically discarded after processing. Offline or batch learning, on the other hand, is an approach where all the data are available and are processed at the same time. In offline learning, the dataset is static and does not change with time. In settings with large data volume, SGD is advantageous as observations can be processed and then discarded, reducing memory demands. Thus, SGD can be used to process data exhaust[1] in real-time from a retail website with large traffic volumes or to process data from a medical database that's too large to read into memory all at once. Online learning can present additional challenges if there is temporal dependence and/or the sampling design is driven by a model being estimated using SGD.

2.2 Setup and Notation

Suppose that the available data are $Z_1, \ldots, Z_n \in \mathscr{Z} \subseteq \mathbb{R}^p$, which comprise n i.i.d. draws from a fixed but unknown distribution P. The goal is to use this sample to estimate parameter vector $\theta^* \in \Theta \subseteq \mathbb{R}^d$ which satisfies:

$$\theta^* = \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} \mathbb{E}_P[f(\theta; Z)] = \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} \int_z f(\theta; z) \mathrm{d}P(z),$$

[1] Data exhaust or exhaust data refers to the trail of data resulting from online activities.

where $f : \Theta \times \mathscr{Z} \to \mathbb{R}$ is a loss function that is differentiable in θ for almost every $z \in \mathscr{Z}$.

As P is generally unknown, an estimator of θ^*, denoted by $\hat{\theta}$, is obtained by minimizing the empirical risk function. The estimator is given by

$$\hat{\theta} = \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} \ell(\theta), \quad \ell(\theta) = \frac{1}{n} \sum_{i=1}^{n} f(\theta; Z_i).$$

As a running illustrative example, we use a generalized linear model (GLM) where the data are of the form $Z = (X, Y)$. Consider a GLM such that $\mathbb{E}(Y|X = \mathbf{x}) = g^{-1}(\mathbf{x}^T\theta)$ with distribution

$$y|\mathbf{x} \sim \exp\Big\{\frac{y\mathbf{x}^T\theta - b(\mathbf{x}^T\theta)}{a(\psi)} + c(y, \psi)\Big\},$$

where y is the output variable, $\mathbf{x}$ is the vector of inputs, θ is the parameter vector of the model, g is the link function, ψ is the dispersion parameter, and $a(\cdot)$, $b(\cdot)$, $c(\cdot, \cdot)$ are real-valued functions. The negative log-likelihood for θ is proportional to

$$\ell(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \Big\{ y_i \mathbf{x}_i^T\theta - b(\mathbf{x}_i^T\theta)\Big\}.$$

It will be useful in what follows to recall the following properties of GLMs.

- $\mathbb{E}(y|\mathbf{x}) = h(\mathbf{x}^T\theta) = \nabla_\theta b(\mathbf{x}^T\theta)$,
- $\nabla_\theta \ell(\theta) = \{h(\mathbf{x}^T\theta) - y\}\mathbf{x}$,
- $\nabla^2_\theta \ell(\theta) = \nabla_\theta h(\mathbf{x}^T\theta)\mathbf{x}$,

where $h(\cdot) = g^{-1}(\cdot)$. In the specific case of logistic regression, the loss function to be optimized is

$$\ell(\theta) = \frac{1}{n} \sum_{i=1}^{n} \Big\{\log(1 + e^{\mathbf{x}_i^T\theta}) - y_i \mathbf{x}_i^T\theta\Big\}.$$

The gradient and Hessian under the logistic regression model are respectively

- $\nabla_\theta \ell(\theta) = \sum_{i=1}^{n} \big\{h\big(\mathbf{x}_i^T\theta\big) - y_i\big\}\mathbf{x}_i$,
- $\nabla^2_\theta \ell(\theta) = \sum_{i=1}^{n} h\big(\mathbf{x}_i^T\theta\big)\big\{1 - h\big(\mathbf{x}_i^T\theta\big)\big\}\mathbf{x}_i\mathbf{x}_i^T$,

where $h(u) = 1/(1+e^{-u})$.

It is standard in software designed to fit GLMs to use gradient descent or the Newton-Raphson method to construct $\hat{\theta}$ but these methods can be cumbersome in problems with high-dimension, high-volume, or both. In such cases, SGD and its variants provide a scalable and practicable alternative due to their computational and memory efficiency.

2.3 Standard (Vanilla) SGD

When applied in the offline learning setting, SGD methods either cycle through the set of data sequentially or choose an observation at random, after which the observation is processed and then an update made. In the online setting, a new data point is observed at each iteration, processed to make an update, and then discarded. In general, there are three popular ways of selecting observations when implementing SGD on static datasets (Bottou 2009).

- *Random Sampling*: Data points are drawn uniformly with replacement from the dataset at each iteration.
- *Cycling*: Data points are chosen sequentially from a randomly permuted dataset so that in each epoch observations are processed in the same order. An epoch typically refers to a complete pass through a static dataset.
- *Random Reshuffling*: Multiple passes are made through the dataset, but before each pass, the dataset is first shuffled, after which the data points are selected sequentially to be processed.

We can operationalize each of these modes of selection using the function $S : \mathbb{N} \to \mathbb{N}$ which determines the index of the observation processed at each iteration so that the data are processed in a stream as follows: $Z_{S(1)}, Z_{S(2)}, Z_{S(3)}, \ldots$. This allows for multiple passes through a fixed dataset as well as data accumulating in real-time, or a mixture of both.

The standard SGD update at the tth iteration is

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \epsilon_t \nabla_\theta f\left(\hat{\theta}_t; Z_{S(t)}\right),$$

where ϵ_t is the stepsize at iteration t, $\hat{\theta}_t$ denotes the parameter estimate at iteration t, $Z_{S(t)}$ denotes the data point being processed at iteration t and $\nabla_\theta f(\hat{\theta}_t; Z_{S(t)})$ denotes the gradient of $f(\theta; Z)$ with respect to θ evaluated at $\hat{\theta}_t$ and $Z_{S(t)}$. If expectation and differentiation are interchangeable, $\mathbb{E}\big[\partial/\partial\theta\{f(\theta; Z_{S(t)})\}|_{\theta=\hat{\theta}_t}\big] = \partial/\partial\theta\{\ell(\theta)\}|_{\theta=\hat{\theta}_t}$, where the expectation is taken with respect to the observation. In the context of logistic regression, the standard SGD update is

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \epsilon_t \Big\{ h\big(\mathbf{x}_{S(t)}^T \hat{\theta}_t\big) - y_{S(t)} \Big\} \mathbf{x}_{S(t)}.$$

Empirical studies have shown that SGD methods often make faster progress per floating point operation (FLOP) early in the optimization process than gradient descent. The heuristic explanation often provided for this phenomenon is that SGD uses a noisy gradient that is inexpensive to compute and is 'good enough' in that the averaging effects implicit in parameter updates will ensure the algorithm heads in the right direction. However, SGD tends to slow down considerably during the latter stages of convergence when the estimate is approaching an optimal solution because of the diminishing step size (which entails a considerable amount of fine-tuning to determine the appropriate rate of diminution) (Bertsekas 1997a; Friedlander and Schmidt 2012).

Figure 2.1 illustrates the performance of GD and SGD by comparing the number of FLOPs per iteration under the logistic regression setup. The y-axis shows the square of the Euclidean distance between each parameter iterate and the desired optimum denoted by $||\hat{\theta}_t - \hat{\theta}^*||^2$, where $\hat{\theta}^*$ is the maximum likelihood estimate (MLE) obtained using the Newton-Raphson method. With respect to FLOPs, one iteration of SGD and GD costs $\mathcal{O}(p)$ and $\mathcal{O}(np)$ respectively. Thus, for example, a 1000 iterations of SGD are far less computationally expensive than a 1000 iterations of GD. It can be observed in Figure 1 that during the early stages of optimization, SGD outperforms GD in all three cases of differing p presented, but as the algorithms progress, GD surpasses SGD.

When a small constant step size is used with SGD methods, there is an oscillation within each data cycle, as shown in Luo (1991). In contrast, gradient descent achieves a steady linear convergence rate for convex functions even with a constant step size, if the gradient of the objective is Lipschitz continuous (Polyak 1987). Wilson and Martinez (2003) use empirical results on classification and speech recognition tasks to argue that training neural networks using SGD methods with a constant learning rate is more efficient than using GD.

Empirical studies suggest that the cycling and random reshuffling methods converge

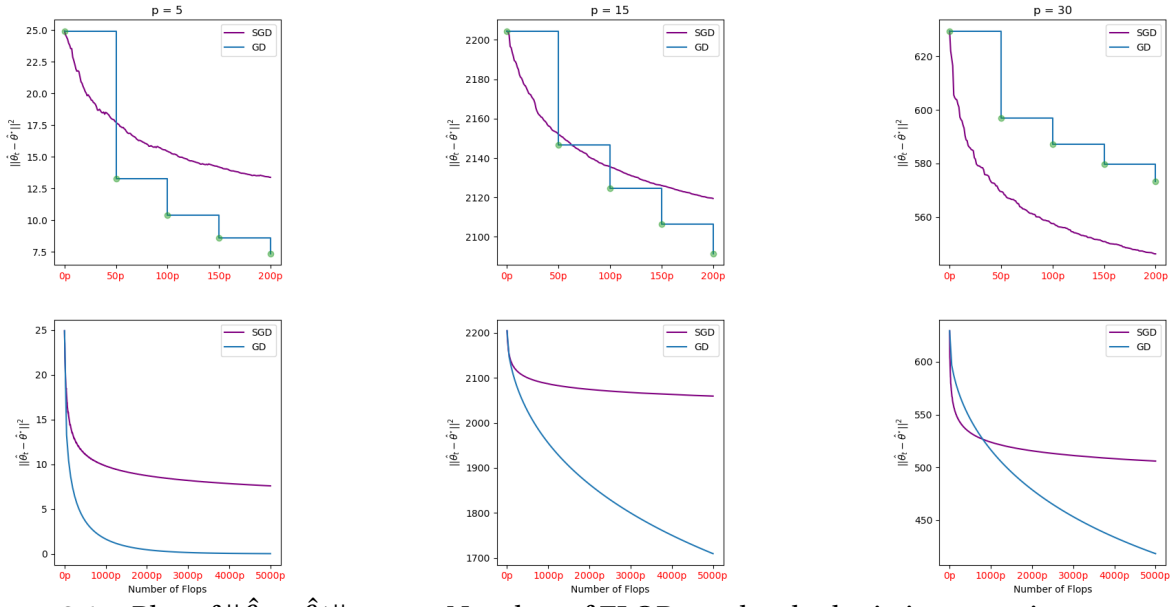

Figure 2.1: Plot of $||\hat{\theta}_t - \hat{\theta}^*||$ versus Number of FLOPs under the logistic regression set-up. The number of observations, n, is fixed at 50 in all three cases. The top panel shows the first few iterations for a clearer picture of the progress made by SGD and GD in the initial stages.

more rapidly than the random sampling method because each observation is processed once in each epoch thereby covering the entire dataset more evenly (Bottou 2009). The convergence rate of SGD with diminishing step sizes is $\mathcal{O}(1/\sqrt{T})$ for general convex functions (Zhang 2004; Bottou et al. 2018) and $\mathcal{O}(1/T)$ for strongly convex functions (Hazan et al. 2007; Nemirovski et al. 2009; Rakhlin et al. 2011; Nesterov 2013), where T denotes the total number of iterations. While Vanilla SGD is simple to implement and has low memory requirements, it is plagued by oscillations about the optimum as iterates approach a solution. This has given rise to several modifications of the standard algorithm in an attempt to alleviate this problem. We present some of these methods in the ensuing section.

2.4 Variants of SGD

2.4.1 Mini-Batch SGD (MB-SGD)

Despite its relatively low computational cost per iteration, Vanilla SGD is characterized by noisy gradient estimates that reduce the rate of convergence. Whereas GD makes updates

using a full gradient, i.e. the gradient of the loss function computed using the complete dataset, SGD methods compute the gradient (referred to as a stochastic gradient) using a randomly sampled data point. A natural middle ground between GD and SGD is to compute gradient updates with a subset of the data somewhere between a single observation and the entire data set. MB-SGD uses a random selection of multiple observations referred to as a mini-batch, instead of a single observation, to compute the gradient. In this context, since multiple data points are processed at each iteration, the data stream function is redefined as $S : \mathbb{N} \rightarrow \mathbb{N}^k$, where $k > 1$. Thus, MB-SGD combines the desirable properties of GD and SGD and serves as a compromise between GD and standard SGD. Using mini-batches reduces the noise in the gradient associated with the standard SGD update, while reducing the high computational cost of GD. It has been shown in Dekel et al. (2012) that the rate of convergence of mini-batch SGD is $\mathcal{O}(1/T + 1/\sqrt{b\,T})$, where b is the size of the mini-batch and $1 \leq b \leq T$. Dekel et al. (2012) also show that convergence of MB-SGD in both the serial and distributed settings is dependent on the variance of the stochastic gradient for smooth convex loss functions.

Figure 2.2 compares MB-SGD with different batch sizes to SGD and GD using an L2-regularized logistic regression loss function. The design matrix is generated from a multivariate normal under an AR1 covariance matrix with a correlation coefficient of 0.8. The true value of the parameter vector, θ^*, used in the data generation process is an array of evenly spaced values over the interval -2 to 2 for both $p = 10$ and $p = 100$. The performance metric used in the comparison is $||\hat{\theta}_t - \hat{\theta}^*||^2$, where $\hat{\theta}^*$ is the MLE. It can be observed that in both of the cases presented GD performs best and the larger mini-batch sizes tend to outperform the smaller mini-batch sizes. This can be attributed to the reduction in the noisiness of the stochastic gradient as more observations are used in the computation of the gradient. However, it is worth noting that in the $p = 10$ case, using a mini-batch size of 50 versus a mini-batch size of 10 does not result in an appreciable reduction in the distance between the iterates and the MLE. Similarly, there does not appear to be much difference between a mini-batch size of 100 and 200 in the $p = 100$ case. In both cases, the choice of a larger mini-batch results in a relatively small improvement at the expense of a larger computational cost. Empirical studies have shown that larger mini-batch sizes do not always do better than smaller mini-batch sizes (LeCun et al. 2012; Dekel et al. 2012; Keskar et al. 2016). This suggests that the right choice of mini-batch size is largely problem-dependent and is a hyper-parameter that benefits from tuning.

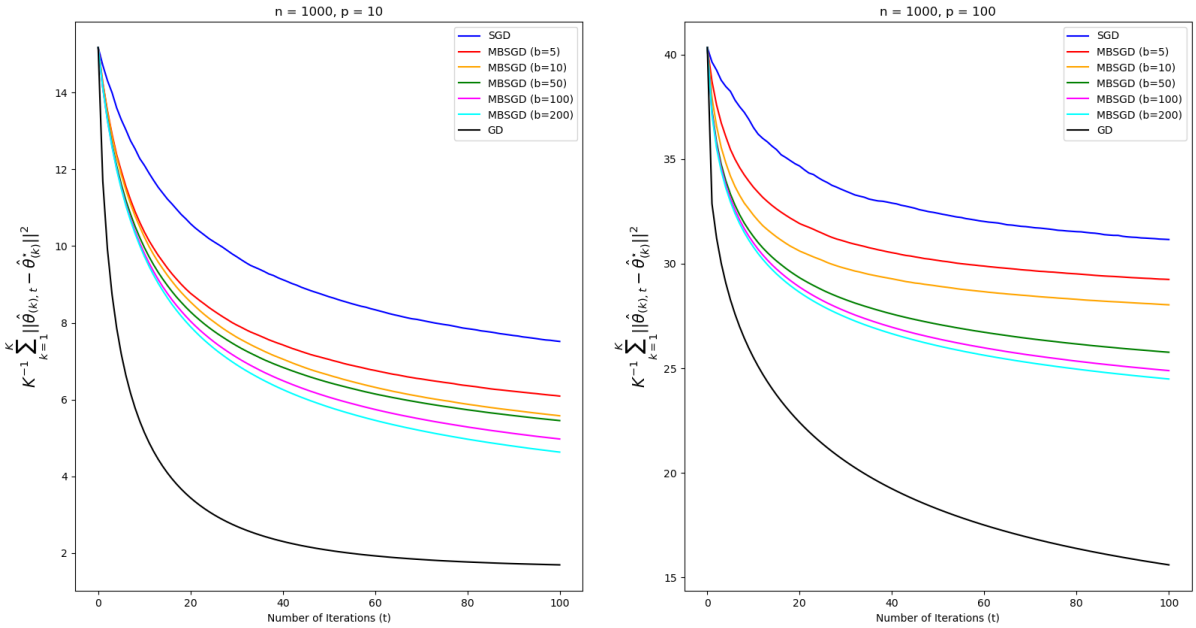

Figure 2.2: Plot of $K^{-1}\sum_{k=1}^{K}||\hat{\theta}_{(k),t}-\hat{\theta}^{*}_{(k)}||$ versus Number of Iterations for SGD, MB-SGD with varying mini-batch sizes, and GD using an L2-regularized logistic loss function. k and t index the dataset and iteration number, respectively. The number of datasets, K, is 200.

2.4.2 SGD with Momentum

Standard Momentum

Although SGD has proved generally successful in optimizing a wide array of functions, the method is characterized by slow progress in regions where the curvature of the objective function is steeper in some dimensions than others. The introduction of *momentum* into the standard SGD algorithm, attributed to Polyak (1964), aims to accelerate progress in such situations. The method incorporates an additional term called momentum, denoted by v_t, that captures the direction and speed of the parameter vector. The momentum term is a weighted average of previous gradients and decays exponentially with time. The extent to which past gradients influence the direction of the parameter vector depends on the magnitude of a velocity parameter, denoted by β, which governs the influence of the momentum on each update. Initializing $\hat{\theta}_0$ and $\hat{v}_0$ arbitrarily to some finite values, SGD updates with momentum are of the form:

$$\hat{v}_{t+1} = \beta\,\hat{v}_t - \epsilon_t \nabla_\theta f\big(\hat{\theta}_t; Z_{S(t)}\big),$$
$$\hat{\theta}_{t+1} = \hat{\theta}_t + \hat{v}_{t+1}.$$

The update under logistic regression is thus

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \beta\,\hat{v}_t - \epsilon_t \Big\{ h\big(\mathbf{x}_{S(t)}^T \hat{\theta}_t\big) - y_{S(t)} \Big\} \mathbf{x}_{S(t)}.$$

The application of momentum helps to dampen the oscillations associated with Vanilla SGD by assigning larger updates to dimensions with consistent gradients while reducing the size of updates corresponding to dimensions whose gradients point in opposite directions. Typical values of β are 0.5, 0.9, and 0.99, or β can be adaptively increased over time (Goodfellow et al. 2016).

Figure 2.3 compares Standard SGD to SGD with Momentum by considering $||\hat{\theta} - \hat{\theta}^*||^2$ under the same setup used in Figure 2. It can be observed that Vanilla SGD performs better than SGD with Momentum in the initial stages of optimization. This can be explained by the tendency of SGD with Momentum to take too-large steps at times, which may result in large deviations from the trajectory toward the desired goal. However, SGD with Momentum swiftly closes the gap and surpasses Standard SGD by working its way back

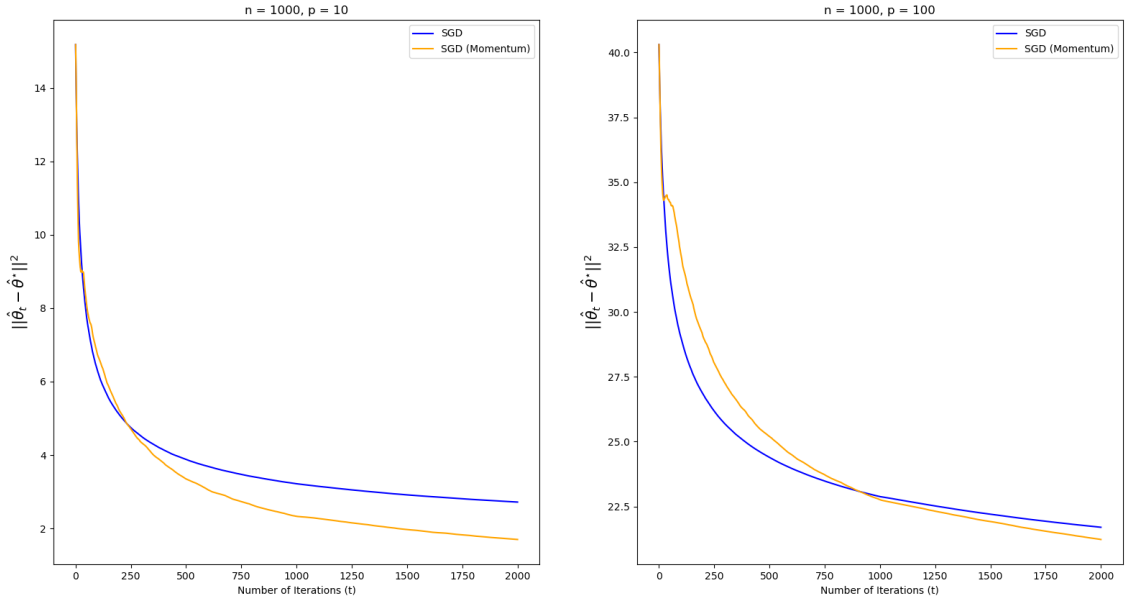

Figure 2.3: Plot of $||\hat{\theta}_t - \hat{\theta}^*||$ versus Number of Iterations for SGD and SGD with Momentum using an L2-regularized logistic loss function. Results are averaged over 200 datasets.

toward the optimum. Thus, on the whole, the application of momentum to SGD results in faster progress toward the optimum compared to SGD without momentum.

Nesterov Momentum

Much as the application of momentum to SGD helps to accelerate progress in the relevant direction, in some cases, too-large steps may result in overshooting the optimum. The incorporation of Nesterov momentum into SGD, introduced by Sutskevar et al. (2013), allows a correction of the large steps that are a feature of the standard momentum algorithm. This idea was inspired by Nesterov's Accelerated Gradient (Nesterov 1983). When standard momentum results in a sub-optimal location of the parameter vector, Nesterov momentum attempts a correction to reduce the size of the step. After setting $\hat{\theta}_0$ and $\hat{v}_0$ to arbitrary finite initial values, the SGD updates with Nesterov momentum are given by

$$\hat{v}_{t+1} = \beta\,\hat{v}_t - \epsilon_t \nabla_\theta f\left(\hat{\theta}_t + \beta\,\hat{v}_t; Z_{S(t)}\right),$$
$$\hat{\theta}_{t+1} = \hat{\theta}_t + \hat{v}_{t+1}.$$

The parameter update for the logistic regression case is

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \beta\,\hat{v}_t - \epsilon_t\Big[h\big\{\mathbf{x}_{S(t)}^T(\hat{\theta} + \beta\,\hat{v}_t)\big\} - y_{S(t)}\Big]\mathbf{x}_{S(t)}.$$

The difference between Nesterov momentum and standard momentum lies in the way the gradient is evaluated. Whereas standard momentum evaluates the gradient at the current parameter value before applying the velocity, Nesterov momentum evaluates the gradient after the current velocity has been applied to the parameter vector as a way of anticipating the approximate future location of the parameter vector. Thus, Nesterov momentum adds an element of foresight that standard momentum lacks.

Figures 2.4 and 2.5 compare SGD with Momentum and SGD with Nesterov Momentum by considering the trajectories of individual components of the parameter vector as they journey from their starting points toward the optimum. The results are obtained under an L2-regularized logistic regression set-up with design matrices generated from a multivariate normal under two different covariance structures, specifically an AR1 matrix and an equi-correlated matrix (a matrix with 1 on the diagonals and equal correlation everywhere else). The red star represents the true parameter value denoted by θ^*, and the green star represents the MLE using the Newton-Raphson method, denoted by $\hat{\theta}^*$. The parameter vector, θ^*, is an array of evenly spaced values over the interval -2 to 2. The y-axis of each sub-plot is rescaled so that the lower and upper limits depict $\pm k$ times the standard error of the parameter component, where k is a positive constant chosen such that the full trajectory of the largest parameter estimate is enclosed within the specified limits. The green shaded regions represent the MLE of the specified parameter component plus or minus 3 times its standard error. It can be observed that, overall, SGD with standard Momentum takes larger steps and tends to overshoot the optimum by a larger margin than SGD with Nesterov Momentum. Consequently, SGD with standard Momentum has a farther distance to travel than its Nesterov counterpart in backtracking toward the desired goal. It is worth noting that the choice of the velocity parameter is crucial to the performance of SGD with either standard or Nesterov momentum because a relatively small velocity parameter results in slow progress toward the optimum, while too-large a velocity results in greatly overshooting the mark.

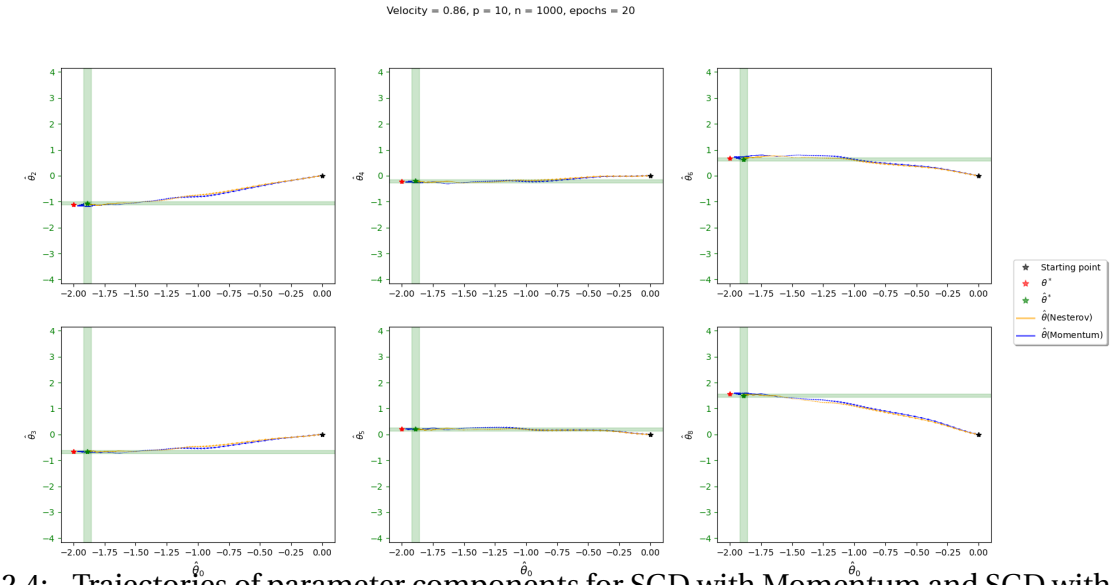

Figure 2.4: Trajectories of parameter components for SGD with Momentum and SGD with Nesterov Momentum under an L2-regularized logistic loss function and a design matrix with an AR1 covariance structure ($\hat{\theta}_0 = \mathbf{0}$). Results depict the parameter trajectories averaged over 200 datasets.

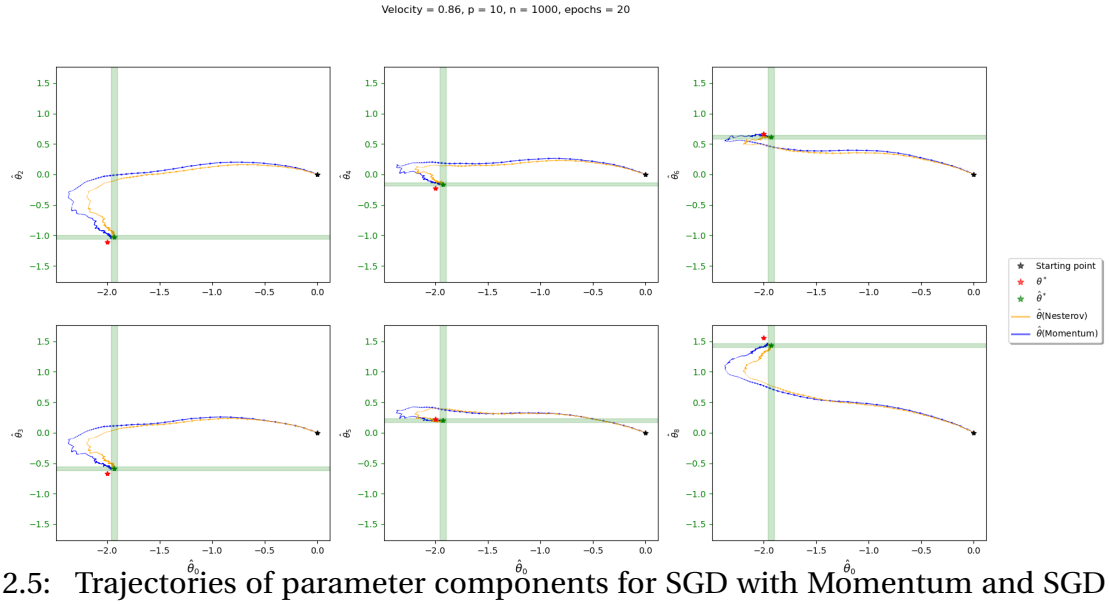

Figure 2.5: Trajectories of parameter components for SGD with Momentum and SGD with Nesterov Momentum under an L2-regularized logistic loss function and a design matrix with an equi-correlated covariance structure ($\hat{\theta}_0 = \mathbf{0}$). Results depict the parameter trajectories averaged over 200 datasets.

2.4.3 Iterate Averaging Methods

Averaged Stochastic Gradient Descent - ASGD

ASGD, also referred to as Polyak-Ruppert averaging, involves using the average of all past iterates as the final estimate of the desired parameter. The idea of averaging the iterates was developed independently by Ruppert (1988) and Polyak (1990). The former proved asymptotic normality in the one-dimensional case, while the latter showed mean square convergence of multi-dimensional estimates for non-linear functions. Polyak and Juditsky (1992) established asymptotic normality of multi-dimensional estimates for non-linear functions. Given the standard SGD update

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \epsilon_t \nabla_\theta f\left(\hat{\theta}_t; Z_{S(t)}\right),$$

the final estimate of θ at the end of T iterations is taken as the average of all the iterates and is given by

$$\bar{\hat{\theta}}_T = \frac{1}{T}\sum_{t=1}^{T} \hat{\theta}_t,$$

which can be computed recursively as

$$\bar{\hat{\theta}}_t = \left(1 - \frac{1}{t}\right)\bar{\hat{\theta}}_{t-1} + \frac{1}{t}\hat{\theta}_t.$$

Figure 2.6, obtained under the same setup as Figure 2, shows a comparison of Standard SGD with ASGD with respect to $||\hat{\theta}_t - \hat{\theta}^*||$. It can be observed that in both the $p = 10$ and $p = 100$ cases, although SGD appears to have a slight advantage over ASGD in the initial stages of the optimization process, ASGD quickly closes the gap and becomes comparable to SGD.

For general strongly convex functions in the online learning setting, the convergence rate of ASGD is $\mathcal{O}(\log(T)/T)$ (Hazan et al. 2007). It is worth pointing out that the asymptotic convergence rate of ASGD is not necessarily better than that of SGD simply because of the practical benefit of averaging. It may be worse in some cases (Kushner and Yin 2003). Polyak and Juditsky (Polyak 1990; Polyak and Juditsky 1992) show that the asymptotic rate of convergence of ASGD is an improvement on that of the standard SGD method if the sequence ϵ_t goes to zero more slowly than $\mathcal{O}(1/t)$, for example, $\epsilon_t = \mathcal{O}(1/t^\alpha)$, $\quad 0.5 < \alpha < 1$.

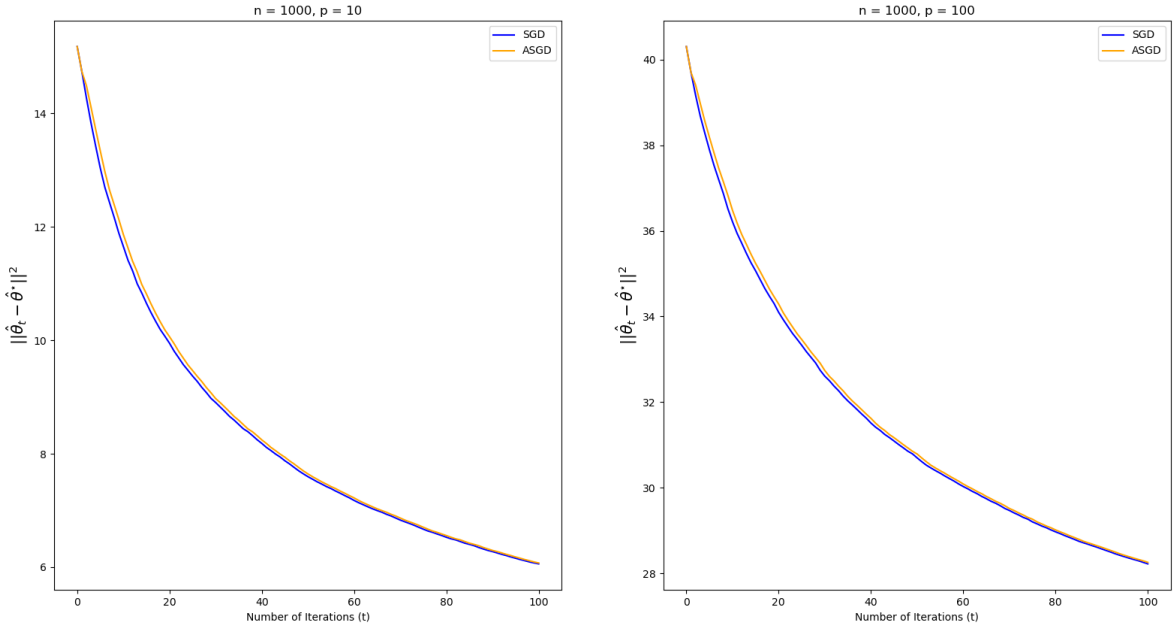

Figure 2.6: Plot of $||\hat{\theta}_t - \hat{\theta}^*||$ versus Number of Iterations for SGD and ASGD using an L2-regularized logistic loss function. Results are averaged over 200 datasets.

Alpha-suffix Averaging

Rakhlin et al. (2011) propose a slight modification to the ASGD method that results in a convergence rate of $\mathcal{O}(1/T)$ for strongly convex but non-smooth loss functions. The authors advocate averaging αT of the iterates to obtain the final estimate of the parameter vector, where α is between 0 and 1, since iterates at the early stages are usually far from the optimum. They refer to their method as α-suffix averaging. The method uses as the final parameter estimate after T iterations,

$$\bar{\hat{\theta}}_T^{(\alpha)} = \frac{\hat{\theta}_{(1-\alpha)T+1} + \ldots + \hat{\theta}_T}{\alpha T},$$

for some $\alpha \in (0,1)$. A possible limitation of this algorithm is that if all the iterates cannot be stored in memory, one needs to know the stopping time T in advance (which in practice is rarely known beforehand) to know at which iterate to begin computing the average. The authors suggest that one way to circumvent this issue is by exponentially increasing the number of iterations in each epoch and then retaining only the average of the current

epoch. The authors also show that when the function is smooth and strongly convex, both standard SGD and ASGD attain a convergence rate of $\mathcal{O}(1/T)$.

Polynomial-decay Averaging

Shamir and Zhang (2013) introduce an averaging variant of SGD that can be computed recursively and attains an optimal convergence rate of $\mathcal{O}(1/T)$ for strongly convex functions and $\mathcal{O}(1/\sqrt{T})$ for general convex functions. The estimate of θ at time t per this algorithm is given by

$$\bar{\theta}_t^{(\eta)} = \left(1 - \frac{\eta+1}{\eta+t}\right)\bar{\theta}_{t-1}^{(\eta)} + \frac{\eta+1}{\eta+t}\hat{\theta}_t,$$

where $\eta \geq 0$ parameterizes the algorithm such that $\eta > 0$ assigns smaller weight to iterates computed at the early stages of the algorithm. A value of $\eta = 0$ yields the standard Polyak-Ruppert averaging method. The authors suggest a value of $\eta = 3$. They also show that without making smoothness assumptions, the last iterate of standard SGD has a convergence rate of $\mathcal{O}(\log(T)/\sqrt{T})$ and $\mathcal{O}(\log(T)/T)$ for convex and strongly convex functions respectively. The authors carry out experiments on some binary classification problems that show that while the proposed method outperforms standard ASGD, its performance is comparable to that of the α-suffix averaging method for $\alpha = 0.5$.

2.4.4 Gradient Averaging Methods

These methods are limited to the offline learning setting where the dataset is fixed and consists of n observations.

Incremental Aggregated Gradient - IAG

SGD methods that use a constant stepsize tend to suffer from oscillations as the iterates approach a solution while those that use a diminishing stepsize sequence require the tuning of hyperparameters. Many hybrid methods that combine SGD and GD also require tuning of the hyperparameter that regulates the rate at which the SGD method evolves into a full gradient method. In an attempt to mitigate the aforementioned difficulties, Blatt et al. (2007) propose IAG, a method that evaluates a single stochastic gradient at each iteration but incorporates the average of n previously computed gradients to perform updates, where n is the number of observations in the dataset. The method uses a constant step size

denoted by ϵ. IAG cycles through the data points in selecting an observation to compute the stochastic gradient at each iteration. The algorithm operates in the following way:

- **Step 1**: Initialize $\hat{\theta}_1, ..., \hat{\theta}_n$ arbitrarily.
- **Step 2**: Compute an aggregated gradient given by $\hat{d}_n = \sum_{i=1}^{n} \nabla_\theta f(\hat{\theta}_i; Z_i)$.
- **Step 3**: For $t \geq n$,

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \epsilon \frac{1}{n} \hat{d}_t,$$
$$\hat{d}_{t+1} = \hat{d}_t - \nabla_\theta f\left(\hat{\theta}_{t+1-n}; Z_{S(t+1)_n}\right) + \nabla_\theta f\left(\hat{\theta}_{t+1}; Z_{S(t+1)_n}\right),$$

where $S(t)_n$ denotes t modulo n. $\hat{d}_t$ is scaled by $1/n$ to make it comparable to the magnitude of the descent direction of standard SGD. The update for the auxiliary parameter $\hat{d}$ for $t \geq n$ under logistic regression (after obtaining $\hat{d}_n$) is

$$\hat{d}_{t+1} = \hat{d}_t - \left\{ h\left(\mathbf{x}_{S(t+1)_n}^T \hat{\theta}_{t+1-n}\right) - h\left(\mathbf{x}_{S(t+1)_n}^T \hat{\theta}_{t+1}\right) \right\} \mathbf{x}_{S(t+1)_n}.$$

The authors suggest initializations such as $\hat{\theta}_1 = ... = \hat{\theta}_n$ or using standard SGD to obtain the first n iterates. Unlike standard SGD, convergence of the IAG algorithm can be attained without a diminishing stepsize sequence, a consequence of including an average of past gradients. The authors show that for the class of strongly convex functions, IAG has a global linear rate of convergence.

Stochastic Average Gradient - SAG

Roux et al. (2012) develop SAG, a method for optimizing strongly convex functions that employs a single observation to compute the gradient at each iteration but incorporates a gradient for each data point in the actual parameter update. The primary motivation behind the development of SAG was the search for an algorithm that retains the low cost per iteration of SGD methods while attaining the linear convergence rate associated with full gradient methods. SAG differs from IAG by how observations are selected for the update. While IAG cycles through the data points, SAG selects an observation at random at each iteration. SAG iterates are of the form

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \frac{\epsilon_t}{n} \sum_{i=1}^{n} \hat{w}_i^t,$$

where at each iteration $S(t)$ is an index randomly sampled with replacement from the set of data indices $\{1,...,n\}$ and

$$\hat{w}_i^t = \begin{cases} \nabla_\theta f(\hat{\theta}_t; Z_i) & \text{if} \quad i = S(t) \\ \hat{w}_i^{t-1} & \text{otherwise.} \end{cases}$$

$\hat{w}_i^0$ is typically initialized to 0 for all $i \in \{1,...,n\}$. In the case of logistic regression, $\hat{w}$ is updated via

$$\hat{w}_i^t = \begin{cases} \left\{ h(\mathbf{x}_i^T \hat{\theta}) - y_i \right\} \mathbf{x}_i & \text{if} \quad i = S(t) \\ \hat{w}_i^{t-1} & \text{otherwise.} \end{cases}$$

The authors provide an explicit rate of convergence for strongly convex functions that is similar to the $\mathcal{O}(1/T)$ of GD, but with a constant that is proportional to n. To mitigate the effect of the sample size on the rate of convergence, they suggest that the initial parameter vector be set to the average of n standard SGD iterates. Due to the requirement of storing previous gradients, SAG is best suited for problems such as linear regression, where the gradients are simply the feature vectors weighted by the residuals. In such instances, only the weights need to be stored. The authors conduct experiments on some binary classification problems to compare SAG to a variety of full gradient and stochastic gradient methods including GD, Nesterov Accelerated GD, L-BFGS, IAG, SGD, ASGD, and SGD with Momentum. The experiments reveal that, in general, the SGD methods outperform the full gradient methods during the first few passes through the data after which progress slows down substantially. The full gradient methods progress steadily and eventually surpass the SGD methods. However, SAG appears to do better than the other SGD methods in that it continues to make consistent progress even after the first few passes through the data, which suggests that the method may be preferable when only a limited number of passes through the data can be afforded. The results also suggest that for sufficiently large datasets, SAG can use much larger step sizes compared to IAG to attain a linear convergence rate.

Stochastic Variance Reduced Gradient (SVRG)

Johnson and Zhang (2013) propose a method for reducing the variability inherent in SGD iterates due to the approximate gradients, which tends to slow down convergence. Stochastic Variance Reduced Gradient (SVRG) stores a version of $\hat{\theta}$ denoted by $\tilde{\theta}$, which is an estimate

of $\hat{\theta}$ after every m SGD iterations, as well as an average gradient $\tilde{\mu} = n^{-1}\sum_{i=1}^{n}\nabla f(\tilde{\theta}; Z_i)$. The general update rule (after the first m SGD iterations) is given by

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \epsilon_t\{\nabla_\theta f(\hat{\theta}_t; Z_{S(t)}) - \nabla_\theta f(\tilde{\theta}; Z_{S(t)}) + \tilde{\mu}\},$$

where $S(t)$ is a random draw with replacement from $\{1, 2, ..., n\}$ at iteration t. Under the logistic regression model, after m SGD iterations to obtain $\tilde{\theta}$ and $\tilde{\mu}$, the parameter vector is updated using

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \epsilon_t\left[\left\{h(\mathbf{x}_{S(t)}^T\hat{\theta}_t) - h(\mathbf{x}_{S(t)}^T\tilde{\theta}_t)\right\}\mathbf{x}_{S(t)} + \frac{1}{n}\sum_{i=1}^{n}\left\{h(\mathbf{x}_{S(t)}^T\tilde{\theta}_t) - y_i\right\}\mathbf{x}_i\right].$$

The authors suggest that a value of m on the same order of n but a little larger be used. For instance, they use $m = 2n$ for convex functions and $m = 5n$ for non-convex functions. The authors establish a linear convergence rate in the case of strongly convex functions, and a rate of $\mathcal{O}(1/T)$ for smooth but not strongly convex functions. The method is also applicable to non-convex problems like neural networks, though in this case, the authors advocate the use of a starting vector obtained from standard SGD, after which the SVRG procedure can then be applied to accelerate convergence.

Semi-Stochastic Gradient Descent(S2GD)

An algorithm for optimizing smooth convex loss functions is proposed by Konečnỳ and Richtárik (2013). The method, called S2GD, is a blend of the computational efficiency of SGD and the ability of GD to yield stable gradients, thereby accelerating convergence. S2GD involves the computation of a single full gradient and a random number of stochastic gradients according to a geometric law over one or more epochs (where j indexes an epoch). The number of stochastic gradients, t_j, to be computed is then obtained by sampling an integer between 1 and m with probability $(1-\nu\epsilon)^{m-t}/\beta$, where m is the maximum number of stochastic steps per epoch, ϵ is a constant stepsize, ν is the lower bound on the strong convexity constant[2], and $\beta = \sum_{t=1}^{m}(1-\nu\epsilon)^{m-t}$. The S2GD algorithm operates in the following way.

- **Step 1**: Initialize $\hat{\theta}_0$ to some finite value,
- **Step 2**: For $j = 0, 1, ...,$

[2]The method can still be applied if the strong convexity constant is unknown. In this case, ν is set to 0.

i. $\hat{g}_j = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta f(\hat{\theta}_j; Z_i)$.

ii. $\hat{w}_{j,0} = \hat{\theta}_j$.

iii. For $t = 0$ to $t_j - 1$, where $t_j \in \{1, 2, ..., m\}$,

Pick $S(t) \in \{1, 2, ..., n\}$ uniformly at random with replacement.

$\hat{w}_{j,t+1} = \hat{w}_{j,t} - \epsilon\{\hat{g}_j + \nabla_\theta f(\hat{w}_{j,t}; Z_{S(t)}) - \nabla_\theta f(\hat{\theta}_j; Z_{S(t)})\}$.

iv. $\hat{\theta}_{j+1} = \hat{w}_{j,t_j}$.

The stochastic gradient updates of the auxiliary parameter $\hat{w}_j$ under logistic regression is

$$\hat{w}_{j,t+1} = \hat{w}_{j,t} - \epsilon\left[\frac{1}{n}\sum_{i=1}^{n}\left\{h\left(\mathbf{x}_i^T\hat{\theta}_j\right) - y_i\right\}\mathbf{x}_i + \left\{h\left(\mathbf{x}_{S(t)}^T\hat{w}_{j,t}\right) - h\left(\mathbf{x}_{S(t)}^T\hat{\theta}_t\right)\right\}\mathbf{x}_{S(t)}\right].$$

In effect, a full gradient is computed at the start of each epoch and then within each epoch j, a total of t_j stochastic gradient updates are made to obtain the parameter estimate. The SVRG algorithm (Johnson and Zhang 2013) arises as a special case when $\nu = 0$. The authors show that optimal choices of m and ν are $\mathcal{O}(\kappa)$ and μ respectively, where κ is the condition number and μ is the strong convexity constant. For strongly convex functions, the algorithm attains a linear rate of convergence. If the objective is simply convex, the convergence rate reduces to $\tilde{\mathcal{O}}(1/T)$.

SAGA Algorithm

Inspired by SVRG (Johnson and Zhang 2013) and SAG (Roux et al. 2012), Defazio et al. (2014) develop a method designed to optimize functions that consist of a finite sum of convex functions and a potentially non-differentiable regularization function. The method was conceived in an effort to improve upon the theoretical convergence rates of SAG and SVRG. The algorithm, called SAGA, uses a form of gradient averaging with a proximal operation on the regularizer. The SAGA algorithm is outlined below.

- **Step 1**: Initialize $\hat{\theta}_0 \in \mathbb{R}^p$ and $\nabla_\theta f(\hat{\phi}_i^0, Z_i)$ with $\hat{\phi}_i^0 = \hat{\theta}_0$ for each $\{1, ..., n\}$ and store derivatives in a data structure.

- **Step 2**: For $t \geq 0$,

  i. Sample index $S(t)$ uniformly at random with replacement from $\{1, ..., n\}$.

ii. $\hat{\phi}_{S(t)}^{t+1} = \hat{\theta}_t$ and substitute $\nabla_\theta f\left(\hat{\phi}_{S(t)}^{t}, Z_{S(t)}\right)$ with $\nabla_\theta f\left(\hat{\phi}_{S(t)}^{t+1}, Z_{S(t)}\right)$ in the data structure.

iii. $\hat{w}_{t+1} = \hat{\theta}_t - \epsilon\left\{\nabla_\theta f\left(\hat{\phi}_{S(t)}^{t+1}, Z_{S(t)}\right) - \nabla_\theta f\left(\hat{\phi}_{S(t)}^{t}, Z_{S(t)}\right) + \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta f\left(\phi_i^t, X_i\right)\right\}$.

iv. $\hat{\theta}_{t+1} = \text{prox}_\epsilon^h(\hat{w}_{t+1})$,

where $\text{prox}_\epsilon^h(y) = \text{argmin}_{\theta\in\mathbb{R}^p}\left\{r(\theta) + \frac{1}{2\epsilon}||\theta - y||^2\right\}$ and $r(\theta)$ is the regularization term. Under the logistic regression model, the auxiliary parameter $\hat{w}$ is updated according to

$$\hat{w}_{t+1} = \hat{\theta}_t - \epsilon\left[\left\{h\left(\mathbf{x}_{S(t)}^T \hat{\phi}_{S(t)}^{t+1}\right) - h\left(\mathbf{x}_{S(t)}^T \hat{\phi}_{S(t)}^{t}\right)\right\}\mathbf{x}_{S(t)} + \frac{1}{n}\sum_{i=1}^{n}\left\{h\left(\mathbf{x}_i^T \hat{\phi}_i^t\right) - y_i\right\}\mathbf{x}_i\right].$$

The value of the auxiliary parameter $\hat{\phi}_i$ is only updated when the ith data index is selected. Although SAGA lacks the advantage of low storage cost, it utilizes a constant stepsize (denoted by ϵ) and has a linear convergence rate for strongly convex functions. In the case of non-strongly convex functions, the algorithm can be easily applied without alteration and attains a convergence rate of $\mathcal{O}(1/T)$. Like SAG, SAGA performs best when applied to functions with a gradient structure similar to that of linear regression or logistic regression, where only weighting constants need to be stored.

2.4.5 Accelerated SGD Methods

Accelerated Stochastic Approximation

Kesten et al. (1958) propose an SA method that relies on the number of sign changes in successive gradient estimates to determine the magnitude of the step sizes. As the name suggests, this method was developed to speed up the convergence of SGD, which is known to slow down as iterates draw closer to a solution. The intuition behind this method is that if the number of sign changes are few then it means that the current iterate is far from the solution and so large step sizes should be taken. However, when the sign changes are frequent, it implies that the iterates are close to the solution and so the step sizes should be reduced accordingly. Typically, a constant step size is used up until when the inner-product of the current and previous gradient estimates is negative, at which point the step size is reduced.

Kesten's algorithm updates iterates as follows:

$$\hat{\theta}_t = \hat{\theta}_{t-1} - \epsilon_t \nabla_\theta f\big(\hat{\theta}_{t-1}, Z_{S(t-1)}\big).$$

The stepsize is determined in the following way.

$$\begin{aligned} s_{t+1} &= s_t + \mathbb{I}\Big[\nabla_\theta f\big(\hat{\theta}_t, Z_{S(t)}\big)^T \nabla_\theta f\big(\hat{\theta}_{t-1}, Z_{S(t-1)}\big) < 0\Big], \\ \epsilon_{t+1} &= \epsilon(s_{t+1}). \end{aligned}$$

where s_t counts the number of sign changes and $\epsilon(s_t)$ is a deterministic sequence that depends on the size of s_t. In effect, small values of s_t translate into large step sizes and large values of s_t result in small step sizes. The parameter s in the context of logistic regression is obtained via

$$s_{t+1} = s_t + \mathbb{I}\Big[\Big\{h\big(\mathbf{x}_{S(t)}^T \hat{\theta}_t\big) - y_{S(t)}\Big\}\Big\{h\big(\mathbf{x}_{S(t-1)}^T \hat{\theta}_{t-1}\big) - y_{S(t-1)}\Big\}\mathbf{x}_t^T \mathbf{x}_{t-1} < 0\Big].$$

Kesten proves almost sure convergence of the estimator for the one-dimensional case. Delyon and Juditsky (1993) extend Kesten's method to the multidimensional case and prove asymptotic normality of the resulting estimator.

2.4.6 Adaptive Learning Rate Methods

These are methods that generally employ a different learning rate for each element of the parameter vector instead of a global learning rate and are well-suited to online learning.

Adaptive Gradient - ADAGRAD

Duchi et al. (2011) introduce ADAGRAD, a family of algorithms that takes into account the attributes of the observed data in the assignment of learning rates. These methods assign higher rates to infrequent[3] features while more frequently observed features are given lower learning rates. The purpose of this mode of assigning learning rates is to identify more easily features that are relatively uncommon but are highly informative. ADAGRAD aims at facilitating the selection of a learning rate by employing different learning rates for each

[3]Infrequent features refer to those features which are rarely non-zero within very high-dimensional feature vectors but tend to be very informative when they attain non-zero values.

parameter that are updated on the fly, thereby speeding up convergence. The general form of the parameter update is

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \epsilon G_t^{-1/2} g_t,$$

where ϵ is a constant step size, $g_t = \nabla_\theta f(\hat{\theta}_t, Z_{S(t)})$, and $G_t = \sum_{\tau=1}^{t} g_\tau g_\tau^T$ is the sum of the outer product of the gradient vectors up to iteration t. However, due to the high cost associated with computing the inverse of the outer product matrix in higher dimensional problems, a more commonly used version ofADAGRAD is

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \epsilon \left\{ \text{diag}(G_t) + \eta \right\}^{-1/2} \odot g_t,$$

where η is a constant that prevents division by zero. The symbol $\odot$ denotes the Hadamard product. Under the logistic regression model, G_t is given by

$$G_t = \sum_{\tau=1}^{t} \left\{ h\left(\mathbf{x}_{S(\tau)}^T \hat{\theta}_\tau\right) - y_{S(\tau)} \right\}^2 \mathbf{x}_{S(\tau)} \mathbf{x}_{S(\tau)}^T.$$

The ADAGRAD algorithms are similar to second order GD in that they use an approximation of the Hessian of the loss function to update the parameter vector. ADAGRAD eliminates the need to tune a global learning rate since each component of the parameter vector has its own learning rate. The authors show empirically that this family of algorithms outperforms standard SGD when the gradient is sparse. ADAGRAD is a great improvement on Vanilla SGD as evidenced by its successful application to a variety of tasks including image recognition and word embedding. However, the algorithm is plagued by a consistently growing sum of accumulated gradients resulting in very small learning rates, which slows down learning.

Root Mean Square Propagation - RMSProp

To combat the drastic reduction of the learning rates that is a byproduct of ADAGRAD, Hinton et al. (2012) propose another adaptive learning rate method called RMSProp. This algorithm employs an exponentially decaying average of squared gradients to adapt the learning rate of each component of the parameter vector. After initializing $\hat{\theta}_0$ to some finite

value and setting $\hat{v}_0$ to the zero vector, the RMSProp algorithm proceeds as follows:

$$\hat{v}_t = \beta\,\hat{v}_{t-1} + (1-\beta) g_t \odot g_t,$$
$$\hat{\theta}_{t+1} = \hat{\theta}_t - \frac{\epsilon}{\sqrt{\hat{v}_t + \eta}} \odot g_t,$$

where $g_t = \nabla_\theta f\left(\hat{\theta}_t, Z_{S(t)}\right)$, ϵ is a constant learning rate, v_t is the running average of past squared gradients up to iteration t, and β is a hyper-parameter typically set at 0.9. Division and square root are element-wise operations. η is a smoothing term that prevents division by zero and is usually fixed at 0.001. Under the logistic regression model the parameter v is updated using

$$\hat{v}_t = \beta\,\hat{v}_{t-1} + (1-\beta)\left\{h\left(\mathbf{x}_{S(t)}^T \hat{\theta}_t\right) - y_{S(t)}\right\}^2 \mathbf{x}_{S(t)} \odot \mathbf{x}_{S(t)}.$$

Though the method is unpublished, RMSProp remains a widely used algorithm, especially in the training of neural networks.

Adaptive Moment Estimation - Adam

Adam, proposed by Kingma and Ba (2014), is a method that combines Adagrad's ability to handle sparse gradient vectors and RMSProp's knack for handling non-stationary objective functions. Adam adapts the learning rate of each parameter using bias-corrected estimates of the first and (uncentered) second moments of the gradient vector. The updates involved are as follows:

$$\hat{m}_{t+1} = \beta_1 \hat{m}_t + (1-\beta_1) g_t,$$
$$\hat{v}_{t+1} = \beta_2 \hat{v}_t + (1-\beta_2) g_t \odot g_t,$$
$$\hat{\theta}_{t+1} = \hat{\theta}_t - \epsilon \frac{\sqrt{1-\beta_2^t}}{(1-\beta_1^t)(\sqrt{\hat{v}_{t+1}} + \eta)} \hat{m}_{t+1},$$

where $g_t = \nabla_\theta f\left(\hat{\theta}_t, Z_{S(t)}\right)$, ϵ is a constant step size, $\hat{m}_t$ and $\hat{v}_t$ are running averages of the gradient and squared gradient respectively, and are initialized to the zero vector at the start. β_1 and β_2 are hyper-parameters that regulate the rate of decay of the running averages. η is a positive constant that prevents division by zero and is typically set at 10^{-8}. Division and square root are element-wise operations. The authors suggest a value of 0.9 for β_1 and 0.999 for β_2. The auxiliary parameters m and v within the context of logistic regression are

updated as follows:

$$\hat{m}_{t+1} = \beta_1 \hat{m}_t + (1-\beta_1)\Big\{h\big(\mathbf{x}_{S(t)}^T \hat{\theta}_t\big) - y_{S(t)}\Big\}\mathbf{x}_{S(t)},$$
$$\hat{v}_{t+1} = \beta_2 \hat{v}_t + (1-\beta_2)\Big\{h\big(\mathbf{x}_{S(t)}^T \hat{\theta}_t\big) - y_{S(t)}\Big\}^2 \mathbf{x}_{S(t)} \odot \mathbf{x}_{S(t)}.$$

Adam scales extremely well for large datasets and/or high-dimensional feature vectors, requires little memory, and is easy to implement. The authors assert that Adam is widely applicable to a variety of machine learning problems that deal with non-convex objective functions.

2.4.7 Second Order/Quasi-Newton SG Methods

These are methods that attempt to mirror Newton's method by incorporating curvature information into parameter updates using some (usually inexpensive) approximation of the Hessian matrix.

Online BFGS

Schraudolph et al. (2007) modify the popular BFGS algorithm to accommodate convex optimization in online settings. After initializing $\hat{\theta}_0$ to some finite value and the Hessian approximation H with $\hat{H}_0 = \eta\mathbf{I}$ for some $\eta > 0$, the proposed online BFGS method makes parameter updates of the form

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \frac{\epsilon_t}{c}\hat{H}_t \nabla_\theta f\big(\hat{\theta}_t, Z_{S(t)}\big),$$

where $0 < c \leq 1$. The Hessian approximation at time t, $\hat{H}_t$, is updated using

$$\hat{H}_{t+1} = \big(\mathbf{I} - \hat{\rho}_t \hat{s}_t \hat{u}_t^T\big)\hat{H}_t\big(\mathbf{I} - \hat{\rho}_t \hat{u}_t \hat{s}_t^T\big) + c\,\hat{\rho}_t \hat{s}_t \hat{s}_t^T,$$

where $\hat{s}_t = -c^{-1}\epsilon_t \hat{H}_t \nabla_\theta f\big(\hat{\theta}_t, Z_{S(t)}\big)$, $\hat{u}_t = \nabla_\theta f\big(\hat{\theta}_{t+1}, Z_{S(t)}\big) - \nabla_\theta f\big(\hat{\theta}_t, Z_{S(t)}\big) + \alpha \hat{s}_t$ ($\alpha \geq 0$), and $\hat{\rho}_t = (\hat{s}_t^T \hat{u}_t)^{-1}$.

The online method differs from the standard BFGS algorithm in that it eliminates the line search used in the original method to find an acceptable stepsize and scales down the last term in the update of $\hat{H}$ by a factor c. It also uses consistent gradient measurements that involve the same data point to compute the differences in gradients, i.e., $\nabla_\theta f\big(\hat{\theta}_{t+1}; Z_{S(t)}\big) -$

$\nabla_\theta f\big(\hat{\theta}_t; Z_{S(t)}\big)$ instead of $\nabla_\theta f\big(\hat{\theta}_{t+1}; Z_{S(t+1)}\big) - \nabla_\theta f\big(\hat{\theta}_t; Z_{S(t)}\big)$. Under logistic regression, the terms u and s for computing the Hessian are updated in the following way.

$$\hat{s}_t = -c^{-1}\epsilon_t \hat{H}_t \Big\{ h\big(\mathbf{x}_{S(t)}^T \hat{\theta}_t\big) - y_{S(t)} \Big\} \mathbf{x}_{S(t)},$$
$$\hat{u}_t = \Big\{ h\big(\mathbf{x}_{S(t)}^T \hat{\theta}_{t+1}\big) - h\big(\mathbf{x}_{S(t)}^T \hat{\theta}_t\big) \Big\} \mathbf{x}_{S(t)} + \alpha \hat{s}_t.$$

The authors argue that the additional computational cost incurred from computing the term $f\big(\hat{\theta}_{t+1}; Z_{S(t)}\big)$ in the Hessian approximation is mostly offset by the noise reduction gained by using the difference between gradients computed with the same data sample, $Z_{S(t)}$. The authors perform experiments on natural language processing tasks that indicate that online BFGS significantly outperforms SGD and the natural gradient descent method of Amari et al. (2000).

Quasi-Newton SGD (SGD-QN)

SGD-QN is a method by Bordes et al. (2009) designed primarily to solve linear SVM problems. This algorithm is motivated by the online BFGS method. The method employs an update that involves estimating a diagonal rescaling matrix to approximate the Hessian of an L2-regularized loss function with regularization parameter λ.

The individual elements of the diagonal matrix are computed in the following manner

$$\hat{H}_{ii} = \frac{[\hat{\theta}_t - \hat{\theta}_{t-1}]_i}{\big[\nabla_\theta f\big(\hat{\theta}_t; Z_{S(t-1)}\big) - \nabla_\theta f\big(\hat{\theta}_{t-1}; Z_{S(t-1)}\big)\big]_i},$$

and updated only after m number of iterations. The parameter updates are broken up into two parts; one involving the actual data points with updates given by

$$\hat{\theta}_{t+1} = \hat{\theta}_t - (t + t_0)^{-1} \hat{H} \nabla_\theta f\big(\hat{\theta}_t; Z_{S(t)}\big),$$

where $t_0 > 0$ is a hyper-parameter of the learning rate schedule, and the other involving only the regularization term which occurs after m iterations with updates given by

$$\hat{\theta}_{t+1} = \hat{\theta}_{t+1} - m(t + t_0)^{-1} \lambda \hat{H} \hat{\theta}_{t+1}.$$

m is fixed at c/s, where c is a predetermined constant (authors suggest $c = 16$), and s is the average proportion of non-zero coefficients present in the feature vectors. The resulting

algorithm, though slightly slower per iteration than traditional SGD due to the additional computation introduced by the Hessian approximation, requires fewer iterations in total to attain the same level of accuracy.

Following the discovery of a flaw in the original SGD-QN algorithm which prevents the method from achieving its design objective of improving the learning speed of SGD, the authors issued an erratum (Bordes et al. 2010) proposing a fix. The flaw unearthed is particularly evident in the impact on dense[4] data, where the learning rates frequently remain the same for all features. This defeats the purpose of the algorithm since the updates based on the original method are equivalent to using a global learning rate for all the features. The correction proposed consists of a change in the way that the learning rates are adapted. The learning rate for the ith coordinate of the parameter vector based on the original method is of the form

$$\epsilon_{i,t} = \frac{\hat{H}_{ii}}{t_0 + t},$$

while the corrected version employs learning rates of the form

$$\epsilon_{i,t} = \frac{1}{\lambda t_0 + \sum_{k=1}^{t-1} r_{i,k}}, \quad \text{where} \quad r_{i,t} = \frac{\left[\nabla_\theta f\left(\hat{\theta}_t; Z_{S(t)}\right) - \nabla_\theta f\left(\hat{\theta}_{t-1}; Z_{S(t)}\right)\right]_i}{\left[\hat{\theta}_t - \hat{\theta}_{t-1}\right]_i}.$$

Under logistic regression, $r_{i,t}$ is computed as follows:

$$r_{i,t} = \frac{\left[\left\{h\left(\mathbf{x}_{S(t)}^T \hat{\theta}_t\right) - h\left(\mathbf{x}_{S(t)}^T \hat{\theta}_{t-1}\right)\right\}\mathbf{x}_{S(t)}\right]_i}{\left[\hat{\theta}_t - \hat{\theta}_{t-1}\right]_i}.$$

Specifically, the corrected version of the algorithm adapts the learning rates for each feature at different speeds which improves performance, especially on ill-conditioned data. The method can be adapted to non-linear models for which analytical forms of the gradient can be obtained.

Stochastic Quasi-Newton (SQN) Method

SQN (Byrd et al. 2016) is a method designed for convex functions that employs the limited memory version of the standard BFGS method to incorporate curvature information into parameter updates. Unlike the classical BFGS method, SQN does not approximate

[4]Dense data consist of observations whose inputs are predominantly real values other than zero.

the Hessian using differences between gradients. Rather, the method uses Hessian-vector products computed at regular intervals. In addition, it uses a different and larger subset of randomly selected data points than that used for computing the gradient to approximate the Hessian. Moreover, the curvature estimates are not updated at each iteration but intermittently at equally spaced intervals. This is done to control the additional computational cost associated with the Hessian approximation. The algorithm makes updates of the form

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \frac{\epsilon_t}{|B|}\hat{H}_t \sum_{i \in B} \nabla_\theta f\big(\hat{\theta}_t; Z_i\big),$$

where $\hat{H}_t$ is a symmetric positive definite approximation to the Hessian of the loss function, the subscript $B \subset \{1, 2, .., n\}$ denotes the indices of the subset of observations used to compute the gradient, and $|B|$ denotes the number of elements in B. Specifically, the Hessian is updated via

$$\hat{H}_{t+1} = \big(\mathbf{I} - \hat{\rho}_t \hat{s}_t \hat{u}_t^T\big)\hat{H}_t\big(\mathbf{I} - \hat{\rho}_t \hat{u}_t \hat{s}_t^T\big) + \hat{\rho}_t \hat{s}_t \hat{s}_t^T, \quad \hat{\rho} = 1/(\hat{u}_t^T \hat{s}_t).$$

The correction terms for updating the Hessian are computed every L iterations using

$$\hat{s}_k = \bar{\theta}_k - \bar{\theta}_{k-1}, \quad \hat{u}_k = \frac{1}{|B_H|} \sum_{i \in B_H} \nabla_\theta^2 f\big(\bar{\theta}_k; Z_i\big)\hat{s}_k,$$

where $\bar{\theta}_k = \frac{1}{L}\sum_{j=t-L+1}^{t} \hat{\theta}_j$ and B_H denotes the indices of the subset of observations that is used to compute the Hessian approximation to distinguish it from B, the subset used in the gradient computation. The subscript k is used to keep track of the times when a new correction pair (s_k, u_k) is computed while t enumerates the gradient computations and parameter updates. In the case of logistic regression, the correction term u_k is updated using

$$\hat{u}_k = \frac{1}{|B_H|} \sum_{i \in B_H} h\big(\mathbf{x}_i^T \bar{\theta}_k\big)\Big(1 - h\big(\mathbf{x}_i^T \bar{\theta}_k\big)\Big)\mathbf{x}_i \mathbf{x}_i^T.$$

The authors suggest values of $L = 10$ or 20, $|B| = 50, 100$ or greater, and $|B_H| \geq 300$. They also establish global convergence of the algorithm for strongly convex functions. Though the convergence rate of SQN is not an improvement on that of standard SGD, numerical experiments conducted by the authors show that SQN outperforms the SGD and online BFGS algorithms, and requires fewer iterations to attain similar levels of accuracy. SQN can easily be applied to non-convex functions on condition that $s_k^T u_k > 0$ is guaranteed for

each correction pair.

2.4.8 Hybrid Methods

Hybrid methods are a blend of components of incremental gradient methods and full gradient methods in order to take advantage of the low computational cost per iteration of the former class of methods while exploiting the ultimately faster convergence rate of the latter.

Hybrid Online-Batch Backpropagation

Grippo (1994) introduces a hybrid method primarily for training neural networks. The method exploits the computational efficiency of the incremental gradient method while redefining a constant step size (denoted by ϵ) at the start of each epoch. The hybrid online-batch method, as it is called, periodically modifies the step size based on an acceptance criterion which measures the reduction in the objective function. Thus, the method requires computing the full objective intermittently in order to determine what the stepsize should be. The basic form of the algorithm consists of the following main steps.

- **Step 0**: Initialize $\hat{\theta}_0$ and select the hyper-parameters $0 < \gamma_1 < 1$, $\gamma_2 > 0$, $0 < \delta < 1$, and $\Delta_2 \geq \Delta_1 > 0$.
- **Step 1**: $t = 0$, $\hat{W} = \sum_{i=1}^{n} f(\theta_0; Z_i)$.
- **Step 2**: $\hat{\psi}_1 = \hat{\theta}_t$ and $j = 0$; Choose Δ_t such that $\Delta_2 \geq \Delta_t \geq \Delta_1 > 0$ and $\epsilon = \Delta_t$.
- **Step 3**: For $i = 1, ..., n$

  $$\hat{\psi}_{i+1} = \hat{\psi}_i - \epsilon \nabla_\theta f(\hat{\psi}_i; Z_i).$$

- **Step 4**: Compute $\ell(\hat{\psi}_{n+1}) = \sum_{i=1}^{n} f(\hat{\psi}_{n+1}; Z_i)$

  i. If $\ell(\hat{\psi}_{n+1}) \leq \hat{W} - \gamma_1 \epsilon \| \sum_{i=1}^{n} \nabla_\theta f(\hat{\psi}_i; Z_i) \|^2 - \gamma_2 \epsilon^2 \sum_{i=1}^{n} \| \nabla_\theta f(\hat{\psi}_i; Z_i) \|^2$,
  $\hat{\theta}_{t+i} = \hat{\psi}_{i+1}$ for $i = 1, ..., n$, $\epsilon_t = \epsilon$, and $\hat{W} = \ell(\hat{\psi}_{n+1})$
  $t = t + n$ and return to **Step 2**.

  ii. Else, $\epsilon = \delta \epsilon$, $j = j + 1$ and return to **Step 3**.

The acceptance criterion (in Step 4) under the logistic regression model is given by

$$\sum_{i=1}^{n}\Big\{\log\big(1+e^{\mathbf{x}_i^T\hat{\psi}_{n+1}}\big)-y_i\mathbf{x}_i^T\hat{\psi}_{n+1}\Big\}\leq\sum_{i=1}^{n}\Big\{\log\big(1+e^{\mathbf{x}_i^T\hat{\theta}_0}\big)-y_i\mathbf{x}_i^T\hat{\theta}_0\Big\}$$
$$-\gamma_1\epsilon\left\|\sum_{i=1}^{n}\Big\{h\big(\mathbf{x}_i^T\hat{\psi}_i\big)-y_i\Big\}\mathbf{x}_i\right\|^2-\gamma_2\epsilon^2\sum_{i=1}^{n}\Big\{h\big(\mathbf{x}_i^T\hat{\psi}_i\big)-y_i\Big\}^2\mathbf{x}_i\mathbf{x}_i^T.$$

Two notable advantages of the proposed method are it ensures a consistent reduction of the objective function and does not require a decaying stepsize schedule. On the other hand, evaluating the objective function to determine the bounds on the constant step sizes, however periodic, imposes additional computational costs that may be prohibitive.

A class of IG methods for Least Squares

A class of hybrid methods for least squares problems is proposed by Bertsekas (1997a). These methods combine SGD with GD in a way that leverages the fast convergence of the former method in the early stages of optimization and the much faster linear convergence rate of the latter in the final stages. The proposed class of methods is parameterized by a non-negative constant μ that controls the extent to which the method approaches a full gradient method. In effect, a special weighting scheme which is a function of μ is imposed on the observations. The general method proceeds as follows.

- **Step 1**: Fix a scalar $\mu>0$, $t=0$, and initialize $\hat{\theta}_0$.
- **Step 2**: $\hat{\psi}_0=\hat{\theta}_t$.
- **Step 3**: $\hat{h}_0=0$ and for $i=1,...,n$,
  $w_i(\mu)=\frac{1}{1+\mu+...+\mu^{n-i}}$.
  $\hat{h}_i=\mu\hat{h}_{i-1}+\sum_{j=1}^{i}w_j(\mu)\nabla_\theta f(\hat{\psi}_{j-1};Z_j)$.
  $\hat{\psi}_i=\hat{\theta}_t-\epsilon_t\hat{h}_i$.
- **Step 4**: $t=t+1$, $\hat{\theta}_{t+1}=\hat{\psi}_n$ and return to **Step 2**.

The term $\hat{h}$ (for each $i=1,...,n$) under the logistic regression model is updated as follows:

$$\hat{h}_i=\mu\hat{h}_{i-1}+\sum_{j=1}^{i}w_j(\mu)\Big\{h\big(\mathbf{x}_j^T\hat{\psi}_{j-1}\big)-y_j\Big\}\mathbf{x}_j.$$

It can be observed that a value of $\mu = 0$ yields the standard SGD method while the method approaches GD as $\mu \to \infty$. In addition, the author proposes a time-varying version of the method which involves gradually incrementing μ so that the method starts out as an incremental method and approaches a full gradient method as μ gets larger. For the time-varying version, when the dataset is large, the computation of $w_i(\mu)$ is prone to considerable numerical error when μ is much bigger than 1. Therefore, to circumvent this problem, the author recommends that each time μ is incremented to a value greater than 1, more observations be grouped together to reduce the number of data batches to be cycled through. This hybrid method uses a constant stepsize without suffering the oscillation associated with the standard SGD method and is especially suitable for training neural networks. The author shows that this class of methods achieves a linear rate of convergence for quadratic functions.

Hybrid Deterministic-Stochastic Method

Friedlander and Schmidt (2012) present a method for optimizing a finite sum of functions, that gradually increases the number of data points used in evaluating the gradient as the algorithm progresses. Specifically, the method begins with a sample size of 1 and then grows linearly to include all the observations according to the formula

$$|B_{t+1}| = \lceil \min\{1.1|B_t| + 1, n\} \rceil,$$

where B_t is the batch at iteration t and n is the sample size. The authors provide theoretical analyses of their method that show that it is possible to obtain the faster convergence rates of full gradient methods if the sample size is adapted in the aforementioned fashion. They implement the proposed method of incrementing the batch size in conjunction with a second order method of updating the parameter iterates that uses the L-BFGS quasi-Newton approximation to the Hessian. The authors conduct experiments involving binary and multinomial logistic regression, as well as conditional random fields to compare their method to SGD and standard L-BFGS. The results suggest that both SGD and the hybrid method progress quickly at the initial stages. However, unlike SGD which slows down considerably after a number of passes through the data, the hybrid method maintains steady progress like L-BFGS.

Big Batch SGD (BB-SGD)

A technique for adaptively increasing the batch sizes used in SGD updates is proposed by De et al. (2016). A particular advantage of big batch methods is that they obviate the need for a decaying learning rate schedule since larger batches improve the accuracy of the approximate gradient.

The proposed method relies on the premise that $-\nabla \ell_B(\theta)$ is expected to be a descent direction reasonably often if

$$k^2 \|\nabla_\theta \ell_B(\theta)\|^2 \geq \frac{1}{|B|}\mathbb{E}\Big\{ \Big\|\nabla_\theta f(\theta;Z) - \nabla_\theta \ell(\theta)\Big\|^2 \Big\}$$

for some $k < 1$ and batch B, where $\nabla_\theta \ell_B(\theta) = |B|^{-1}\sum_{Z\in B} \nabla_\theta f(\theta;Z)$. The algorithm proceeds by starting with an initial batch size greater than 1. The batch size is increased until such time as the condition, $\|\nabla_\theta \ell_B(\theta_t)\|^2 > V_B/|B|$ is satisfied, where

$$V_B = \frac{1}{|B|-1}\sum_{Z\in B}\Big\|\nabla_\theta f(\hat{\theta}_t;Z) - \nabla_\theta \ell_B(\hat{\theta}_t)\Big\|^2.$$

Once the above stipulation is met, the parameter vector is updated as follows:

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \epsilon \frac{1}{|B|}\sum_{Z\in B}\nabla_\theta f(\hat{\theta}_t;Z).$$

In the case of logistic regression, V_B is given by

$$V_B = \frac{1}{|B|-1}\sum_{(\mathbf{x},y)\in B}\left\|\Big\{h(\mathbf{x}^T\hat{\theta}_t) - y\Big\}\mathbf{x} - \sum_{(\mathbf{x},y)\in B}\Big\{h(\mathbf{x}^T\hat{\theta}_t) - y\Big\}\mathbf{x}\right\|^2.$$

The batch size is incremented according to the rule $B \leftarrow B + 0.1B$. The authors prove that when the objective function has Lipschitz gradients and satisfies the Polyak-Lojasiewicz Inequality,[5] the algorithm attains a linear convergence rate (albeit slower than gradient descent by a factor of $\{k^2+(1-k)^2\}/(1-k)^2$, $k \in (0,1)$). The convergence rate of this method for smooth convex functions is $\mathcal{O}(1/T)$. The authors assert that the method is applicable to non-convex functions and performs well on neural networks. They provide details on the practical implementation of their method including the use of BB-SGD with the Barzilai-

[5] $\|\nabla \ell(\theta)\|^2 \geq 2\mu(\ell(\theta) - \ell(\theta^*))$ for $\mu > 0$ (this does not require convexity but implies that every stationary point is a global minimizer).

Borwein step size method (Barzilai and Borwein 1988) and backtracking line search. They compare BB-SGD to GD, standard SGD with a decaying learning rate, and a first order version of the hybrid method by Friedlander and Schmidt using experiments conducted on linear and logistic regression problems. Their results show that the big batch methods perform better than the other methods. They also use image classification problems using convolutional neural networks to compare BB-SGD against SGD with fixed and decaying learning rates, and the ADADELTA method of Zeiler (Zeiler 2012) that indicate that BB-SGD outperforms the other algorithms.

2.4.9 Other Stochastic Gradient Methods

There is a plethora of algorithms that are related to stochastic approximation with proven convergence rates. One such method is Pegasos, a stochastic sub-gradient descent optimization method developed by Shalev-Shwartz et al. (2011) for SVM problems that has a convergence rate of $\mathcal{O}(1/T)$ and is particularly useful for solving text classification problems using large datasets. Pegasos can be adapted to fit other prediction problems and loss functions. Other stochastic gradient methods that utilize sub-gradient averaging include Nesterov's dual averaging method (Nesterov 2009) for optimizing non-smooth convex functions, which achieves optimal convergence rates, and Xiao's extension (Xiao 2010), which exploits the structure of regularization in the online setting by using the average of all previous sub-gradients as well as the whole regularization term, and not just its sub-gradient. Xiao proves that his method has a faster convergence rate of $\mathcal{O}(\log T/T)$ when the regularization term is strongly convex. Related works can be found in Zinkevich (2003), Hazan et al. (2007), Hazan and Kale (2011), Moulines and Bach (2011), Agarwal et al. (2009). Additional work on adaptive learning rates on stochastic optimization can be found in Zeiler (2012), Hayashi et al. (2016), and Dozat (2016).

A comprehensive theoretical analysis of stochastic approximation can be found in Kushner and Yin (2003). Further work on convergence analyses of incremental gradient methods can be found in Ljung (1977), Tsitsiklis et al. (1986), Bertsekas and Tsitsiklis (1989), White (1989), Luo (1991), Mangasarian (1993), Gaivoronski (1994), Grippo (1994), Mangasarian and Solodov (1994), Zhi-Quan and Paul (1994), Bertsekas and Tsitsiklis (1996), Bertsekas (1997b).

Table 2.1: Table showing SGD methods and type of data to which method is applicable

Method	Offline setting	Online setting
Standard SGD	✓	✓
Mini-batch SGD	✓	
SGD with Momentum	✓	✓
SGD with Nesterov Momentum	✓	✓
ASGD	✓	✓
α-Suffix	✓	✓
Polynomial-decay	✓	✓
IAG	✓	
SAG	✓	
SAGA	✓	
AdaGrad	✓	✓
RMSProp	✓	✓
Adam	✓	✓
Online BFGS	✓	✓
SGD-QN	✓	✓
SQN	✓	
Hybrid Online-Batch	✓	
IG methods for LS	✓	
Hybrid Deterministic-Stochastic	✓	

CHAPTER

3

SOME APPLICATIONS OF SGD IN STATISTICS

SGD methods are used extensively in many areas of Statistics, including design of experiments, reinforcement learning, and classical regression and classification problems. SGD and its variants are arguably the most popular algorithms used in large scale and online machine learning problems. In the following sections, we outline a number of ways in which SGD is used in different statistical fields.

3.1 Design of Experiments

Design of Experiments, aka Experimental Design, encompasses techniques and methods for efficiently studying how response variables depend on experimental (controlled) factors. It involves systematically varying the levels of different factors in a controlled environment, recording the changes that are observed in the response, and analyzing the data collected in order to arrive at some conclusion. It is widely applied in manufacturing and scientific

research, including bioassay and clinical trials. For example, in the development of a new drug, one of the main objectives is to find the maximum tolerable dose, that is, the highest dose level that will result in toxicity at a predetermined probability, or the median effective dose, the dose required to achieve a desired effect in half of the population. The question of estimating the median effective dose (or some other percentile, say the α-percentile, of the dose response curve) can be expressed in the form of a problem that SA is designed to solve. Thus, the dose-finding problem becomes one of finding the solution to an equation of the form:

$$F(X)-\alpha=0,$$

where $F(X)$ is the probability of toxicity occurring at dose X. The SA procedure for the above scenario is given by

$$X_{t+1}=X_t-\epsilon_t(Y_t-\alpha),$$

where X_t is the dose administered to the tth subject and Y_t is a binary random variable which takes the value of 1 to indicate the presence of toxicity in the subject and 0 otherwise. Stochastic approximation, though seldom used directly in recent times, has been applied with some success in such settings where sequential designs are feasible.

Wetherill (1963) uses an empirical study to investigate the performance of the SA procedure when applied to estimating the quantiles of a dose-response curve in a bioassay setting. He explores the efficiency of the Robbins-Monro (RM) estimator of the median quantile and the behaviour of its variance in small samples. By computing the mean squared error of the estimator for different combinations of sample sizes, initial values, and step sizes, he concludes that the RM estimator for the median is robust to sub-optimal choices of the initial value and the constant factor, c, of the decaying learning schedule (of the form $\epsilon_t=c/t$). Numerical results of the study also suggest that the small sample variance of the estimator differs very little from the asymptotic variance. The results, however, indicate that the RM method is liable to result in large bias in the estimators of extreme percentiles.

Cochran and Davis (1965) investigate the use of the RM method to estimate the median lethal dose (LD50) in bioassay via sequential experiments. Specifically, the authors seek to explore whether the asymptotic properties of the RM procedure hold for sample sizes less than 50, as well as its efficacy when initial values of the LD50 and the standard deviation of the tolerance distribution are poor. The experiment begins by making an initial guess, X_1, of the median lethal dose and administering it to m_1 test subjects (animals). The dose level

administered to the $(t+1)$th group of test subjects is given by

$$X_{t+1} = X_t - \epsilon_t \left(p_t - \frac{1}{2} \right),$$

where $p_t = k_t/m_t$ with k_t being the number of test subjects who deteriorate after being given the dose and m_t, the number of test subjects to whom the tth dose is given. When the experiment is completed, the estimate of the LD50 is obtained as the dose level which the researcher would have administered to the next group of subjects had the study not being terminated. The authors posit that this sequential method is useful when subjects respond to the drug or stimulus within a short period of its being dispensed. They also suggest that a merit of the method is that it may require fewer test subjects to estimate the LD50 with a specified precision as compared to non-sequential methods.

To explore the usefulness of the RM procedure, the authors compute the bias, variance, and MSE of the desired estimate using the expressions derived by Hodges and Lehmann (Hodges Jr and Lehmann 1956) for finite n. Cochran and Davis' experimental results suggest that when the starting value is poorly selected, smaller step sizes have more adverse effects on the resulting estimate than larger ones, and result in large mean squared error due to large bias.

To avoid such large biases, the authors suggest the use of the delayed RM process. The delayed method maintains a constant step size to determine the next dose level as long as all test subjects either survive or perish after being given the current dose. The diminution of step sizes is introduced when a change is observed in the proportion of deaths that occur. Another option is to use Kesten's accelerated method (Kesten et al. 1958), which adapts the step size only when the difference between the current dose and the previous dose $(X_t - X_{t-1})$ is of opposite sign to the difference between the two preceding doses $(X_{t-1} - X_{t-2})$. In the event that the researcher has no reliable initial guess of the lethal dose, the authors propose a two-stage plan, where the first stage involves using single test subjects at equally spaced dose levels to obtain a preliminary guess about the location of the LD50, and the second stage utilizes the standard RM method. Cochran and Davis' study indicates that the MSE of the resulting estimator is less sensitive to sub-optimal choices of the initial dose level if larger step sizes are used.

Anbar (1976) analyzes the asymptotic optimality of the Robbins-Monro estimator for the median effective dose, where optimality refers to attaining minimum asymptotic variance.

This optimality can be achieved when the slope of the dose response curve is known, which is unfeasible in many situations. The RM method under consideration is of the form

$$X_{t+1} = X_t - \frac{a_t}{t}(Y_t - \alpha),$$

where a_t^{-1} is a sequence that approaches β, the slope of the dose response function. The author discusses adaptive methods that are available for cases where the slope is unknown. The first, Venter's method (Venter 1967), requires the use of pairs of observations, Y_t' and Y_t'' taken at $X_t + c_t$ and $X_t - c_t$ respectively such that Y_t in the above RM recursion is substituted with $Y_t' + Y_t''$, where c_t is an appropriately chosen sequence satisfying $c_t = c\,t^{-\gamma},\ c > 0,\ 0 \le \gamma \le 0.5$. Venter adaptively incorporates into the recursion process a slope estimate of the form $a_t^{-1} = \beta_t = t^{-1}\sum_{j=1}^{t}(Y_j' - Y_j'')/2c_j$. The second method, proposed by Anbar (Anbar 1978), is basically to use the inverse of the standard slope estimator of the linear regression model as the slope estimate of the dose response curve, i.e. $a_t^{-1} = \beta_t = \sum_{j=1}^{t} Y_j(X_j - \bar{X}_t)/\sum_{j=1}^{t}(X_j - \bar{X}_t)^2$, where $\bar{X}_t = t^{-1}\sum_{j=1}^{t} X_j$. Both Venter's and Anbar's methods with appropriate truncation have been shown to yield quantile estimates that have minimal asymptotic variance. Based on numerical studies by the author, it appears that Anbar's method is less sensitive to poor choices of the starting dose level as compared to Venter's method.

Khuri et al. (2006) present a comprehensive review on the different approaches to solving the problem of selecting a design for a generalized linear model. They discuss the SA method as an approach to developing an optimal design in sequential analysis. They consider SA in the context of estimating the quantiles of a dose-response curve. Using the works of Wu (1985), and Frees and Ruppert (1990), they evaluate the deficiency of SA for efficient estimation of the dose levels in small samples, especially since the effectiveness of the procedure is highly dependent on the initial choice of dose level.

Cheung and Elkind (2009) develop an algorithm based on the RM procedure to determine the maximum tolerated dose, the highest dose of a drug that does not lead to unacceptable side effects, but yields the desired results. The method uses continuous measurements rather than the typical binary outcomes often used in Phase I trials. The authors emphasize that the proposed method, by virtue of its dependence on SA, is relatively simple and shares the convergence properties of the RM procedure.

The algorithm relies on what they refer to as a virtual observation, which is the actual observation perturbed by some measure of the difference between the assigned dose and

the dose which is actually administered. This approach is meant to serve as a means of circumventing the need for a continuum of doses, a requirement of the SA procedure which is often impracticable especially in phase I trials. An outline of the procedure is below.

Consider a trial with m patients in each cohort. Let $Y_{ij} = M(X_i) + \sigma(X_i)\epsilon_{ij}$ be the safety measurement of the jth patient in the ith cohort, where $M(x)$ and $\sigma(x)$ are the mean and standard deviation of the distribution of outcomes at dose x, and ϵ_{ij} are independently distributed from a distribution G, with zero mean and unit variance. Let X_i^* be the assigned dose and X_i, the dose to be administered to cohort i, which, in practice, is limited to a set of discrete values say, $1, ..., K$. The goal is to find x such that $P(Y_{ij} > t_0 | X_i = x) = p$, where t_0 is some pre-specified threshold. Let $U_i = \bar{Y}_i + [E\{S_i/\sigma(X_i)\}]^{-1} z_p S_i$ where $\bar{Y}_i$ and S_i are the respective sample mean and standard deviation of cohort i, and z_p is the upper pth percentile of G. The virtual observation for cohort i made at X_i^* is defined as $V_i = U_i + \beta(X_i^* - X_i)$ for some $\beta > 0$. The following SA recursion is then used to obtain the next assigned dose X_{i+1}^*:

$$X_{i+1}^* = X_i^* - \frac{1}{i\beta}(V_i - t_0)$$

and the next actual dose is given by $X_{i+1} = C(X_{i+1}^*)$, where $C(x)$ is the rounded value of x if $0.5 \leq x < K + 0.5$, and $x = 1$ if $x < 0.5$ and $x = K$ if $x > .5$. The doses could be instantiated at the lowest dose, that is $X_i^* = X_i = 1$, or at some predetermined maximum tolerated dose. Results of a simulation study show that the proposed method is superior to the commonly used continual reassessment method (O'Quigley et al. 1990), as well as robust under misspecification of the distribution G.

Cheung (2010) explores the similarities and differences between dose-finding in clinical trials and stochastic approximation methods. The author provides a detailed analysis of the SA method for percentile estimation as well as modern model-based methods for dose-finding proposed by O'Quigley et al. (1990), Ratain et al. (1993), and Eisenhauer et al. (2000). Cheung is of the view that as modern methods become more complex making their theoretical properties difficult to study, the class of SA procedures is a wellspring of ideas that can be tapped into and applied to other dose-finding methods where practicable. The author emphasizes that SA with its well-established theory, may yet prove useful in extending the theoretical principles of model-based methods, particularly in specialized clinical settings.

3.2 Classification and Regression

Many statistical problems can be cast as regression or classification problems. Regression involves estimating the relationship between a dependent variable and one or more predictors. Classification is the task of learning a model that predicts the category of a new observation by using data that consists of observations whose categories or labels are known. Some statistical tools for solving regression and classification problems include generalized linear models (GLMs), support vector machines (SVMs), and neural networks. This section discusses the different ways in which SGD has been applied in the context of GLMs, SVMs, and neural nets to solve statistical problems.

3.2.1 Generalized Linear Models (GLMs)

Generalized Linear Models, previously introduced in Section 2, are a generalization of linear regression such that an appropriate transformation of the conditional expectation of the response given the predictors results in a linear function of the predictors. GLMs are based on the assumption that the response conditioned on the predictors comes from a distribution in the exponential family. Commonly used GLMs include the well-known linear regression model for continuous responses, logistic regression model for binary responses, and the Poisson model for count responses. A number of ways in which SGD methods have proved useful in analyzing or solving problems cast as GLMs are outlined below.

Györfi and Walk (1996) investigate convergence properties of the ASGD estimator of the parameter vector of a linear regression model under a constant step size rule. They show that for a stationary and ergodic sequence of observations, the averaged estimator converges almost surely to the optimum parameter value, with smaller step sizes resulting in smaller biases. They also prove strong consistency and asymptotic unbiasedess of the estimator under independence and martingale assumptions on the observations. The authors further show asymptotic normality of the estimator under the same assumptions.

Tsuruoka et al. (2009) propose a method for training log-linear models used in natural language processing tasks that uses a modification of L1 regularization coupled with the clipping SGD method of Carpenter (Carpenter 2008). Their method updates the parameter vector of the model, denoted by θ, at each iteration using the difference between the total penalty and the cumulative value of the actual penalties that have been applied to each parameter up until that point. Only the parameters associated with the features present in

the observation sampled at the tth iteration are updated. The following way of updating the jth component of the parameter vector is proposed.

$$\hat{\theta}_{t+\frac{1}{2},j} = \hat{\theta}_{t,j} + \epsilon_t \frac{\partial \ell\big(\theta; Z_{S(t)}\big)}{\partial \theta_j}\bigg|_{\theta=\hat{\theta}_t}$$

$$\text{if} \quad \hat{\theta}_{t+\frac{1}{2},j} > 0 \quad \text{then} \quad \hat{\theta}_{t+1,j} = \max\big\{0, \hat{\theta}_{t+\frac{1}{2},j} - \big(u_t + \hat{q}_{t-1,j}\big)\big\}$$

$$\text{else if} \quad \hat{\theta}_{t+\frac{1}{2},j} < 0 \quad \text{then} \quad \hat{\theta}_{t+1,j} = \min\big\{0, \hat{\theta}_{t+\frac{1}{2},j} - \big(u_t - \hat{q}_{t-1,j}\big)\big\},$$

where $\hat{q}_{t,j} = \sum_{k=1}^{t}\big(\hat{\theta}_{t+1,j} - \hat{\theta}_{t+\frac{1}{2},j}\big)$ is the total L1 penalty that $\hat{\theta}_j$ has accumulated up until time t, $u_t = \lambda/n \sum_{k=1}^{t} \epsilon_k$, and λ is the regularization parameter. The authors evaluate the efficacy of their method using experiments involving text chunking, named entity recognition, and part-of-speech tagging. The results suggest that their SGD method with exponential decay stepsize schedule[1] produces compact and more accurate models within a shorter span of time compared to other L1-regularized SGD methods including the popular Quasi-Newton method, OWL-QN, introduced by Andrew and Gao (2007).

Xu (2011) presents a way of setting the learning rate that requires fewer observations or fewer passes over the dataset to reach convergence compared to ASGD, which typically takes a very large number of data points or several passes over the dataset to achieve convergence. The proposed learning rate schedule is of the form

$$\epsilon_t = \frac{\epsilon_0}{(1 + a\epsilon_0 t)^c}, \ \epsilon_0 > 0, \ a > 0, \ c \in [0,1],$$

and can be applied to classification and regression problems with convex loss functions. The author uses simulation studies to compare ASGD with the recommended learning rate schedule to standard SGD and ASGD with the typical decaying step size (e.g. $1/(1+t)^{(-1/2)}$) schedule. The studies show the effect of the learning schedule on the convergence of ASGD and reveal the superiority of ASGD implemented with the proposed learning rate over the other methods. Experiments using real-world datasets show that ASGD with the recommended learning rate outperforms standard SGD and other variants of SGD including Pegasos, SGD-QN, and the limited memory version of oBFGS by Schraudolph et al. (2007), as well as LIBLINEAR, the dual coordinate descent method by Fan et al. (2008). Compared

[1]The authors found that using the learning rate schedule, $\epsilon_t = \epsilon_0 \alpha^{-t/n}$ yielded better results in their experiments than the typical choice of $\epsilon_t = \epsilon_0/(1 + t/n)$.

to these other methods, ASGD with the proposed learning rate attains good accuracy with respect to test error after one pass of the data set in almost all the results presented.

Bach (2014) leverages the self-concordance[2] properties of generalized linear models to show that the convergence rate of the ASGD method improves from the usual $O(1/\sqrt{T})$ to $O(R^2/\mu T)$, where R is a uniform bound on the features and μ is the smallest eigenvalue of the Hessian at the global minimum, which need not be known in advance. The analysis is conducted in the online setting and focuses on the logistic loss function but can be extended to all generalized linear models with uniformly bounded features. The author shows that AGSD when applied to GLMs naturally adapts to local strong convexity since the stepsize utilized is proportional to $1/R^2\sqrt{n}$ and does not depend on the constant μ.

Toulis et al. (2014) develop an algorithm for the efficient computation of implicit SGD updates for GLMs. The updates are called implicit because the next iterate shows up in both sides of the update equation. Specifically, the implicit updates are given by

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \epsilon_t \left\{ y_{S(t)} - h\left(\hat{\theta}_{t+1}^T \mathbf{x}_{S(t)}\right)\right\}\mathbf{x}_{S(t)},$$

where h is the inverse of the link function of the GLM, $\mathbf{x}$ is the feature vector and y is the response value. The implicit SGD update can be likened to a stochastic form of the proximal gradient method since the update can be expressed in the following way:

$$\hat{\theta}_{t+1} = \underset{\theta}{\operatorname{argmin}} \left\{ f\left(\theta; Z_{S(t)}\right) + \frac{1}{2\epsilon_t}||\theta - \hat{\theta}_t||^2 \right\}.$$

Implicit SGD normalizes the learning rate with the norm of the input which makes it more numerically stable than its explicit counterpart, standard SGD. The authors propose a method for efficiently computing implicit updates that reduces the multi-dimensional implicit equation to a one-dimensional one by exploiting the structure of the gradient of the log-likelihood functions of GLMs. They derive analytical formulas for the bias and variance of both the standard and implicit SGD estimators in the context of online learning where the data are assumed to come from a known exponential family distribution. They show that the asymptotic properties of both the explicit and implicit SGD estimates are identical with respect to bias, variance, and convergence rate. They also observe through empirical studies that though the implicit method has more biased estimates for small samples than

[2]Generalized self-concordance of f (Bach et al. 2010): Suppose f is a function defined on a Hilbert space $\mathcal{H}$. $\forall \theta_1, \theta_2 \in \mathcal{H}$, the function $\psi : t \mapsto f[\theta_1 + t(\theta_2 - \theta_1)]$ satisfies: $\forall t \in \mathbb{R}, |\psi'''(t)| \leq R||\theta_1 - \theta_2||\psi''(t)$.

the standard method, the implicit method has smaller variance and is more robust to poor choices of the learning rate. In an ensuing paper, Toulis et al. (2016) propose averaging of the implicit SGD iterates to increase statistical efficiency. They show that the resulting estimator achieves unbiasedness under strong convexity.

Cohen et al. (2017) consider the problem of least squares regression under the streaming data setting. They propose a variant of SGD called Projected SGD with Weighted Averaging (PSGD-WA) that involves a projection of unconstrained SGD iterates onto a constraint set followed by a weighted average of all iterates up to the current iteration. The authors assume that the researcher has prior knowledge that the optimal parameter lies in the interior of some subset of $\mathbb{R}^d$, say Θ. The weights are assigned such that the most recent iterates have greater weights and the weights sum up to one. The next iterate of the projected parameter estimate is obtained by

$$\hat{\theta}_{t+1} = \underset{\theta \in \Theta}{\operatorname{argmin}} \left\{ \epsilon_t \nabla_\theta f\left(\hat{\theta}_t; Z_{S(t)}\right)^T \left(\theta - \hat{\theta}\right) + \frac{1}{2} \left\| \theta - \hat{\theta}_t \right\|^2 \right\},$$

after which the weighted average can be computed recursively using

$$\bar{\theta}_t = \frac{W_{t-1}}{W_t} \bar{\theta}_t + \left(1 - \frac{W_{t-1}}{W_t}\right) \hat{\theta}_t,$$

where $W_t = W_{t-1} + 1/\alpha_t$, $W_{t-1} = \sum_{r=0}^{t-1} 1/\alpha_r$ and α_r is a decreasing sequence in r. The authors provide an explicit $O(1/T)$ upper bound on the convergence rate of the weighted average parameter estimate that depends on the variance of the noise in the observations and the size of the constraint set. The method is applicable to unconstrained least squares regression, in which case, the iterate update reverts to the standard SGD update, after which the weighted averaging is applied. The authors carry out experiments on both synthetic and real data under the linear regression setup to compare PSGD-WA with a projected version of the standard SGD and a projected version of the unweighted averaging SGD method proposed by Défossez and Bach (2015). Their experiments suggest that PSGD-WA outperforms the aforementioned SGD variants even under static data settings.

3.2.2 Support Vector Machines

Support Vector Machines (SVM) are a popular tool for tackling classification and regression problems. When applied to classification tasks, the goal of the SVM model is to estimate a hyperplane that attains the maximum distance between data points of any class that are closest to each other. The regression analogue consists of finding a predictive model such that for each training example, the deviation between the observed response and the predicted is at most some tolerance ε, while ensuring that the function is as flat as possible. Below are a number of ways in which SGD has been applied with SVM.

Zhang (2004) analyzes the finite-sample convergence behavior of SGD algorithms when applied to regularized and unregularized linear prediction problems. He explores the effects of early stopping and averaging on the learning performance of SGD and ASGD using convergence bounds and empirical studies. Consequently, the author posits that the standard SGD estimate can be superior to its averaged counterpart in some cases because it requires fewer iterations over the data to achieve a pre-specified level of accuracy. Furthermore, the desired accuracy can be attained with a sufficiently small constant learning rate. The studies also reveal that the performance of SGD under both regularized and unregularized settings are comparable. This suggests that the application of SGD to unregularized objective functions has an implicit regularization effect. In particular, the author's analysis points to the fact that multiple passes through the dataset serves as a form of bias-variance trade-off that results in an implicit form of regularization parameterized by the total number of iterations, T. Thus, the choice of T determines the balance between bias and variance that yields optimal results. Text classification experiments involving large high-dimensional datasets with SVM loss functions are used to compare ASGD and SGD to perceptron-based methods that have proved successful in training natural language models. Both standard and averaged SGD methods outperform their perceptron-based analogues.

Shalev-Shwartz et al. (2011) develop a stochastic subgradient-based algorithm for optimizing the objective function cast by SVM. They show that their method, Pegasos (Primal Estimated sub-GrAdient SOlver for SVM), achieves an ε-accurate solution in $\tilde{O}(1/\lambda\varepsilon)$ iterations (where λ is the regularization parameter), which is an order of magnitude better than other SVM learning methods. The convergence of the method is independent of the number of data points and as such is well-suited to large-scale text classification problems. The authors show that their method can be easily adapted to incorporate the use of non-linear kernels without resorting to optimizing the dual of the objective function. However,

this extension leads to an increase in the runtime of the algorithm that depends on the number of observations. Comparisons of Pegasos with other SVM algorithms, notably SVM-Light (Joachims 1999), SVM-Perf (Joachims 2006), LASVM (Bordes et al. 2005), and SDCA (Hsieh et al. 2008), show that in most of the classification experiments conducted by the authors, Pegasos outperforms the others with respect to runtime, primal suboptimality, and test classification error when implemented with linear kernels. However, when used with the Gaussian kernel, though Pegasos performs appreciably well, it does not outdo other state-of-the-art SVM methods. Additional advantages of the algorithm include its ability to handle other convex loss functions and the fact that it uses a predefined stepsize schedule, $1/(\lambda t)$, which renders the search for a suitable learning rate unnecessary. It is worth noting that very small values of λ result in long runtimes of the algorithm.

3.2.3 Neural Networks

Neural networks are mathematical models that are used to approximate a wide array of functions and can be represented by a system of interconnected units. The idea of neural networks was formalized in the seminal paper by McCulloch and Pitts (1943). Their goal was to mimic the ability of the human brain to learn tasks by looking at several examples of data. The most basic form of a neural network is the perceptron. It is a model that takes in inputs and returns a binary output depending on the sign of the weighted sum of the inputs. The weights are real numbered values that represent the importance of the inputs in determining the output. Neural networks are characterized by their architecture, which refers to the number of layers between the input layer and the output layer, as well as the number of nodes in each layer. Each connection between nodes is associated with a real-valued number called a weight. A single-layer perceptron consists only of an input layer and an output layer. There is no layer of nodes between the input and output layers. In general, the input layer is not counted as one of the layers of the network, hence the name single-layer perceptron although it is technically composed of two layers. Neural networks are trained by minimizing some specified loss function to obtain the optimal weights that result in a pre-determined accuracy.

The most common type of neural network is the feed-forward network, which consists of a directed string of functions of weighted inputs transferred from one node to another. Feed-forward neural networks, also referred to as multi-layer perceptrons (MLPs), are a generalization of the single-layer perceptron because they have one or more layers of nodes

between the input and output layers known as hidden layers. Backpropagation refers to the method used to compute the gradient of the loss function. It is so-called because it involves applying the chain rule and iterating backward through the layers of the network from the output layer to the input layer to obtain the derivatives with respect to each weight. The weights are then updated using some gradient-based optimization method. This is repeated until the desired level of accuracy is achieved. Neural networks can be applied to a wide range of problems, including speech recognition and computer vision, as well as classical regression and classification problems.

In their review paper on neural networks, Cheng and Titterington (1994) provide an in-depth analysis of neural networks from a statistical viewpoint. They draw parallels between several statistical methods and their equivalent methods in the neural network community. They also elaborate on the various ANN architectures that are applied in different problems and tie them to their statistical analogues. For example, they provide neural network representations of classical statistical approaches to regression, discriminant analysis, and clustering. Sarle (1994) also compares statistical models to their equivalent forms in the neural network literature and provides an extensive list that maps statistical terminology to neural network jargon.

Currently, the most popular methods for training neural networks are variants of SGD, typically RMSProp (Hinton et al. 2012) and Adam (Kingma and Ba 2014). RMSProp was derived from RProp (Resilient Propagation) (Riedmiller and Braun 1993), which is a batch learning method that uses the sign of the gradient to make individual updates to each weight. This obviates the need for a global learning rate and makes it a very fast training method. However, the main disadvantage of RProp is that it cannot easily be adapted to mini-batches as it does not take into account variations in mini-batches. RMSProp is a method that leverages the ability of RProp to learn quickly, with the added advantage that it can be applied to mini-batches.

Adam is a method derived from the amalgamation of the desirable properties of AdaGrad (Duchi et al. 2011) and RMSProp. AdaGrad is well-suited to learning with sparse feature vectors, which are prevalent in NLP tasks. AdaGrad shares some similarities with the Adaptive Regularization of Weights (AROW) algorithm (Crammer et al. 2009), which is, in turn, related to the Confidence-Weighted (CW) learning algorithm (Dredze et al. 2008).

The CW algorithm is a linear classification algorithm that maintains a probabilistic measure of confidence for each parameter associated with a feature. These confidence

parameters are governed by a Gaussian distribution with some mean vector and covariance matrix. The means encapsulate the learner's knowledge about the parameter of each feature, and the variances are estimates of the confidence in each parameter. The means and variances are updated with each observation that is received, such that the probability of correctly classifying that data point under the updated distribution meets some pre-specified value, usually greater than 0.5. Parameters associated with low confidence values are updated more aggressively than those with higher confidence values. The AROW algorithm reformulates the constrained objective under CW learning into an unconstrained problem by regularizing the constraint. The effect is an algorithm that is less aggressive in parameter updates and more robust to label noise[3]. AROW, unlike the CW method, can be adapted to handle other problems such as regression.

Since their conception, there have been many efforts to speed up the training of neural networks. Stochastic Meta Descent (SMD) (Schraudolph 1999) is one such algorithm developed to improve the speed of convergence of neural networks. It belongs to the class of gain adaptation algorithms, which generally adapt learning rates by keeping track of consecutive gradients. SMD is an extension of Sutton's work (Sutton 1992) on linear systems to the non-linear case, and provides a more general framework for many existing local gain adaptation algorithms. The SMD algorithm makes parameter updates using

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \eta_t \odot \nabla_\theta f\big(\hat{\theta}_t, Z_{S(t)}\big).$$

The step size or gain vector, η_t is adapted in the following way:

$$\eta_t = \eta_{t-1} \odot \max\left(\frac{1}{2}, 1 - \mu \nabla_\theta f\big(\hat{\theta}_t, Z_{S(t)}\big) \odot v_t\right),$$

where $\mu \geq 0$ is a scalar tuning parameter, the max is an element-wise operation, and the vector v is obtained iteratively using

$$\hat{v}_{t+1} = \lambda \hat{v}_t - \eta_t \odot \big[\nabla f\big(\hat{\theta}_t, Z_{S(t)}\big) + \lambda H_t \hat{v}_t\big],$$

where $0 \leq \lambda \leq 1$ is also a scalar tuning parameter, and $H_t = \nabla^2_\theta f\big(\hat{\theta}_t; Z_{S(t)}\big)$ is the instantaneous Hessian at the tth iteration. The exponential averaging of past gradients is obtained

[3]Label noise refers to errors in the labels that arise due to a number of reasons such as insufficient information required to correctly classify data and encoding mistakes.

when $H_t = \lambda \mathbf{I}$. Basically, the SMD method assigns individual step sizes to each component of the parameter vector, thus accelerating convergence.

Another method developed to accelerate convergence of neural networks is the batch normalization transform by Ioffe and Szegedy (2015). Batch normalization transforms the inputs of any layer by centering and scaling using the mean and the inverse of the standard deviation of each mini-batch, respectively. This method was introduced to address the internal covariate shift problem that is inherent in deep layer neural networks. In deep networks, a small change in the parameters results in a greater change in inputs because the inputs to each layer are influenced by the parameters of all preceding layers. Thus, the distribution of the inputs to each layer changes during the course of training. This phenomenon is referred to as the internal covariate shift. Batch normalization fixes the mean and variance of each layer which speeds up the training of the network.

Weight normalization (Salimans and Kingma 2016) is a method inspired by batch normalization that reparameterizes the weights of the network to speed up the optimization process. Weight normalization separates the magnitude of the weight vector from its direction while projecting the gradient away from the current weight. Specifically, the reparameterization is of the form,

$$\mathbf{w} = \frac{g}{||\mathbf{v}||}\mathbf{v},$$

where $\mathbf{w}$ is a p-dimensional vector of the original weights, $\mathbf{v}$ is a p-dimensional vector, g is a scalar, and $||.||$ is the Euclidean norm. This improves the conditioning of the objective function and increases the speed of convergence. The model is then trained using SGD methods on the new parameters $\mathbf{v}$ and g. Weight normalization is independent of the observations within mini-batches and more robust to the choice of the learning rate.

Loshchilov and Hutter (2016) propose another way of speeding up training of deep neural networks referred to as SGD with warm restarts. This technique involves implementing multiple runs of training, where each run comprises T_k epochs (k indexes the run). Within each run, the learning rate is decayed according to the following schedule:

$$\epsilon_t = \epsilon^k_{\min} + \frac{1}{2}\left(\epsilon^k_{\max} - \epsilon^k_{\min}\right)\left(1 + \cos\left(\frac{T_{\text{cur}}}{T_k}\pi\right)\right),$$

where $\epsilon^k_{\min}$ and $\epsilon^k_{\max}$ are ranges of the learning rate, and T_{cur} tracks the number of epochs that have been executed since the last restart. At the start of a new run, the warm restart

is implemented by increasing the learning rate ϵ_t and using the last iterate $\hat{\theta}_t$ from the previous run as the starting value of θ. The authors suggest that a small value of T_k be used at the beginning and gradually incremented by a fixed factor at each restart.

Additional work on accelerating convergence in neural network training can be found in Schwenk and Bengio (2000), Johnson and Zhang (2013), and De et al. (2016).

3.3 Reinforcement Learning

Reinforcement Learning (RL) can be described as learning the best actions to take to achieve a goal by interacting with an environment. The goal of most RL tasks is to maximize the long-term cumulative expected reward. The agent is the learner and decision maker. The environment is everything outside of the agent. Typically, the agent has no control over the environment. The reward is a numerical value that defines the goal of the RL task. The agent selects actions in a given state of the environment, receives a reward, and then the environment presents new states to the agent. An important feature of RL is the trade-off between exploitation and exploration that arises as a result of trying to maximize the expected total reward. Specifically, the agent has to exploit previously selected actions that have yielded good results but it must also explore the actions available in order to ensure that optimal actions are not overlooked. The sequential decision problem cast by RL is formalized using a *Markov decision process (MDP)*, described by a tuple (S, A, T, r, γ).

- S denotes the set of states that the environment can be in;
- A denotes the set of actions available to the agent;
- $T : S \times A \times S \to \mathbb{R}_+$ is a transition function which gives the probability distribution of the next state given the current state and action;
- $r : S \times A \to \mathbb{R}$ is the reward function;
- γ is a discount rate.

Usually, the goal in RL tasks is to maximize the sum of future discounted rewards, defined by

$$G_t := \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$$

Most solutions to RL problems involve finding an optimal policy. A policy is a mapping from states to actions available to the agent when in those states and is given by

$$\pi : S \rightarrow A.$$

An optimal policy is a policy that is at least as good as all the others. Value functions facilitate the task of finding an optimal policy. A value function is the total amount of reward that an agent can expect to accumulate in the long term given that he started from a particular state (state-value function) or given that he started from a specific state and took a specific action (action-value function). The state-value function under a policy, π is defined as

$$V_\pi(s) := \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left\{\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s\right\}.$$

The action-value function under a policy, π is given by

$$Q_\pi(s,a) := \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left\{\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s, A_t = a\right\}.$$

A policy π is said to be better than another policy π' if and only if $V_\pi(s) \geq V_{\pi'}(s),\ \forall s \in S$. Optimal policies share the same optimal state-value function or action-value function.

The application of SA in RL is evident in several of the approaches that are used to solve RL tasks. In particular, SA is used in the updates to the value functions and policy parameters, as well as in the convergence proofs of RL algorithms.

RL methods for finding the optimal policy can be grouped into two main categories, namely model-based and model-free methods. Model-based approaches are methods that involve the estimation of a model of the environment. Model-free approaches involve learning via trial and error. Two popular classes of methods that are used for solving RL problems are Monte Carlo (MC) methods and Temporal Difference (TD) learning algorithms.

MC methods are model-free methods that learn using sequences of states, actions, and rewards obtained from interacting with the environment. MC methods estimate the value of a state under a policy by averaging the returns following the first visit to the state or every visit to the state. These methods are specifically applied to episodic tasks as the value functions are updated at the end of each episode (a subsequence of states, actions, and rewards that end in a specific state). The general form of an MC method is outlined below.

- Given episodic data consisting of $S_1, A_1, R_1, S_2 ...., R_T$,
- For each state S_t with return G_t,
  i. $N(S_t) \leftarrow N(S_t) + 1$,
  ii. $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t)}\Big\{G_t - Q(S_t, A_t)\Big\}$,

where $N(S_t)$ counts the number of times that S is visited in the episode (for the case of every-visit MC).

TD methods, like MC methods, learn from experience and do not require a model of the environment. These methods update estimates based on learned estimates. Unlike MC methods, updates are made at each time step. The simplest TD learning algorithm, called TD(0), makes updates of the form

$$V(S_t) \leftarrow V(S_t) + \alpha\{R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\}.$$

$R_{t+1} + \gamma V(S_{t+1})$ is called the target and $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the TD error.

For environments with a large number of states and/or actions, it is expedient to represent the value function as a function parameterized by some weight vector, w instead of in a tabular form. Specifically, the value function $V_\pi(S)$ is approximated by $\hat{V}_\pi(S, w)$.

A special case of function approximation is where $V(.; w)$ is a linear function of the weight vector. The approximate value function is given by

$$\hat{V}(S, w) = w^T \boldsymbol{\phi}(S),$$

where $\boldsymbol{\phi}(S)$ is a vector of features that characterize state S.

SGD methods are commonly used to learn the approximate value function by finding the optimal parameter vector and are appropriate for online tasks. A general SGD update of the parameter vector of the value function is given by

$$\hat{w}_{t+1} = \hat{w}_t + \epsilon\Big[U_t - \hat{V}_\pi(S_t, \hat{w}_t)\Big]\nabla_w \hat{V}_\pi(S_t, \hat{w}_t),$$

where U_t is some noisy version of $V_\pi(S_t)$. For example, when the approximate value function is linear in the weights, the SGD update of the weight vector is of the form

$$\hat{w}_{t+1} = \hat{w}_t + \epsilon\Big[U_t - \hat{V}_\pi(S_t, \hat{w}_t)\Big]\boldsymbol{\phi}(S_t).$$

One very widely used TD method for solving RL problems is Q-Learning. Q-Learning is a model-free method that estimates the optimal action-value function, Q_*, irrespective of the policy being followed. The procedural form of Q-Learning at each time step is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big[R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\big].$$

Given a differentiable parametric form of the value function, $Q(.; w)$ we can obtain an estimating equation for the weight vector w, of the value function using the Bellman equations as outlined below.

$$\begin{aligned}
&\mathbb{E}\big[R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a'; w) | S_t = s, A_t = a\big] = Q(s, a; w) \\
\implies &\mathbb{E}\big[R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a'; w) - Q(s, a; w) | S_t = s, A_t = a\big] = 0 \\
\implies &\mathbb{E}\big[\{R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a'; w) - Q(s, a; w)\}\nabla_w Q(s, a; w) | S_t = s, A_t = a\big] = 0 \\
\implies &\mathbb{E}\big[\mathbb{E}\big[\{R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a'; w) - Q(s, a; w)\}\nabla_w Q(s, a; w) | S_t = s, A_t = a\big]\big] = 0 \\
\implies &\mathbb{E}\big[\{R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(s, a)\}\nabla_w Q(s, a; w)\big] = 0,
\end{aligned}$$

where $\nabla_w Q(s, a; w)$ is the derivative of the action-value function with respect to its parameter vector w.

Thus, for a sequence of states, actions, and rewards obtained under policy π, indexed from $1, ..., T$, an estimating equation for the parameter vector, w is

$$P_n\left[\sum_{t=1}^{T}\Big\{R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a'; w) - Q(s, a; w)\Big\}\nabla_w Q(s, a; w)\right] = 0.$$

Policy Gradient methods are another class of methods for finding the optimal policy of an RL problem. This class of methods learn directly a policy parameterized by some vector θ and do not require a value function to select actions. Specifically, the policy is denoted by $\pi_\theta(a|s)$ which is the probability of choosing action a given that the agent is in state s with parameter vector θ. Policy gradient methods aim at maximizing some performance metric, $J(\theta)$ with respect to the policy parameter. For example, for episodic tasks, a logical performance measure is the value function of the starting state S_0, defined by

$$J(\theta) := V_{\pi_\theta}(S_0).$$

In general, the stochastic gradient updates to the parameter vector of the policy are of the form

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \epsilon \nabla_\theta J(\hat{\theta}),$$

where $\nabla_\theta J(\theta)$ is the gradient of the performance measure with respect to θ.

Tsitsiklis (1994) extends and applies the classical theory of SA to prove the convergence of Q-Learning under more general assumptions than those made in the proof by Watkins and Dayan (1992). The author develops this proof for the case of undiscounted problems where you do not assume that all policies must lead to a zero-cost absorbing state. The proof involves a combination of SA theory and the convergence results on asynchronous parallel algorithms by Bertsekas (1982), and Bertsekas and Tsitsiklis (1989). Jaakkola et al. (1994) independently develop a rigorous proof of the convergence of a class of convergent iterative algorithms including the Q-Learning and TD(λ) algorithms which exploits the relationship between dynamic programming[4] and SA theory. Sutton and Barto (2018) provide an extensive and comprehensible summary of concepts in RL.

[4]Dynamic programming refers to a class of algorithms for solving RL problems that assumes a perfect model of the environment is available in the form of an MDP.

CHAPTER

4

HYPER-PARAMETER TUNING USING THE DOUBLE BOOTSTRAP AND SPSA

Due to early stopping, using SGD usually results in estimates that are not exactly equal to the true minimizer of the empirical loss function. The difference between the SGD estimator and the true minimizer depends on the observed data, the tuning parameters of the SGD method, and the stopping criterion. We use a simulation study to illustrate the impact of the choice of the step size schedule on the accuracy of SGD estimates. The goal is to compute a confidence region with a specified coverage probability in the context of linear regression. The covariance matrix of the parameter vector is computed using an adaptation of the plug-in method by Chen et al. (2016).

The coverage probability relies largely on how good an estimate of the parameter vector is obtained. The speed of convergence of the SGD iterates to the desired optimum and the accuracy of the resulting parameter estimate depend on the choice of step size schedule and the pre-specified convergence tolerance. The heatmaps in Figures 4.1 and 4.2 show the initial coverage probabilities associated with selected values of t_0 and t_1 under a squared

error loss function and a design matrix with an AR1 covariance structure with different correlation coefficients. We observe that bad choices of these tuning parameters can result in poor coverage. It is worth pointing out that when one is going to employ SGD to optimize a function, there's no guidance as to how to choose these hyper-parameters in order to get good coverage. That is, the same choice of t_0 and t_1 are unlikely to yield the same coverage for two different problems as is evident in the heatmaps. It can be observed that for the same class of models, i.e. the linear model, the same values of t_0 and t_1 yield different coverage probabilities depending on the strength of the correlations among the covariates. This emphasizes that t_0 and t_1 are largely problem-dependent.

To determine the right choice of the parameters of the step size function required to attain a coverage probability that is as close to the nominal confidence level as possible, we employ the double bootstrap method in conjunction with the Simultaneous Perturbation Stochastic Approximation (SPSA) method (Spall 1998) to adapt the hyper-parameters, t_0 and t_1, with a pre-specified tolerance, τ. One variation of this proposed technique is to update both parameters of the step size schedule simultaneously. Another way is to keep either t_0 or t_1 fixed while updating the other. In both variations, one instance of a Monte Carlo simulation consists of a repetition of the double bootstrap while adapting the hyper-parameters with SPSA until such time as the actual coverage probability is at least as large as the nominal confidence level.

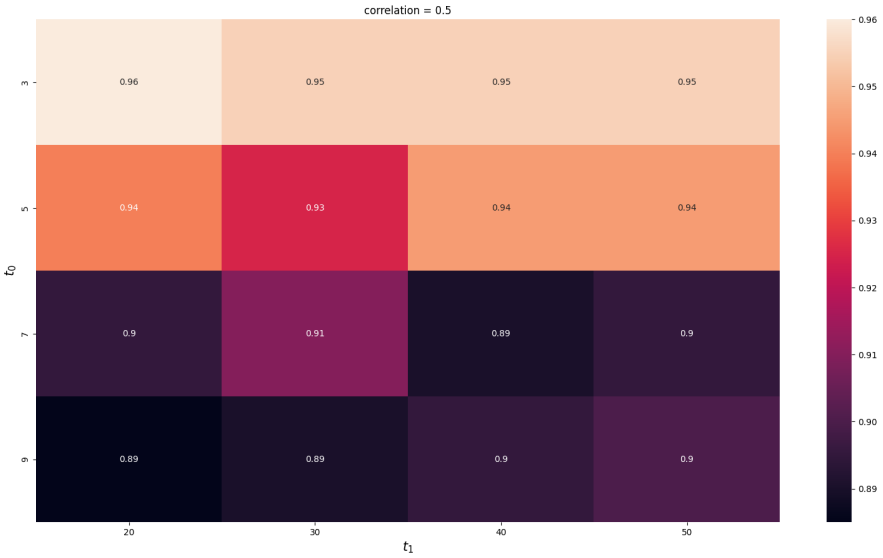

Figure 4.1: Heat map of initial coverage probabilities under a squared error loss function and a design matrix with an AR1 covariance structure with a correlation coefficient of 0.5.

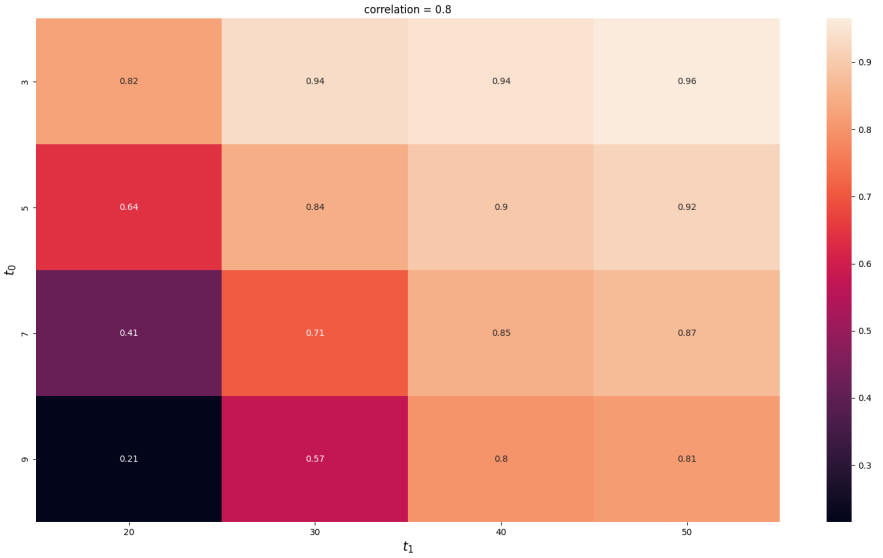

Figure 4.2: Heat map of initial coverage probabilities under a squared error loss function and a design matrix with an AR1 covariance structure with a correlation coefficient of 0.8.

4.1 Outline of the Adaptive Method

We provide a basic description of the fully adaptive tuning method below.

Consider a dataset denoted by D_{orig} consisting of a design matrix $\mathbf{X}$ and a response vector $\mathbf{Y}$, where $\mathbf{Y}$ is generated according to $\mathbf{Y} = \mathbf{X}\theta^* + \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Thus, D_{orig} consists of $\{\mathbf{x}_i, Y_i\}_{i=1}^n$.

(1) Set t_0 and t_1 to initial values.

(2) Run one instance of SGD on D_{orig} and obtain $\hat{\theta}_{SGD}$ and the vector of residuals, $\mathbf{e}_{SGD} = \mathbf{Y} - \mathbf{X}\hat{\theta}_{SGD}$.

(3) Initialize P_B to 0. While $P_B <$ nominal coverage:

(I) For $b = 1, ..., B$:

(A) Generate dataset, D_b according to $Y_{b,i} = \mathbf{x}_i^T \hat{\theta}_{SGD} + e_{b,i}$, where $e_{b,i}$ is sampled with replacement from the components of $\mathbf{e}_{SGD}$.

(B) Run SGD on $D_b\{\mathbf{x}_i, Y_{b,i}\}_{i=1}^n$ to obtain $\hat{\theta}_b$ and $\mathbf{e}_{BOOT} = \mathbf{Y}_b - \mathbf{X}\hat{\theta}_b$

(C) For $c = 1, .., C$:

(i) Generate dataset, D_c according to $Y_{c,i} = \mathbf{x}_i^T \hat{\theta}_b + e_{c,i}$, where $e_{c,i}$ is sampled with replacement from the components of $\mathbf{e}_{BOOT}$.

(ii) Run SGD on $D_c\{\mathbf{x}_i, Y_{c,i}\}_{i=1}^n$ to obtain $\hat{\theta}_c$.

(iii) Compute $I_c = \mathbb{1}\{(\hat{\theta}_c - \hat{\theta}_b)^T \hat{\mathbf{V}}_c^{-1} (\hat{\theta}_c - \hat{\theta}_b) \leq p F_{1-\alpha}(p, n-p)\}$, where $\hat{\mathbf{V}}_c = (n-p)^{-1} \sum_{i=1}^n (Y_i^c - \mathbf{x}_i^T \hat{\theta}_c)^2 (\mathbf{X}^T\mathbf{X})^{-1}$, p is the number of parameters, n is the number of observations, and $F_{1-\alpha}$ is the $(1-\alpha)$ percentile of the F-distribution.

(D) Compute $P_C = C^{-1} \sum_{c=1}^C I_c$

(II) Compute $P_B = B^{-1} \sum_{b=1}^B P_C$

(III) Update t_0 and t_1 using the SPSA procedure and repeat Step (3) starting from (I).

We use the SPSA algorithm because it is a gradient estimation scheme that requires only two function evaluations at perturbed parameter values to estimate the gradient of the coverage generating function whose analytic form is unknown, unlike its counterpart, the K-W algorithm, which requires $2p$ function evaluations. We incorporate the double bootstrap because, in practice, we do not know the true parameter θ^*, so instead, we take

$\hat{\theta}_{SGD}$ to be the truth. We then consider the behavior of $\hat{\theta}_c - \hat{\theta}_b$ in the bootstrap world, which mirrors that of $\hat{\theta}_{SGD} - \theta^*$ in the real world.

4.2 Simulation Study

For the simulation study, the design matrix is simulated from a multivariate normal distribution under an AR1 covariance structure with a correlation coefficient of 0.8. The data consists of 200 observations and 9 variables. The inclusion of an intercept results in a 10-dimensional parameter vector. The true parameter vector, θ^* is an array of evenly spaced values over the interval -2 to 2. The values t_0 and t_1 are first initialized to some positive values. Each iteration of the Monte Carlo simulation begins with generating an initial dataset, D_{orig} according to $Y_{(200\times1)} = \mathbf{X}_{(200\times10)}\theta^*_{(10\times1)} + \boldsymbol{\varepsilon}_{200\times1}$, where $\mathbf{X} \sim MVN(\mathbf{0}, \Sigma)$ and $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \mathbf{I})$, after which one of the proposed methods is applied. The R-squared for this model is 0.96. The number of iterations in the first and second levels of the double bootstrap are 20 and 10, respectively, that is, $B = 20$ and $C = 10$. The nominal confidence level used is 0.95.

4.2.1 Results of Adaptive Method

Table 4.1 presents the results of using the fully adaptive method described above to tune t_0 and t_1 to obtain values of these hyper-parameters that will yield a coverage probability that is at least as large as the nominal or as close to it as possible. The simulation results suggest that the coverage probability is sensitive to the choice of t_0 and t_1, as is evident by the values of the initial coverage probability for the various choices of t_0 and t_1. In particular, it can be observed that in all but the first row of the table, the initial coverage probability is less than the nominal. Also, it can be observed that among the cases presented, certain combinations of initial t_0 and t_1 values yield better initial coverage probabilities than others, thus underscoring the notion that these hyper-parameters have a significant impact on estimation and inference using SGD. However, after adapting the hyper-parameters, the resulting values of t_0 and t_1 yield coverage probabilities that are not statistically significantly different from the desired nominal coverage. The McNemar test to detect a difference between the initial and final coverage probabilities is statistically significant in all the cases presented, with the exception of the case of $t_0 = 3$ and $t_1 = 40$, where the initial coverage probability already surpasses the desired coverage probability.

Table 4.1: Results of hyper-parameter tuning using fully adaptive SPSA method with the double bootstrap. The results presented are averaged over 200 simulated datasets. Numerical values in parentheses are standard errors of the estimates. "Sig." refers to a statistically significant result. The McNemar test statistic is compared to $\chi^2_{0.95,1} = 3.84$. The stopping criterion used for the SGD method is $||\hat{\theta}_t - \hat{\theta}_{t-1}||^2/||\hat{\theta}_{t-1}||^2 < \tau$, where τ is set to 10^{-7}.

Initial t_0	Initial t_1	Initial Coverage	Final t_0	Final t_1	Final Coverage	McNemar Test Stat.	Avg. Time (mins.)
3	40	0.96 (0.01)	2.87 (0.03)	40.07 (0.03)	0.98 (0.01)	1.29 (Not Sig.)	4
5	40	0.90 (0.02)	3.34 (0.09)	41.57 (0.26)	0.96 (0.01)	8.89 (Sig.)	11
7	40	0.89 (0.02)	4.97 (0.1)	41.51 (0.22)	0.95 (0.02)	8.05 (Sig.)	57
9	40	0.85 (0.03)	3.47 (0.13)	45.98 (0.62)	0.95 (0.02)	13.33 (Sig.)	112
11	40	0.80 (0.03)	3.72 (0.15)	49.94 (1.03)	0.95 (0.02)	24.64 (Sig.)	176
13	40	0.66 (0.03)	4.57 (0.22)	54.88 (1.20)	0.94 (0.02)	52.27 (Sig.)	294
15	40	0.51 (0.04)	5.66 (0.26)	59.09 (1.28)	0.96 (0.01)	88.04 (Sig.)	490

4.2.2 Results of Mixture Method

Table 4.2 presents the results of using the mixture method to tune t_0 to obtain values that will yield a coverage probability that is at least as large as the nominal or as close to it as possible. Specifically, we keep t_1 fixed while adapting t_0. Similar to the fully adaptive case, it can be observed that in all but the first row of the table, the initial coverage probability is less than the nominal. However, after adapting t_0, the resulting values of t_0 combined with the fixed value of t_1 yield coverage probabilities that are not statistically significantly different from the desired nominal coverage. The McNemar test is statistically significant in all but the first row.

4.3 Concluding Remarks

We have presented a collection of stochastic gradient descent methods and their applications in some areas of Statistics. Though SGD is extensively used with a great deal of success, it can be very erratic and requires a significant amount of hyper-parameter tuning to obtain desirable results.

It is worth noting that though the fully adaptive and mixture methods proposed for hyper-parameter tuning result in final values of the hyper-parameters that yield coverage probabilities that are close to the nominal, it appears that the fully adaptive method takes a relatively shorter time as compared to the mixture method. This is particularly evident in the cases where the coverage associated with initial values of t_0 and t_1 is quite poor.

In spite of their shortcomings, SGD and its variants have provided a much-needed solution to the problem of optimization in this era of high-volume and streaming data.

Table 4.2: Results of hyper-parameter tuning using mixture version of adaptive SPSA method. The results presented are averaged over 200 simulated datasets. Numerical values in parentheses are standard errors of the estimates. "Sig." refers to a statistically significant result. The McNemar test statistic is compared to $\chi^2_{0.95,1} = 3.84$. The stopping criterion used for the SGD method is $||\hat{\theta}_t - \hat{\theta}_{t-1}||^2/||\hat{\theta}_{t-1}||^2 < \tau$, where τ is set to 10^{-7}.

Initial t_0	Initial t_1	Initial Coverage	Final t_0	Final t_1	Final Coverage	McNemar Test Stat.	Avg. Time (mins.)
3	40	0.96 (0.01)	2.89 (0.04)	40	0.98 (0.01)	0.14 (Not Sig.)	8
5	40	0.90 (0.02)	3.95 (0.08)	40	0.95 (0.02)	4.55 (Sig.)	85
7	40	0.89 (0.02)	4.32 (0.10)	40	0.95 (0.02)	6.0 (Sig.)	109
9	40	0.85 (0.03)	3.55 (0.11)	40	0.95 (0.02)	15.38 (Sig.)	465
11	40	0.80 (0.03)	4.29 (0.12)	40	0.98 (0.01)	15.38 (Sig.)	996
13	40	0.66 (0.03)	4.77 (0.14)	40	0.97 (0.01)	52.27 (Sig.)	2478

CHAPTER

5

OPTIMAL EXPERIMENTAL DESIGNS FOR PRECISION MEDICINE WITH MULTI-COMPONENT TREATMENTS

5.1 Introduction

Treatment decisions for complex diseases and disorders often involve a combination of multiple therapeutic options. Cancer treatment is one area in which the use of such treatments is increasingly becoming common(Garcia-Aguilar et al. 2016; Sambi et al. 2019; Philippou et al. 2020). For example, a cancer treatment could consist of a combination of chemotherapy, radiation, and drugs. In light of this, there is a need to understand better how to optimally combine treatment components to promote the patient's long-term well-being. We refer to treatments that consist of a combination of therapeutic options as multi-component treatments. We propose a method for evaluating multi-component treatments that exploits a classical optimal design strategy to maximize statistical power

for analyses that involve comparing multi-component treatments against each other and against a control. The goal is to allocate treatments to maximize power for a specified primary analysis while ensuring enough information is retained to conduct secondary analyses.

Most clinical trials are designed with multiple objectives. For example, a trial may have the objectives of comparing multiple treatments to a placebo, and at the same time, minimizing the exposure of patients to the less efficacious treatments. Randomization, a crucial component of clinical trials, prevents the intentional or unintentional assignment of patients with better prognoses or higher recovery rates to the more effective treatment (Ning and Huang 2010; Rosenberger et al. 2012). Balanced randomization, which ensures that a roughly equal number of participants are assigned to each treatment, is a popular mode of randomization in clinical trials. A common criticism of this method is that it does not guarantee that most of the participants are treated effectively (Berry and Eick 1995). This disadvantage precipitated the initial study of adaptive procedures for treatment assignment in clinical trials as an alternative to conventional methods (Thompson 1933; Feldman 1962; Zelen 1969; Sobel and Weiss 1971). A comparative study of balanced randomization and adaptive randomization can be found in Berry and Eick (1995). Rosenberger et al. (2012) review several adaptive randomization procedures for clinical trials. Rosenberger et al. (2001) propose an adaptive design for the optimal allocation of treatments that minimizes the expected number of treatment failures for a fixed variance of a test statistic in a binary response clinical trial. One type of adaptive randomization that is related to our strategy is Covariate-adaptive (CA) randomization.

It is widely acknowledged that improving balance within important covariates across different treatment groups is advantageous because it potentially increases the statistical power of a clinical trial (Kernan et al. 1999; Weir and Lees 2003). CA randomization is a method that uses past treatment assignments and patient characteristics in determining future allocation probabilities to achieve covariate balance across treatment groups. Hu et al. (2014) review CA randomization and recent theoretical developments of the method. The most commonly used CA scheme is the minimization method introduced by Taves (1974) and extended by Pocock and Simon (1975). Our method automatically considers patient characteristics by incorporating covariate information when determining the optimal allocation of the multi-component treatments. Specifically, we optimize the treatment assignments over feature vectors that include interactions between treatment factors and covariates.

Another method related to our proposed strategy is the Multi-phase Optimization Strategy (MOST) (Collins et al. 2005). MOST is an efficient way of optimizing multi-component interventions, that draws on engineering principles to determine which components in an intervention are active and contribute to positive outcomes. It incorporates a standard randomized control trial (RCT) to evaluate the optimized intervention. MOST is composed of three phases. The first phase is the screening phase, where a randomized experiment is used to select the active components that contribute positively to patient outcomes and should be included in the intervention. The second stage of the MOST process is the refining phase. This phase involves fine-tuning the components identified during the screening phase to determine their optimal levels. The final phase, the confirming phase, consists of evaluating the efficacy of the interventions obtained in the second phase using a standard RCT. Collins et al. (2007) provide an exposition on MOSTs and SMARTs and the possibility of integrating the two. Our strategy is related to MOST in that it condenses all three phases into one step by optimally allocating multi-component treatments to patients with reference to the primary hypothesis under consideration, given patients' baseline covariates.

Optimal designs have been successfully applied in many different fields, including pharmacology and precision medicine. Optimal designs provide a principled approach to creating custom designs in situations where a standard design may be inadequate to satisfy the requirements of the problem under investigation. Early research on design optimality can be traced to the works of Smith (Smith 1918), Kiefer (Kiefer 1959, 1961), and Kiefer and Wolfowitz (Kiefer and Wolfowitz 1959). Optimal designs can be grouped into two main categories: designs that focus on optimization with respect to the regression coefficients and designs concerned with optimizing the prediction variance of the response. The designs in the first group include the A, c, D, and E optimal designs. The second group comprises G, I, and V optimal designs. A review of optimal designs can be found in Atkinson and Donev (1992) and Johnson et al. (2011). Our proposed method uses the c and D optimality criteria in constructing the composite objective function to be minimized to obtain the optimal allocation of the multi-component treatments. The c-optimality criterion minimizes the variance of a linear unbiased estimator of a specified linear combination of the unknown model parameters, while the D-optimality criterion minimizes the volume of the joint confidence region of the parameter estimates. We use a composite criterion because if we focus only on maximizing power for the primary hypotheses, we will have little or no information left to power the secondary analyses.

5.2 Setup and Notation

We focus on a single-stage RCT. The trial will generate data of the form $(\mathbf{X}_i, \mathbf{A}_i, Y_i)$, $i = 1, ..., n$, obtained from n subjects enrolled in a trial. The vector $\mathbf{X}$ denotes the subject's baseline covariate information, $\mathbf{A}$ is the option in the treatment set, $\mathscr{A}$, that is assigned at the decision point under consideration, and Y is a scalar response on each subject after the decision has been implemented.

We consider settings where the treatment option $\mathbf{A} \in \mathscr{A}$ consists of q binary components so that $\mathbf{A}$ takes values in $\mathscr{A} = \{0,1\}^q$, where 1 indicates that a component is present and 0, a component is absent. For example, for $q = 3$, a multi-component treatment vector could be $(1,0,1)$. In the case of our cancer example, this will translate into a treatment combination of the form (chemotherapy, no radiation, drugs).

We posit a model of the form $\mathbb{E}(Y|\mathbf{X} = \mathbf{x}, \mathbf{A} = \mathbf{a}) = \psi(\mathbf{x}, \mathbf{a})^T \beta^*$, where $\psi : \mathscr{X} \times \mathscr{A} \to \mathbb{R}^s$ is a feature vector that might consist of main effects and low-order interactions. Define $\phi(\mathbf{a}) := \mathbb{E}\{\psi(\mathbf{X}, \mathbf{a})\}$, the expected value of $\psi(\mathbf{X}, \mathbf{a})$ with respect to the distribution of $\mathbf{X}$ so that $\phi(\mathbf{a})^T \beta^*$ is the mean outcome under treatment $\mathbf{A} = \mathbf{a}$.

For the primary analyses, we consider the following hypotheses.

- $H_{0,1} : \{\phi(\mathbf{a}_0) - \phi(\mathbf{a}_0')\}^T \beta^* = 0$, where $\mathbf{a}_0, \mathbf{a}_0' \in \mathscr{A}$ are two fixed treatments of interest. This is a test of no difference in mean outcomes under two fixed treatments $\mathbf{a}_0$ and $\mathbf{a}_0'$.

- $H_{0,2} : \phi(\mathbf{a}_0)^T \beta^* \leq \mu_C$, where a_0 is a treatment of interest and μ_C is the mean outcome under control. This is a test that the mean outcome under some fixed treatment $\mathbf{a}_0$ is not better than the mean outcome under the control. Here, we treat μ_C as a known quantity.

- $H_{0,3} : \max_{\mathbf{a} \in \mathscr{A}} \phi(\mathbf{a})^T \beta^* \leq \mu_C$. This is a test that the mean outcome associated with the treatment that yields the best outcome among all the treatment combinations is worse than the mean outcome under the control.

For example, let a^1, a^2, a^3, a^4 be the indicators of four treatment factors whose levels can be combined so that we have a total of 2^4 possible treatment combinations of the form $\mathbf{a} = (a^1, a^2, a^3, a^4)^T$. Suppose we include a single covariate $X \in \mathbb{R}$ with $\mathbb{E}(X) = 0$ and we choose $\psi(x, \mathbf{a})$ to be a feature vector composed of all main effects and two-way interactions so that $\psi(x, \mathbf{a}) = (1, x, a^1, a^2, a^3, a^4, xa^1, xa^2, xa^3, xa^4, a^1a^2, a^1a^3, ..., a^3a^4)^T$.
Then, $\phi(\mathbf{a}) = (1, 0, a^1, a^2, a^3, a^4, 0, ..., 0, a^1a^2, a^1a^3, ..., a^3a^4)^T$. A primary analysis we may be

interested in is the comparison of a minimally intensive treatment $\mathbf{a}_0 = (1,0,0,0)^T$ and the maximally intensive treatment $\mathbf{a}_0' = (1,1,1,1)^T$. This comparison does not involve all the components of β^*, so a design that maximizes power for such a test does not require that all the components of β^* be identified. Let $\hat{\beta}_n$ denote the OLS estimator for β^*. Since the right-hand side of both $H_{0,1}$ and $H_{0,2}$ are of the form $\nu^T\beta^*$ for some fixed vector ν, we consider designs that will minimize the variance of the test statistic $\nu^T\hat{\beta}$. However, we are not only interested in maximizing power for a specific test but also in ensuring that we have sufficient information about the other multi-component treatments in the treatment set. For example, if we were to focus on maximizing power for $H_{0,2}$, a c-optimal design would result in an allocation of all subjects to $\mathbf{a}_0$, leaving no information about other treatments. Thus, we consider a criterion that consists of the c-optimality and D-optimality criteria. The c-optimality criterion is concerned with minimizing the variance of a linear combination of model parameters. Most test statistics associated with the tests to be considered consist of a linear combination of the model parameters. Thus, minimizing the c-optimality criterion is equivalent to minimizing the variance of the test statistic and will result in higher statistical power. The D-optimality criterion is concerned with controlling the volume of the joint confidence region of the estimates of the model parameters. Minimizing this volume results in more precise parameter estimates. Given ν, we obtain our optimal design $\mathbb{D}_c^{\lambda}(\nu) = \{\mathbf{a}^{(1)}, ..., \mathbf{a}^{(n)}\}$, by solving

$$\min_{\mathbf{a}_1,...,\mathbf{a}_n \in \mathscr{A}} \left[\nu^T \left\{ \sum_{i=1}^{n} \psi(\mathbf{X}_i, \mathbf{a}_i)\psi(\mathbf{X}_i, \mathbf{a}_i)^T \right\}^{-1} \nu + \lambda \det\left\{ \sum_{i=1}^{n} \psi(\mathbf{X}_i, \mathbf{a}_i)\psi(\mathbf{X}_i, \mathbf{a}_i)^T \right\}^{-1} \right].$$

The parameter $\lambda \geq 0$ governs the trade-off between the c-optimality and D-optimality criteria. Larger values of λ result in lower power for the hypothesis test under consideration but more information about β^*. We can inform the choice of λ by computing the power for a range of values of λ for a given effect size. For example, if we assume $Y = \psi(\mathbf{X},\mathbf{A})^T\beta + \epsilon$, where $\epsilon \sim \mathscr{N}(0,\sigma^2)$ then, under $H_{0,2}$, the power for an α-level test of $\nu^T\beta^* \leq \mu_C$ with test statistic $\nu^T\hat{\beta}$ is given by $\Phi\Big(z_\alpha - \Delta/\Big[\nu^T\Big\{\sum_{i=1}^{n}\psi[\mathbf{X}_i, \mathbb{D}_{c,i}^{\lambda}(\nu)]\psi[\mathbf{X}_i, \mathbb{D}_{c,i}^{\lambda}(\nu)]^T\Big\}^{-1}\nu\Big]^{1/2}\Big)$, where z_α is the $\alpha \times 100$ percentile of the standard normal distribution and $\Delta = (\nu^T\beta^* - \mu_C)/\sigma$ is the standardized effect size.

For the third hypothesis $H_{0,3}$, we consider a minimal design under a least favorable configuration because it is a composite hypothesis, and there exist several possible values of β that can satisfy the specified hypothesis. Again, we assume $Y = \psi(\mathbf{X},\mathbf{A})^T\beta +$

ϵ, where $\epsilon \sim \mathcal{N}(0,\sigma^2)$. Let $\mathbb{D} = \{\mathbf{a}^{(1)},...,\mathbf{a}^{(n)}\} \in \mathscr{A}^n$ denote an arbitrary design, $\Sigma(\mathbb{D}) = \sum_{i=1}^n \psi(\mathbf{x}_i,\mathbb{D}_i)\psi(\mathbf{x}_i,\mathbb{D}_i)^T$. Under mild moment and eigenvalue conditions (Lai and Wei 1982), the OLS estimator $\hat{\beta}_n \sim \mathcal{N}\{\beta^*,\sigma^2\Sigma(\mathbb{D})^{-1}\}$. The joint distribution of $\left\{\phi(\mathbf{a})^T\hat{\beta}_n\right\}_{a\in\mathscr{A}}$ is multivariate normal with mean $\left\{\phi(\mathbf{a})^T\beta^*\right\}_{a\in\mathscr{A}}$ and covariance $\zeta\{\Sigma(\mathbb{D}),\sigma\}$, where ζ is a smooth (matrix-valued) function. The variance of the test statistic $W := \max_{\mathbf{a}\in\mathscr{A}}\phi(\mathbf{a})^T\hat{\beta}_n$ is

$$\mathbb{V}_n(\beta^*,\mathbb{D}) = \text{Var}(W) = E(W^2) - [E(W)]^2.$$

Define $\mathscr{B} \subset \{\beta \in \mathbb{R}^s : \max_{\mathbf{a}\in\mathscr{A}}\phi(\mathbf{a})^T\beta \leq \mu_C\}$ to be the set of allowable configurations. For any $\lambda \geq 0$, define the optimal design against a least favorable configuration for $H_{0,3}$ as $\mathbb{D}_M^\lambda = \{\mathbf{a}^{(1)},...,\mathbf{a}^{(n)}\} \in \mathscr{A}^n$ solving

$$\min_{\mathbf{a}_1,...,\mathbf{a}_n\in\mathscr{A}}\left[\sup_{\beta\in\mathscr{B}}\mathbb{V}_n(\beta,\{\mathbf{a}_1,...,\mathbf{a}_n\}) + \lambda\det\left\{\sum_{i=1}^n \psi(\mathbf{X}_i,\mathbf{a}_i)\psi(\mathbf{X}_i,\mathbf{a}_i)^T\right\}^{-1}\right],$$

where λ regulates the trade-off between maximizing power for $H_{0,3}$ and obtaining enough information on β^*. We obtain the variance of the test statistic under the worst case by finding the $\beta \in \mathscr{B}$ that maximizes $\mathbb{V}_n(\beta,\{\mathbf{a}_1,...,\mathbf{a}_n\})$. Thus, setting λ to zero maximizes power because we are left with finding a design that minimizes the variance of the test statistic under a least favorable configuration. Similar to $H_{0,1}$ and $H_{0,2}$, the choice of λ can be informed by computing the power for a range of λ values. Let τ_α denote a critical value for an α-level test of $H_{0,3}$, so that the test rejects if $W \geq \tau_\alpha$. Guidance for the construction of τ_α can be found in Tsiatis et al. (2019).

5.3 Simulations

We consider the case where we have k binary treatment factors resulting in 2^k treatment combinations and ℓ covariates. We choose $\psi(\mathbf{X},\mathbf{A})$ to include all main effects and two-way interactions. Thus, the total number of parameters, p, is $k + \binom{k}{2} + \ell + k\ell + 1$, including an intercept. We use a stochastic search method to determine the design that minimizes the design criterion. The data-generating model is given by

$$Y_i = \psi(\mathbf{X}_i,\mathbf{A}_i)^T\beta + \epsilon_i, \quad i = 1,...,n,$$

where $\mathbf{X}_i \sim \mathcal{N}\{0, \Omega_{AR1}(0.5)\}$ and $\epsilon_i \sim \mathcal{N}(0,1)$.

For our simulations for hypotheses 1 and 2, we consider sample sizes of 250 and 500, $k = 8$ treatment factors at two levels each, and $\ell = 4$ covariates (resulting in $p = 73$ parameters). The covariates are centered so that they have mean 0. The feature vector $\psi(\mathbf{x}, \mathbf{a})$, comprising all main effects and pairwise interactions, is of the form

$$(1, a^1, a^2, ..., a^8, a^1a^2, ..., a^1a^8, ..., a^7a^8, x^1, ..., x^4, a^1x^1, ..., a^1x^4, a^2x^1, ..., a^8x^4).$$

For hypothesis 3, we consider sample sizes of 50 and 100, $k = 3$ binary treatment factors, and $\ell = 2$ centered covariates because the maximization step involved in optimizing the composite criterion is very computationally costly. The feature vector $\psi(\mathbf{x}, \mathbf{a})$ comprises all main effects and pairwise interactions. All initial power and log determinant values in the ensuing results are based on a random allocation of the treatments. The final power and log determinant values are the results obtained based on the optimal allocation method proposed.

5.3.1 Hypothesis 1

Let $\mathbf{a}_0 = (1,0,0,0,0,0,0,0)^T$ and $\mathbf{a}_0' = (1,0,1,0,1,0,1,0)^T$.
Consider $H_0 : \{\phi(\mathbf{a}_0) - \phi(\mathbf{a}_0')\}^T \beta^* = 0$. We obtain the optimal design by solving the following objective function.

$$\min_{\mathbf{a}_1,...,\mathbf{a}_n \in \mathscr{A}} \nu^T \left\{ \frac{1}{n} \sum_{i=1}^{n} \psi(\mathbf{X}_i, \mathbf{a}_i)\psi(\mathbf{X}_i, \mathbf{a}_i)^T \right\}^{-1} \nu + \lambda \log\left[\det\left\{ \frac{1}{n} \sum_{i=1}^{n} \psi(\mathbf{X}_i, \mathbf{a}_i)\psi(\mathbf{X}_i, \mathbf{a}_i)^T \right\}^{-1} \right],$$

where $\nu = \{\phi(\mathbf{a}_0) - \phi(\mathbf{a}_0')\}$.

The results of the above hypothesis test are shown in Table 5.1 and Figure 5.1. It can be observed that smaller values of λ yield higher power as well as relatively larger values of the log determinant of $\{n^{-1}\Psi(\mathbf{X}, \mathbf{A})^T \Psi(\mathbf{X}, \mathbf{A})\}^{-1}$. This supports the assumption that a greater weight on the c-optimality criterion results in higher power while a greater weight on the D-optimality criterion results in more information about β^*. An increase in the sample size from 250 to 500 results in higher power as expected, though it can be observed that there is no appreciable difference between the log determinants for the same values of λ and the effect size upon increasing the sample size from 250 to 500.

Table 5.1: Averaged results over 200 Monte Carlo replicates for $H_{0,1}$; $k = 8$ and $\ell = 4$; LogDet refers to the log of the determinant of $\{n^{-1}\Psi(\mathbf{X},\mathbf{A})^T\Psi(\mathbf{X},\mathbf{A})\}^{-1}$.

Sample size	Effect size	λ	Initial Power	Initial Type I Error	Initial LogDet	Final Power	Final Type I Error	Final LogDet	Avg.Time (mins.)
250	0.5	0.05	0.21	0.06	154.09	0.9	0.04	184.06	286
250	0.5	0.1	0.21	0.06	154.09	0.84	0.01	171.02	316
250	0.5	0.5	0.21	0.06	154.09	0.68	0.06	153.61	326
250	0.5	1	0.21	0.06	154.09	0.48	0.04	149.69	326
250	0.5	2	0.21	0.06	154.09	0.43	0.06	147.05	410
500	0.5	0.05	0.24	0.06	147.02	1.0	0.07	182.05	667
500	0.5	0.1	0.24	0.06	147.02	0.99	0.04	169.16	666
500	0.5	0.5	0.24	0.06	147.02	0.95	0.04	151.90	710
500	0.5	1	0.24	0.06	147.02	0.82	0.06	148.06	701
500	0.5	2	0.24	0.06	147.02	0.74	0.06	145.44	709

5.3.2 Hypothesis 2

Let $\mathbf{a}_0 = (1,1,0,1,1,0,1,1)^T$ and $H_0 : \phi(\mathbf{a}_0)^T\beta^* \leq \mu_C$, where μ_C is the mean outcome under control which we set at some fixed value. In our simulations, we use $\mu_C = 0.3$. Similar to the preceding hypothesis, we obtain the optimal design by solving the objective function below.

$$\min_{\mathbf{a}_1,\ldots,\mathbf{a}_n \in \mathscr{A}} v^T \left\{ \frac{1}{n}\sum_{i=1}^{n} \psi(\mathbf{X}_i,\mathbf{a}_i)\psi(\mathbf{X}_i,\mathbf{a}_i)^T \right\}^{-1} v + \lambda \log\left[\det\left\{ \frac{1}{n}\sum_{i=1}^{n} \psi(\mathbf{X}_i,\mathbf{a}_i)\psi(\mathbf{X}_i,\mathbf{a}_i)^T \right\}^{-1} \right],$$

where $v = \phi(\mathbf{a}_0)$.

From Table 5.2 and Figure 5.2, it can be observed that similar to the first hypothesis, smaller values of λ yield higher power as well as relatively larger values of the log determinant. Thus, a greater weight on the D-optimality criterion results in lower power but more information about the parameter vector and, subsequently, better inference to facilitate secondary analyses.

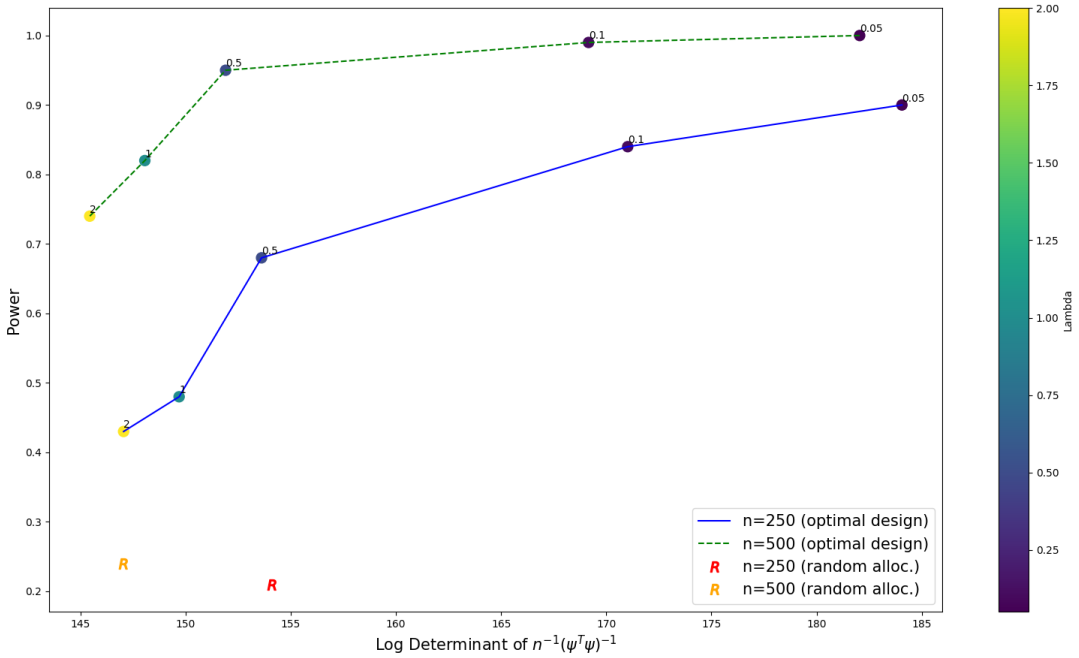

Figure 5.1: Power versus Log Determinant of $\{n^{-1}\Psi(\mathbf{X},\mathbf{A})^T\Psi(\mathbf{X},\mathbf{A})\}^{-1}$ for different values of λ. Results are averaged over 200 Monte Carlo replicates. The solid and dashed lines correspond to sample sizes of 250 and 500, respectively. (Text above each point is the value of λ used in the composite criterion; $k=8$ and $\ell=4$)

Table 5.2: Averaged results over 200 Monte Carlo replicates for $H_{0,2}$; $k=8$ and $\ell=4$; LogDet refers to the log of the determinant of $\{n^{-1}\Psi(\mathbf{X},\mathbf{A})^T\Psi(\mathbf{X},\mathbf{A})\}^{-1}$.

Sample size	Effect size	λ	Initial Power	Initial Type I Error	Initial LogDet	Final Power	Final Type I Error	Final LogDet	Avg.Time (mins.)
250	0.3	0.05	0.21	0.06	154.09	1.0	0.03	164.94	313
250	0.3	0.1	0.21	0.06	154.09	0.82	0.02	157.73	300
250	0.3	0.5	0.21	0.06	154.09	0.67	0.05	148.67	308
250	0.3	1	0.21	0.06	154.09	0.55	0.03	146.58	320
250	0.3	2	0.21	0.06	154.09	0.45	0.05	145.18	324
500	0.3	0.05	0.27	0.06	147.02	1.0	0.07	163.15	695
500	0.3	0.1	0.27	0.06	147.02	1.0	0.05	156.0	679
500	0.3	0.5	0.27	0.06	147.02	0.93	0.09	147.03	695
500	0.3	1	0.27	0.06	147.02	0.84	0.05	144.98	653
500	0.3	2	0.27	0.06	147.02	0.83	0.08	143.58	677

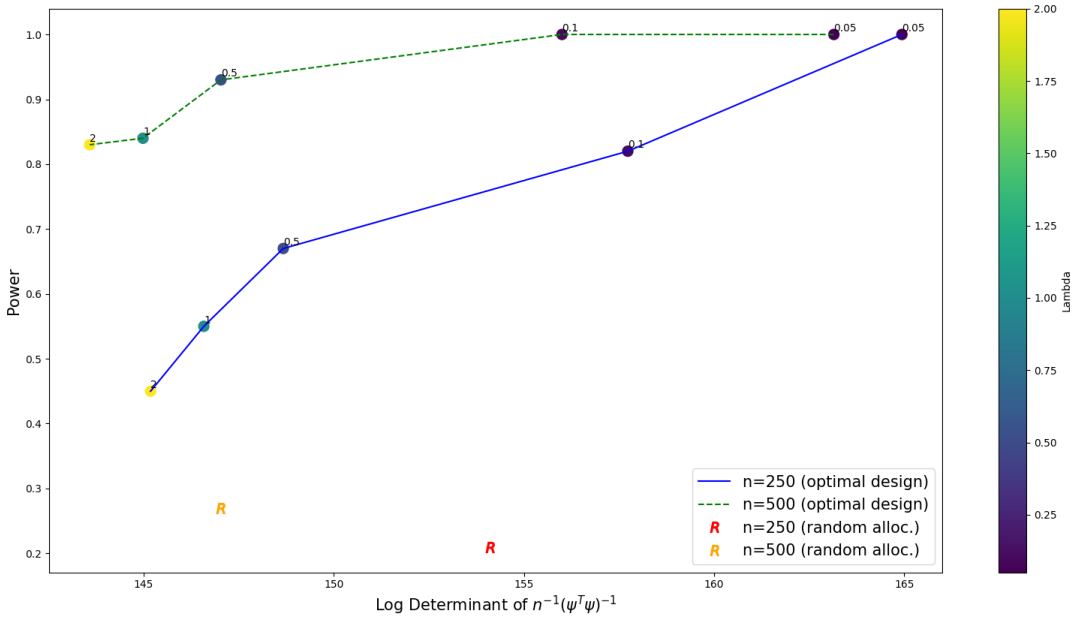

Figure 5.2: Power versus Log Determinant of $\{n^{-1}\Psi(\mathbf{X},\mathbf{A})^T\Psi(\mathbf{X},\mathbf{A})\}^{-1}$ for different values of λ. Results are averaged over 200 Monte Carlo replicates. The solid and dashed lines correspond to sample sizes of 250 and 500, respectively. (Text above each point is the value of λ used in the composite criterion; $k = 8$ and $\ell = 4$)

Table 5.3: Averaged results over 200 Monte Carlo replicates for $H_{0,3}$; $k = 3$ and $\ell = 2$; LogDet refers to the log of the determinant of $\{n^{-1}\Psi(\mathbf{X},\mathbf{A})^T\Psi(\mathbf{X},\mathbf{A})\}^{-1}$.

Sample size	Effect size	λ	Initial Power	Initial Type I Error	Initial LogDet	Final Power	Final Type I Error	Final LogDet	Avg.Time (mins.)
50	0.8	1.4×10^{-5}	0.56	0.06	25.16	0.84	0.05	23.60	811
50	0.8	2.8×10^{-5}	0.56	0.06	25.16	0.83	0.06	23.53	416
50	0.8	1×10^{-4}	0.56	0.06	25.16	0.82	0.06	23.33	432
50	0.8	3×10^{-4}	0.56	0.06	25.16	0.80	0.04	23.02	434
50	0.8	6×10^{-4}	0.56	0.06	25.16	0.81	0.05	22.82	519
100	0.8	1.4×10^{-5}	0.98	0.03	23.40	0.99	0.06	22.82	666
100	0.8	2.8×10^{-5}	0.98	0.03	23.40	0.99	0.05	22.73	629
100	0.8	1×10^{-4}	0.98	0.03	23.40	0.99	0.05	22.49	808
100	0.8	3×10^{-4}	0.98	0.03	23.40	0.98	0.04	22.30	756
100	0.8	6×10^{-4}	0.98	0.03	23.40	0.99	0.04	22.22	830

5.3.3 Hypothesis 3

The hypothesis under consideration is $H_{0,3} : \max_{\mathbf{a}\in\mathscr{A}} \phi(\mathbf{a})^T\beta^* \leq \mu_C$. As previously mentioned, this is a composite null hypothesis, so we consider a minimal design under a least favorable configuration. For $H_{0,3}$, we obtain the optimal design by solving

$$\min_{\mathbf{a}_1,\ldots,\mathbf{a}_n\in\mathscr{A}}\left(\sup_{\beta\in\mathscr{B}}\left[\frac{1}{n}\mathbb{V}_n(\beta,\{\mathbf{a}_1,\ldots,\mathbf{a}_n\})\right]+\lambda\det\left\{\frac{1}{n}\sum_{i=1}^{n}\psi(\mathbf{X}_i,\mathbf{a}_i)\psi(\mathbf{X}_i,\mathbf{a}_i)^T\right\}^{-1}\right),$$

where $\mathscr{B}\subset\{\beta\in\mathbb{R}^s : \max_{\mathbf{a}\in\mathscr{A}}\phi(\mathbf{a})^T\beta\leq\mu_C\}$ is defined to be the set of allowable configurations. It can be observed from Table 5.3 and Figure 5.3 that there is an increase in the power of the test after optimizing the composite criterion above. This is more evident in the case of $n = 50$ than in the $n = 100$ case, where the initial power obtained via random allocation is already very close to 1. However, with regard to the final log determinant, the same trend is observed as in the previous hypotheses. That is, an increase in the weights on the D-optimality criterion (as indicated by the magnitude of λ) results in a diminution of the log determinant after optimization in both the $n = 50$ and $n = 100$ cases. Thus, a greater value of λ leads to more information about the parameter vector.

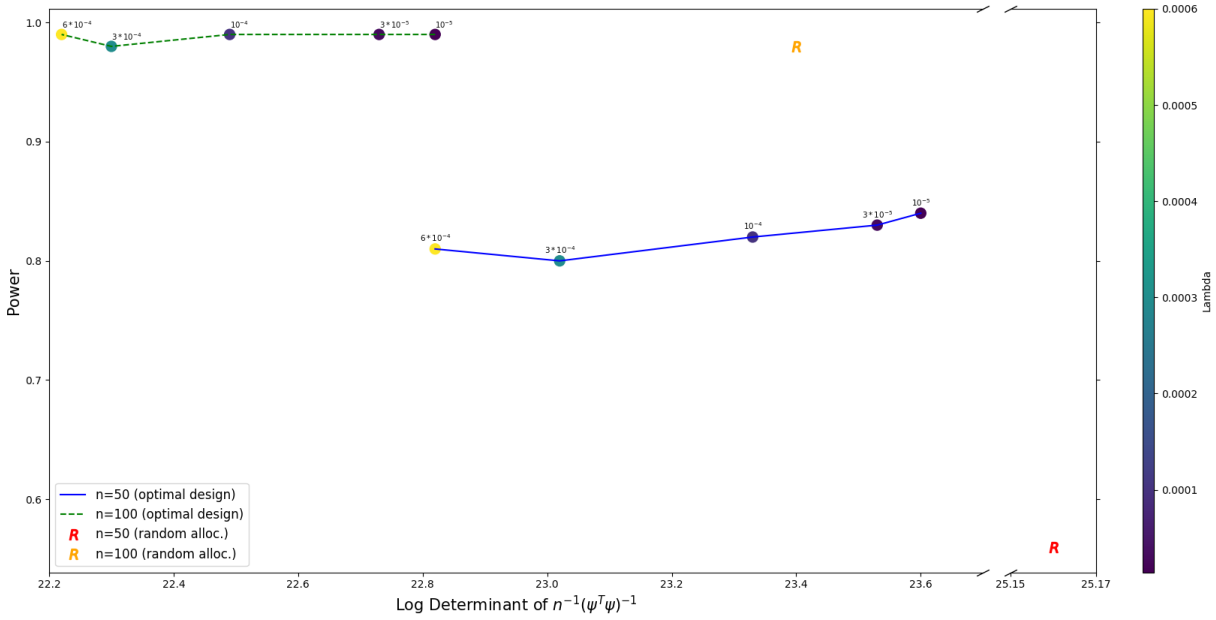

Figure 5.3: Power versus Log Determinant of $\{n^{-1}\Psi(\mathbf{X},\mathbf{A})^T\Psi(\mathbf{X},\mathbf{A})\}^{-1}$ for different values of λ. Results are averaged over 200 Monte Carlo replicates. The solid and dashed lines correspond to sample sizes of 50 and 100, respectively. (Text above each point is the value of λ used in the composite criterion; $k=3$ and $\ell=2$)

5.3.4 Concluding Remarks

We have proposed a novel method for optimally allocating treatments that consist of multiple therapeutic options. The strategy exploits the theory of classical optimal design and automatically incorporates patient characteristics in the optimization process. The simulation results of the hypotheses considered suggest that, depending on the choice of λ, we are able to obtain higher statistical power when the above-mentioned approach is used to allocate treatments as compared to randomly allocating treatments.

CHAPTER

6

CONCLUSIONS

6.1 Discussion and Conclusions

6.1.1 SGD and Applications in Statistics

We have presented a collection of stochastic gradient descent methods and their applications in some areas of Statistics. We have shown that the conditions on the step size sequence are sufficient for the convergence of SGD using the example of the mean. We have also illustrated through examples and with our proposed method for tuning hyper-parameters that though SGD is extensively used with a great deal of success, it can be very erratic and requires a significant amount of hyper-parameter tuning to obtain desirable results. In spite of their shortcomings, SGD and its variants have provided a much-needed solution to the problem of optimization in this era of high-volume and streaming data.

6.1.2 Optimal Experimental Designs for Precision Medicine with Multi-component Treatments

We have introduced a novel method for optimally allocating multi-component treatments that leverages classical optimal design methods by optimizing a composite criterion to maximize power for primary analyses while retaining sufficient information to conduct secondary analyses. Our preliminary findings are promising and suggest that the proposed method is superior relative to a random assignment of the treatments, as indicated by the results of our simulation studies.

6.1.3 Programming Language

All the code used in the simulations and plots is written in Python and can be made available on request.

6.2 Future Work

6.2.1 SGD and Applications in Statistics

We would like to evaluate our hyper-parameter tuning method on a different class of statistical models, such as the generalized linear model. We also plan to investigate ways to improve the computational speed of our method. In addition, we intend to compare the performance of the tuning method when used with other variants of SGD other than Vanilla SGD.

6.2.2 Optimal Experimental Designs for Precision Medicine with Multi-component Treatments

We intend to conduct more simulations under a wider array of settings than those presented in this work. We would also like to evaluate the performance of our proposed method for allocating multi-component treatments when applied to real-world data. Our future plans involve extending our analysis to multi-decision trials. In addition, we would like to explore alternative methods of optimizing the proposed composite criterion.

REFERENCES

Agarwal, Alekh; Wainwright, Martin J; Bartlett, Peter L, and Ravikumar, Pradeep K. Information-theoretic lower bounds on the oracle complexity of convex optimization. In *Advances in Neural Information Processing Systems*, pages 1–9, 2009.

Amari, Shun-Ichi; Park, Hyeyoung, and Fukumizu, Kenji. Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural computation*, 12(6):1399–1409, 2000.

Anbar, Dan. The application of stochastic approximation methods to the bio-assay problem. Technical report, CASE WESTERN RESERVE UNIV CLEVELAND OHIO DEPT OF MATHEMATICS AND STATISTICS, 1976.

Anbar, Dan. A stochastic newton-raphson method. *Journal of Statistical Planning and Inference*, 2(2):153–163, 1978.

Andrew, Galen and Gao, Jianfeng. Scalable training of l 1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, pages 33–40. ACM, 2007.

Atkinson, Anthony Curtis and Donev, Alexander N. *Optimum experimental designs*, volume 5. Clarendon Press, 1992.

Bach, Francis. Adaptivity of averaged stochastic gradient descent to local strong convexity for logistic regression. *The Journal of Machine Learning Research*, 15(1):595–627, 2014.

Bach, Francis and others, . Self-concordant analysis for logistic regression. *Electronic Journal of Statistics*, 4:384–414, 2010.

Barzilai, Jonathan and Borwein, Jonathan M. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988.

Berry, Donald A and Eick, Stephen G. Adaptive assignment versus balanced randomization in clinical trials: a decision analysis. *Statistics in medicine*, 14(3):231–246, 1995.

Bertsekas, Dimitri. Distributed dynamic programming. *IEEE transactions on Automatic Control*, 27(3):610–616, 1982.

Bertsekas, Dimitri P. A new class of incremental gradient methods for least squares problems. *SIAM Journal on Optimization*, 7(4):913–926, 1997a.

Bertsekas, Dimitri P. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997b.

Bertsekas, Dimitri P and Tsitsiklis, John N. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

Bertsekas, Dimitri P and Tsitsiklis, John N. *Neuro-dynamic programming*, volume 5. Athena Scientific Belmont, MA, 1996.

Bishop, Chris M. Training with noise is equivalent to Tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

Blatt, Doron; Hero, Alfred O, and Gauchman, Hillel. A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, 18(1):29–51, 2007.

Bordes, Antoine; Ertekin, Seyda; Weston, Jason; Botton, Léon, and Cristianini, Nello. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(9), 2005.

Bordes, Antoine; Bottou, Léon, and Gallinari, Patrick. Sgd-qn: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 10(Jul):1737–1754, 2009.

Bordes, Antoine; Bottou, Léon; Gallinari, Patrick; Chang, Jonathan, and Smith, S Alex. Erratum: Sgdqn is less careful than expected. *Journal of Machine Learning Research*, 11(Aug): 2229–2240, 2010.

Bottou, Léon. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991.

Bottou, Léon. Curiously fast convergence of some stochastic gradient descent algorithms. In *Proceedings of the symposium on learning and data science, Paris*, 2009.

Bottou, Léon and Cun, Yann L. Large scale online learning. In *Advances in neural information processing systems*, pages 217–224, 2004.

Bottou, Léon; Curtis, Frank E, and Nocedal, Jorge. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

Byrd, Richard H; Hansen, Samantha L; Nocedal, Jorge, and Singer, Yoram. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26 (2):1008–1031, 2016.

Carpenter, Bob. Lazy sparse stochastic gradient descent for regularized multinomial logistic regression. *Alias-i, Inc., Tech. Rep*, pages 1–20, 2008.

Chen, Han-Fu; Duncan, Tyrone E, and Pasik-Duncan, Bozenna. A kiefer-wolfowitz algorithm with randomized differences. *IEEE Transactions on Automatic Control*, 44(3): 442–453, 1999.

Chen, Xi; Lee, Jason D; Tong, Xin T, and Zhang, Yichen. Statistical inference for model parameters in stochastic gradient descent. *arXiv preprint arXiv:1610.08637*, 2016.

Cheng, Bing and Titterington, D Michael. Neural networks: A review from a statistical perspective. *Statistical science*, pages 2–30, 1994.

Cheung, Ying Kuen. Stochastic approximation and modern model-based designs for dose-finding clinical trials. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 25(2):191, 2010.

Cheung, Ying Kuen and Elkind, Mitchell SV. Stochastic approximation with virtual observations for dose-finding on discrete levels. *Biometrika*, 97(1):109–121, 2009.

Chowdhury, Mashrur; Apon, Amy, and Dey, Kakan. *Data analytics for intelligent transportation systems*. Elsevier, 2017.

Cochran, William G and Davis, Miles. The robbins–monro method for estimating the median lethal dose. *Journal of the Royal Statistical Society: Series B (Methodological)*, 27 (1):28–44, 1965.

Cohen, Kobi; Nedić, Angelia, and Srikant, Rayadurgam. On projected stochastic gradient descent algorithm with weighted averaging for least squares regression. *IEEE Transactions on Automatic Control*, 62(11):5974–5981, 2017.

Collins, Linda M; Murphy, Susan A; Nair, Vijay N, and Strecher, Victor J. A strategy for optimizing and evaluating behavioral interventions. *Annals of Behavioral Medicine*, 30 (1):65–73, 2005.

Collins, Linda M; Murphy, Susan A, and Strecher, Victor. The multiphase optimization strategy (most) and the sequential multiple assignment randomized trial (smart): new methods for more potent ehealth interventions. *American journal of preventive medicine*, 32(5):S112–S118, 2007.

Crammer, Koby; Kulesza, Alex, and Dredze, Mark. Adaptive regularization of weight vectors. In *Advances in neural information processing systems*, pages 414–422, 2009.

De, Soham; Yadav, Abhay; Jacobs, David, and Goldstein, Tom. Big batch sgd: Automated inference using adaptive batch sizes. *arXiv preprint arXiv:1610.05792*, 2016.

Defazio, Aaron; Bach, Francis, and Lacoste-Julien, Simon. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646–1654, 2014.

Défossez, Alexandre and Bach, Francis. Averaged least-mean-squares: Bias-variance trade-offs and optimal sampling distributions. In *Artificial Intelligence and Statistics*, pages 205–213. PMLR, 2015.

Dekel, Ofer; Gilad-Bachrach, Ran; Shamir, Ohad, and Xiao, Lin. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(Jan):165–202, 2012.

Delyon, Bernard and Juditsky, Anatoli. Accelerated stochastic approximation. *SIAM Journal on Optimization*, 3(4):868–881, 1993.

Dozat, Timothy. Incorporating nesterov momentum into adam. 2016.

Dredze, Mark; Crammer, Koby, and Pereira, Fernando. Confidence-weighted linear classification. In *Proceedings of the 25th international conference on Machine learning*, pages 264–271. ACM, 2008.

Duchi, John; Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul): 2121–2159, 2011.

Eisenhauer, EA; O'dwyer, PJ; Christian, M, and Humphrey, JS. Phase i clinical trial design in cancer drug development. *Journal of Clinical Oncology*, 18(3):684–684, 2000.

Fan, Rong-En; Chang, Kai-Wei; Hsieh, Cho-Jui; Wang, Xiang-Rui, and Lin, Chih-Jen. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, 9: 1871–1874, 2008.

Feldman, Dorian. Contributions to the" two-armed bandit" problem. *The Annals of Mathematical Statistics*, 33(3):847–856, 1962.

Frees, Edward W and Ruppert, David. Estimation following a sequentially designed experiment. *Journal of the American Statistical Association*, 85(412):1123–1129, 1990.

Friedlander, Michael P and Schmidt, Mark. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.

Gaivoronski, AA. Convergence analysis of parallel backpropagation algorithm for neural networks. *Optimization Methods and Software*, 4:117–134, 1994.

Garcia-Aguilar, Julio; Glynne-Jones, Rob, and Schrag, Deborah. Multimodal rectal cancer treatment: in some cases, less may be more. *American Society of Clinical Oncology Educational Book*, 36:92–102, 2016.

Goodfellow, Ian; Bengio, Yoshua, and Courville, Aaron. *Deep learning*. MIT press, 2016.

Grippo, Luigi. A class of unconstrained minimization methods for neural network training. *Optimization Methods and Software*, 4(2):135–150, 1994.

Györfi, László and Walk, Harro. On the averaged stochastic approximation for linear regression. *SIAM Journal on Control and Optimization*, 34(1):31–61, 1996.

Hayashi, Hiroaki; Koushik, Jayanth, and Neubig, Graham. Eve: A gradient based optimization method with locally and globally adaptive learning rates. *arXiv preprint arXiv:1611.01505*, 2016.

Hazan, Elad and Kale, Satyen. Better algorithms for benign bandits. *Journal of Machine Learning Research*, 12(Apr):1287–1311, 2011.

Hazan, Elad; Agarwal, Amit, and Kale, Satyen. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

Hinton, Geoffrey; Srivastava, Nitish, and Swersky, Kevin. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14, 2012.

Hodges Jr, J and Lehmann, Erich Leo. Two approximations to the robbins monro process. Technical report, UNIVERSITY OF CALIFORNIA Oakland United States, 1956.

Hsieh, Cho-Jui; Chang, Kai-Wei; Lin, Chih-Jen; Keerthi, S Sathiya, and Sundararajan, Sellamanickam. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415, 2008.

Hu, Feifang; Hu, Yanqing; Ma, Zhenjun, and Rosenberger, William F. Adaptive randomization for balancing over covariates. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(4):288–303, 2014.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Jaakkola, Tommi; Jordan, Michael I, and Singh, Satinder P. Convergence of stochastic iterative dynamic programming algorithms. In *Advances in neural information processing systems*, pages 703–710, 1994.

Joachims, Thorsten. Making large-scale support vector machine learning practical, advances in kernel methods. *Support vector learning*, 1999.

Joachims, Thorsten. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, 2006.

Johnson, Rachel T; Montgomery, Douglas C, and Jones, Bradley A. An expository paper on optimal design. *Quality Engineering*, 23(3):287–301, 2011.

Johnson, Rie and Zhang, Tong. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

Kernan, Walter N; Viscoli, Catherine M; Makuch, Robert W; Brass, Lawrence M, and Horwitz, Ralph I. Stratified randomization for clinical trials. *Journal of clinical epidemiology*, 52 (1):19–26, 1999.

Keskar, Nitish Shirish; Mudigere, Dheevatsa; Nocedal, Jorge; Smelyanskiy, Mikhail, and Tang, Ping Tak Peter. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Kesten, Harry and others, . Accelerated stochastic approximation. *The Annals of Mathematical Statistics*, 29(1):41–59, 1958.

Khuri, André I; Mukherjee, Bhramar; Sinha, Bikas K, and Ghosh, Malay. Design issues for generalized linear models: A review. *Statistical Science*, pages 376–399, 2006.

Kiefer, J. Optimum designs in regression problems, ii. *The Annals of Mathematical Statistics*, 32(1):298–325, 1961.

Kiefer, Jack. Optimum experimental designs. *Journal of the Royal Statistical Society: Series B (Methodological)*, 21(2):272–304, 1959.

Kiefer, Jack and Wolfowitz, Jacob. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

Kiefer, Jack and Wolfowitz, Jacob. Optimum designs in regression problems. *The annals of mathematical statistics*, 30(2):271–294, 1959.

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Konečnỳ, Jakub and Richtárik, Peter. Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*, 2013.

Kushner, Harold and Yin, G George. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

Lai, Tze Leung and Wei, Ching Zong. Least squares estimates in stochastic regression models with applications to identification and control of dynamic systems. *The Annals of Statistics*, 10(1):154–166, 1982.

LeCun, Yann A; Bottou, Léon; Orr, Genevieve B, and Müller, Klaus-Robert. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

Ljung, Lennart. Analysis of recursive stochastic algorithms. *IEEE transactions on automatic control*, 22(4):551–575, 1977.

Loshchilov, Ilya and Hutter, Frank. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Luo, Zhi-Quan. On the convergence of the lms algorithm with adaptive learning rate for linear feedforward networks. *Neural Computation*, 3(2):226–245, 1991.

Mangasarian, Olvi L. Mathematical programming in neural networks. *ORSA Journal on Computing*, 5(4):349–360, 1993.

Mangasarian, Olvi L and Solodov, Mikhail V. Serial and parallel backpropagation convergence via nonmonotone perturbed minimization. *Optimization Methods and Software*, 4(2):103–116, 1994.

McCulloch, Warren S and Pitts, Walter. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

Moulines, Eric and Bach, Francis R. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459, 2011.

Nemirovski, Arkadi; Juditsky, Anatoli; Lan, Guanghui, and Shapiro, Alexander. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

Nesterov, Yurii. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady AN USSR*, volume 269, pages 543–547, 1983.

Nesterov, Yurii. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

Nesterov, Yurii. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Newton, David; Pasupathy, Raghu, and Yousefian, Farzad. Recent trends in stochastic gradient descent for machine learning and big data. In *2018 Winter Simulation Conference (WSC)*, pages 366–380. IEEE, 2018.

Ning, Jing and Huang, Xuelin. Response-adaptive randomization for clinical trials with adjustment for covariate imbalance. *Statistics in medicine*, 29(17):1761–1768, 2010.

O'Quigley, John; Pepe, Margaret, and Fisher, Lloyd. Continual reassessment method: a practical design for phase 1 clinical trials in cancer. *Biometrics*, pages 33–48, 1990.

Philippou, Yiannis; Sjoberg, Hanna; Lamb, Alastair D; Camilleri, Philip, and Bryant, Richard J. Harnessing the potential of multimodal radiotherapy in prostate cancer. *Nature Reviews Urology*, 17(6):321–338, 2020.

Pocock, Stuart J and Simon, Richard. Sequential treatment assignment with balancing for prognostic factors in the controlled clinical trial. *Biometrics*, pages 103–115, 1975.

Polyak, Boris T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Polyak, Boris T. Introduction to optimization. optimization software. *Inc., Publications Division, New York*, 1, 1987.

Polyak, Boris T and Juditsky, Anatoli B. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Polyak, Boris Teodorovich. New method of stochastic approximation type. *Automation and remote control*, 51(7 pt 2):937–946, 1990.

Rakhlin, Alexander; Shamir, Ohad, and Sridharan, Karthik. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

Ratain, Mark J; Mick, Rosemarie; Schilsky, Richard L, and Siegler, Mark. Statistical and ethical issues in the design and conduct of phase i and ii clinical trials of new anticancer agents. *JNCI: Journal of the National Cancer Institute*, 85(20):1637–1643, 1993.

Riedmiller, Martin and Braun, Heinrich. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Proceedings of the IEEE international conference on neural networks*, volume 1993, pages 586–591. San Francisco, 1993.

Robbins, Herbert and Monro, Sutton. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Rosenberger, William F; Stallard, Nigel; Ivanova, Anastasia; Harper, Cherice N, and Ricks, Michelle L. Optimal adaptive designs for binary response trials. *Biometrics*, 57(3):909–913, 2001.

Rosenberger, William F; Sverdlov, Oleksandr, and Hu, Feifang. Adaptive randomization for clinical trials. *Journal of biopharmaceutical statistics*, 22(4):719–736, 2012.

Roux, Nicolas L; Schmidt, Mark, and Bach, Francis R. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in neural information processing systems*, pages 2663–2671, 2012.

Ruppert, David. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

Salimans, Tim and Kingma, Durk P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016.

Sambi, Manpreet; Bagheri, Leila, and Szewczuk, Myron R. Current challenges in cancer immunotherapy: multimodal approaches to improve efficacy and patient response rates. *Journal of oncology*, 2019.

Sarle, Warren S. Neural networks and statistical models. 1994.

Schraudolph, Nicol N. Local gain adaptation in stochastic gradient descent. 1999.

Schraudolph, Nicol N; Yu, Jin, and Günter, Simon. A stochastic quasi-newton method for online convex optimization. In *Artificial intelligence and statistics*, pages 436–443, 2007.

Schwenk, Holger and Bengio, Yoshua. Boosting neural networks. *Neural computation*, 12 (8):1869–1887, 2000.

Shalev-Shwartz, Shai; Singer, Yoram; Srebro, Nathan, and Cotter, Andrew. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

Shamir, Ohad and Zhang, Tong. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning*, pages 71–79, 2013.

Smith, Kirstine. On the standard deviations of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards a proper choice of the distribution of observations. *Biometrika*, 12(1/2):1–85, 1918.

Sobel, Milton and Weiss, George H. Play-the-winner rule and inverse sampling in selecting the better of two binomial populations. *Journal of the American Statistical Association*, 66(335):545–551, 1971.

Spall, James C. Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on aerospace and electronic systems*, 34(3):817–823, 1998.

Sutton, Richard S. Gain adaptation beats least squares. In *Proceedings of the 7th Yale workshop on adaptive and learning systems*, volume 161168, 1992.

Sutton, Richard S and Barto, Andrew G. *Reinforcement learning: An introduction*. MIT press, 2018.

Taves, Donald R. Minimization: a new method of assigning patients to treatment and control groups. *Clinical Pharmacology & Therapeutics*, 15(5):443–453, 1974.

Thompson, William R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

Toulis, Panagiotis; Airoldi, Edoardo, and Rennie, Jason. Statistical analysis of stochastic gradient methods for generalized linear models. In *International Conference on Machine Learning*, pages 667–675, 2014.

Toulis, Panos; Tran, Dustin, and Airoldi, Edo. Towards stability and optimality in stochastic gradient descent. In *Artificial Intelligence and Statistics*, pages 1290–1298, 2016.

Tsiatis, Anastasios A; Davidian, Marie; Holloway, Shannon T, and Laber, Eric B. *Dynamic treatment regimes: Statistical methods for precision medicine*. Chapman and Hall/CRC, 2019.

Tsitsiklis, John; Bertsekas, Dimitri, and Athans, Michael. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.

Tsitsiklis, John N. Asynchronous stochastic approximation and q-learning. *Machine learning*, 16(3):185–202, 1994.

Tsuruoka, Yoshimasa; Tsujii, Jun'ichi, and Ananiadou, Sophia. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 477–485. Association for Computational Linguistics, 2009.

Venter, JH. An extension of the robbins-monro procedure. *The Annals of Mathematical Statistics*, 38(1):181–190, 1967.

Watkins, Christopher JCH and Dayan, Peter. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Weir, Christopher J and Lees, Kennedy R. Comparison of stratification and adaptive methods for treatment allocation in an acute stroke clinical trial. *Statistics in medicine*, 22(5):705–726, 2003.

Wetherill, GB. Sequential estimation of quantal response curves. *Journal of the Royal Statistical Society: Series B (Methodological)*, 25(1):1–38, 1963.

White, Halbert. Some asymptotic results for learning in single hidden-layer feedforward network models. *Journal of the American Statistical Association*, 84(408):1003–1013, 1989.

White, Kyle R; Stefanski, Leonard A, and Wu, Yichao. Variable selection in kernel regression using measurement error selection likelihoods. *Journal of the American Statistical Association*, 112(520):1587–1597, 2017.

Wilson, D Randall and Martinez, Tony R. The general inefficiency of batch training for gradient descent learning. *Neural networks*, 16(10):1429–1451, 2003.

Wu, CF Jeff. Efficient sequential designs with binary data. *Journal of the American Statistical Association*, 80(392):974–984, 1985.

Xiao, Lin. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11(Oct):2543–2596, 2010.

Xu, Wei. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *arXiv preprint arXiv:1107.2490*, 2011.

Zeiler, Matthew D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zelen, Marvin. Play the winner rule and the controlled clinical trial. *Journal of the American Statistical Association*, 64(325):131–146, 1969.

Zhang, Tong. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116. ACM, 2004.

Zhi-Quan, Luo and Paul, Tseng. Analysis of an approximate gradient projection method with applications to the backpropagation algorithm. *Optimization Methods and Software*, 4(2):85–101, 1994.

Zinkevich, Martin. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 928–936, 2003.

APPENDICES

APPENDIX

A

FURTHER COMPARISONS OF SOME SGD METHODS

A.1 Standard Momentum and Nesterov Momentum

We present some additional cases comparing the performance of SGD with Standard Momentum and SGD with Nesterov Momentum using a simulation study under the logistic regression setup. The design matrices used are generated from a multivariate normal under two different covariance structures, specifically an AR1 covariance matrix and an equi-correlated matrix (a matrix with 1 on the diagonals and equal correlation everywhere else). The value of the correlation used in all cases is 0.8. The data consists of 1000 observations and 9 variables. The results shown are averaged over 200 datasets.

We plot the trajectories of the individual components of the parameter vector to map the path of the parameter vector as it journeys from the initial point toward the desired goal. The figures presented below show the trajectories of the coefficients of some of the coefficients plotted against the intercept parameter. The black star represents the starting

point, the red star represents the true parameter value and the green star represents the maximum likelihood estimate using the Newton-Raphson method.

Figures A.1 and A.2 show the trajectories of components of a parameter vector consisting of a mix of both negative and positive values for an AR1 covariance design matrix and equi-correlated design matrix respectively. The velocity parameter used is 0.75. It can be observed that in the case of the AR1 structure, a value of 0.75 is not large enough to reach the desired optimum in 20 passes through the data. However, the velocity of 0.75 appears to be a good choice in the case of the equi-correlated covariance structure. The setup used in Figures A.3 and A.4 is equivalent to that of A.1 and A.2 respectively with the exception that the velocity parameter used in these examples is 0.9. In both the AR1 and equi-correlated covariance structures, the velocity parameter of 0.9 leads to an overshooting of the desired target. In all four cases, $\theta^* = [-2, -1.56, -1.11, -0.67, -0.22, 0.22, 0.67, 1.11, 1.56, 2]$.

In Figures A.5 and A.6, the setup is as described above but the parameter vector is a vector of all positive values, i.e. $\theta^* = [1, 1.11, 1.22, 1.33, 1.44, 1.56, 1.67, 1.78, 1.89, 2]$ and the velocity parameter is 0.95. The setups used in Figures A.7 and A.8 are the same as those of A.5 and A.6 respectively except that $\theta^* = [-3, -2.78, -2.56, -2.33, -2.11, -1.89, -1.67, -1.44, -1.22, -1]$. It can be observed that in these cases as well, a velocity parameter of 0.95 appears too large and tends to miss the mark, resulting in more iterations required to arrive at the optimum.

A.2 Vanilla SGD and SGD with Standard Momentum

From Figures A.9 and A.10, it can be observed that SGD with Momentum outstrips Vanilla SGD in journeying toward the desired optimum under both types of covariance structures. In these examples, $\theta^* = [-2, -1.56, -1.11, -0.67, -0.22, 0.22, 0.67, 1.11, 1.56, 2]$. Similar to the examples on SGD with Momentum, the black star represents the starting point, the red star represents the true parameter value and the green star represents the maximum likelihood estimate using the Newton-Raphson method. After 20 complete passes through the data, SGD with Standard Momentum is within reach of the goal while Vanilla SGD lags far behind and will require several more passes through the data to attain the optimum.

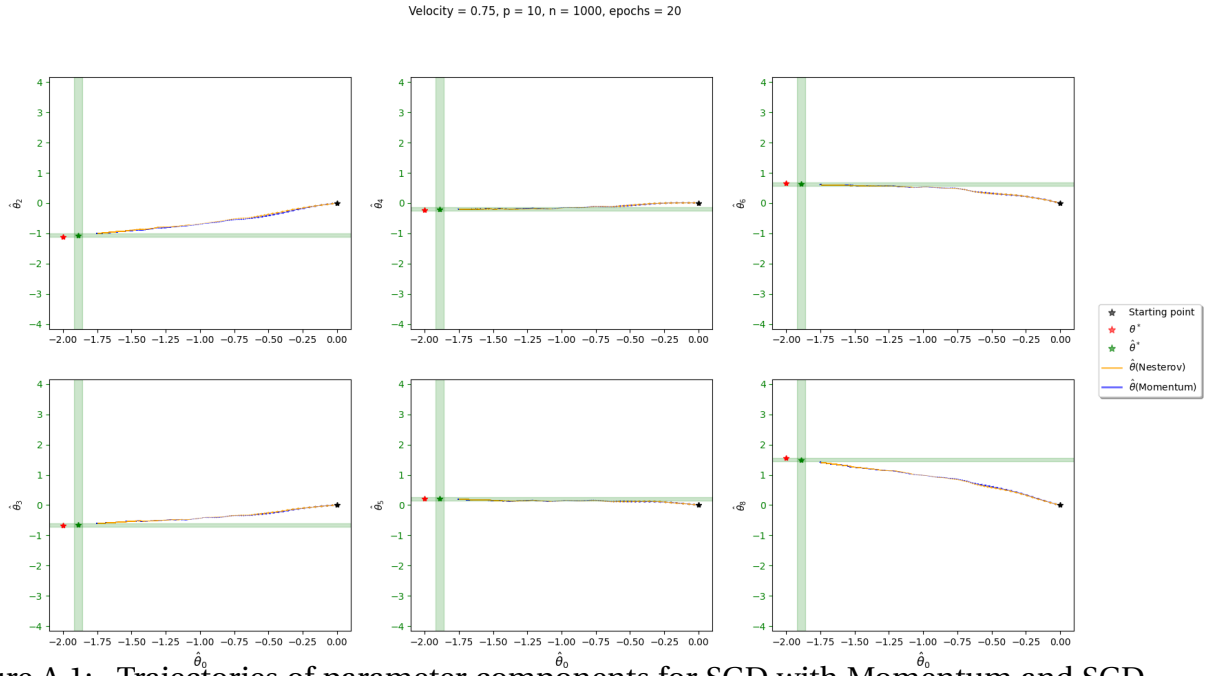

Figure A.1: Trajectories of parameter components for SGD with Momentum and SGD with Nesterov Momentum under an L2-regularized logistic loss function and a design matrix with an AR1 covariance structure ($\hat{\theta}_0 = \mathbf{0}$). Results depict the parameter trajectories averaged over 200 datasets.

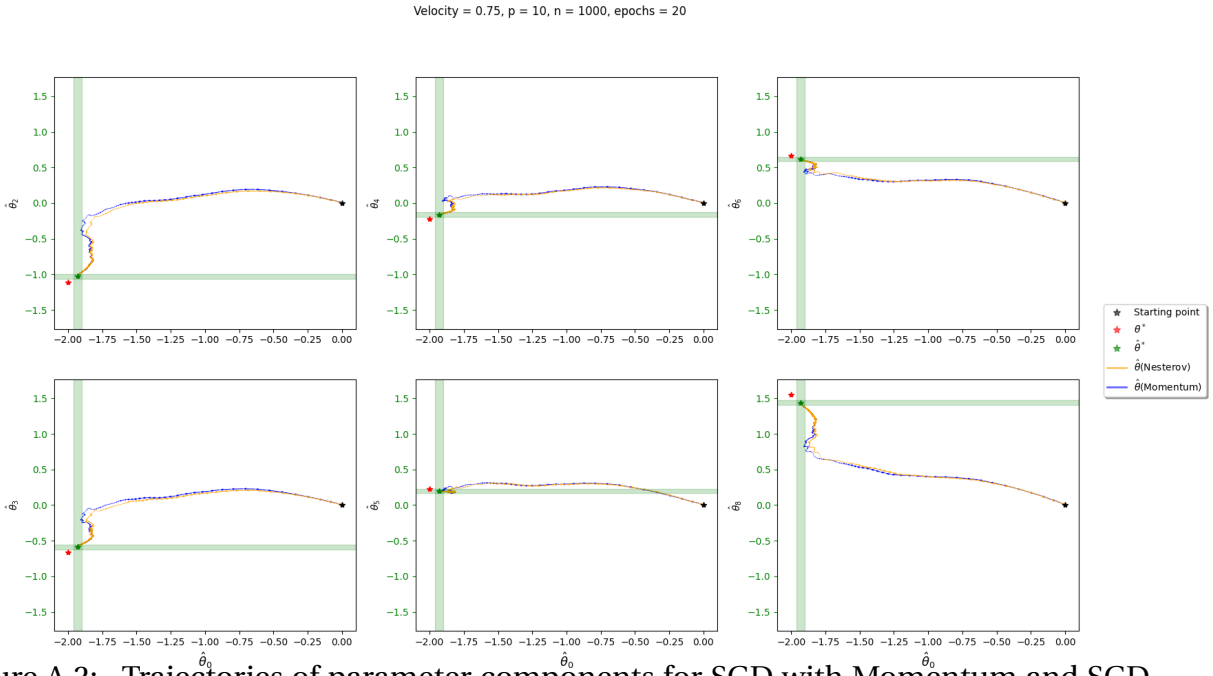

Figure A.2: Trajectories of parameter components for SGD with Momentum and SGD with Nesterov Momentum under an L2-regularized logistic loss function and a design matrix with an equi-correlated covariance structure ($\hat{\theta}_0 = \mathbf{0}$). Results depict the parameter trajectories averaged over 200 datasets.

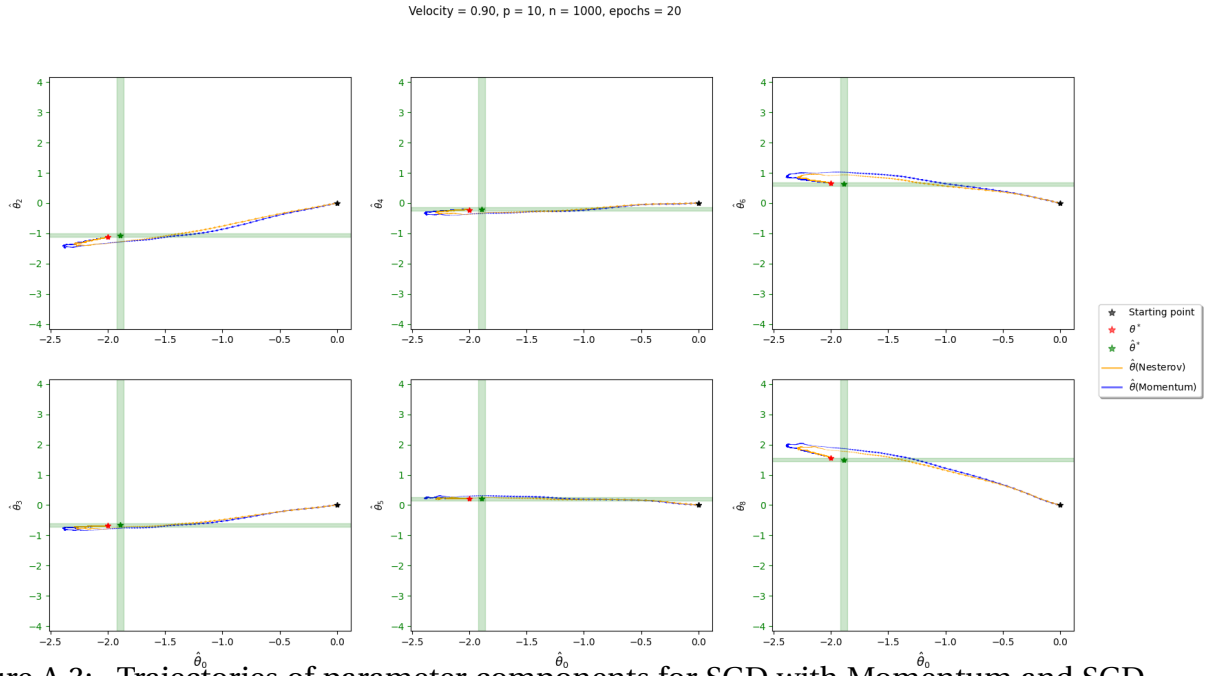

Figure A.3: Trajectories of parameter components for SGD with Momentum and SGD with Nesterov Momentum under an L2-regularized logistic loss function and a design matrix with an AR1 covariance structure ($\hat{\theta}_0 = \mathbf{0}$). Results depict the parameter trajectories averaged over 200 datasets.

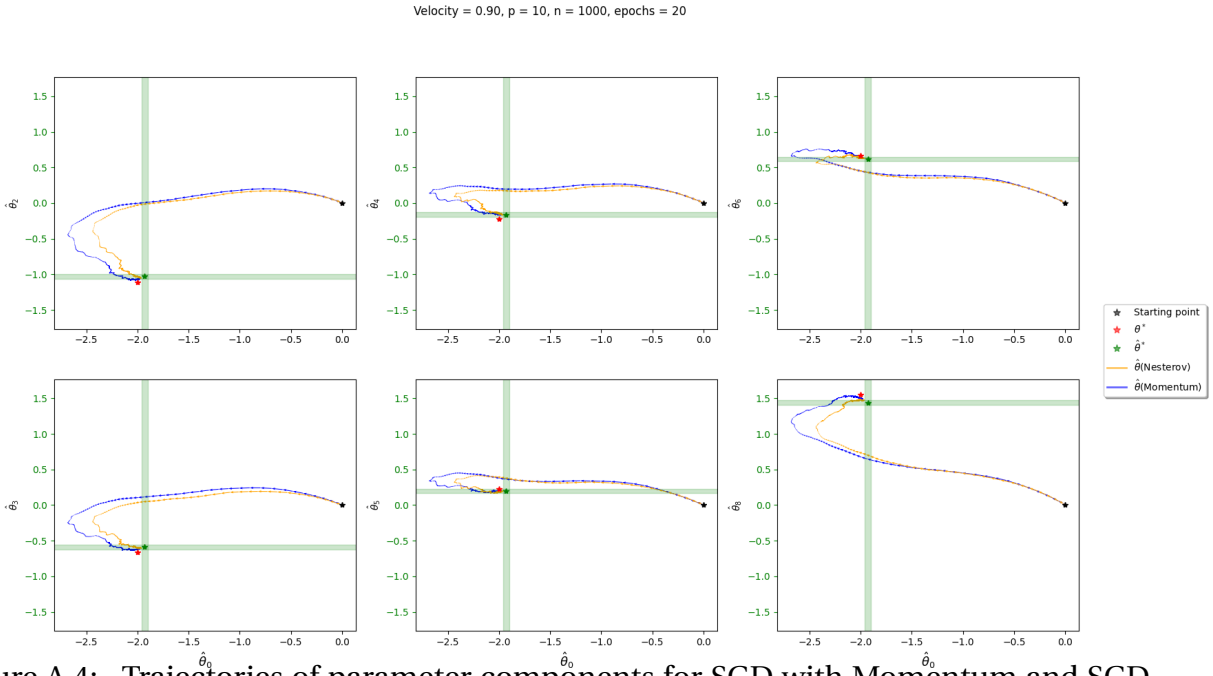

Figure A.4: Trajectories of parameter components for SGD with Momentum and SGD with Nesterov Momentum under an L2-regularized logistic loss function and a design matrix with an equi-correlated covariance structure ($\hat{\theta}_0 = \mathbf{0}$). Results depict the parameter trajectories averaged over 200 datasets.

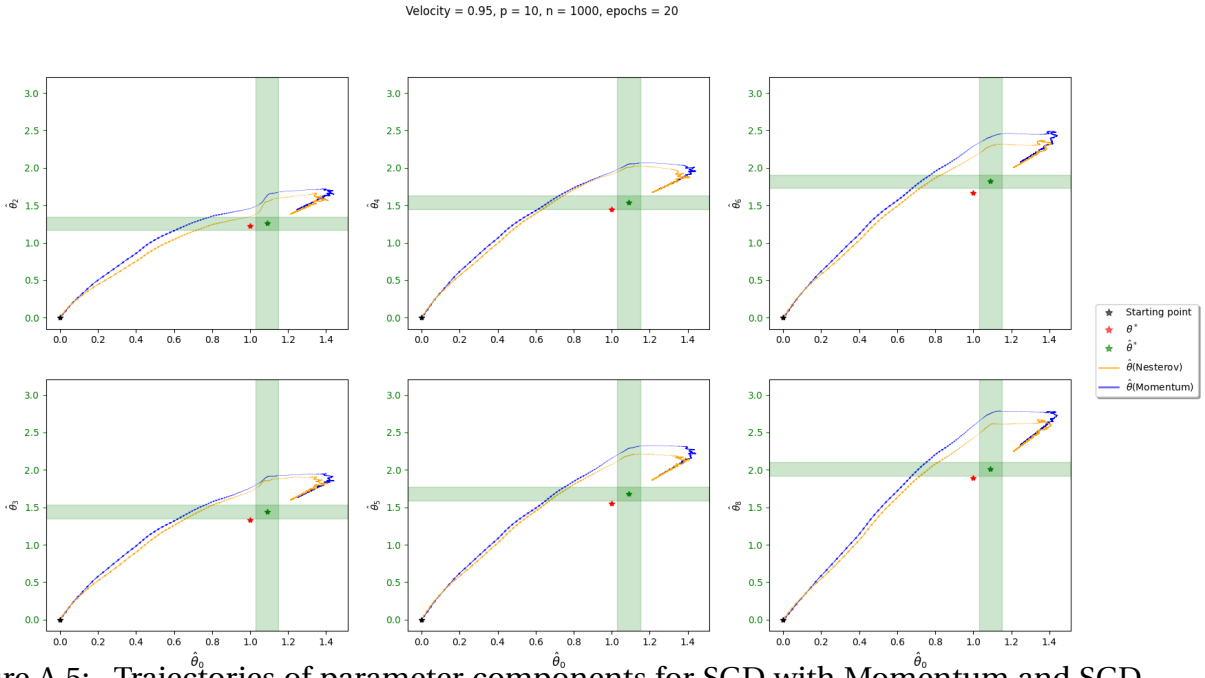

Figure A.5: Trajectories of parameter components for SGD with Momentum and SGD with Nesterov Momentum under an L2-regularized logistic loss function and a design matrix with an AR1 covariance structure ($\hat{\theta}_0 = \mathbf{0}$). Results depict the parameter trajectories averaged over 200 datasets.

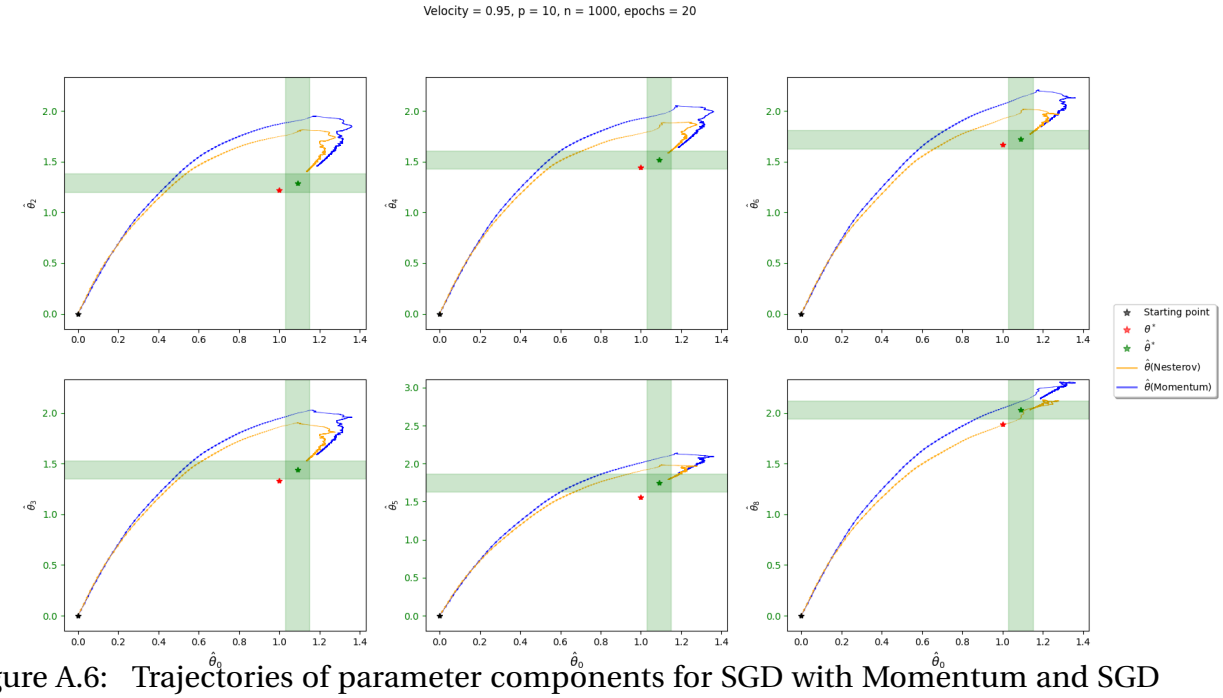

Figure A.6: Trajectories of parameter components for SGD with Momentum and SGD with Nesterov Momentum under an L2-regularized logistic loss function and a design matrix with an equi-correlated covariance structure ($\hat{\theta}_0 = \mathbf{0}$). Results depict the parameter trajectories averaged over 200 datasets.

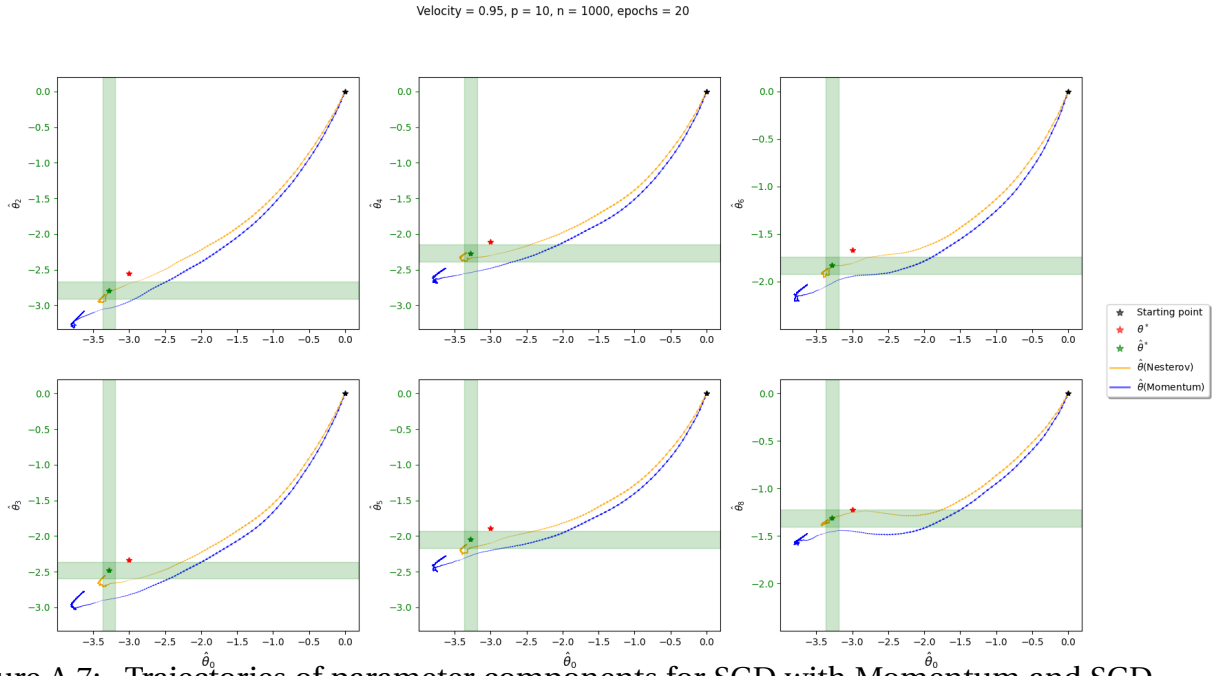

Figure A.7: Trajectories of parameter components for SGD with Momentum and SGD with Nesterov Momentum under an L2-regularized logistic loss function and a design matrix with an AR1 covariance structure ($\hat{\theta}_0 = \mathbf{0}$). Results depict the parameter trajectories averaged over 200 datasets.

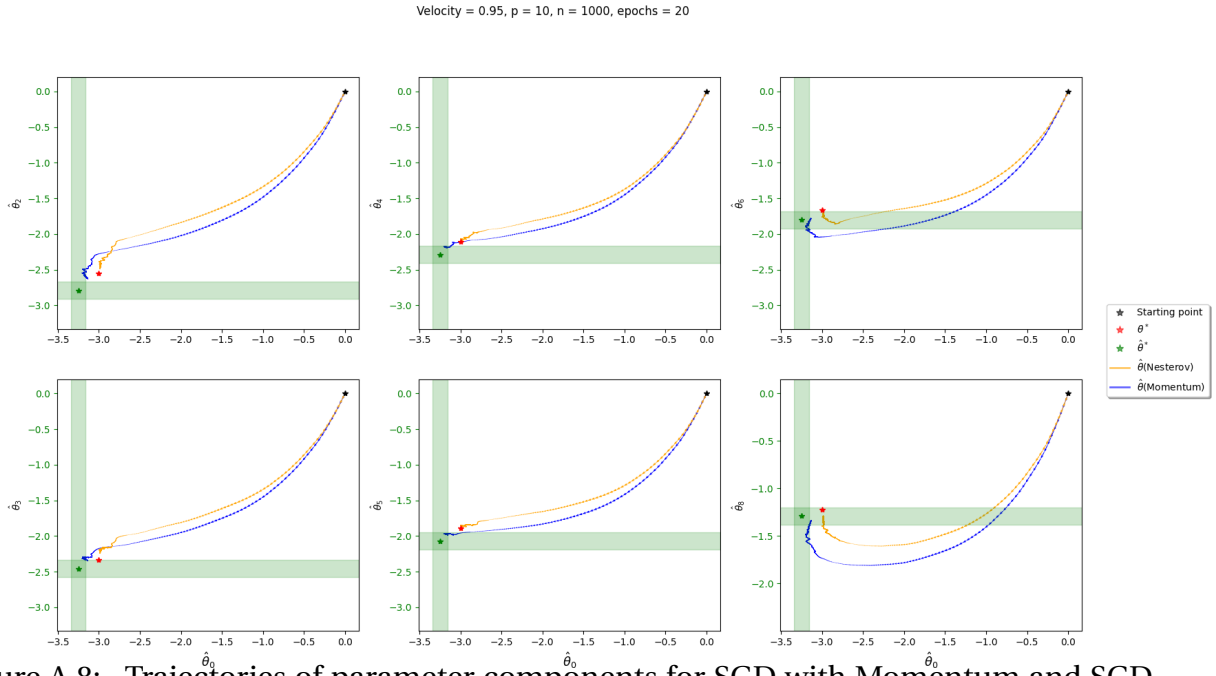

Figure A.8: Trajectories of parameter components for SGD with Momentum and SGD with Nesterov Momentum under an L2-regularized logistic loss function and a design matrix with an equi-correlated covariance structure ($\hat{\theta}_0 = \mathbf{0}$). Results depict the parameter trajectories averaged over 200 datasets.

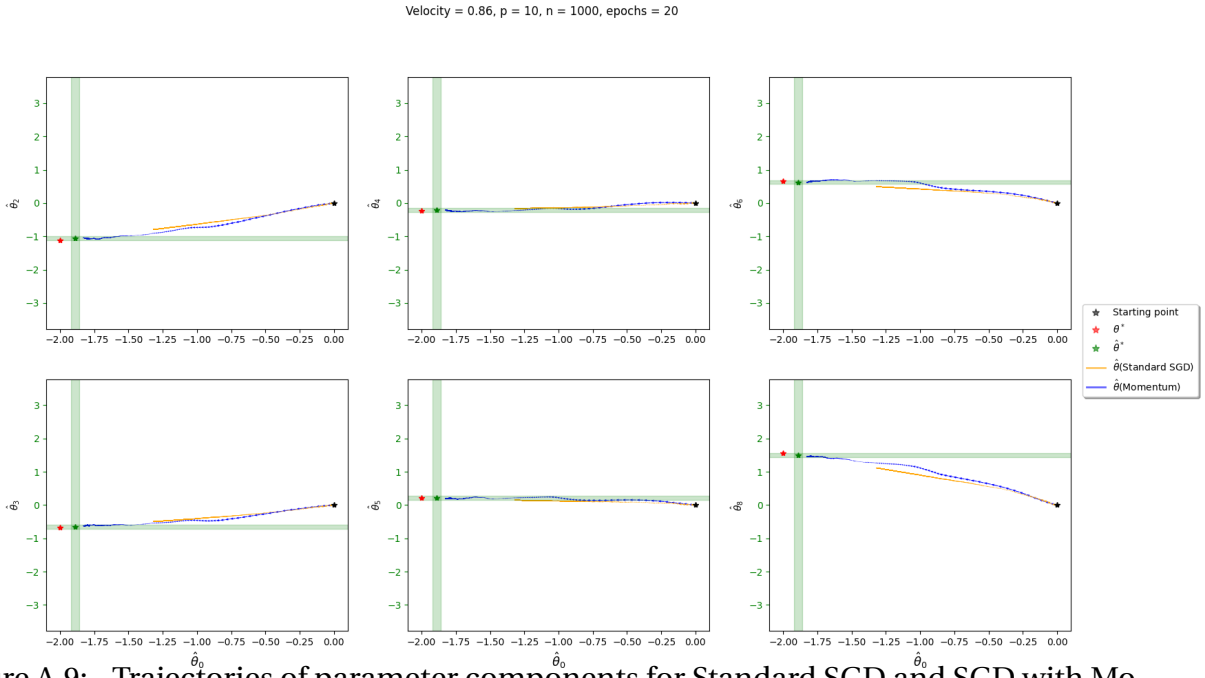

Figure A.9: Trajectories of parameter components for Standard SGD and SGD with Momentum under an L2-regularized logistic loss function and a design matrix with an AR1 covariance structure ($\hat{\theta}_0 = \mathbf{0}$). Results depict the parameter trajectories averaged over 200 datasets.

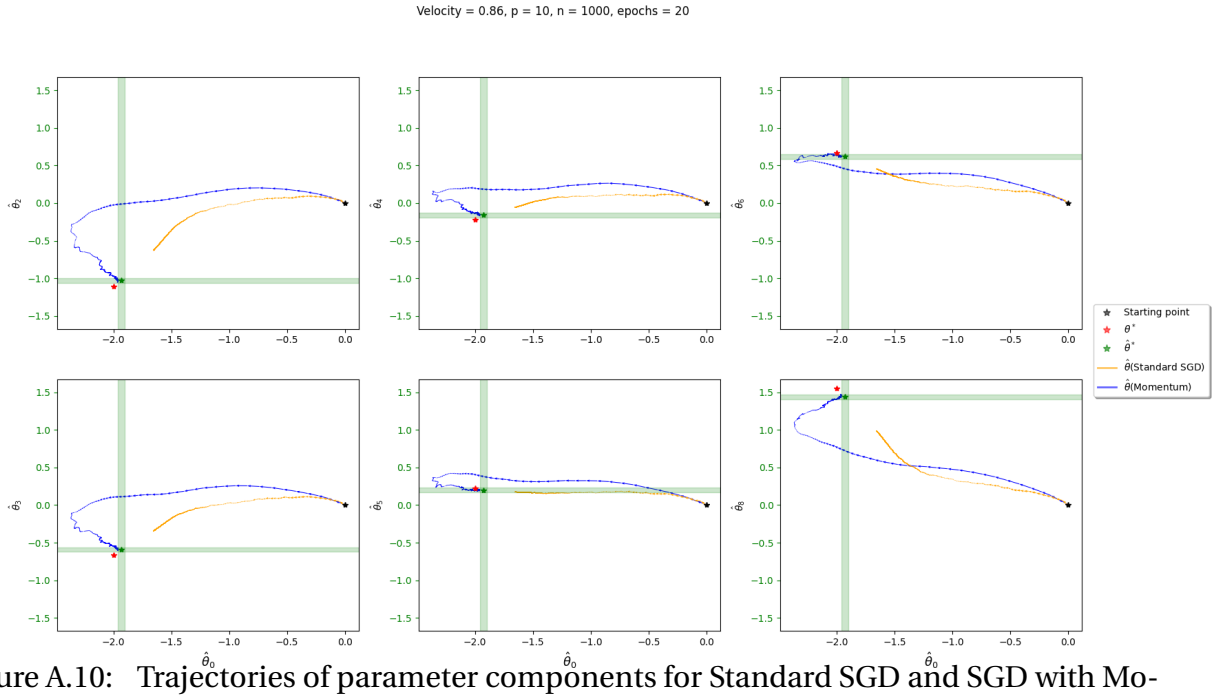

Figure A.10: Trajectories of parameter components for Standard SGD and SGD with Momentum under an L2-regularized logistic loss function and a design matrix with an equicorrelated covariance structure ($\hat{\theta}_0 = \mathbf{0}$). Results depict the parameter trajectories averaged over 200 datasets.

On the whole, it can be observed that the choice of the velocity parameter is essential to the performance of SGD with Momentum since a relatively small velocity parameter results in slow progress toward the optimum, while too-large a velocity results in overshooting the mark. Also, a good choice of the velocity parameter appears to be problem dependent and thus, it is a parameter that may require tuning. It is worth noting that when the velocity parameter is relatively large resulting in overshooting of the desired goal, standard momentum overshoots by a larger margin than Nesterov momentum, and thus has a farther distance to travel in backtracking toward the optimum.

APPENDIX

B

ADDITIONAL NOTES AND RESULTS ON OPTIMAL DESIGNS WITH MULTI-COMPONENT TREATMENTS

B.0.1 Outline of Stochastic Search Procedure

To jump-start the optimization, we begin with an initial treatment allocation to subjects that consists of a random assignment with replacement from the set of 2^k possible treatment combinations (k is the number of treatment factors), after which the stochastic search is used to find the optimal allocation of treatments to patients. A basic description of the stochastic search procedure is as follows.

1. Randomly assign treatments to subjects from the set of treatments.
2. While the stopping criterion has not been achieved:
    - Permute the subjects' indices $\{1, ..., n\}$.

- For $i = 1, ..., n$:
  - Swap treatment combination of subject i with alternative options in the treatment set while keeping treatment combinations of all other subjects fixed.
  - Select the treatment combination for subject i for which the criterion is least.

The search is terminated when there are no more changes to be made by swapping the treatment of any subject.

For our simulations, we use the log of the determinant for reasons of numerical stability and to facilitate the selection of the range of values of λ to explore. We also use the Xpress software for finding the parameter vector in the set of allowable configurations that maximizes the variance of the test statistic under Hypothesis 3. This is a Mixed Non-Linear Integer Programming problem and often results in solutions that are locally optimal.

B.0.2 More Results on Hypotheses 2 and 3

Table B.1 and Figure B.1 show some additional results for Hypothesis Test 2, i.e. $H_{0,2}$: $\phi(\mathbf{a}_0)^T \beta^* \leq \mu_C$. In the cases presented, μ_C is fixed at 0.3. It can be observed that the optimal allocation of treatments results in higher power relative to the random allocation. The impact of the magnitude of the λ values is seen in the final log determinant values, in that, higher values of λ lead to smaller log determinants after optimization.

Table B.2 shows more results for $H_{0,3} : \max_{a \in \mathscr{A}} \phi(\mathbf{a})^T \beta^* \leq \mu_C$. In these cases as well, μ_C is set to 0.3. In general, we can observe a similar influence of the magnitude of the λ values on the log determinants as is observed in all the results previously shown. The results show that the power of the test is greater when treatments are allocated optimally.

Table B.1: Averaged results over 200 Monte Carlo replicates for $H_{0,2}$; Number of treatment factors, k, and number of covariates, ℓ, are 8 and 4 respectively; LogDet refers to the log of the determinant of $\{n^{-1}\Psi(\mathbf{X},\mathbf{A})^T\Psi(\mathbf{X},\mathbf{A})\}^{-1}$.

Sample size	Effect size	λ	Initial Power	Initial Type I Error	Initial LogDet	Final Power	Final Type I Error	Final LogDet	Avg.Time (mins.)
250	0.5	0.05	0.33	0.06	154.09	1.0	0.03	164.94	329
250	0.5	0.1	0.33	0.06	154.09	1.0	0.02	157.73	323
250	0.5	0.5	0.33	0.06	154.09	0.97	0.05	148.67	317
250	0.5	1	0.33	0.06	154.09	0.93	0.03	146.58	340
250	0.5	2	0.33	0.06	154.09	0.78	0.05	145.18	382
500	0.5	0.05	0.55	0.06	147.02	1.0	0.07	163.15	829
500	0.5	0.1	0.55	0.06	147.02	1.0	0.05	156.0	708
500	0.5	0.5	0.55	0.06	147.02	1.0	0.09	147.03	688
500	0.5	1	0.55	0.06	147.02	1.0	0.04	144.98	704
500	0.5	2	0.55	0.06	147.02	1.0	0.08	143.58	701

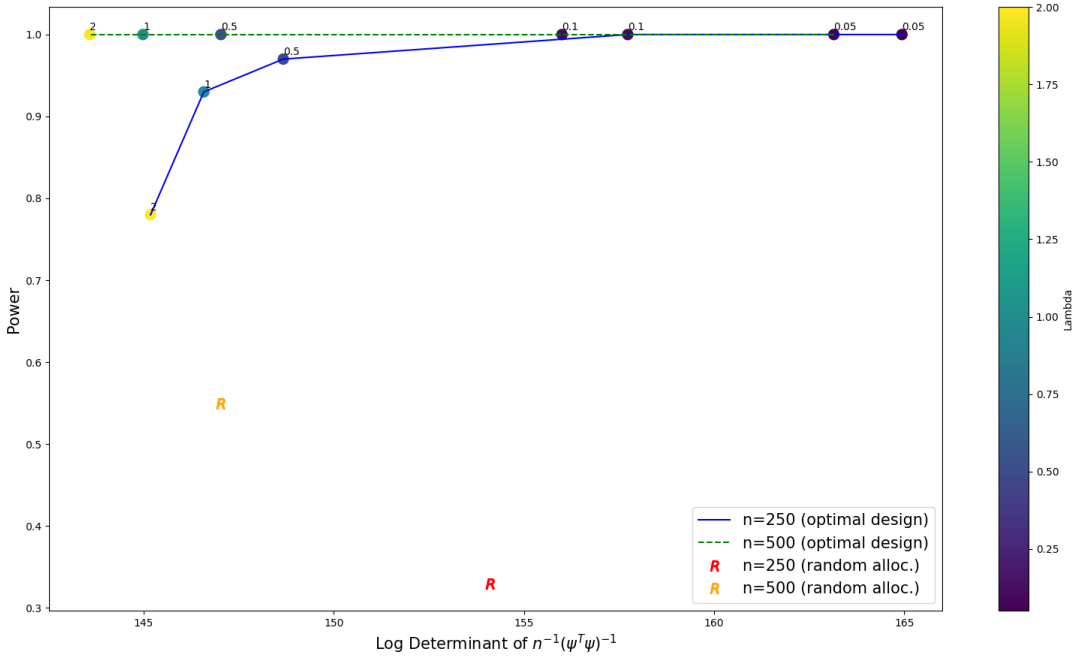

Figure B.1: Power versus Log Determinant of $\{n^{-1}\Psi(\mathbf{X},\mathbf{A})^T\Psi(\mathbf{X},\mathbf{A})\}^{-1}$ for different values of λ. Results are averaged over 200 Monte Carlo replicates. The solid and dashed lines correspond to sample sizes of 250 and 500, respectively. (Text above each point is the value of λ used in the composite criterion; $k=8$ and $\ell=4$)

Table B.2: Averaged results over 200 Monte Carlo replicates for $H_{0,3}$; Number of treatment factors, k, and number of covariates, ℓ, are 3 and 2 respectively; LogDet refers to the log of the determinant of $\{n^{-1}\Psi(\mathbf{X},\mathbf{A})^T\Psi(\mathbf{X},\mathbf{A})\}^{-1}$.

Sample size	Effect size	λ	Initial Power	Initial Type I Error	Initial LogDet	Final Power	Final Type I Error	Final LogDet	Avg.Time (mins.)
50	0.8	0.005	0.56	0.06	25.16	0.87	0.04	22.63	781
50	0.8	0.01	0.56	0.06	25.16	0.86	0.04	22.63	863
50	0.8	0.5	0.56	0.06	25.16	0.84	0.05	22.62	922
50	0.8	1	0.56	0.06	25.16	0.84	0.03	22.62	1007
50	0.8	5	0.56	0.06	25.16	0.83	0.04	22.62	927
50	0.5	1×10^{-4}	0.29	0.06	25.16	0.44	0.06	23.32	448
50	0.5	2.4×10^{-4}	0.29	0.06	25.16	0.38	0.07	23.06	423
50	0.5	2.8×10^{-4}	0.29	0.06	25.16	0.43	0.04	23.02	486
50	0.5	1.4×10^{-3}	0.29	0.06	25.16	0.39	0.03	22.70	547