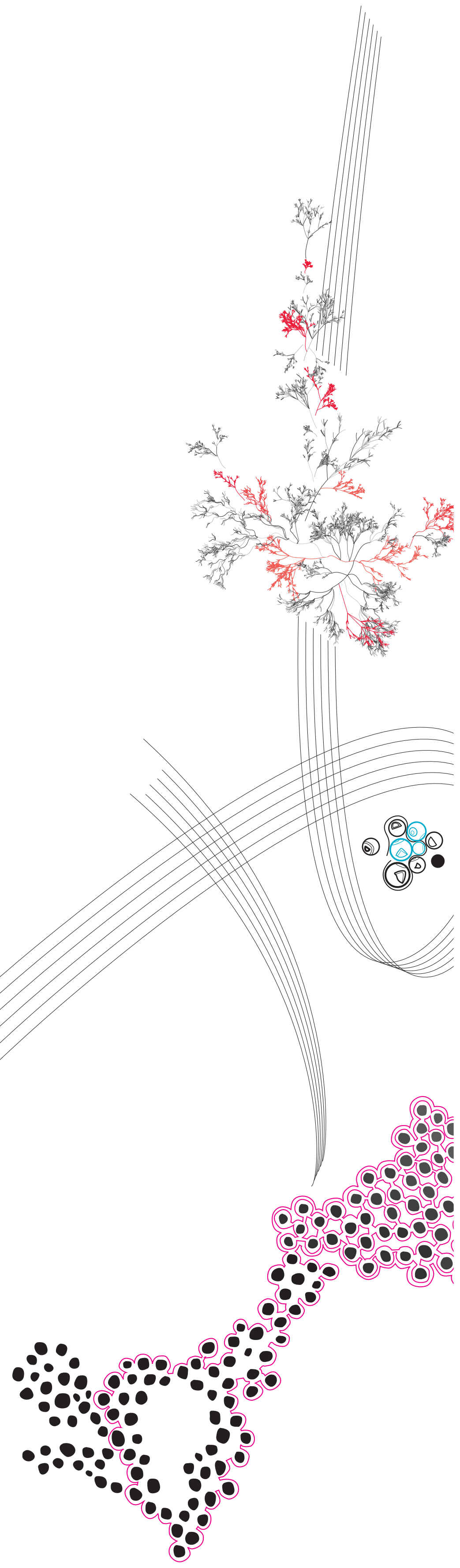


BSc Thesis Applied Mathematics

Gradient Descent with Random Minibatches in the Linear Model

Yazan Mash'al

Supervisors: G. Clara, J. Schmidt-Hieber, A. Faizan

June, 2024

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Acknowledgements

I am grateful to everyone who supported and guided me through the completion of this bachelor thesis. I want to thank Gabriel Clara. His unwavering support, keen insights, and patient mentorship have been indispensable throughout this journey. His exceptional guidance has not only enriched this research but also significantly shaped my academic growth and understanding of the field.

I would also like to extend my sincere thanks to Dr. Johannes Schmidt-Hieber and Dr. Ahmed Faizan. Their expertise and feedback in the theoretical and practical underpinnings of this paper have been invaluable, providing additional perspectives and enhancing the quality of this work. I appreciate their contributions and the opportunities they offered for thoughtful discussions.

Furthermore, I am deeply thankful to my family and friends, whose constant encouragement and belief in my abilities have been a tremendous source of strength. Your support has been instrumental in navigating the challenges of this project.

Lastly, I wish to acknowledge the body of existing research and other scholars' pioneering work in machine learning and optimization. Their foundational contributions have been integral to the development of this thesis.

Gradient Descent with Random Minibatches in the Linear Model

Yazan Mash'al*

June, 2024

Abstract

This thesis explores the dynamics and convergence properties of gradient descent algorithms when integrated with dropout techniques and random minibatch sampling in the context of linear regression. Dropout, a form of regularization, and minibatch gradient descent are both crucial in developing robust machine learning models that generalize well to unseen data. This paper aims to showcase the theoretical properties and practical implications of using these two methods.

Keywords: Gradient Descent, Dropout, Random Minibatch Sampling, Linear Regression, Machine Learning, Optimization

1 Introduction

In machine learning, regularization is used for developing models that are not only predictive but also robust and generalizable to unseen data. Regularization techniques, such as lasso (L_1) regression and ridge (L_2) regression, are used to prevent models from overfitting by penalizing the coefficients associated with each feature in the model. These regularization methods, well-established in the statistical literature [3], focus on controlling the model complexity through direct penalties on the coefficients. In contrast, modern algorithmic regularization methods, such as dropout, originate from the machine learning domain and are less understood but effective in practice. This distinction highlights a qualitative difference in how the two communities — statistics and machine learning — approach regularization.

Despite the effectiveness of L_1 and L_2 regularization, these techniques have limitations, especially when dealing with high-dimensional data. High-dimensional datasets are characterized by many features, which require complex models to capture the underlying patterns. As the number of features increases, the volume of the feature space grows exponentially. Specifically, each additional feature increases the dimensionality of the dataset, vastly expanding the search space for potential solutions.

In light of these challenges, alternative approaches to regularization have been explored. One notable method, introduced by [12], is the use of dropout as a form of regularization that involves randomly omitting units from a neural network during training. This technique can be viewed as adding noise to the training process, preventing units from co-adapting too closely.

*Email: y.mashal@student.utwente.nl

While dropout can be compared to L_2 regularization in certain contexts, particularly in linear regression models, it induces a more complex form of regularization known as the ℓ_2-path norm in deep linear networks, as discussed in Proposition 2.1 of [7]. This regularizer is distinct from the traditional L_2-penalty and highlights the nuanced effect of dropout in deeper and more complex networks.

This thesis aims to delve deeper into the mechanism of dropout and its implications for model training and regularization. By exploring the theoretical underpinnings of dropout and its practical applications, this paper seeks to showcase how the introduction of noise through dropout can mimic traditional L_2 regularization effects and the conditions that lead to statistically optimal outcomes. Furthermore, this research extends the investigation to the convergence properties of gradient descent with minibatches and dropout in the context of linear regression models. Such an aspect is crucial for understanding how the dynamics of model training change with the incorporation of minibatches, a common practice in the training of large-scale machine learning models. Through thoroughly examining these themes, this thesis contributes to the broader understanding of regularization techniques in machine learning, offering insights into how models can be made more generalisable and less prone to over-fitting in the face of high-dimensional data challenges.

1.1 Outline

The research begins by providing an introduction to dropout, including its various schemes and how it influences the optimization process within neural networks. We then go into the specifics of gradient descent augmented with dropout, highlighting the impact on model regularization and convergence.

Following this, we examine the effects of integrating random minibatch sampling with gradient descent and dropout. Through mathematical analysis and Monte Carlo simulations, we demonstrate how these techniques affect the convergence speed and stability of gradient descent in specific settings.

The primary aim in Section 5 is to simulate model averaging procedures traditionally performed with Markov Chain Monte Carlo (MCMC) methods, without the substantial computational overhead these methods typically entail. MCMC methods can be computationally intensive and impractical for very large datasets and complex models, such as those frequently encountered in deep learning. By simulating model averaging through dropout, this method provides a computationally efficient way to integrate out over different network architectures, promoting the development of robust features. This approach has shown promise, particularly in the training of deep neural networks, where it encourages the model to learn features that are useful across many different random subsets of the other neurons.

Our findings reveal that incorporating dropout with minibatch sampling (random minibatch sampling) mitigates the risk of overfitting and improves the convergence behaviour of gradient descent. The simulations in Section 5 validate the theoretical models, showing that dropout and minibatch sampling effectively regularize the training process and reduce model variance.

This thesis contributes to the broader understanding of regularization and optimization in machine learning, offering insights into how these techniques can be effectively combined to tackle the challenges posed by high-dimensional data. Future work will explore extending these methodologies to more complex models and larger datasets, aiming to further enhance their applicability and effectiveness in real-world scenarios.

1.2 Research questions

The main research question of this paper is: "How does the integration of random minibatches in gradient descent affect the convergence behaviour and covariance structure of the parameter estimates?"

To effectively explore this main question, we will break it down into several sub-research questions that are more focused and manageable within the scope of this paper.

Firstly, we will address the foundational question: "What is dropout, and how does it work?". This question, in Section 2, provides an opportunity to introduce the dropout technique, discussing its theoretical foundation and implementation in the training of neural networks.

Secondly, we will examine the relationship between dropout and gradient descent by asking: "How does dropout affect the convergence speed of gradient descent in the linear model?". This exploration, in Section 3, will focus on understanding how the inclusion of dropout influences the gradient descent process to converge, and how different dropout rates impact this convergence.

Lastly, we will extend our investigation to consider the combined effects of dropout and minibatches with the question: "How does random minibatch sampling affect the convergence behaviour and covariance structure of the parameter estimates in the linear model?". Here, in Section 4, we will analyze whether combining dropout with minibatch training enhances the convergence of the parameter estimates.

These questions aim to provide a comprehensive understanding of how dropout, especially when integrated with minibatches, influences the training dynamics and performance of gradient descent in the linear model. We extend our validation in Section 5 to neural networks, specifically in the context of the MNIST dataset.

1.3 Literature review

This review will cover the concept of dropout, its theoretical foundations, variations and implementations, as well as its impact on the field of deep learning.

Dropout, introduced by [12], emerged as a regularization technique designed to reduce the overfitting phenomenon in deep learning models. Unlike classical regularization methods, dropout provides a novel approach by randomly deactivating a subset of neurons during the training process, thereby reducing the model's reliance on any single neuron and encouraging a more distributed and robust feature representation.

At its core, dropout operates under the following principle: during each training iteration, a random fraction of the nodes (neurons) in the network layers are "dropped out," meaning they are temporarily removed along with all their incoming and outgoing connections [12]. This randomness introduces a form of noise that helps to break up co-adaptations of neurons, forcing the network to learn more robust features that are not dependent on specific neural pathways. The mathematical formulation of dropout involves adjusting the network's weights during training, which can be viewed as training a "thinned" network, offering insights into its effectiveness as a regularization strategy.

The adaptability of the dropout concept has led to the development of several variants aimed at optimizing its regularization capabilities across different neural network architectures. These include DropConnect in [13], DropBlock in [2], and selective dropout techniques, each tailored to specific network structures or training challenges. For instance, DropConnect modifies the connectivity pattern within the network, while DropBlock focuses on dropping contiguous regions of feature maps in convolutional neural networks (CNNs). These variations underscore the flexibility of dropout as a regularization concept, capable of being customized to enhance model performance across a wide array of deep learning applications.

The introduction of dropout has had a profound impact on the field of deep learning, significantly advancing the state-of-the-art (SOTA) across various domains [6]. An important contribution in the recognition of dropout's potential was its application in the AlexNet model, which achieved a high performance in the ImageNet challenge, as detailed in [5]. This work generated significant interest in dropout due to its role in enhancing model performance in large-scale image classification tasks.

The effectiveness of dropout in improving model generalization has been demonstrated through empirical research, including performance comparisons using benchmark datasets, as done in [12]. Dropout techniques have been shown to enhance the robustness of models against input noise and variance, contributing to the development of more reliable and stable deep learning systems.

Moreover, the exploration of dropout in various network architectures, such as recurrent neural networks (RNNs) and transformers, highlights its versatility. Dropout's potential to improve model performance extends beyond traditional feedforward networks, offering benefits across a wide range of deep learning applications.

1.4 Notation and definitions

Column vectors are denoted in bold, e.g., $\mathbf{x} = (x_1, \ldots, x_d)^\top$. The Euclidean norm of a vector $\mathbf{x}$ is defined as $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^\top \mathbf{x}}$, but since we are only going to use the Euclidean norm, the subscript is going to be emitted for brevity. The $d \times d$ identity matrix is denoted by I_d, or simply I when the dimension d is clear from the context. For a given square matrix $A \in \mathbb{R}^{n \times n}$, $\operatorname{Diag}(A)$ extracts the diagonal elements of A and forms a diagonal matrix D. If $A = [a_{ij}]$, then $\operatorname{Diag}(A) = D$ is defined as:

$$D = \operatorname{Diag}(A) \quad \text{where} \quad D_{ij} = \begin{cases} a_{ii} & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

Additionally, we define $\overline{A} = A - \operatorname{Diag}(A)$. For a scalar $p \in (0, 1)$, the matrix A_p is defined as $A_p = pA + (1 - p) \operatorname{Diag}(A)$. This expression rescales the off-diagonal entries of A by p while retaining the diagonal entries. The smallest eigenvalue of a symmetric matrix A is denoted by $\lambda_{\min}(A)$. We also define the Gram matrix, denoted by $\mathbb{X} := X^\top X$.

The operator norm of a linear operator $T : V \to W$ between normed linear spaces is given by $\|T\|_{\text{op}} = \sup_{\|v\|_V \leq 1} \|Tv\|_W$. The spectral norm, which is also known as the operator norm for matrices, is denoted $\|\cdot\|$ and equals the maximum singular value.

2 What is dropout?

Dropout is a regularisation technique used in the training of machine learning models, which was first introduced in [12]. It involves adding noise during the training epoch to reduce the generalisation error. In an abstract sense, dropout can be considered as approximating a Bayesian sampling method, where all the sub-graphs of the neural network are averaged and the final model is assembled [12]. An example of dropping neurons during the training epoch and its effect on the network can be seen in Figure 1.

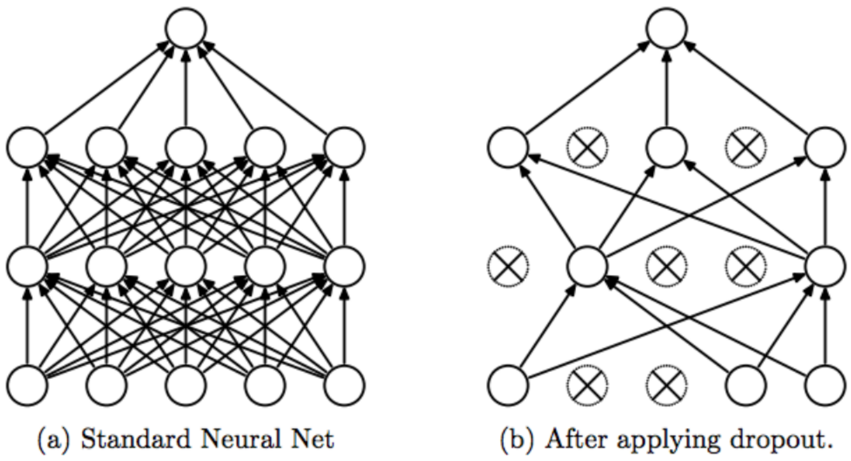


FIGURE 1: Dropout applied to a neural network. Panel (a) shows a standard neural network, and panel (b) shows a neural network after applying dropout (Adapted from [12])

2.1 Description of the neural network model with dropout

This section describes the dropout neural network model. More formally, as presented in [12], consider a neural network with L hidden layers. Let $l \in \{1, \dots, L\}$ index the hidden layers of the network. Denote by $\mathbf{z}^{(l)}$ the vector of inputs into layer l, and by $\mathbf{Y}^{(l)}$ the vector of outputs from layer l (note that $\mathbf{Y}^{(0)} = x$ is the input). $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ represent the weights and biases at layer l, respectively. The feed-forward operation of a standard neural network can be described as, for $l \in \{0, \dots, L-1\}$ and any hidden unit i,

$$\begin{aligned} \mathbf{z}_i^{(l+1)} &= \mathbf{W}_i^{(l+1)}\mathbf{Y}^{(l)} + b_i^{(l+1)}, \\ \mathbf{Y}_i^{(l+1)} &= f(\mathbf{z}_i^{(l+1)}), \end{aligned}$$

where f is an activation function, for instance, $f(x) = \frac{1}{1+\exp(-x)}$. With dropout, the feed-forward operation becomes:

$$\begin{aligned} r_j^{(l)} &\sim \text{Bernoulli}(p), \\ \tilde{y}^{(l)} &= r^{(l)} \circ \mathbf{Y}^{(l)}, \\ z_i^{(l+1)} &= W_i^{(l+1)}\tilde{y}^{(l)} + b_i^{(l+1)}, \\ \mathbf{Y}_i^{(l+1)} &= f(z_i^{(l+1)}), \end{aligned}$$

where $\circ$ denotes the element-wise product, and $r^{(l)}$ is a vector of Bernoulli random variables, which are used to "drop" (zero out) elements of $\mathbf{Y}^{(l)}$. However, given that this model is beyond the scope of this paper, we are going to observe the emerging properties in a linear regression model instead.

3 Gradient descent with dropout in linear regression

We commence by examining a linear regression model characterized by a fixed $n \times d$ design matrix X and an $n \times 1$ outcome vector $\mathbf{Y}$, shown by the relationship:

$$\mathbf{Y} = X\boldsymbol{\beta}^* + \boldsymbol{\epsilon}, \tag{1}$$

where $\boldsymbol{\beta}^*$ is an $d \times 1$ vector that symbolizes the unknown parameter vector that characterizes the regression model and $\boldsymbol{\epsilon}$ is normally distributed noise with $\mathbb{E}[\boldsymbol{\epsilon}] = 0$ and $\mathrm{Cov}(\boldsymbol{\epsilon}) = I_n$. The objective is to deduce $\boldsymbol{\beta}^*$ from the given data $(X, \mathbf{Y})$. Should X incorporate a null column, its corresponding regression coefficient would be non-influential to $\mathbf{Y}$. Thus, excluding both the null column and its associated coefficient simplifies the model.

The estimator of $\boldsymbol{\beta}^*$, $\tilde{\boldsymbol{\beta}}$, uses the following least squares criterion as the cost function:

$$\boldsymbol{\beta} \mapsto \frac{1}{2}||\mathbf{Y} - X\boldsymbol{\beta}||^2.$$

Using a constant learning rate $\alpha > 0$, gradient descent iteratively refines the estimate according to the following update function:

$$\tilde{\boldsymbol{\beta}}_{k+1} = \tilde{\boldsymbol{\beta}}_k - \alpha\nabla_{\tilde{\boldsymbol{\beta}}_k}\frac{1}{2}||\mathbf{Y} - X\tilde{\boldsymbol{\beta}}_k||^2 = \tilde{\boldsymbol{\beta}}_k + \alpha X^\top(\mathbf{Y} - X\tilde{\boldsymbol{\beta}}_k), \tag{2}$$

where k indexes the iteration steps, initiating from a potentially stochastic baseline $\boldsymbol{\beta}_0$.

Dropout, as shown in [12], modifies the standard gradient descent by incorporating gradients from a stochastically reduced model in each iteration. We define a $d \times d$ diagonal dropout matrix D as a p-dropout matrix, where $D_{ii} \sim \mathrm{Bernoulli}(p)$ for $p \in (0, 1)$. This results in approximately pd entries of D being 1, with the remainder zero. In theory, the dropout methodology aims to approximate Bayesian model averaging across all possible network configurations, as discussed in [12].

To implement dropout, we introduce a sequence of *i.i.d.* dropout matrices $\{D_k\}$, where D_k represents the matrix used in the k^{th} iteration. The gradient descent update function in (2), when incorporated with dropout for any vector $\tilde{\boldsymbol{\beta}}_{k+1}$ under a dropout matrix D_{k+1}, becomes:

$$\tilde{\boldsymbol{\beta}}_{k+1} = \tilde{\boldsymbol{\beta}}_k - \alpha\nabla_{\tilde{\boldsymbol{\beta}}_k}\frac{1}{2}||\mathbf{Y} - XD_{k+1}\tilde{\boldsymbol{\beta}}_k||^2 = \tilde{\boldsymbol{\beta}}_k + \alpha D_{k+1}X^\top(\mathbf{Y} - XD_{k+1}\tilde{\boldsymbol{\beta}}_k), \tag{3}$$

indicating the adoption of a model reduced by D_{k+1}.

3.1 The different dropout schemes

We are going to mention two dropout approaches, as was discussed in [1]: one applies dropout directly within the gradient descent step, and another applies dropout to the gradient before the update step. The first method alters the design matrix X to XD_{k+1} in iteration $k + 1$, affecting the gradient computation by the randomness of D_{k+1}. The latter, simplified dropout, only modifies the gradient post-computation with D_{k+1}.

When the columns of X are orthogonal, the dropout approaches coincide, as diagonal matrices commute. The effectiveness of dropout, motivated from a model averaging perspective, aims at approximating Bayesian model averaging over network configurations. This method has been supported by empirical successes across various network architectures and tasks.

3.2 Averaged dropout in gradient descent

In the linear regression framework introduced in (1), the implementation of dropout aims to reduce over-fitting by randomly reducing the model complexity during training iterations. While the end goal remains to minimize the original loss function $\boldsymbol{\beta} \mapsto \mathbb{E}[\|\mathbf{Y} - X\boldsymbol{\beta}\|^2|\mathbf{Y}]$, the introduction of dropout transforms this into a regularized surrogate loss. The expected loss to be minimized is given by

$$\boldsymbol{\beta} \mapsto \mathbb{E}\left[\|\mathbf{Y} - XD\boldsymbol{\beta}\|^2 \,\middle|\, \mathbf{Y}\right], \tag{4}$$

where D represents the dropout mask applied to the matrix X. The dropout mask D varies across iterations, introducing stochasticity into the training process. Contrary to the initial expectation that the law of large numbers would average out the stochastic effects of dropout, it is observed that the process does not simply converge to a minimizer of the expected loss in the traditional sense [1]. Instead, the randomness induced by dropout leads to a more complex form of regularization, shaping the optimization landscape in a way that encourages better generalization.

For a detailed discussion on how dropout affects the convergence and optimization process, refer to Section 3 in [1]. This updated discussion provides insights into why the averaging effect of the law of large numbers does not straightforwardly apply and how dropout instead alters the path and properties of the convergence in a nuanced manner.

3.2.1 Analysis of the averaged loss with dropout

Consider the Gram matrix $\mathbb{X} := X^\top X$, then the squared norm under dropout can be expressed as:

$$\|\mathbf{Y} - XD\boldsymbol{\beta}\|^2 = \|\mathbf{Y}\|^2 - 2\mathbf{Y}^\top XD\boldsymbol{\beta} + \boldsymbol{\beta}^\top D\mathbb{X}D\boldsymbol{\beta}. \tag{5}$$

Taking the expectation over D, and noting that $\mathbb{E}[D] = pI$ and

$$\begin{aligned} \mathbb{E}[DAD] &= \mathbb{E}\left[D\overline{A}D + D\mathrm{Diag}(A)D\right] \\ &= p^2\overline{A} + p\mathrm{Diag}(A) = p\left(pA - p\mathrm{Diag}(A) + \mathrm{Diag}(A)\right) = pA_p. \end{aligned} \tag{6}$$

Here, we have that $A_p = pA + (1-p)\,\mathrm{Diag}(A)$. We now arrive at:

$$\mathbb{E}\left[\|\mathbf{Y} - XD\boldsymbol{\beta}\|^2\right] = \|\mathbf{Y}\|^2 - 2p\mathbf{Y}^\top X\boldsymbol{\beta} + p\boldsymbol{\beta}^\top\mathbb{X}\boldsymbol{\beta} + p(1-p)\boldsymbol{\beta}^\top\mathrm{Diag}(\mathbb{X})\boldsymbol{\beta}. \tag{7}$$

This formulation aligns with a Tikhonov regularization framework, asserting that the minimizer $\boldsymbol{\beta}$ can be determined by differentiating the averaged loss to find:

$$\tilde{\boldsymbol{\beta}} := \underset{\boldsymbol{\beta}\in R^d}{argmin}\ \mathbb{E}\left[\|\mathbf{Y} - XD\boldsymbol{\beta}\|^2|\mathbf{Y}\right] = \mathbb{X}_p^{-1}X^\top\mathbf{Y}, \tag{8}$$

where $\mathbb{X}_p := p\mathbb{X} + (1-p)\mathrm{Diag}(\mathbb{X})$ acts as the effective design matrix under dropout. This result provides an insightful interpretation of how dropout influences the design matrix, effectively averaging the original matrix $\mathbb{X}$ with its diagonal component weighted by $(1-p)$ and off-diagonal entries by p. For a detailed analysis of this estimator and its properties, refer to Section 2 of [1], which studies this estimator comprehensively.

3.3 Analysis of iterative dropout schemes

Gradient descent within the linear model framework, particularly with a small yet fixed learning rate as previously discussed, ensures exponential convergence over iterations. This section delves into the details of iterative dropout schemes and how convergence comes about with a fixed learning rate α.

3.4 Convergence analysis of gradient descent with dropout

We examine the statistical characteristics of the dropout approach shown in (3):

$$\tilde{\boldsymbol{\beta}}_{k+1} = \tilde{\boldsymbol{\beta}}_k + \alpha D_{k+1} X^\top (\mathbf{Y} - X D_{k+1} \tilde{\boldsymbol{\beta}}_k).$$

We are first going to rewrite this update equation in order to facilitate the study of its characteristics. Utilizing the idempotent nature of D_k, and recognizing the commutative property of diagonal matrices, we have that:

$$\tilde{\boldsymbol{\beta}}_{k+1} - \tilde{\boldsymbol{\beta}} = (I - \alpha D_{k+1} \mathbb{X} D_{k+1})(\tilde{\boldsymbol{\beta}}_k - \tilde{\boldsymbol{\beta}}) + \alpha D_{k+1} \overline{\mathbb{X}} (pI - D_{k+1}) \tilde{\boldsymbol{\beta}}.$$

Herein, the secondary term inherently has a zero mean, independent across iterations, attributable to the stochastic independence of D_{k+1} and the vector pair $(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\beta}}_{k+1})$.

3.5 Contractive mapping and convergence guarantees

Assumption 3.1. *The learning rate α and the dropout probability p are chosen such that $\alpha p \|\mathbb{X}\| < 1$, the initialization $\boldsymbol{\beta}_0$ is a square-integrable random vector that is independent of the data $\mathbf{Y}$ and the model is in reduced form, meaning that X does not have zero columns.*

As was shown in [1], we are going to utilise Assumption 3.1 to meet the convergence condition. The expectations of the random matrices $(I - \alpha D_k X D_k)$ and $(I - \alpha D_k \mathbb{X}_p)$, under a sufficiently small learning rate α, act as contractive mappings. Assumption 3.1 stipulates the conditions under which this contractivity ensures convergence of dropout iterates. This is taken from Lemma 1 of [1].

Theorem 3.2. *Given Assumption 3.1, $\|I - \alpha p \mathbb{X}_p\| \leq 1 - \alpha p (1-p) \min_i \mathbb{X}_{ii} < 1$, and for any iteration k,*

$$\|\mathbb{E}[\tilde{\boldsymbol{\beta}}_k - \tilde{\boldsymbol{\beta}}]\| \leq \|I - \alpha p \mathbb{X}_p\|^k \|\mathbb{E}[\boldsymbol{\beta}_0 - \tilde{\boldsymbol{\beta}}]\|.$$

This lemma showcases the expected exponential rate at which the expectation of dropout iterates approach $\tilde{\boldsymbol{\beta}}$. While exponential convergence is generally characteristic of linear models, dropout does not inherently enhance this convergence rate. In fact, dropout introduces additional noise and effectively reduces the learning rate, which can slow down convergence compared to standard gradient descent. Therefore, while dropout contributes to regularization and improved generalization, it does so at a slower convergence rate due to the reduced effective learning rate.

For now, we are not going to analyze the behaviour of the second moment, given by $\mathbb{E}[(\tilde{\boldsymbol{\beta}}_k - \tilde{\boldsymbol{\beta}})(\tilde{\boldsymbol{\beta}}_k - \tilde{\boldsymbol{\beta}})^\top]$. These are analyzed in [1], but for the scope of this paper, the previous results introduced aim to aid our analysis in the coming section. We are going to move on to a different version of dropout, one that incorporates random minibatches during the update iterations.

4 Gradient descent with random minibatch sampling

We are now going to reinterpret random minibatch sampling within the gradient descent algorithm as a form of dropout. Instead of treating dropout and random minibatch sampling as separate techniques, we view random minibatch sampling as a dropout mechanism. This perspective allows us to analyze random minibatch sampling using the theoretical frameworks established for dropout. By employing random subsets of the dataset in each iteration, random minibatch sampling introduces a form of stochasticity similar to dropout, leading to comparable effects on the optimization process.

Again, suppose we have a dataset $(X, \mathbf{Y})$, where X is $n \times d$ and the response vector $\mathbf{Y}$ is $n \times 1$. The objective function remains the same and it is to minimize the expected loss:

$$\underset{\boldsymbol{\beta} \in R^d}{argmin}\ \mathbb{E}\left[\|(\mathbf{Y} - X\boldsymbol{\beta})\|^2 | \mathbf{Y}\right]. \tag{9}$$

4.1 Batch stochastic gradient framework

At its essence, the classical stochastic gradient method would pick a point from the dataset at random and then compute the gradient iterates. The idea behind this process is that using a single data point at a time results in updates that are cheaper than a full gradient step. However, as proposed in [11], using a single component does not necessarily lead to convergence. Therefore, we consider gradient updates that utilise a subset of the dataset. This allows for quicker gradient updates, as opposed to utilising the whole dataset, and also allows for specific convergent properties to arise.

In the gradient updates, a random subset of $(\mathbf{Y} - X\boldsymbol{\beta})$ is taken in every step when multiplying by an $n \times n$ dropout matrix D. The equation for such implementation of random minibatch gradient descent is given by:

$$\bar{\boldsymbol{\beta}}_{k+1} = \bar{\boldsymbol{\beta}}_k - \alpha \nabla_{\bar{\boldsymbol{\beta}}_k} \frac{1}{2} \|D_{k+1}(\mathbf{Y} - X\bar{\boldsymbol{\beta}}_k)\|^2 = \bar{\boldsymbol{\beta}}_0 - \alpha \sum_{l=1}^{k} \nabla_{\bar{\boldsymbol{\beta}}_l} \frac{1}{2} [D_{l+1} \|\mathbf{Y} - X\bar{\boldsymbol{\beta}}_l\|^2]. \tag{10}$$

Here, the dropout matrices function to select a random subset of data points in each iteration, reflecting principles akin to randomly weighted least squares and resampling methodologies. This method is also related to randomly weighted least squares and resampling techniques. Overall, two batch regimes can be distinguished:

- D_k has a dense diagonal, which has a cost essentially equivalent to that of a full gradient update.
- D_k has a sparse diagonal. This results in random minibatch sampling.

Section 3 of [1] provides insights into the averaging effect of the law of large numbers and how dropout instead alters the path and properties of the convergence in a nuanced manner.

4.2 Analysis of averaged dropout with minibatches

Since $D^\top = D$ and $D^2 = D$, the squared norm under dropout of stochastic gradient descent with minibatches can be expressed as:

$$\begin{aligned}\|D(\mathbf{Y} - X\boldsymbol{\beta})\|^2 &= (D(\mathbf{Y} - X\boldsymbol{\beta}))^\top D(\mathbf{Y} - X\boldsymbol{\beta}) \\ &= (\mathbf{Y} - X\boldsymbol{\beta})^\top D(\mathbf{Y} - X\boldsymbol{\beta}).\end{aligned}$$

Taking the expectation over D, and noting that $\mathbb{E}[D] = pI$, we arrive at:

$$\begin{aligned}\mathbb{E}\left[\|D(\mathbf{Y} - X\boldsymbol{\beta})\|^2|\mathbf{Y}\right] &= \mathbb{E}[(\mathbf{Y} - X\boldsymbol{\beta})^\top D(\mathbf{Y} - X\boldsymbol{\beta})|\mathbf{Y}] \\ &= p\mathbb{E}[||\mathbf{Y} - X\boldsymbol{\beta})||^2|\mathbf{Y}].\end{aligned}$$

The solution to (9) minimizes the expectation such that

$$\bar{\boldsymbol{\beta}} := \underset{\boldsymbol{\beta}\in R^d}{argmin}\ \mathbb{E}[||(\mathbf{Y} - X\boldsymbol{\beta})||^2|\mathbf{Y}] = \mathbb{X}^{-1}X^\top\mathbf{Y}.$$

In this case, the estimator $\bar{\boldsymbol{\beta}}$ corresponds to the usual linear least squares estimator, given by $\hat{\boldsymbol{\beta}} = \mathbb{X}^{-1}X^\top\mathbf{Y}$, assuming that $\mathbb{X}$ is invertible.

However, if $\mathbb{X}$ is not invertible, the linear least squares problem does not have a unique solution. In such cases, the most natural (but not unique) minimizer is given by $X^+\mathbf{Y}$, where X^+ denotes the Moore-Penrose pseudo-inverse of X. This pseudo-inverse provides a generalized solution to the least squares problem.

While a detailed discussion on the Moore-Penrose pseudo-inverse is beyond the scope of this section, it serves as an essential tool in handling cases where $\mathbb{X}$ is singular or not full rank.

4.3 Analysis of iterative minibatch dropout schemes

In equation (10), the gradient with respect to $\bar{\boldsymbol{\beta}}_k$ is $\nabla_{\bar{\boldsymbol{\beta}}_k}\frac{1}{2}\|D_{k+1}(\mathbf{Y}-X\bar{\boldsymbol{\beta}}_k)\|^2 = -X^\top D_{k+1}(\mathbf{Y}-X\bar{\boldsymbol{\beta}}_k)$. We can therefore rewrite the minibatch iterates in (10) as follows:

$$\bar{\boldsymbol{\beta}}_{k+1} = \bar{\boldsymbol{\beta}}_k + \alpha X^\top D_{k+1}(\mathbf{Y} - X\bar{\boldsymbol{\beta}}_k).$$

As we did in the previous section, we look at the statistical properties of the dropout scheme given by (10). The update mechanism can be reformulated as:

$$\begin{aligned}\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}} &= \bar{\boldsymbol{\beta}}_k + \alpha X^\top D_{k+1}(\mathbf{Y} - X\bar{\boldsymbol{\beta}}_k) - \bar{\boldsymbol{\beta}} \\ &= \bar{\boldsymbol{\beta}}_k + \alpha X^\top D_{k+1}(\mathbf{Y} - X\bar{\boldsymbol{\beta}}_k) - \bar{\boldsymbol{\beta}} + \alpha X^\top D_{k+1}(\mathbf{Y} - X\bar{\boldsymbol{\beta}}) - \alpha X^\top D_{k+1}(\mathbf{Y} - X\bar{\boldsymbol{\beta}}) \\ &= (I - \alpha X^\top D_{k+1}X)(\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}}) + \alpha X^\top D_{k+1}(\mathbf{Y} - X\bar{\boldsymbol{\beta}}),\end{aligned} \tag{11}$$

where $\bar{\boldsymbol{\beta}}$ is a least squares estimator solving the normal equations $X^\top\mathbf{Y} = \mathbb{X}\bar{\boldsymbol{\beta}}$. This reformulation reveals the stochastic character of minibatch dropout in linear regression frameworks. For small learning rates α, the term $(I - \alpha X^\top D_{k+1}X)$ acts as a contractive map in expectation. The expected value is $\mathbb{E}[I - \alpha X^\top D_{k+1}X] = I - \alpha p\mathbb{X}$. Thereby, this gives us the same condition as in Section 3 in Assumption 3.1, where $\alpha p\|\mathbb{X}\| < 1$ is a necessary condition to ensure the scheme converges in expectation. However, while Assumption 3.1 is necessary for the convergence of $\tilde{\boldsymbol{\beta}}_k$, it is not sufficient for $\bar{\boldsymbol{\beta}}_k$. The following condition must also hold such that $\bar{\boldsymbol{\beta}}_k$ converges.

Assumption 4.1. *The inverse $\mathbb{X}^{-1}$ must exist such that the estimator converges as $k \to \infty$.*

More specifically, the property we aim to derive is that the null space of $\mathbb{X}$ is empty, which can be achieved when $\mathbb{X}$ has full rank. This means that all eigenvalues λ of $\mathbb{X}$ are greater than zero ($\lambda > 0$ for all λ). We shall assume that the null space of $\mathbb{X}$ is trivially empty, meaning that there does not exist any non-zero vector v such that $\mathbb{X}v = 0$.

To clarify, an empty null space implies that the matrix $\mathbb{X}$ is invertible, which ensures that every non-zero vector v is mapped to a non-zero vector under $\mathbb{X}$. If some $v \neq 0$ is in the null space of $\mathbb{X}$, then the matrix $\mathbb{X}$ would fail to be invertible.

This is important because if such a $v \neq 0$ exists, then for the expression $(I - \alpha p\mathbb{X})v$, we have:

$$(I - \alpha p\mathbb{X})v = v - \alpha p\mathbb{X}v = v.$$

This indicates that the mapping is not contractive. For the iterative process in (12), this non-contractiveness implies that the norm of the iterations does not converge to zero, preventing the convergence of the iterative solution.

There exists a relationship between the null space, eigenvalues, and the norm for ensuring convergence. In particular, the matrix $\mathbb{X}$ being invertible implies that its eigenvalues are all non-zero, which is directly related to the contractive nature of the mapping $(I - \alpha p\mathbb{X})$. For a more detailed discussion about these properties, see Theorems 5.6.9 and 5.6.12 in [4], which provide deeper insights into matrix invertibility and contraction mappings.

Given Assumption 3.1 and that $\mathbb{X}$ is invertible, we are going to show that these two conditions are in fact sufficient. To show this directly, we use the fact that if the spectral radius of a matrix A is less than 1, then $I - A$ is invertible and its norm is less than 1.

To understand the convergence properties of the iterative method, we analyze the spectral properties of the matrix $I - \alpha p\mathbb{X}$.

Lemma 4.2. *Given that $\alpha p\|\mathbb{X}\| < 1$ and $\mathbb{X}$ is a normal matrix, the spectral radius of $I - \alpha p\mathbb{X}$ is less than 1. Consequently, $\|I - \alpha p\mathbb{X}\| < 1$, implying that $I - \alpha p\mathbb{X}$ is a contractive mapping.*

Proof. Let λ be an eigenvalue of $\alpha p\mathbb{X}$ with corresponding eigenvector v. Then, the corresponding eigenvalue of $I - \alpha p\mathbb{X}$ is $1 - \lambda$. Given that $\alpha p\|\mathbb{X}\| < 1$, all eigenvalues λ of $\alpha p\mathbb{X}$ satisfy $0 < \lambda < 1$. Since the eigenvalues of $I - \alpha p\mathbb{X}$ are $1 - \lambda$, this ensures that the spectral radius of $I - \alpha p\mathbb{X}$ is less than 1. Then, we have $\alpha p\mathbb{X}v = \lambda v$. Consider the matrix $I - \alpha p\mathbb{X}$ acting on v:

$$(I - \alpha p\mathbb{X})v = v - \alpha p\mathbb{X}v = v - \lambda v = (1 - \lambda)v.$$

Thus, the eigenvalue of $I - \alpha p\mathbb{X}$ corresponding to λ is $1 - \lambda$.

Given $\alpha p\|\mathbb{X}\| < 1$, the eigenvalues λ of $\alpha p\mathbb{X}$ satisfy $|\lambda| < 1$. Since $\mathbb{X}$ is invertible (implying $\mathbb{X}$ has full rank and all its eigenvalues are positive), we have $0 < \lambda < 1$.

Therefore, for all eigenvalues λ of $\alpha p\mathbb{X}$, the corresponding eigenvalues $1 - \lambda$ of $I - \alpha p\mathbb{X}$ satisfy $0 < 1 - \lambda < 1$. This means that the spectral radius of $I - \alpha p\mathbb{X}$, defined as the maximum absolute value of its eigenvalues, is less than 1.

When $\mathbb{X}$ is a normal matrix (which includes symmetric matrices), we can infer from the spectral radius condition that:

$$\|I - \alpha p\mathbb{X}\| < 1.$$

This ensures that $I - \alpha p \mathbb{X}$ is a contractive mapping. $\square$

To assert the convergence of $\mathbb{E}[\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}}]$, we want to establish a decaying upper bound on the expectation. This will ensure that the expected dropout iterates converge exponentially fast in the number of iterations to the expectation of the estimation $\bar{\boldsymbol{\beta}}$. We show that the second term in (11) has the following expectation:

$$\begin{aligned}\mathbb{E}[\alpha X^\top D_{k+1}(\mathbf{Y} - X\bar{\boldsymbol{\beta}})|\mathbf{Y}] &= \alpha p \mathbb{E}[X^\top(\mathbf{Y} - X\bar{\boldsymbol{\beta}})|\mathbf{Y}] \\ &= \alpha p \mathbb{E}[X^\top \mathbf{Y} - \mathbb{X}\bar{\boldsymbol{\beta}}|\mathbf{Y}] \\ &= \alpha p \mathbb{E}[\mathbb{X}\bar{\boldsymbol{\beta}} - \mathbb{X}\bar{\boldsymbol{\beta}}|\mathbf{Y}] \\ &= 0\end{aligned}$$

We now condition on all randomness except D_{k+1} in (11), and get the following

$$\mathbb{E}[\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}}|\bar{\boldsymbol{\beta}}, \bar{\boldsymbol{\beta}}_k] = (I - \alpha p \mathbb{X})(\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}}). \tag{12}$$

By the tower rule, we have that $\mathbb{E}[\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}}] = (I - \alpha p \mathbb{X})\mathbb{E}[\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}}]$, so induction on k gives us

$$\mathbb{E}[\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}}] = (I - \alpha p \mathbb{X})^{k+1}\mathbb{E}[\bar{\boldsymbol{\beta}}_0 - \bar{\boldsymbol{\beta}}],$$

where $\bar{\boldsymbol{\beta}}_0$ is a potentially stochastic baseline. Sub-multiplicativity of the spectral norm implies $\|(I - \alpha p \mathbb{X})^{k+1}\| \leq \|(I - \alpha p \mathbb{X})\|^{k+1}$, proving the following result.

Theorem 4.3. *For every $k = 0, 1, 2, \ldots$*

$$\|\mathbb{E}[\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}}]\| \leq \|(I - \alpha p \mathbb{X})\|^{k+1}\|\mathbb{E}[\bar{\boldsymbol{\beta}}_0 - \bar{\boldsymbol{\beta}}]\|. \tag{13}$$

4.3.1 Analysis of the second moment dynamics

We next turn our attention to covariance analysis and try to establish bounds on the covariance matrix of $\bar{\boldsymbol{\beta}}_k$.

Lemma 4.4. *The covariance of $AZ = -ADu + ADv$ is given by:*

$$\begin{aligned}\mathrm{Cov}(AZ) &= A\,\mathrm{Cov}(Z)A^\top \\ &= Ap(1-p)\left(\mathrm{Diag}(B_u) + \mathrm{Diag}(B_{uv}) + \mathrm{Diag}(B_{vu}) + \mathrm{Diag}(B_v)\right)A^\top,\end{aligned}$$

where $B_{uv} = uv^\top$, $B_u = uu^\top$, $B_v = vv^\top$, $B_{vu} = vu^\top$, and A is a deterministic matrix.

Proof. To find the covariance of $Z = -Du + Dv$, we use the definition of covariance:

$$\mathrm{Cov}(Z) = \mathbb{E}[ZZ^\top] - \mathbb{E}[Z]\mathbb{E}[Z^\top].$$

First, we calculate $\mathbb{E}[Z]$:

$$\mathbb{E}[Z] = \mathbb{E}[-Du + Dv] = -pu + pv = p(-u + v).$$

Next, we compute $\mathbb{E}[ZZ^\top]$. Given $Z = D(-u + v)$, we expand:

$$\mathbb{E}[ZZ^\top] = \mathbb{E}[D(-u + v)(-u + v)^\top D].$$

We use the identity for the expectation of DAD in (6) where D is a dropout matrix, which states that:

$$\mathbb{E}[DAD] = pA_p,$$

where $A_p = pA - p\,\mathrm{Diag}(A) + \mathrm{Diag}(A)$, and A is a matrix. This identity simplifies our computation. Applying it, we get:

$$\mathbb{E}[ZZ^\top] = p((B_u)_p - (B_{uv})_p - (B_{vu})_p + (B_v)_p).$$

Subtracting the mean product $\mathbb{E}[Z]\mathbb{E}[Z^\top]$, we have:

$$\mathrm{Cov}(Z) = p((B_u)_p - (B_{uv})_p - (B_{vu})_p + (B_v)_p) - p^2(-u+v)(-u+v)^\top.$$

Simplifying the mean product term:

$$p^2(-u+v)(-u+v)^\top = p^2(B_u - B_{uv} - B_{vu} + B_v).$$

Combining these, we get:

$$\mathrm{Cov}(Z) = p((B_u)_p - (B_{uv})_p - (B_{vu})_p + (B_v)_p) - p^2(B_u - B_{uv} - B_{vu} + B_v).$$

After further simplification, and noting that $p(B_u)_p - p^2 B_u = p(1-p)\,\mathrm{Diag}(B_u)$ we obtain:

$$\mathrm{Cov}(Z) = p(1-p)(\mathrm{Diag}(B_u) + \mathrm{Diag}(B_{uv}) + \mathrm{Diag}(B_{vu}) + \mathrm{Diag}(B_v)).$$

This completes the proof. $\square$

We determine the second moment using the law of total covariance. Given a random vector U and a random element V, observe that

$$\mathbb{E}[\mathrm{Cov}(U \mid V)] = \mathbb{E}[UU^\top] - \mathbb{E}[\mathbb{E}[U \mid V]\mathbb{E}[U \mid V]^\top].$$

Substituting $U = \bar{\boldsymbol{\beta}}_{\boldsymbol{k+1}} - \bar{\boldsymbol{\beta}}$ and $V = (\bar{\boldsymbol{\beta}}_{\boldsymbol{k}}, \bar{\boldsymbol{\beta}})$, as well as defining the second moment as

$$A_{k+1} := \mathbb{E}[(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}})^\top].$$

By the law of total covariance, we have that

$$A_{k+1} = \mathbb{E}\left[\mathbb{E}\left[\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}\right]\mathbb{E}\left[\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}\right]^\top\right] + \mathbb{E}\left[\mathrm{Cov}\left(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}\right)\right]$$

Recall $\mathbb{E}\left[\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}, \bar{\boldsymbol{\beta}}_k\right] = (I - \alpha p \mathbb{X})\left(\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}}\right)$ from (12), and so

$$\mathbb{E}\left[\mathbb{E}\left[\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}\right]\mathbb{E}\left[\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}\right]^\top\right] = (I - \alpha p \mathbb{X})\,A_k\,(I - \alpha p \mathbb{X})$$

Evaluating the conditional variance $\mathrm{Cov}(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}})$ is the more challenging part. Recall that the expression in (11) can be used as follows

$$\mathrm{Cov}(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}) = \mathrm{Cov}\left((I - \alpha X^\top D_{k+1} X)(\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}}) + \alpha X^\top D_{k+1}(\mathbf{Y} - X\bar{\boldsymbol{\beta}}) \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}\right)$$

Since the covariance is invariant under shifts and sign flips, so

$$
\begin{aligned}
\operatorname{Cov}(\bar{\boldsymbol{\beta}}_{k+1}-\bar{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}) &= \operatorname{Cov}\left((I-\alpha X^\top D_{k+1}X)(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})+\alpha X^\top D_{k+1}(\mathbf{Y}-X\bar{\boldsymbol{\beta}}) \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}\right) \\
&= \alpha^2 \operatorname{Cov}\left(-X^\top D_{k+1}X(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})+X^\top D_{k+1}(\mathbf{Y}-X\bar{\boldsymbol{\beta}}) \mid \bar{\boldsymbol{\beta}}_k, \hat{\boldsymbol{\beta}}\right)
\end{aligned}
$$

In doing this, we can create an expression for the $\mathbb{E}[\operatorname{Cov}(\bar{\boldsymbol{\beta}}_{k+1}-\bar{\boldsymbol{\beta}}|\bar{\boldsymbol{\beta}}_k,\bar{\boldsymbol{\beta}})]$. We have that by using Lemma 4.4 and setting $A:=X^\top, u:=X(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}}), v:=(\mathbf{Y}-X\bar{\boldsymbol{\beta}})$ along with the fact that the $\mathbb{E}[D]=p$:

$$
\begin{aligned}
A_{k+1} &= \mathbb{E}\left[\mathbb{E}\left[\bar{\boldsymbol{\beta}}_{k+1}-\hat{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \hat{\boldsymbol{\beta}}\right]\mathbb{E}\left[\bar{\boldsymbol{\beta}}_{k+1}-\hat{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \hat{\boldsymbol{\beta}}\right]^\top\right]+\mathbb{E}\left[\operatorname{Cov}\left(\bar{\boldsymbol{\beta}}_{k+1}-\hat{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \hat{\boldsymbol{\beta}}\right)\right] \\
&= (I-\alpha p\mathbb{X})\,A_k\,(I-\alpha p\mathbb{X}) \\
&+ \alpha^2 p(1-p)X^\top\mathbb{E}[\operatorname{Diag}(B_u)+\operatorname{Diag}(B_{uv})+\operatorname{Diag}(B_{vu})+\operatorname{Diag}(B_v)]X.
\end{aligned}
$$

The term in the square brackets is going to be expanded as follows: $\operatorname{Diag}(B_u)+\operatorname{Diag}(B_{uv})+\operatorname{Diag}(B_{vu})+\operatorname{Diag}(B_v)=\operatorname{Diag}(X(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})^\top X^\top)+\operatorname{Diag}(X(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})(\mathbf{Y}^\top-\bar{\boldsymbol{\beta}}^\top X^\top))+\operatorname{Diag}((\mathbf{Y}-X\bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})^\top X^\top)+\operatorname{Diag}((\mathbf{Y}-X\bar{\boldsymbol{\beta}})(\mathbf{Y}^\top-\bar{\boldsymbol{\beta}}^\top X^\top))$. Then, we have that,

$$
\begin{aligned}
A_{k+1} &= \mathbb{E}\left[\mathbb{E}\left[\bar{\boldsymbol{\beta}}_{k+1}-\bar{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}\right]\mathbb{E}\left[\bar{\boldsymbol{\beta}}_{k+1}-\bar{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}\right]^\top\right]+\mathbb{E}\Big[\operatorname{Cov}\left(\bar{\boldsymbol{\beta}}_{k+1}-\bar{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}\right)\Big] \\
&= (I-\alpha p\mathbb{X})\,A_k\,(I-\alpha p\mathbb{X})+\alpha^2 p(1-p)X^\top\mathbb{E}\Big[\Big(\operatorname{Diag}(X(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})^\top X^\top) \\
&+ \operatorname{Diag}(X(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})(\mathbf{Y}^\top-\bar{\boldsymbol{\beta}}^\top X^\top))+\operatorname{Diag}((\mathbf{Y}-X\bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})^\top X^\top) \\
&+ \operatorname{Diag}((\mathbf{Y}-X\bar{\boldsymbol{\beta}})(\mathbf{Y}^\top-\bar{\boldsymbol{\beta}}^\top X^\top))\Big)\Big]X.
\end{aligned}
$$

Since the expectation is a linear operator, we have that $\mathbb{E}[\operatorname{Diag}(A)]=\operatorname{Diag}(\mathbb{E}[A])$. By claiming that, we can rewrite the expression for A_{k+1} as follows:

$$
\begin{aligned}
A_{k+1} &= (I-\alpha p\mathbb{X})\,A_k\,(I-\alpha p\mathbb{X})+\alpha^2 p(1-p)X^\top\Big(\operatorname{Diag}(\mathbb{E}[X(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})^\top X^\top]) \\
&+ \operatorname{Diag}(\mathbb{E}[X(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})(\mathbf{Y}^\top-\bar{\boldsymbol{\beta}}^\top X^\top)])+\operatorname{Diag}(\mathbb{E}[(\mathbf{Y}-X\bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_k-\bar{\boldsymbol{\beta}})^\top X^\top]) \\
&+ \operatorname{Diag}(\mathbb{E}[(\mathbf{Y}-X\bar{\boldsymbol{\beta}})(\mathbf{Y}^\top-\bar{\boldsymbol{\beta}}^\top X^\top)])\Big)X.
\end{aligned}
\tag{14}
$$

Theorem 4.5. *(Second Moment - Recursive Formula). Under Assumption 3.1, for every $k=1,2,\cdots$*

$$
\begin{aligned}
&\|\mathbb{E}[(\bar{\boldsymbol{\beta}}_{k+1}-\bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_{k+1}-\bar{\boldsymbol{\beta}})^\top]-S(\mathbb{E}[(\bar{\boldsymbol{\beta}}_{k+1}-\bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_{k+1}-\bar{\boldsymbol{\beta}})^\top])\| \\
&\le 2\alpha^2 p\|X\|^3\|B_0\|\|(I-\alpha p\mathbb{X})\|^k,
\end{aligned}
\tag{15}
$$

where the affine operator $S:\mathbb{R}^{d\times d}\to\mathbb{R}^{d\times d}$ is defined as

$$
\begin{aligned}
S(A) :=& (I-\alpha p\mathbb{X})A(I-\alpha p\mathbb{X}) \\
&+ \alpha^2 p(1-p)X^\top\Big(\operatorname{Diag}(XAX^\top)+\operatorname{Diag}(I-X\mathbb{X}^{-1}X^\top)\Big)X.
\end{aligned}
\tag{16}
$$

We claim that the second moment evolves as the affine operator up to an exponentially decaying remainder.

Proof. To see this, we note that $\bar{\boldsymbol{\beta}} = \mathbb{X}^{-1}X^\top\mathbf{Y} = \mathbb{X}^{-1}X^\top(X\boldsymbol{\beta}_* + \boldsymbol{\varepsilon}) = \boldsymbol{\beta}_* + \mathbb{X}^{-1}X^\top\boldsymbol{\varepsilon}$. We can now use this to compute $\mathbf{Y} - X\bar{\boldsymbol{\beta}} = X\boldsymbol{\beta}_* + \boldsymbol{\varepsilon} - X\boldsymbol{\beta}_* - X\mathbb{X}^{-1}X^\top\boldsymbol{\varepsilon} = (I - X\mathbb{X}^{-1}X^\top)\boldsymbol{\varepsilon}$. If X is square and invertible, then this expression equals zero. The $\mathbb{E}[\mathbf{Y} - X\bar{\boldsymbol{\beta}}] = E[(I - X\mathbb{X}^{-1}X^\top)\boldsymbol{\varepsilon}] = 0$. Along with that, we also want to define the $\mathbb{E}[(\mathbf{Y} - X\bar{\boldsymbol{\beta}})(\mathbf{Y} - X\bar{\boldsymbol{\beta}})^\top]$ such that $\mathbb{E}[(\mathbf{Y} - X\bar{\boldsymbol{\beta}})(\mathbf{Y} - X\bar{\boldsymbol{\beta}})^\top] = \mathbb{E}[(I - X\mathbb{X}^{-1}X^\top)\boldsymbol{\varepsilon}((I - X\mathbb{X}^{-1}X^\top)\boldsymbol{\varepsilon})^\top] = (I - X\mathbb{X}^{-1}X^\top)(I - X\mathbb{X}^{-1}X^\top)^\top = (I - X\mathbb{X}^{-1}X^\top)$. This expression involves a projection matrix. Let $P = X(X^\top X)^{-1}X^\top$ be the projection matrix onto the column space of X. $I - P$ is the projection matrix onto the orthogonal complement of the column space of X. Thereby, $I - P$ is idempotent and symmetric such that

$$(I - X(X^\top X)^{-1}X^\top)(I - X(X^\top X)^{-1}X^\top)^\top = I - X(X^\top X)^{-1}X^\top.$$

This result shows that the product of the projection matrix $I - P$ with its transpose simplifies to the projection matrix itself, confirming its idempotent nature. We also note that multiplying by $X^\top$ and X results in 0 since $X^\top(I - X(X^\top X)^{-1}X^\top)X = X^\top X - X^\top X = 0$.

We also need one more result, and that is to bound $B_k := \mathbb{E}[(\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}})(\mathbf{Y} - X\bar{\boldsymbol{\beta}})]$. From (11), we can rewrite our expression of interest as follows:

$$\begin{aligned}
B_{k+1} &= \mathbb{E}[(I - \alpha X^\top D_{k+1}X)(\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}})(\mathbf{Y} - X\bar{\boldsymbol{\beta}})] + \mathbb{E}[\alpha X^\top D_{k+1}(\mathbf{Y} - X\bar{\boldsymbol{\beta}})(\mathbf{Y} - X\bar{\boldsymbol{\beta}})] \\
&= (I - \alpha p\mathbb{X})B_k + \alpha p X^\top(I - X\mathbb{X}^{-1}X^\top) \\
&= (I - \alpha p\mathbb{X})B_k.
\end{aligned}$$

Induction on k now leads to $B_{k+1} = (I - \alpha p\mathbb{X})^{k+1}B_0$. Therefore, sub-multiplicativity of the spectral norm and Assumption 3.1 imply $\|B_k\| \to 0$ since $\|B_k\| \leq \|(I - \alpha p\mathbb{X})\|^k \|\mathbb{E}[(\bar{\boldsymbol{\beta}}_0 - \bar{\boldsymbol{\beta}})(\mathbf{Y} - X\bar{\boldsymbol{\beta}})]\|$.

Together with (14), we have that $\mathbb{E}\left[\operatorname{Cov}\left(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}} \mid \bar{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}\right)\right] = S(A_{k+1}) - (I - \alpha p\mathbb{X})A_{k+1}(I - \alpha p\mathbb{X}) + \rho_{k+1}$, where

$$\begin{aligned}
\rho_{k+1} = \alpha^2 p(1-p)X^\top \Big(&\operatorname{Diag}(\mathbb{E}[X(\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}})(\mathbf{Y}^\top - \bar{\boldsymbol{\beta}}^\top X^\top)]) \\
&+ \operatorname{Diag}(\mathbb{E}[(\mathbf{Y} - X\bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}})^\top X^\top])\Big)X.
\end{aligned}$$

To ensure convergence, we still need to assert the decaying of the remainder. To do so, we use the fact that $\|\operatorname{Diag}(A)\| \leq \|A\|$, which has proof that can be found in lemma 13 of [1]. We prove that $\|\rho_{k+1}\| \to 0$ as $k \to \infty$ using the triangle inequality as follows:

$$\begin{aligned}
\|\rho_{k+1}\| &= \|\alpha^2 p(1-p)X^\top(\operatorname{Diag}(XB_k) + \operatorname{Diag}((XB_k)^\top))X\| \\
&\leq \alpha^2 p(1-p)\|X\|^2(\|\operatorname{Diag}(XB_k)\| + \|\operatorname{Diag}((XB_k)^\top))\|) \\
&\leq \alpha^2 p(1-p)\|X\|^2(\|XB_k\| + \|(XB_k)^\top\|) \\
&\leq \alpha^2 p(1-p)\|X\|^3(\|[I - \alpha p\mathbb{X}]\|^k\|B_0\| + \|[I - \alpha p\mathbb{X}]\|^k\|B_0\|) \\
&= 2\alpha^2 p(1-p)\|X\|^3\|B_0\|\|[I - \alpha p\mathbb{X}]\|^k.
\end{aligned}$$

□

In (16), α acts as the learning rate and is a key hyperparameter that controls the step size of the gradient descent updates. As described in lemma 2 of [1], α must be chosen such that the operator norm of S_{lin} is less than 1.

We now move on to defining the value of α that ensures convergence. Define $S_0 := S(0)$ and $S_{lin}(A) := S(A) - S_0$. Therefore, $S_0 = \alpha^2 p(1-p)X^\top(\mathrm{Diag}(I - X\mathbb{X}^{-1}X))X$, and $S_{lin}(A) = (I - \alpha p\mathbb{X})A(I - \alpha p\mathbb{X}) + \alpha^2 p(1-p)X^\top(\mathrm{Diag}(XAX^\top))X$. We are going to try to establish a bound on the linear part of the S operator.

Lemma 4.6. *The linear operator S_{lin} satisfies $\|S_{lin}\|_{\mathrm{op}} \leq \|I - \alpha p\mathbb{X}\| < 1$, provided that*

$$\alpha < \min\left\{\frac{1}{p\|\mathbb{X}\|}, \frac{\lambda_{\min}(\mathbb{X})}{(2-p)\|X\|^4)}\right\},$$

where $\lambda_{\min}(\mathbb{X})$ denotes the smallest eigenvalue of $\mathbb{X}$.

Proof. For an arbitrary matrix A, the triangle inequality, lemma 13 of [1], and sub-multiplicativity of the spectral norm imply that

$$\begin{aligned}\|S_{lin}(A)\| &= \|(I-\alpha p\mathbb{X})A(I-\alpha p\mathbb{X}) + \alpha^2 p(1-p)X^\top(\mathrm{Diag}(XAX^\top))X\| \\ &\leq \|(I-\alpha p\mathbb{X})\|^2\|A\| + \alpha^2 p(1-p)\|X\|^4\|A\| = (\|(I-\alpha p\mathbb{X})\|^2 + \alpha^2 p(1-p)\|X\|^4)\|A\|.\end{aligned}$$

Note that $\|I - \alpha p\mathbb{X}\| = 1 - \alpha p\lambda_{min}(\mathbb{X})$. Therefore, $\|I - \alpha p\mathbb{X}\|^2 = 1 - 2\alpha p\lambda_{min}(\mathbb{X}) + \alpha^2 p^2\lambda^2_{min}(\mathbb{X}) \leq 1 - 2\alpha p\lambda_{min}(\mathbb{X}) + \alpha^2 p\|X\|^4$. We use the fact that $\lambda_{\min}(\mathbb{X}) \leq \|X\|^2$, which has proof in Appendix A. This results in

$$\begin{aligned}\|S_{lin}(A)\| &\leq (1 - 2\alpha p\lambda_{min}(\mathbb{X}) + \alpha^2 p\|X\|^4 + \alpha^2 p(1-p)\|X\|^4)\|A\| \\ &\leq (1 - 2\alpha p\lambda_{min}(\mathbb{X}) + \alpha^2 p((2-p)\|X\|^4)\|A\|.\end{aligned} \tag{17}$$

If we let $\alpha < \frac{\lambda_{min}(\mathbb{X})}{(2-p)\|X\|^4}$, then $\alpha^2 p((2-p)\|X\|^4) \leq \alpha p\lambda_{min}(\mathbb{X})$ and in turn $\|S_{lin}\|_{op} \leq \|I - \alpha p\mathbb{X}\|$. The constraint $\alpha < 1/(p\|X\|)$ enforces that $\alpha p\|X\| < 1$ such that $\|I - \alpha p\mathbb{X}\| < 1$. $\square$

From a geometric perspective, we observe that the choice of α in (17) leads to a quadratic formulation. Specifically, the term $\alpha^2 p((2-p)\|X\|^4)$ behaves quadratically with respect to α. By bounding α, we effectively control this quadratic term, preventing it from becoming excessively large. This control ensures that the expression $1 - 2\alpha p\lambda_{\min}(\mathbb{X}) + \alpha^2 p((2-p)\|X\|^4)$ remains less than 1. Consequently, this bound guarantees the boundedness of the spectral norm $\|S_{lin}\|$.

Figure 2 illustrates the quadratic dependence on α from (17). In this graph, we demonstrate how varying the hyperparameters α and p influences the quadratic term. It is important to note that while we can adjust α and p as hyperparameters, the properties of the data matrix X, such as $\lambda_{\min}(\mathbb{X})$ and $\|X\|^4$, are intrinsic to the dataset and not under our direct control.

The proof of the following theorem essentially follows from the proof of Theorem 2 in [1], with the exception of the constant C.

Theorem 4.7. *(Second Moment - Limit Formula). In addition to Assumption 3.1 suppose $\alpha < \frac{\lambda_{\min}(\mathbb{X})}{(2-p)\|X\|^2)}$, then, for any $k = 1, 2, \ldots$*

$$\left\|\mathbb{E}\left[\left(\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}}\right)\left(\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}}\right)^\top\right] - (\mathrm{id} - S_{lin})^{-1} S_0\right\| \leq Ck\,\|I - \alpha p\mathbb{X}\|^{k-1},$$

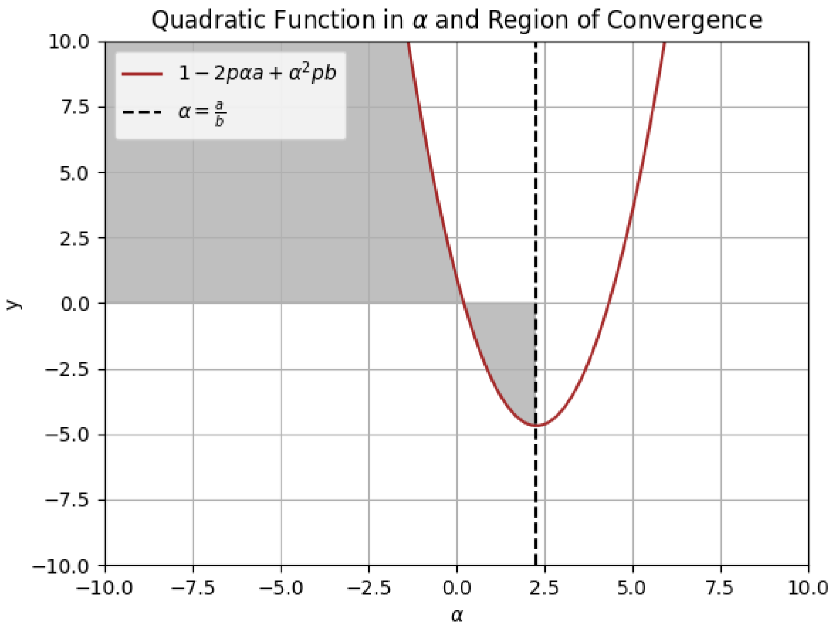


FIGURE 2: The quadratic in α from (17), where the inequality from 4.6 is highlighted in grey.

with constant C given by

$$C := \left\| \mathbb{E}\left[(\bar{\boldsymbol{\beta}}_0 - \bar{\boldsymbol{\beta}}) (\bar{\boldsymbol{\beta}}_0 - \bar{\boldsymbol{\beta}})^\top \right] - (\mathrm{id} - S_{\mathrm{lin}})^{-1} S_0 \right\| + \alpha^2 p(2+p)\|X\|^3 \|B_0\|.$$

This means that, $\mathbb{E}[(\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}})^\top]$ converges exponentially to the limit $(\mathrm{id} - S_{\mathrm{lin}})^{-1} S_0$. Theorem 2 in [1] also establishes that

$$\operatorname{Cov}(\bar{\boldsymbol{\beta}}_k) \to \operatorname{Cov}(\bar{\boldsymbol{\beta}}) + (\mathrm{id} - S_{\mathrm{lin}})^{-1} S_0, \qquad \text{as } k \to \infty,$$

with an exponential rate of convergence. Taking the trace and noting that $|\operatorname{Tr}(A)| \leq d\|A\|$, we obtain a bound for convergence with respect to the squared Euclidean loss,

$$\left| \mathbb{E}\left[\|\bar{\boldsymbol{\beta}}_k - \bar{\boldsymbol{\beta}}\|^2 \right] - \operatorname{Tr}\left((\mathrm{id} - S_{\mathrm{lin}})^{-1} S_0 \right) \right| \leq Cdk \|I - \alpha p \mathbb{X}\|^{k-1}.$$

Since $(\mathrm{id} - S_{\mathrm{lin}})^{-1} S_0$ is a $d \times d$ matrix, the term $\operatorname{Tr}\left((\mathrm{id} - S_{\mathrm{lin}})^{-1} S_0 \right)$ describing the asymptotic suboptimality of the estimator $\bar{\boldsymbol{\beta}}_k$ can be large in high dimensions d, even if the spectral norm of $(\mathrm{id} - S_{\mathrm{lin}})^{-1} S_0$ is small.

5 Verification and validation of the gradient descent models

In this section, we focus on verifying and validating the gradient descent models introduced earlier: gradient descent with dropout and gradient descent with random minibatch sampling. Our aim is to assess the convergence properties of these models and to evaluate their performance under various conditions. Specifically, we have proposed bounds on the convergence of the estimators $(\bar{\boldsymbol{\beta}}_k, \tilde{\boldsymbol{\beta}}_k)$ in expectation and have modelled the behaviour of the covariance of gradient descent with random minibatch sampling. We are going to leverage Theorem 4.3 for the convergence of the estimators $(\bar{\boldsymbol{\beta}}_k, \tilde{\boldsymbol{\beta}}_k)$ along with Theorem 4.7 for validating the behaviour of the covariance of gradient descent with random minibatch sampling.

5.1 Verification and validation using Monte Carlo simulation

Monte Carlo simulations are utilised in statistical analysis to approximate probabilistic processes through random sampling. In this thesis, the simulations evaluate how modifications like minibatch dropout influence the performance and robustness of linear regression models. Monte Carlo simulations are particularly well-suited for this type of analysis because they can effectively mimic the stochastic nature of minibatch dropout in linear regression training. By repeatedly sampling from the input data and applying dropout at different rates, these simulations can provide a detailed distribution of regression outcomes. This allows researchers to quantify how variability in dropout influences the estimates of regression coefficients and the overall model performance.

The main objectives of employing Monte Carlo simulations in our analysis include:

1. Exploring the expected distributions of model parameters ($\tilde{\boldsymbol{\beta}}$, $\bar{\boldsymbol{\beta}}$) to understand the effects of dropout at various rates and varying the dimensionality of the dataset. This is done for both simple dropout and random minibatch sampling.
2. Assessing the variability and stability of model parameters using the bounds derived in (15).

5.2 Methodology

The simulation incorporates a standard linear regression model with parameters adjusted for different rates of dropout, ranging from 0% to 100%, at 5% intervals. This helps in simulating each scenario across numerous training epochs to gather comprehensive performance metrics. In regards to random minibatch sampling, a random subset of the data is sampled in each iteration to converge to the optimal solution efficiently.

We conduct 10,000 iterations for each of the two dropout configurations (simple and minibatch) along with 1,000 samples to ensure statistical reliability. The simulations are performed by maintaining a constant number of features (dimensions), which helps in estimating the optimal dropout rate.

5.2.1 Implementation

The simulation, which can be found in [10], was implemented in Python due to the extensive availability of resources and libraries that facilitate the simulation process. Python is widely recognized for its simplicity and versatility, making it a popular choice for scientific computing and statistical analysis. Several key libraries were utilized to ensure efficient and accurate simulations:

- **NumPy**: This library provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. It is essential for handling the large datasets required for Monte Carlo simulations and for performing linear algebra operations efficiently.

- **Matplotlib**: To visualize the results of the simulations, Matplotlib is used. It is a plotting library that produces high-quality graphs and figures, allowing us to effectively present the distributions of model parameters and other key findings.

The implementation process involves several steps:

1. **Data Generation**: Synthetic datasets are generated to simulate different scenarios of linear regression models. This includes varying the dimensionality of the data and introducing dropout at different rates.
2. **Model Training**: The gradient descent algorithms, both with dropout and random minibatch sampling, are applied to train the linear regression models on the synthetic datasets. During this phase, the parameters $(\tilde{\boldsymbol{\beta}}, \bar{\boldsymbol{\beta}})$ are estimated.
3. **Simulation Execution**: Multiple iterations of the simulation are run to capture the stochastic nature of the dropout and minibatch sampling processes. This involves repeatedly sampling from the datasets and applying the gradient descent algorithms to obtain a distribution of outcomes.
4. **Result Analysis**: The results from the simulations are collected and analyzed to assess the performance and robustness of the models. This includes evaluating the distributions of the model parameters, the convergence properties, and the adherence to the theoretical bounds.
5. **Visualization**: The results are visualized using Matplotlib to create graphs and plots that illustrate the key findings. These visualizations help in interpreting the data and communicating the insights gained from the simulations.

By leveraging Python and its robust ecosystem of libraries, the implementation of the Monte Carlo simulations is both efficient and effective, allowing for a thorough examination of the gradient descent models and their behavior under various conditions.

5.3 Monte Carlo simulation results

5.3.1 Convergence in expectation using gradient descent with dropout in linear regression

Analysis of the results indicates significant variability in model performance with different dropout rates. Here, we define the dropout rate to p, where p is the probability of not retaining a unit. As the dropout rate increases, there tends to be a steady drop in the error, but after $p \approx 0.5$, we start observing an increase again in model error, highlighting the trade-off between robustness and accuracy. The sharp edge at $p \approx 0.5$ indicates the lack of samples, for had there been more samples, the edge would have been smoother. The model achieves the lowest error at a dropout rate of $p \approx 0.5$ and error measurement of $\approx 0.24\%$, which is coherent with the results introduced in [12]. The simulation was carried out with $X \in \mathbb{R}^{3\times3}$.

5.3.2 Convergence in expectation using stochastic minibatch gradient descent in linear regression

In this section, we simulate the influence of implementing stochastic minibatch dropout within a linear regression model.

It is interesting to note from this simulation the steep increase in percentage error beyond a dropout rate of 0.8. This suggests a threshold beyond which dropout becomes detrimental,

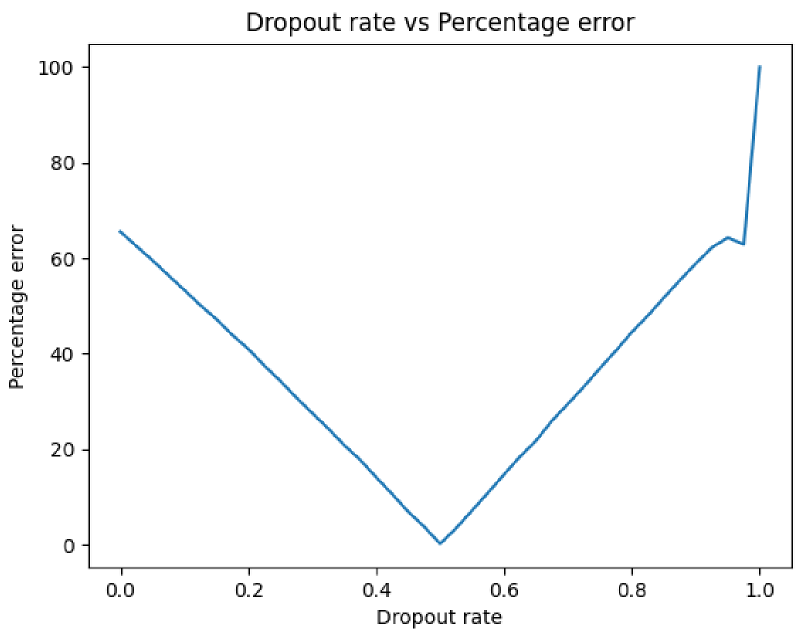


FIGURE 3: Dropout effects on a linear regression model. The Figure depicts the effect of the dropout rate on the percentage error using dropout.

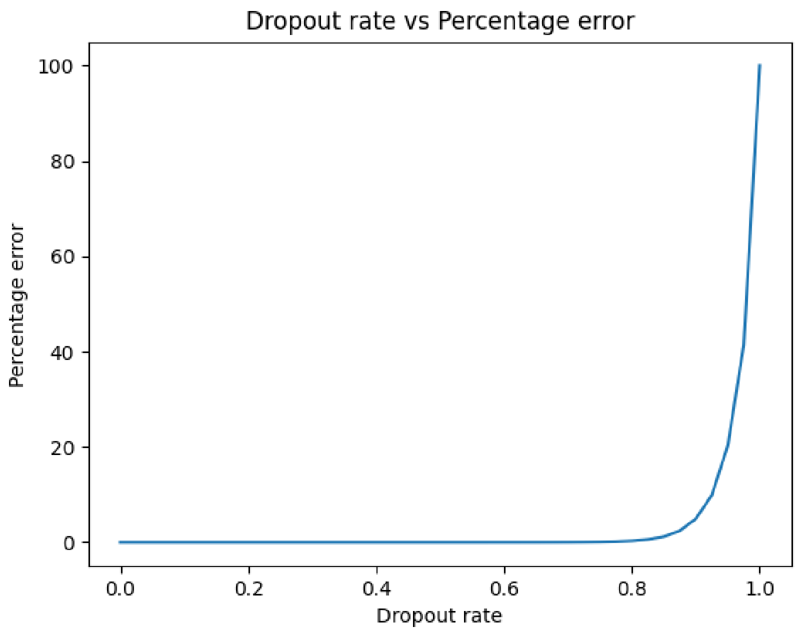
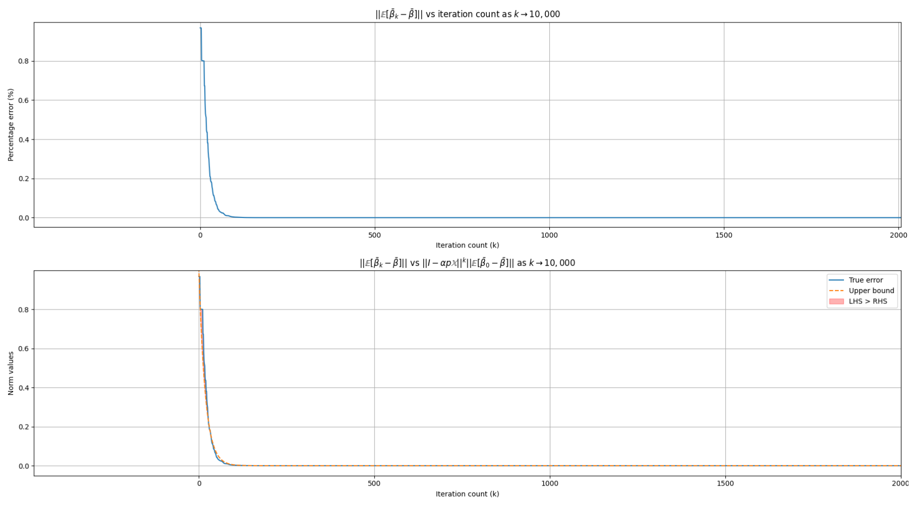


FIGURE 4: Dropout effects on a linear regression model. The Figure depicts the effect of the dropout rate on the percentage error using random minibatch sampling.

underscoring the importance of increasing the number of iterations k as the proportion of information dropped increases.



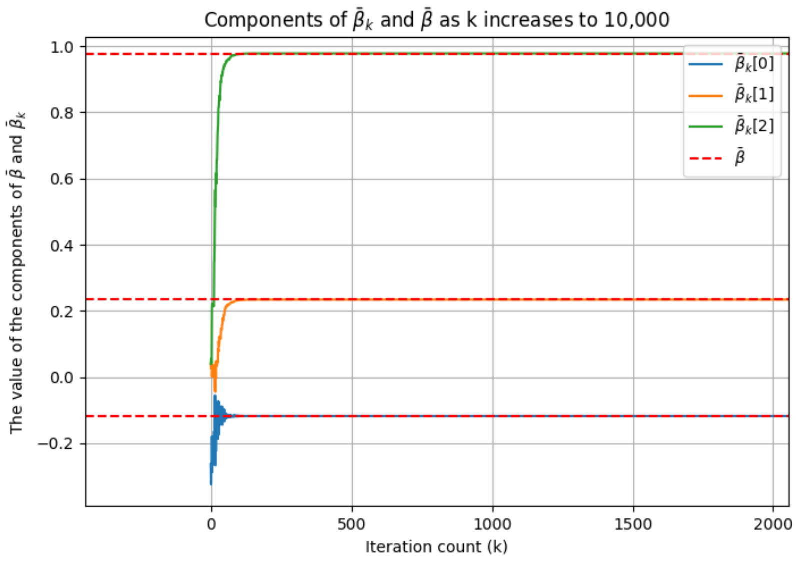


FIGURE 5: Panel (a) depicts the exponential convergence of $\bar{\boldsymbol{\beta}}_k$ to $\bar{\boldsymbol{\beta}}$, along with upper bound on convergence as shown in (13). Panel (b) showcases the different components of $\bar{\boldsymbol{\beta}}_k$ converging to $\bar{\boldsymbol{\beta}}$.

Figure 5 showcases the effects of random minibatch sampling and its effects on a linear regression model. In both panels, the iteration count cuts off at $2,000$, but the simulation was done for $10,000$ iterations. This was done in order to emphasize the initial behaviour of the model, and disregard the behaviour of the iterates after they have arrived at equilibrium.

In panel (a) of Figure 5, the inequality presented in (13) is validated. Initially, for the first 20 iterations, the inequality fails to hold, as can be seen by the red region. This is attributed to the stochastic nature of the model and is treated as a "warm-up" period until the model arrives at equilibrium. For the remaining iterations, the inequality holds. Panel (b) of Figure 5 showcases the convergence of $\bar{\boldsymbol{\beta}}_k$ to $\bar{\boldsymbol{\beta}}$. The components of both vectors are graphed, and convergence can be observed.

5.3.3 Convergence of covariance using stochastic minibatch gradient descent in linear regression

In Figure 6, we aim to verify the inequality introduced in (15). This is done in two ways: We first graph the inequality and highlight in red the region(s) where the inequality does not hold and secondly, we graph the values of $\mathbb{E}[(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}})^\top]$ and $S(\mathbb{E}[(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}})^\top])$.

The behaviour observed in both panels is as expected, as the covariance converges to the affine operator in (16) in the limit. However, as was seen earlier, the inequality posed in (16) does not hold for the first 20 iterations, which can be seen by the red region. This can again be attributed to the stochastic nature of the model, where a dozen or so iterations are needed such that model reaches equilibrium.

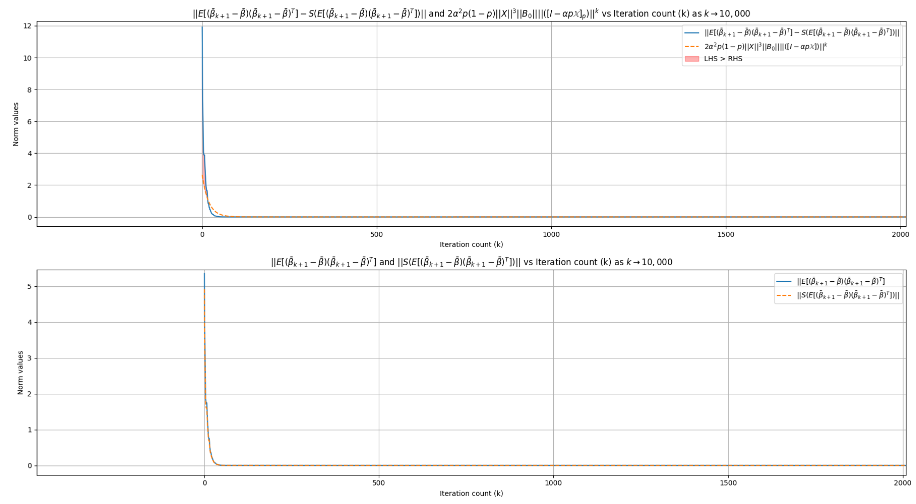


FIGURE 6: The covariance of random minibatch sampling effects on a linear regression model. Panel (a) showcases the inequality posed in (15), where the shaded the region is the region where the inequality does not hold. Panel (b) showcases the values of $\mathbb{E}[(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}})^\top]$ and $S(\mathbb{E}[(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}})(\bar{\boldsymbol{\beta}}_{k+1} - \bar{\boldsymbol{\beta}})^\top])$

5.4 Verification and validation using the MNIST dataset

In order to validate our results to an empirical (real-world) scenario, we chose to train a neural network with both gradient descent with dropout, as described in section 3, along with gradient descent with random minibatch sampling as described in section 4. We trained the neural network using the code provided by [9], where we modified the training function in order to incorporate either dropout, or gradient descent with random minibatch sampling. The intricacies of the neural network model are the same as those discussed in section 2.1. The number of epochs utilised was 30, which was used in order to facilitate convergence. The dataset we used was the MNIST dataset from [8], where the data was split into a training set and a test set. [8] also provides an efficiency rate of 95%, which we are going to use to compare against.

When analysing the effectiveness of gradient descent with dropout, we see that the training accuracy starts at around 30% and rapidly increases in the initial epochs. By epoch 10, the training accuracy stabilizes around 70%, indicating that the model is learning effectively. The close match between training and test accuracy suggests that the model generalizes

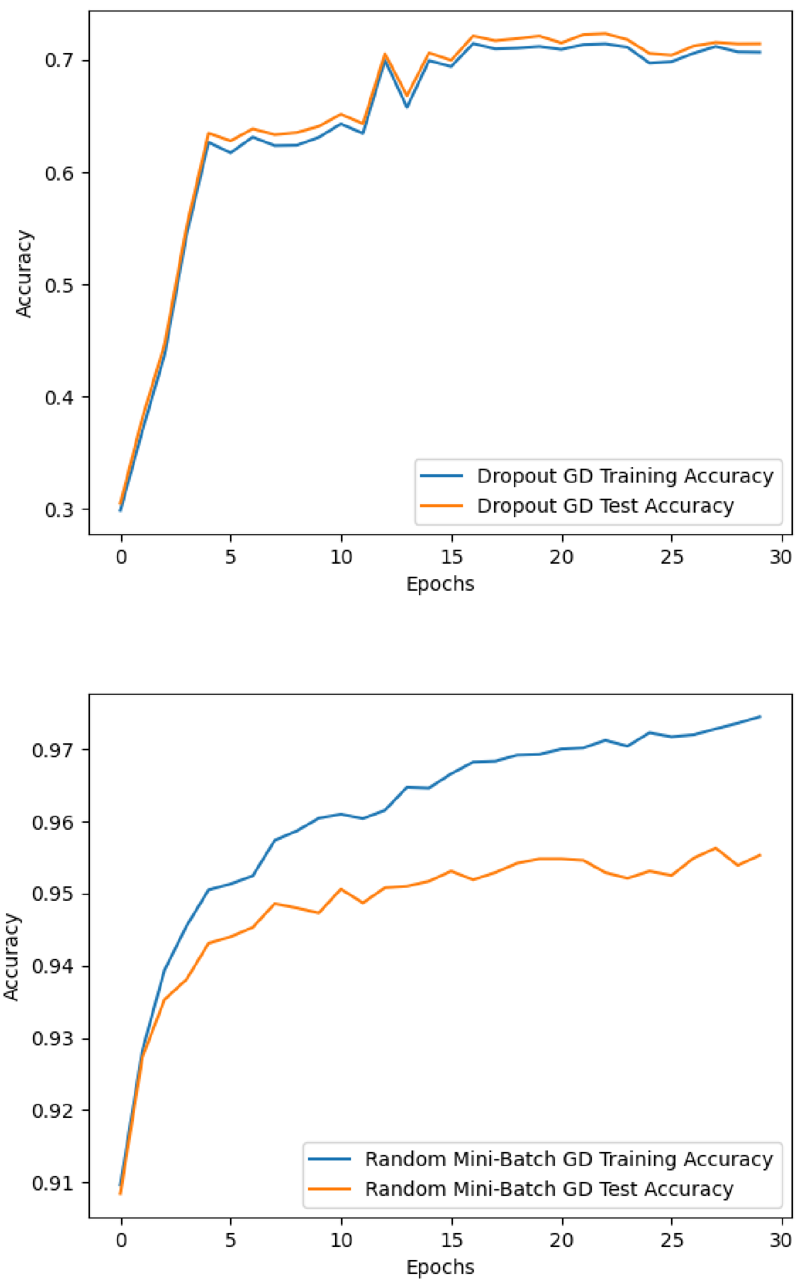


FIGURE 7: The training and test accuracy of gradient descent using dropout (Panel 1) and gradient descent with random minibatch sampling (Panel 2) along with training time of 256.74 and 274.36 seconds respectively.

well to unseen data, due to the dropout regularization.

On the other hand, in the random minibatch gradient descent, the training accuracy starts at around 90% and increases steadily, reaching over 97% by the end of the training. This higher initial accuracy compared to dropout can be attributed to the fact that all neurons are utilized in each iteration, allowing the network to learn more rapidly and effectively during each epoch. The test accuracy follows the training accuracy closely but starts slightly lower at around 91% and reaches approximately 95% by the end of the training. The gap between training and test accuracy is more pronounced than in the dropout case, suggesting a slight overfitting where the model performs better on the training data compared to the test data. This method is slightly slower than dropout, which could be due to the overhead of managing and shuffling mini-batches randomly, but it provides a more direct approach to learning without additional regularization.

In conclusion, the choice of method may depend on the importance of generalization versus achieving high training accuracy.

5.5 Evaluation of the Monte Carlo simulation

Simulation results are compared against theoretical expectations to verify the efficacy of dropout in reducing overfitting. Specifically, we analyze:

Expectation: The expected value of the model parameters under dropout conditions provides insight into the average effect of the dropout process. These theoretical expectations $(\tilde{\boldsymbol{\beta}}, \bar{\boldsymbol{\beta}})$ were derived in sections 3 and 4, and their convergence was verified in Section 5.3.2. For large enough k, which we picked to be $10,000$, the iterates $(\tilde{\boldsymbol{\beta}}_k, \bar{\boldsymbol{\beta}}_k)$ converge to the theoretical expectations $(\tilde{\boldsymbol{\beta}}, \bar{\boldsymbol{\beta}})$.

Covariance analysis: Analyzing the sample covariance matrix of the model parameters across multiple training runs offers insights into the variability introduced by dropout. A smaller covariance in key parameters may suggest that dropout helps in stabilizing learning, whereas a large covariance could indicate increased variability, which might be beneficial or detrimental depending on the specific context and application. For this model, the affine operator defined in (16) aims to model the behavior of the covariance of $\bar{\boldsymbol{\beta}}$. The simulation in 5.3.3 fits the theoretical limits derived in (15).

Errors: asymptotic statement, picking a finite k: Theoretical analysis often involves asymptotic behaviors as model complexity increases. By choosing a finite k—the number of iterations—we can empirically test these asymptotic predictions. This allows us to evaluate how well the dropout technique scales with increasing model complexity and whether the asymptotic benefits (e.g., regularization effects) hold in practical, finite-dimensional settings.

Picking a finite amount of samples: To ensure the robustness of our findings, a finite but sufficiently large sample size is crucial. This helps in approximating the true distribution of model parameters and their performance metrics more accurately. The choice of sample size affects the precision of the estimation and the confidence in the dropout model's effectiveness, particularly in terms of statistical significance and the power of hypothesis tests used in the analysis.

6 Conclusion and Future Work

In this thesis, we explored the connections between gradient descent, dropout, and random minibatch sampling in the context of linear regression. Our investigation focused on understanding how these techniques influence the convergence properties and regularization effects in linear regression models.

We began by introducing the concept of dropout in Section 2, a regularization method widely used in neural networks, and then applied it to the simpler context of gradient descent for linear regression in Section 3. Analyzing dropout in the linear regression model represents a simplified framework. This simplification allows us to gain a deeper understanding of dropout's effects in a controlled setting.

Our analysis demonstrated that dropout, by introducing noise and stochasticity during the training process, effectively reduces overfitting and enhances model generalization. This behaviour aligns with theoretical expectations and underscores dropout's role as a regularizer, akin to traditional methods like L_2 regularization. Although linear models are not as complex or widely used in deep learning, these insights can be extended to more sophisticated models. Understanding dropout in this simplified context paves the way for applying similar principles to more complex and practically relevant neural network architectures.

Furthermore, we integrated random minibatch sampling into the gradient descent algorithm in Section 4. This hybrid approach was rigorously examined through theoretical models and validated using extensive Monte Carlo simulations in Section 5. Our results demonstrated that combining dropout with minibatch sampling not only accelerates the convergence of gradient descent but also maintains stability and reduces variance in the parameter estimates. This contrast between dropout and minibatch sampling proves beneficial in managing the complexity and size of datasets, making the learning process more efficient and scalable.

6.1 Future Work

While this thesis provides significant insights into the convergence and regularization effects of combining dropout with random minibatch sampling, several avenues for future research remain open:

- **Extension to non-linear models**: The current study primarily focuses on linear regression models. Future work could extend the analysis to more complex models such as deep neural networks, where the benefits of dropout and minibatch sampling might be even more pronounced.
- **Dropout variants**: Exploring the impact of various dropout techniques, such as DropConnect or DropBlock, on different types of neural networks could provide further insights into their regularization effects and convergence properties.
- **Dynamic dropout rates**: Developing adaptive schemes for adjusting dropout rates during training, based on the progress of learning and data characteristics, could lead to more efficient and effective training processes.

7 Appendices

A Proof of the Inequality $\lambda_{\min}(\mathbb{X}) \leq \|X\|^2$

Lemma A.1. *Let X be a matrix and $\mathbb{X} = X^T X$. Then, the smallest eigenvalue of $\mathbb{X}$ satisfies:*

$$\lambda_{\min}(\mathbb{X}) \leq \|X\|^2$$

Proof. Let $X \in \mathbb{R}^{m \times n}$ and consider the matrix $\mathbb{X} = X^\top X$.

First, note that $\mathbb{X}$ is symmetric and positive semidefinite. For any vector $v \in \mathbb{R}^n$,

$$v^\top \mathbb{X} v = v^\top (X^\top X) v = (Xv)^\top (Xv) = \|Xv\|_2^2 \geq 0$$

This shows that $\mathbb{X}$ is positive semidefinite, implying that all its eigenvalues are non-negative.

The spectral norm of a matrix X is defined as:

$$\|X\| = \sup_{\|v\|_2 = 1} \|Xv\|_2$$

where $\|v\|_2$ denotes the Euclidean norm of the vector v. This norm is equal to the largest singular value of X.

The singular values of X are the square roots of the eigenvalues of $\mathbb{X} = X^\top X$. Let $\sigma_1, \sigma_2, \ldots, \sigma_n$ be the singular values of X. Then, the eigenvalues of $\mathbb{X}$ are $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$.

The spectral norm $\|X\|$ is the largest singular value of X:

$$\|X\| = \sigma_{\max}$$

where $\sigma_{\max}$ is the largest singular value of X.

Therefore, the smallest eigenvalue of $\mathbb{X}$ is:

$$\lambda_{\min}(\mathbb{X}) = \min \sigma_i^2$$

Since $\sigma_{\max}$ is the largest singular value, we have:

$$\lambda_{\min}(\mathbb{X}) \leq \sigma_{\max}^2$$

By the definition of the spectral norm, we have:

$$\sigma_{\max} = \|X\|$$

Thus, we get:

$$\lambda_{\min}(\mathbb{X}) \leq \|X\|^2$$

This completes the proof. □

References

[1] Gabriel Clara, Sophie Langer, and Johannes Schmidt-Hieber. *Dropout Regularization Versus $\ell_ 2$-Penalization in the Linear Model.* June 18, 2023. DOI: 10.48550/arXiv.2306.10529. arXiv: 2306.10529[math,stat]. URL: http://arxiv.org/abs/2306.10529.

[2] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. *DropBlock: A regularization method for convolutional networks.* Oct. 30, 2018. DOI: 10.48550/arXiv.1810.12890. arXiv: 1810.12890[cs]. URL: http://arxiv.org/abs/1810.12890.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. "Linear Methods for Regression". In: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Ed. by Trevor Hastie, Robert Tibshirani, and Jerome Friedman. New York, NY: Springer, 2009, pp. 43–99. ISBN: 978-0-387-84858-7. DOI: 10.1007/978-0-387-84858-7_3. URL: https://doi.org/10.1007/978-0-387-84858-7_3.

[4] Roger A. Horn and Charles R. Johnson. "Matrix Analysis | Algebra". In: *Higher Education from Cambridge University Press.* ISBN: 9781139020411 Publisher: Cambridge University Press. Oct. 22, 2012. DOI: 10.1017/CBO9781139020411. URL: https://www.cambridge.org/highereducation/books/matrix-analysis/FDA3627DC2B9F5C3DF2FD8C3CC136B48.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (May 24, 2017), pp. 84–90. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386. URL: https://dl.acm.org/doi/10.1145/3065386.

[6] Johanna Hedlund Lindmar, Chang Gao, and Shih-Chii Liu. "Intrinsic sparse LSTM using structured targeted dropout for efficient hardware inference". In: *Lindmar, Johanna Hedlund; Gao, Chang; Liu, Shih-Chii (2022). Intrinsic sparse LSTM using structured targeted dropout for efficient hardware inference. In: AICAS 2022, Incheon, Korea, 13 June 2022 - 15 June 2022.* AICAS 2022. Incheon, Korea: University of Zurich, June 15, 2022. DOI: 10.1109/AICAS54282.2022.9869988. URL: https://www.zora.uzh.ch/id/eprint/219685/.

[7] Poorya Mianjy and Raman Arora. *On Dropout and Nuclear Norm Regularization.* May 28, 2019. DOI: 10.48550/arXiv.1905.11887. arXiv: 1905.11887[cs,stat]. URL: http://arxiv.org/abs/1905.11887.

[8] *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges.* URL: http://yann.lecun.com/exdb/mnist/ (visited on 05/31/2024).

[9] Michael A. Nielsen. "Neural Networks and Deep Learning". In: (2015). Publisher: Determination Press. URL: http://neuralnetworksanddeeplearning.com (visited on 05/31/2024).

[10] *projects/bachelor_thesis/src at main · yazanmashal03/projects.* URL: https://github.com/yazanmashal03/projects/tree/main/bachelor_thesis/src (visited on 06/19/2024).

[11] Clement W Royer. "Lecture notes on stochastic gradient methods". Dec. 10, 2023. URL: https://www.lamsade.dauphine.fr/~croyer/ensdocs/SG/LectureNotesOML-SG.pdf.

[12] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. ISSN: 1533-7928. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[13] Li Wan et al. “Regularization of Neural Networks using DropConnect”. In: *Proceedings of the 30th International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228. PMLR, May 26, 2013, pp. 1058–1066. URL: `https://proceedings.mlr.press/v28/wan13.html`.