UNIVERSITY OF SOUTHAMPTON

Improving Training and Inference for Embedded Machine Learning

by

Jia Bi

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Engineering and Physical Science
School of Electronics and Computer Science

September 2020
Many emerging applications are driving the development of Artificial Intelligence (AI) for embedded systems that require AI models to operate in resource constrained environments. Desirable characteristics of these models are reduced memory, computation and power requirements, that still deliver powerful performance. Deep learning has evolved as the state-of-the-art machine learning paradigm becoming more widespread due to its power in exploiting large datasets for inference. However, deep learning techniques are computationally and memory intensive, which may prevent them from being deployed effectively on embedded platforms with limited resources and power budgets. To address this problem, I focus on improving the efficiency of these algorithms. I show that improved compression and optimization algorithms can be applied to the deep learning framework from training through inference to meet this goal.

This thesis introduces a new compression method that significantly reduces the number of parameters requirements of deep learning models by first-order optimization and sparsity-inducing regularization. This compression method can reduce model size by up to 300× without sacrificing prediction accuracy. To improve the performance of deep learning models, optimization techniques become more important, especially in large-scale applications. As a result, I develop two new first-order optimization algorithms that improve over existing methods by controlling the variance of the gradients, determining optimal batch sizes, scheduling adaptive learning rates, and balancing biased/unbiased estimations of the gradients, which can improve the convergence rate to provide a lower computational complexity.
# Contents

Declaration of Authorship  
Acknowledgements  

1 Introduction  
  1.1 Research Challenges  
  1.2 Research Motivations  
  1.3 Thesis Contributions and Outline  
  1.4 Publications  

I Machine Learning Models on Embedded Systems  

2 Background of Model Compression for Inference  
  2.1 Deep Learning Architecture  
  2.2 Deep Learning on Embedded System  
    2.2.1 Memory Constraints  
    2.2.2 Computation/Power Constraints  
  2.3 Model Compression Techniques  
    2.3.1 Weight Reduction  
    2.3.2 Weight Quantization  
    2.3.3 Weight Regularization  
  2.4 Optimization with Sparse Regularization  
    2.4.1 SGD with $\ell_1$ regularization  
    2.4.2 SGD with cumulative $\ell_1$ regularization  

3 Sparse Deep Neural Networks for Embedded Intelligence  
  3.1 Introduction  
  3.2 Non-convex SVRG for cumulative $\ell_1$ regularization  
    3.2.1 Hybrid Adaptive Learning Rate and Variance Control  
    3.2.2 Learning Rate for Cumulative $\ell_1$ regularization  
    3.2.3 Bias-based Pruning  
  3.3 Experiments  
    3.3.1 Experimental Settings  
    3.3.2 Evaluating the Range of $\lambda$  
    3.3.3 Evaluation of the Trade-off Between Memory Requirements and Performance  
    3.3.4 Comparison with Leading Results
CONTENTS

3.4 Discussion .................................................. 31

II Improving Optimization Algorithms for Model Learning 33

4 Background of Model Optimization for Training 35
4.1 Problem Setting ............................................. 35
4.2 Convex and Non-convex Optimization .......................... 36
4.3 Performance Evaluation ...................................... 37
4.3.1 Convergence Criteria .................................. 37
4.3.2 Black Box Oracles ...................................... 38
4.4 Optimization Techniques .................................... 38
4.4.1 Gradient Descent ...................................... 38
4.4.2 Stochastic Gradient Descent ............................ 39
4.4.3 Stochastic Variance Reduced Gradient ............... 39
4.5 Performance of Non-convex Optimization Techniques ...... 40
4.5.1 Unbiased and Biased Estimations ...................... 40
4.5.2 Batch and Mini-batch Size ............................. 40
4.5.3 Learning Rates ......................................... 41

5 A Stochastic Gradient Method with Hybrid Estimation 43
5.1 Introduction .................................................. 43
5.2 Preliminaries ................................................ 45
5.3 Related works ............................................... 46
5.4 Integrated SVRG with biased estimation ....................... 47
5.4.1 Weighted unbiased estimator analysis .................. 47
5.4.2 Biased estimator analysis ............................... 49
5.4.3 Variance control and combined biased and unbiased estimation 52
5.4.4 Comparison of the convergence rates ................. 53
5.5 Application ................................................... 54
5.6 Discussion .................................................. 55
5.7 Proof details ................................................ 56

6 A Variance Controlled Stochastic Method with Biased Estimation 69
6.1 Introduction .................................................. 69
6.2 Preliminaries ................................................ 71
6.3 Variance controlled SVRG with a combined unbiased/biased estimation ........................................... 72
6.3.1 Weighted unbiased estimator with one-epoch analysis .... 73
6.3.2 Biased estimator on one-epoch analysis ................ 75
6.3.3 Convergence analysis for smooth non-convex objectives 76
6.3.4 Best of two worlds ................................... 79
6.4 Application .................................................. 82
6.5 Discussion .................................................. 83
6.6 Proof details ................................................ 83

7 Conclusions and Further Work 101
# List of Figures

1.1 Ranking the limitation of computational cost in different hardware architectures if DL models are applied. ........................................ 2
1.2 Thesis contributions focus on improving the efficiency and sparsity of deep learning models from multiple aspects. .......................... 4

2.1 Artificial neuron. ........................................................................ 9
2.2 Comparison Neural Network between Single and Multiple Hidden Layer. ................................................................. 11
2.3 LeNet-5 [LeCun et al. 2001] Architecture. .................................... 11
2.4 VGG-16 [Simonyan and Zisserman 2014a] Architecture. ............... 12
2.5 The process of pruning method [Han et al. 2016b]. ......................... 14
2.6 Comparison of Lasso family in a neural network. ............................. 17

3.1 Effects of $\lambda$ on the performance and convergence rate when $\eta$ is decayed. ................................................................. 27
3.2 Effects of $\lambda$ on the performance and convergence rate when $\eta$ is fixed. ................................................................. 28
3.3 The performance of SSVRG evaluation with other two baseline methods. ................................................................. 29
3.4 Comparison of the convergence rate of SSVRG and MSVRG. ............. 30

5.1 Comparison of rates of convergence in three approaches, including SGD, MSVRG and $J\text{SVRG}^+$ via test loss/error. ................. 56

6.1 Comparison of rates of convergence in four approaches, including SGD, SVRG, SCSG and VCSG. .................................................. 81
6.2 Visualization of test error of four approaches, including SGD, SVRG and SCSG and VCSG against time consumption. ...................... 82

7.1 Summary of the thesis. ................................................................ 102
List of Tables

3.1 Comparison of five compression methods on the MNIST data-set. . . . . . . 29
3.2 SSVRG compression results by layer on two DL models for MNIST data-set. 31
5.1 Comparison of the IFO complexity of different algorithms. . . . . . . . . . . . . 45
6.1 Comparison of results on IFO calls (defined in Definition 4.3.2) of gradient methods for smooth non-convex problems. . . . . . . . . . . . . . . . . . . . . . . 71
Declaration of Authorship

I, Jia Bi declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

Improving Training and Inference for Embedded Machine Learning

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed by myself;
7. Either none of this work has been published before submission, or parts of this work have been published as: see Section 1.3.

Signed:

Date:
Acknowledgements

First and foremost, I would like to express my deepest gratitude to my Ph.D. advisor, Professor Steve R. Gunn, for his unwavering support throughout my journey at University of Southampton. Steve has been an exceptional advisor. In retrospect, I learnt from Steve how to estimate real problems in year one, solve these problems in year two and three, and spread the discovery in year four. His brilliance, sharp insights and amazing expertise in almost every aspect of machine learning and embedded system has been a great source of inspiration for me. His words and deeds let me know a qualified research scientist who should have the attitude towards science such as patience, honest, humility, respect for logic and critical thinking. I would also thank my co-advisor, Dr. Terrence Mak. Terrence offered me invaluable advice and diligently guided me through challenging problems.

I gave my sincere thanks to Professor Mahesan Niranjan, who is my course lecturer in machine learning. He opened machine learning world for me. His ambition and foresight ignited my passion for my further Ph.D. research in machine learning.

I have been extremely fortunate to collaborate with a wonderful set of colleagues: Chenyang Zhu, Runshan Hu, Jie Zhan, Rafael Gutierrez Nuno, Dr. Liang Wang, Wei Zhen Leong, Amin Sabet, Dr. Kueh Si Mon and Dr. Lorenzo Servadei. Special mention goes to Chenyang Zhu and Runshan Hu. Their exceptional insights and ideas, and strong background of machine learning have been instrumental in my research.

And I give sincere thanks to my fiance, Chuan Ma who is not only my fiance but also my soulmate, and to his family. Their constant encouragements and support lead me through the valley of darkness. I would also like to thank my friends: Annabelle Yin, Dr. Peijun He, Iris Li, Silei Peng, Dr. Zhibo Qu, Dr. James Byers, Dr. Chiren Jeevi and Yanli Qi, for their invaluable support throughout my life.

Last but most important, I would like to thank my family, Kejun Bi and Lin Gu, for their unconditional love, support and countless sacrifices at every moment of my life. Without their love, care, guidance and support, none of these would have been possible. It is to them I dedicated this thesis.
Deep learning (DL) techniques for artificial intelligence have become an indispensable part of our modern society, which are finding wider application across a number of domains such as computer vision (Szeliski 2010), natural language processing (Deng and Liu 2018) and intelligent robotics (Russell and Norvig 2009). Most of these applications prompt the development of efficient computing machinery in DL techniques to recognize patterns from large-scale data through a training process and then make predictions on unseen data through the process of inference. Due to the exceptional performance DL techniques bring to artificial intelligence there is a requirement to extend their reach to applications powered by embedded devices, such as smart phones, drones, and self-driving cars. However, resource limitations of embedded systems pose some challenges for the deployment of DL training and inference techniques.

1.1 Research Challenges

The first challenge is the limited computational resource and power of embedded systems, since DL models require considerable computational cost in both the training and test process. Figure 1.1 illustrates the increasing computing resource available for different platforms ranging from embedded devices to high performance computers. For example, the energy consumption in micro-controller use micro-Watts unit. However, currently the power in smart mobile phone (Iphone X) is up to 18W. Whilst DL methods have been implemented on some of the more powerful platforms their realisation on more limited resource based platforms such as micro-controllers and embedded systems is still in its infancy. In the training process, the performance of the optimization algorithm is a crucial factor affecting the computational cost of DL models since the goal of training is to stop the training process once the optimization algorithm has converged to an optimal value of parameters for the model (Reddi et al. 2016a). However, since the resulting objective function of DL models are non-convex problems, optimization algorithms for
DL models require significant computation and there are no guarantees of obtaining the global optima \cite{JainKar2017}. Furthermore, the power of DL methods is realised when large datasets are deployed for training which compounds their deployment on embedded devices. As a result, improving the efficiency of non-convex optimization techniques for embedded DL is urgently needed to address this challenge. Moreover, in the inference process the high computational cost is also generated by the complicated structure of DL models. For example, to achieve around 90% top-5 accuracy on ImageNet database, ResNet-50 model on an embedded device Arria-10 FPGA demonstrates 7.74 Giga Operation per second (GOPS) with 27.2 ms of latency breakdown per image. ResNet-152 as a more complicated structure can demonstrate 315.5 GOPS, but higher latency as 71.7ms due to the high computational cost per image. As a result, to save the computational cost that is generated in the inference process, I consider to simplify the structure of DL models, which also can achieve higher accuracy with low latency breakdown per image.

Another key challenge is the memory limitation of embedded systems. To solve more and more complicated AI applications, DL models are designed having more and more number of layers (deeper) so as to achieve good performance. However, deeper DL models would incur larger memory footprints. For example, ResNet-152 (152 layers) improves 1.1% accuracy of ResNet-50 (50 layers) on the ImageNet data-set by increasing $3 \times$ of and the number of layers than ResNet-50 \cite{Hu2018}. In terms of the memory footprints that are calculated by multiplying the number of weights with bits precision, the ResNet-152 has 60.3M number of weights with 32 bits to store each weight; the memory requirement of ResNet-152 is 60.3M \times 4 = 243MB, while for 4 cameras, the total memory requirement of ResNet-152 is up to 1GB, which is prohibitive on many embedded devices. Moreover, in the inference process, the large size of DL models would have high energy consumption due to the high computational cost, which may well exceed the available power budget. DL models are more easily stored on off-chip memory such as DRAM in embedded devices rather than on-chip storage, but require larger DRAM accesses than on-chip. For example, the energy cost per 32 bit operation in a 45nm technology ranges from 3pJ to 640pJ for off-chip memory access \cite{Han2015}. If running DL models with around 1 billion weights on an Iphone XS, which is one of the current advanced smart-phones, at 60HZ would require $60HZ \times 1G \times 640pJ = 38.4W$ just for DRAM access, which is beyond the power envelope of the Iphone XS (18W). As
a result, smaller models can bring two benefits in small memory requirements and low power consumption.

Based on the two challenges in real applications, correspondingly, my research problems are summarised as how to significantly reduce the memory requirements of DL models without performance loss enabling models to be applied for embedded platform; and how to significantly reduce the computational requirements of DL models enabling training on embedded systems to be more attractable.

1.2 Research Motivations

In this thesis, our motivation is to analyse optimization and compression algorithms for embedded DL models, which allow models to become more computationally-efficient and less memory intensive on embedded systems without performance loss. Tesla self-driving cars require periodic updates of the learning models from their public cloud or servers. However, the current communication systems in an over-the-air (OTA) updating rule may have a risk of network latency. Our algorithms can be applied in DL models, to provide increased efficiency to enable them to be directly stored into the car’s embedded systems. Inference and training processes of the DL models can then run on the embedded systems enabling dynamic updates to the DL models from real-time sensor input. Consequently, to achieve these goals, we focus on how to efficiently estimate optimal solutions of DL models in the training process, and how to simplify and compress DL models with the same accuracy of original models for the inference process.

1.3 Thesis Contributions and Outline

This thesis is divided into two parts to introduce model compression and optimization algorithms. The model compression techniques are improved by introducing a weight regularizer in Part I, and the model optimization techniques are improved by accelerating convergence of optimization in Part II. Our contribution in thesis centers around promoting the efficiency of neural networks by advanced optimizations $ISVRG^+$ and $VCSG$ that can accelerate the convergence of optimization in training process, and improving the sparsity and prediction accuracy of networks by our compression method $SSVRG$ that can significantly reduce the number of parameters of trained neural networks without sacrificing accuracy in the inference process, which are illustrated in Figure 1.2. Part I: Chapter 2 provides background of DL models, the limitations of embedded DL models, model compression techniques. Moreover, I list DL architectures, datasets and the hardware platform I used for my experiments in the thesis.
Chapter 3 proposes SSVRG that is an efficient compression method using a $\ell_1$-regularization technique working with a variance controlled optimization algorithm that is based on stochastic variance reduction gradient (SVRG) optimization, which can fast approximately observe the optimal value and also reduce the number of weights of DL models without accuracy loss during the training process. The content of this chapter is based primarily on [Bi and Gunn, 2018]. As we use an empirical method to evaluate the performance of our algorithm, the further work focuses on analysing an improved variance controlled SVRG on non-convex problems to improve the efficiency of optimization in the training process via a theoretical way, which is discussed in the following chapter.

Part II:

Chapter 4 provides the background of the model optimization algorithms on non-convex problems. We introduce three typical first-order optimization techniques including stochastic gradient descent (SGD), gradient descent (GD) and SVRG. Moreover, we also introduce three important factors that would impact on the performance of these optimization algorithms.

Chapter 5 proposes the algorithm $ISVRG^+$ which can speedup the efficiency of training by balancing the trade-off between two versions of estimations of the gradients including biased and unbiased estimations and controlling the variance of gradient in SVRG. $ISVRG^+$ reduces the computational cost of SVRG more than two orders of magnitude. The content of this chapter is based primarily on [Bi and Gunn, 2019]. The further work aims to further accelerate the convergence of $ISVRG^+$ by using batching techniques to reduce the number of training samples $n$, which will be discussed in next chapter.

Chapter 6 proposes an algorithm as variance controlled SVRG ($VCSG$). $VCSG$ use the batching technique on optimization to provide an appropriate subset of training samples instead of whole samples in each iteration, which can significantly save computational cost during the training process. To our best knowledge, $VCSG$ achieves the lowest
computation complexity, which further reduces the computational cost of ISVRG$^+$ by two orders of magnitude. The content of this chapter is based primarily on [Bi and Gunn 2019].

Chapter 7 summaries this thesis and discuss the further work for efficient embedded deep learning.

1.4 Publications

The work in this thesis has contributed to the following publications:


Part I

Machine Learning Models on Embedded Systems
Chapter 2

Background of Model Compression for Inference

Artificial neural networks are a popular machine learning technique for many learning tasks ranging from computer vision, natural language processing to signal processing. Neural networks consist of a number of neurons and connections (known as weights). Input data are transferred to neurons by weights, and then produce outputs via weighted-sum activation functions with bias terms. An activation function is used to introduce non-linearity and mimic the behaviour of a simple neuronal model. Commonly employed activation functions include a Sigmoid, Tanh and Rectified Linear Unit. Neurons are organised by layers and are not connected in the same layer. Figure 2.1 illustrates the operation of a single neuron. The inputs \{x_i\}_{i=1}^n are multiplied by the corresponding weights \{w_i\}_{i=1}^n and are summed with the bias term before passing them through the activation function to produce the output \(y = f(\sum_{i=1}^n w_i x_i + b)\). A typical neural network architecture contains an input layer, a number of hidden layers and an output layer. The hidden layers will contain many neurons and the output layer will contain one neuron for each target. In order to obtain suitable values for the weights the network undergoes a training procedure using a loss function, such as Mean Squared Error (MSE) or Cross Entropy. This provides an objective function which is minimized by a suitable
optimization method to determine the weights and biases in the network. In practice the training process is decomposed into a feed-forward step to compute the error in target predictions, with a back-propagation step to adjust the weights to reduce the error \cite{Rumelhart1987}. This is necessary because a feed-forward process has no ability to determine which of hidden units generate the error by comparing the differences between desired and actual output vector of nets \cite{Bishop1995}. Back-propagation requires a differentiable activation function to ensure that the target error can be propagated back to the relevant weights. Then gradient-based optimization methods are used to evaluate derivatives of the error with respect to the weights. Commonly employed approaches include Gradient Descent (GD) and Stochastic-Gradient (SG)-based optimization methods. As a result, optimization plays an important role in the training process. In contrast, optimization is not required for the inference process which is entirely satisfied by the feed-forward operation.

2.1 Deep Learning Architecture

To improve the accuracy of neural networks especially on large-scale applications, the number of hidden layers is increased, resulting in the structure of the network becoming more complicated and the training process more challenging. Such neural networks are termed deep learning models. In this section, I give an overview of different types of DL models, including Multi-Layer Perceptrons (MLP), Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Typically, a MLP is a three-layer network including an input layer, a hidden layer and a output layer, which is shown in Figure 2.2(a). In MLP models, all layers are fully-connected, and each layer is followed by a non-linear activation function. DNNs as illustrated in Figure 2.2(b) improve the performance of MLP by increasing the number of hidden layers, which enables more robust training to be applied. An early DL model was introduced by \cite{LeCun1998} in the form of LeNet-300-100 is a fully connected network with two hidden layers containing 300 and 100 neurons respectively \cite{LeCun2001}. Subsequently with the development of improved training algorithms the ability to train networks with larger number of layers has been realised. An interesting artefact of DNN models is that often the layers can be seen to learn higher and higher order representations of the object classes. For example, in image processing tasks for visual pattern recognition, neurons in a deep network can recognise more complex shapes, such as rectangles, triangle, etc, and are intrinsically more powerful than shallow networks \cite{Pascanu2013}. However, DNNs face drawbacks associated with the increased number of layers, which increases the computational complexity required to calculate the weights and introduces extra memory requirements to store them \cite{Hinton2006}. CNNs are biologically-inspired variants of MLP which exploit local spatial correlations by processing a local connectivity pattern between neurons of adjacent layers \cite{LeCun2001}. In image recognition,
Chapter 2 Background of Model Compression for Inference

Figure 2.2: Comparison of Neural Networks between Single and Multiple Hidden Layers

CNNs use small spatially contiguous neurons called *receptive fields* to connect multiple layers and process portions of the input image (Krizhevsky et al., 2012). LeNet-5 was an early CNN model which has 5 hidden layers including two convolutional layers and two fully-connected layers, and is shown in Figure 2.3. VGG-nets (Simonyan and Zisserman, 2014a) is a more recent CNN model which was the winner of ImageNet ILSVRC-2014 competition that has strong classification and localisation performance, which is widely used in image classification, segmentation tasks. VGG-16 has 13 convolutional layers and 3 fully-connected layers, which has 138 total million weights, which the architecture is illustrated in Figure 2.4. More recent DL models are capable of learning with over 100 hidden layers such as the Residual Network (ResNet) (Hu et al., 2018). Compared to DNNs, CNNs require less storage for parameters since their convolutional layers are typically sparsely connected with weight and parameter sharing by respective fields. Compared to DNNs and CNNs, RNNs can be used to capture the temporal information of the input signal and share the weights in time. Due to the long time stamp, RNN suffer from both the vanishing and exploring gradient problems (Jozefowicz et al., 2015).
Figure 2.4: VGG-16 (Simonyan and Zisserman, 2014a) Architecture.

which is addressed by Long Short-Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) that enable uninterrupted gradients to be flowed with the hidden cell (Jozefowicz et al., 2015; Han et al., 2016b). Compared to CNN from the hardware perspective, LSTMs have inefficient performance related to a low ratio of operations per weight, which results in the cheap computation per weights but expensive fetching data (Dally, 2017).

2.2 Deep Learning on Embedded System

The increasing number of layers of DL technologies demand greater hardware resources with respect to memory, computation, and power. This makes their deployment on embedded systems more challenging due to the constraints of these hardware resources.

2.2.1 Memory Constraints

It is difficult to store DL models on embedded systems due to the limited memory capacity. To store DL models, memory is required to store input data, weights, and activation outputs. When applying a DL model on an embedded device there are two approaches which can be adopted, one is to store the DL model in the cloud and upload data for inference and retrieve the result, and the second approach stores the model and computes the inference on the embedded system. The advantage of the first approach is that it releases hardware resources on the embedded system, but the disadvantage is that it is now critically dependent on network connectivity and latency. In contrast, the second approach is now dependent on the local hardware resources but independent of the network. To give this some perspective, the total number of weights in AlexNet is 61 million (Krizhevsky et al., 2012), and in VGG is 138 million weights (Simonyan and Zisserman, 2014a). A more recent DL model ResNet-152 has more than 6 billion weights (He et al., 2015). Whilst many personal computers are able to store these network representations and perform training and inference through their CPU and GPU capability, it is not possible to realise these networks on embedded platforms due to
the limited memory size. The memory capacities typically range from 500M weights for a mobile GPU and FPGA platform (e.g. Nvidia Tegra X1 and Angel Eye (Guo et al., 2018)), 336M weights for an ASIC platform (e.g. EIE-256PE (Han et al., 2016a)) and 2K for a micro-controller platform (e.g. ST-Microelectronics-32). Furthermore, the large number of weights demand a high memory bandwidth of the hardware (Ding and Kennedy, 2000).

2.2.2 Computation/Power Constraints

Implementing DL models on embedded systems is also challenging due to the restricted computational capability and limited power budget. The fundamental computational operation required for DL implementation is the Multiply-Accumulate (MAC) operation which is heavily used both in training and inference. Ultimately the number of these operations will be dictated by the network architecture, including the number of layers, number of neurons, and dimension of the input. Furthermore, the mapping of these operations onto the hardware is also of importance. For example, a DNN network (Jouppi et al., 2018) having 5 fully connected layers with 20M weights can exploit 61% workload of a Google Tensor Processing Unit (TPU). Whereas a LSTM network (Wu et al., 2016) having 24 fully-connected layers with 52M weights is only able to exploit 29%. Finally, a CNN network (Silver et al., 2016) having 72 convolutional layers with 100M weights is only able to exploit 5% of the TPU capability (Jouppi et al., 2017). A further factor which can affect the power budget of an embedded system is the limited memory bandwidth, which restricts the load and fetch times. This is relevant to DL models as a large number of parameters including weights and activation function outputs will need to be moved to/from external memory (e.g. DRAM).

2.3 Model Compression Techniques

Deep neural networks are typically over-parameterized, and are redundant (Han et al., 2016b; Cun et al., 1990; Denil et al., 2013), which results in inefficiencies of both computation and memory. From the previous section the memory and computational requirements are related to the size, connectivity and number representation of the network architecture. However, whilst the large number of layers, neurons and connections (weights) enable effective training, there is a possibility that once trained a smaller network could be found that approximates the large network with similar performance (Hinton et al., 2015). As a result, techniques that are able to reduce the size and connectivity of the network architecture are attractive for deploying these models on embedded systems. There are two main approaches to reduce the memory footprint of models by either reducing the number of weights in the model and/or reducing the number of bits to store each weight (Han et al., 2016b; Guo et al., 2016; Gong et al., 2014; Wu et al., 2016).
The number of bits required to store the network representation is given by

\[ M = N \times B, \]  

where \( N \) is the total number of weights and \( B \) is the number of bits required to store each weight.

### 2.3.1 Weight Reduction

An early approach to compress the network was Optimal Brain Damage and Optimal Brain Surgeon proposed by (Cun et al., 1990; Hassibi et al., 1993), which reduced the number of connections based on the Hessian of the loss function. Another idea from matrix factorization was applied to compress parameters in models by finding a low-rank approximation of the weight matrix (Denton et al., 2014). However, in practice whilst it improves computation performance, it does not significantly reduce the memory requirements. Pruning is a popular compression method which is a direct approach to remove connections with small values and emphasize the important connections with large weight values in all layers of the network (Han et al., 2016b). One method called Deep compression (Han et al., 2016b) uses pruning as part of a three-step compression method which is illustrated in Figure 2.5. After training the model, a pruning method removes weights with values below a threshold, converting the dense weight matrix into a sparse matrix. Then the final step is retraining the network where only the remaining sparse weights are updated. An advantage of pruning is that this method is simple to implement. However, the threshold has an important effect on the performance of the final model and must be chosen carefully. A second disadvantage is that the pruning method needs to train models twice resulting in increased computation. A similar pruning method, Dynamic Network Surgery (DNS) uses pruning to remove unimportant weights.

**Figure 2.5:** The process of pruning method (Han et al., 2016b).
which combines with splicing techniques to recover weights in case of over pruning and incorrect pruning (Guo et al., 2016). Another approach called HashedNets exploits the concept of weight sharing by approximating many weights with a single weight and an index table to provide compression. The method bins network connections into hash buckets by a randomized hash function (Chen et al., 2015). Another weight sharing approach uses k-means clustering to identify the shared weights for each layer of a trained network (Han et al., 2016b).

2.3.2 Weight Quantization

Weight quantization for reducing the bit-width to store each weight is another approach to reduce memory requirements of models. It can successfully condense CaffeNet and SqueezeNet to 8-bits with only slight accuracy loss (Gysel et al., 2016). One refinement adapts the bit-depth for each layer quantizing the sparse weight matrix to be an index which encodes in 8-bit for convolutional layers and 5-bit for fully connected layers with only 1% loss in accuracy (Han et al., 2016b). In a more extreme case it is possible to use 1-bit to represent the weights with the added advantage that computation can then be done with binary operations, but the accuracy loss can be over 12% (Rastegari et al., 2016).

2.3.3 Weight Regularization

As part of the training process, regularization in the objective function will penalise the weights, which has often been employed not only to avoid over-fitting for accuracy improvement, but also possible to induce sparsity for model parametrization. Given a supervised learning problem with targets $y_1, y_2, \ldots, y_n$ and corresponding input data $x_1, x_2, \ldots, x_n$, the regularized objective function to estimate the weights $W$ in the model, $f$, is defined by

$$
\min_W \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, f(x_i; W)) + \gamma R(W),
$$

where $\mathcal{L}$ is a loss function to measure the distance between the target and estimate which is calculated by the function $f$. $R(W)$ is a regularization term and $\gamma$ is a hyper-parameter controlling the influence of the regularization term. Weights are updated by optimising the combined loss function and regularizer. A common family of regularizers has the form, $R(W) = ||W||_p$ where $p$ represents as $\ell_p$-norm ranging from 0 to $\infty$.

Weight decay as a penalty was one of techniques for regularization that aims to improve generalization (Krogh and Hertz, 1992), by encouraging a reduction in the magnitude of the weight vector of the neural network (Collins and Kohli, 2014a). However, the regularization term in weight decay using an $\ell^2$-norm ($p = 2$) does not produce sparsity and hence there is no compression. In contrast, the only cases where the regularizer will
provide a sparse solution is when \( p = 0 \) or \( p = 1 \). To directly construct sparse models, a simple regularization operator using the \( \ell_0 \)-norm has been proposed (Collins and Kohli, 2014b), whilst \( \|W\|_0 \) can induce sparsity it is no longer possible to compute the gradient and hence the optimization algorithm on deep learning models becomes an NP hard problem (Polyak, 1963). In contrast \( \ell_1 \)-regularization can not only induce sparsity but retains the attractive property of being able to compute the gradient of the objective function providing an efficient implementation. For example, Least Absolute Shrinkage and Selection Operator (Lasso) estimators as \( \ell_1 \) regularization can penalize weights to zero in the training process (Tibshirani, 1994). Group lasso (Yuan and Lin, 2006) is an efficient regularization-based method for training sparse learning structures. This concept was extended to include a group penalty for network connections with lasso which creates a combined \( \ell_{2,1} \) regularizer to remove neurons (Scardapane et al., 2017), which was shown to reduce the number of weights; a compression rate of 3\times was achieved for a small increase in error. Another approach proposed group-wise brain damage to prune the convolutional kernel tensor in a group-wise fashion (Lebedev and Lempitsky, 2015), and was shown to speed-up the computation of convolutional layer training. The dropout approach which refers to dropping out neurons from visible and hidden layers in the neural network during training can also be shown to be a form of regularization (Hinton et al., 2012). Sparse Variational Dropout (Sparse VD) extends the concept of dropout by providing a different dropout rate for each weight of the model (Molchanov et al., 2017) and is shown to have competitive compression performance.

I now introduce three sparse regularization approaches that employ \( \ell_1 \)-regularization, including Lasso and two variants: Group Lasso (Yuan and Lin, 2006; Schmidt et al., 2007) and Sparse Group Lasso (Scardapane et al., 2017).

**Lasso** is a typical \( \ell_1 \)-regularization which is given by \( R_{\ell_1}(W) = \|W\|_1 \). However, care must be taken with the implementation of the optimisation algorithm in order to handle the function being non-differentiable at zero.

**Group Lasso (GL)** differs from Lasso by imposing sparsity on a neuron-level such that weights connected to the same neuron are treated as a group and regularization will force all weights in a group to become zero (Scardapane et al., 2017). The work defined three different groups of variables including input groups, hidden groups and bias groups, corresponding to different effects of the group-level sparsity. More specifically, input groups correspond to the vector of all output connections from one of the neurons in hidden layers of the network, and bias groups as one-dimensional groups corresponding to the biases of the network. Consequently, if variables of an input group are set to zero, the corresponding feature of input samples can be neglected. If variables of a hidden groups are set to zero, the weights can be removed. Finally, if a variable in a bias group is set to zero, the corresponding bias would be removed from the neuron.
Sparse Group Lasso (SGL) is an improved group lasso by combining $\ell_1$ regularization with $\ell_2$ regularization, which can remove not only grouped weights connected by the same neuron but also other unimportant weights. Figure 2.6 illustrates the differences between Lasso, Group Lasso, and Sparse Group Lasso in a neural network. In light blue dashed lines or circles, we represent the removed connections or neurons (Scardapane et al., 2017).

![Figure 2.6: Comparison of Lasso family in a neural network.](image)

Consequently, regularization methods can obtain a compact model with smaller size than the original model (Tsuruoka et al., 2009). Compared with pruning methods, regularization approaches have the advantage that no retraining is required as shown in Figure 2.5. Compared with matrix factorization, regularization approaches have the advantage that the regularizer updates the weights based on a full rank weight matrix resulting in more accurate values of weights (Scardapane et al., 2017).

### 2.4 Optimization with Sparse Regularization

$\ell_1$-regularization has been shown to achieve good compression (Tsuruoka et al., 2009), but the choice of SGD optimization has slow asymptotic convergence due to the inherent variance (Johnson and Zhang, 2013).

It has two process during weight update, including optimization and regularization. To improve the performance of regularization during, we need consider how regularization works well with optimization. A typical optimization is first-order method: Stochastic Gradient Descent (SGD). In this section, I only focus on how SGD working with $\ell$-1
regularization, but the details of SGD and other advanced optimization algorithms will be introduced in next chapters.

### 2.4.1 SGD with \(\ell_1\) regularization

Although SGD is a simple optimization method which is straightforward to implement, it cannot be efficiently applied to \(\ell_1\)-regularization due to two problems. The first problem is that \(\ell_1\)-regularization is difficult to decay weights going to zero in practice. For example, based on Equation 2.2, weights are updated by \(\ell_1\) regularization as

\[
\begin{align*}
  w_j^{k+1} &= w_j^k - \eta_k \frac{\partial}{\partial w_j} \left( \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i; W)) + \frac{\gamma}{M} \sum_{j=1}^{M} |w_j| \right) \\
  \text{(2.3)}
\end{align*}
\]

where the weights of the network are represented by \(w_j\) and the total number of weights is \(M\). The \(k\) is iteration counter and \(\eta_k\) is learning rate. Due to the second term, \(\frac{\partial}{\partial w_j} \sum_{j=1}^{M} (\gamma |w_j|)\), possibly being non-differentiable when \(w_j = 0\), the weight can be updated by a sign function, which is showed in Equation 2.4.

\[
\begin{align*}
  \text{if } w_i > 0 & \quad \Rightarrow \quad w_j^{k+1} = w_j^k - \eta_k \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L(y_i, f(x_i; W))}{\partial w_j} - \eta_k \gamma M, \\
  \text{else if } w_i < 0 & \quad \Rightarrow \quad w_j^{k+1} = w_j^k - \eta_k \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L(y_i, f(x_i; W))}{\partial w_j} + \eta_k \gamma M. \\
  \text{(2.4)}
\end{align*}
\]

I call this method as SGD\(_\ell_1\). However, due to the high variance generated by SGD, SGD\(_\ell_1\) is hard to settle weights at zero. In the other word, the updated weights can easily being moved away from zero by these fluctuations. To address this problem, Carpenter (2008) proposed a clipping regularization method that is called as Clipping SGD\(_\ell_1\), which is showed in Equation 2.5.

\[
\begin{align*}
  \text{if } w_j^{k+\frac{1}{2}} > 0 & \quad \Rightarrow \quad w_j^{k+1} = \max(0, w_j^{k+\frac{1}{2}} - \eta_k \gamma M), \\
  \text{else if } w_j^{k+\frac{1}{2}} < 0 & \quad \Rightarrow \quad w_j^{k+1} = \min(0, w_j^{k+\frac{1}{2}} + \eta_k \gamma M). \\
  \text{(2.5)}
\end{align*}
\]

SGD\(_\text{clipping}\ell_1\) update weights by optimization and regularization separately, resulting in reducing the chance of weight moving away from zero by optimization. However, this
method also cannot control the penalty of $\ell_1$ regularization, which incurs low performance of model by over-penalty, or remain large number of non-zero weights by insufficient penalty \cite{Tsuruoka et al. 2009}.

### 2.4.2 SGD with cumulative $\ell_1$ regularization

As in practice the regularisation parameter $\gamma$ must be chosen carefully to control the generalisation ability of the model. Furthermore, care must be taken in the optimization approach to handle the non-differentiable element of the $\ell_1$ regularizer. *Cumulative $\ell_1$ regularization* has been proposed to address this by cumulating the $\ell_1$ penalties to resolve these problems \cite{Tsuruoka et al. 2009}. The method clips regularization updates at zero and provides a more stable convergence for the weights. Moreover, the cumulative penalty can reduce the weight to zero more quickly. Cumulative $\ell_1$ regularization can be integrated with SVRG, by using the cumulative update to calculate the gradient associated with the regularization term,

\[
\text{if } w^{s+1}_{t+\frac{1}{2}} > 0 \text{ then } \\
\quad w^{s+1}_{t+1} = \max \left( 0, w^{s+1}_{t+\frac{1}{2}} - (u + q^{s+1}_t) \right),
\]

\[
\text{else if } w^{s+1}_{t+\frac{1}{2}} < 0 \text{ then } \\
\quad w^{s+1}_{t+1} = \min \left( 0, w^{s+1}_{t+\frac{1}{2}} + (u - q^{s+1}_t) \right), \tag{2.6}
\]

where, $u$ is a constant value to control the compression rate of model given by the average value of the total $\ell_1$ penalty multiplied by $\gamma$, $w^{s+1}_{t+\frac{1}{2}}$ represents the weight after Equation 3.2 update, and $q^{s+1}_t$ is the difference between weights over one epoch and is given by

\[
q^{s+1}_t = \sum_{t=0}^{m-1} \left( w^{s+1}_{t+1} - w^{s+1}_{t+\frac{1}{2}} \right). \tag{2.7}
\]

However, SGD-cumulative-$\ell_1$ is still sensitive to the variance of gradients by SGD, which can have a detrimental effect on its performance. To address this problem, a new method that allow an appropriate optimization can work well with $\ell_1$-regularization will be analysed in next chapter.
Chapter 3

Sparse Deep Neural Networks for Embedded Intelligence

Deep learning (DL) is becoming more widespread due to its power in solving complex classification problems. However, DL models often require large memory and energy consumption, which may prevent them from being deployed effectively on embedded platforms, limiting their application. This work addresses the problem of memory requirements by proposing a regularization approach to compress the memory footprints of the models. It is shown that the sparsity-inducing regularization problem can be solved effectively using an enhanced stochastic variance reduced gradient optimization approach. Experimental evaluation of my approach shows that it can reduce the memory requirements both in the convolutional and fully-connected layers by up to 300× without affecting overall predict accuracy.

3.1 Introduction

Artificial intelligence is finding wider application across a number of domains where computational resources can vary from large data centres to mobile devices. However, state-of-the-art techniques such as DL (LeCun et al., 2015) require significant resources, including large memory requirements and energy consumption. Reducing the size of the DL model to a compact model that has a small memory footprint without compromising its performance is a desirable research aim to address the challenges for deploying these leading approaches on mobile and embedded devices.

Given training labels \( y_1, y_2, \ldots, y_n \) and corresponding input data \( x_1, x_2, \ldots, x_n \), the regularized objective function to estimate the weights, \( w \), in the model, \( f \), is defined by

\[
\min_w \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, f(x_i; w)) + \gamma R(w),
\]  

(3.1)
where $\mathcal{L}$ is a loss function, $R(w)$ is the regularizer and $\gamma$ is a hyper-parameter controlling the balance between these two objectives. This optimization problem can be strongly convex or non-convex (Allen-Zhu and Yuan 2016) and is largely determined by the form of the model. In the case of deep neural networks the problem will be non-convex. The choice of regularizer and regularization parameter is important in order to control the generalisation ability of the model. One additional benefit of a regularizer is that it can be used to enforce certain properties on the model parameters. For example, the choice of an $\ell_1$-regularizer as used in a number of learning approaches including Lasso, group Lasso, e.g. Group Lasso (GL) (Wen et al. 2016), and logistic Lasso (Tibshirani 1996) enables model compression by encouraging weights to be zero.

To find the optimal solution of Equation 3.1, Stochastic Variance Reduced Gradient (SVRG) is preferable to stochastic gradient descent (SGD) as SGD optimization has slow convergence asymptotically due to noise (Johnson and Zhang 2013). However, directly applying this to a deep neural network objective with a regularizer with non-continuous derivatives such as $\ell_1$-regularization presents challenges for SVRG. During optimization the discontinuity can effect the robustness of the SVRG optimization algorithm, which slows down the convergence. To address this problem, I propose my compression method Sparse SVRG (SSVRG) by introducing an adaptive learning rate, as an approach to deal with the discontinuity in the derivative and a hyper-parameter to control the balance of the variance reduction. I show that this can improve SVRG optimization for non-convex functions such as those resulting from sparse $\ell_1$-regularization.

### 3.2 Non-convex SVRG for cumulative $\ell_1$ regularization

The focus of this work is around an efficient sparse regularized approach for weight reduction. This could be extended by exploiting the opportunity for weight quantization but this is not explored here. I introduced cumulative $\ell_1$-regularization in Chapter 2. Before introducing my method, I introduce the background to SVRG optimization.

SVRG is a gradient-based optimization approach that updates the weights using a combination of the full batch gradient with a more frequent stochastic update using a subset of the training set as:

$$w_{t+1}^{s+1} = w_t^{s+1} - \eta_t \left( \frac{1}{n} \sum_{i=1}^{n} \nabla f_i (w_t^{s+1}) - \nabla f_i (\tilde{w}_s) + \tilde{g}_s^{t+1} \right), \tag{3.2}$$

where $0 \leq t \leq m - 1$ and $0 \leq s \leq S - 1$ control the number of iterations and the frequency of the stochastic updates, $i_t$ is randomly selected from $\{1, ..., n\}$, $\tilde{w}$ is the weight after every period iteration and $\tilde{g}_s^{t+1} = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i (\tilde{w}_s)$ is the average value of full gradients. Gradients in Equation 3.2 are generated by two terms, $\frac{1}{n} \sum_{i=1}^{n} (\nabla f_i (w_t^{s+1}) - \nabla f_i (\tilde{w}))$ and $\tilde{g}_s^{t+1}$. They ensure that a reduction of variance helps initial weights close to a
global minima boost the convergence rate of SGD in strongly convex problems. However, for non-strongly convex problems, global minimization of a non-convex function is NP-hard \cite{Allen Zhu and Hazan 2016}. It has been argued that SVRG can also be applied in non-convex neural network objectives to accelerate the local convergence rate of SGD \cite{Johnson and Zhang 2013}. An improved version of SVRG was proposed that was shown to have non-asymptotic rates of convergence for non-convex optimization and provably faster than SGD \cite{Allen Zhu and Hazan 2016}. The details are described in Alg. 3.1.

Algorithm 3.1: SVRG \((w_0, \eta, \{p_i\}_{i=0}^m, m, S)\)

**Input**: Update epoch length \(m\), learning rate \(\eta\), number of epochs \(S\), discrete probability distribution \(\{p_i\}_{i=0}^m\) and initialize \(\tilde{w}^0 = w_0^m = w^0\).

```
1 for s=0 to S-1 do
2 \quad w_{s+1}^0 = w_s^m;
3 \quad g_{s+1} = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\tilde{w}^s);
4 \quad for t = 0 to m - 1 do
5 \quad \quad randomly select \(i_t\) from \(\{1,...,n\}\);
6 \quad \quad v_{t+s+1} = \nabla f_{i_t}(w_{t+s+1}) - \nabla f_{i_t}(\tilde{w}^s) + \tilde{g}_{s+1};
7 \quad \quad w_{t+1} = w_{t+s+1} - \eta v_{t+s+1};
8 \quad \quad \tilde{w}^s_{t+1} = \sum_{i=0}^m p_i w_{t+s+1}^i;
9 \quad end for
10 end for
11 Output: \(w_a\) chosen uniformly randomly from \(\\{\{w_{t+s+1}^i\}_{i=0}^{m-1}\}_{s=0}^{S-1}\)
```

There are two problems when directly combining SVRG with cumulative \(\ell_1\)-regularization that is introduced in last chapter for non-convex problems: (1) adding the \(\ell_1\)-regularizer to SVRG limits the theoretical and practical performance of the optimization \cite{Allen Zhu and Hazan 2016}, (2) the trade-off in variance reduction versus the sparsity of the solution; when many weights become zero the robustness and convergence rate can be reduced. To address these two problems my compression method SSVRG introduces a hyper-parameter \(\lambda\) in Equation 3.2 alongside an adaptive learning rate \(\eta\). The weight update then becomes

\[
\begin{align*}
    w_{t+1}^{s+1} &= w_{t+1}^{s+1} - \eta \left( \frac{1 - \lambda}{n} \sum_{i=1}^n \nabla f_i(w_{t+1}^{s+1}) - \nabla f_i(\tilde{w}) \right) + \lambda g_{s+1} \\
    &\quad \text{if } w_{t+1/2}^{s+1} > 0 \\
    &= \max \left( 0, w_{t+1/2}^{s+1} - \left( u + q_{t+1}^{s+1} + \tilde{b} \right) \right), \quad (3.3) \\
    &\quad \text{else if } w_{t+1/2}^{s+1} < 0 \\
    &= \min \left( 0, w_{t+1/2}^{s+1} + u - q_{t+1}^{s+1} - \tilde{b} \right), \quad \text{where } 0 \leq \lambda \leq 1 \text{ is a hyper-parameter that balances the gradients and } \tilde{b} \text{ is the smallest bias term amongst all layers. The full procedure is illustrated in Alg. 3.2.}
\end{align*}
\]
sections describe the main contributions associated with the introduction of the adaptive learning rate and the hyper-parameter controlling the variance reduction, as well as supplementary details on the cumulative learning schedule and bias pruning.

**Algorithm 3.2: SSVRG($w^0, \{p_i\}_{i=0}^m \cdot m, S$)**

**Input**: Update epoch length $m$, learning rate $\eta$, number of epochs $S$, discrete probability distribution $\{p_i\}_{i=0}^m$ and $u = 0$;

1. Initialize $\tilde{w}^0 = w^0$; $q_i = 0$ for all $i$;

2. for $s=0$ to $S-1$ do

3. $w_{s+1}^0 = w_s^0$;

4. $\tilde{g}^{s+1} = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\tilde{w}^s)$;

5. for $t = 0$ to $m - 1$ do

6. randomly select $i_t$ from $\{1, \ldots, n\}$;

7. $v_t^{s+1} = (1 - \lambda) \left( \nabla f_i(u_{t+1}^{s+1}) - \nabla f_i(\tilde{w}^s) \right) + \lambda \tilde{g}^{s+1};$

8. $u_{t+\frac{1}{2}}^{s+1} = u_{t+1}^{s+1} - \eta v_t^{s+1};$

9. $z = u_{t+\frac{1}{2}}^{s+1};$

10. $u \leftarrow u + \omega_t \gamma / n$;

11. $b$ is minimal bias in all layers;

12. if $u_{t+\frac{1}{2}}^{s+1} > 0$ then

13. $\tilde{w}_{t+1}^{s+1} = \max \left( 0, u_{t+1}^{s+1} - (u + q_t^{s+1}) - \tilde{b} \right)$

14. ;

15. else if $u_{t+\frac{1}{2}}^{s+1} < 0$ then

16. $\tilde{w}_{t+1}^{s+1} = \min \left( 0, u_{t+1}^{s+1} + (u - q_t^{s+1}) + \tilde{b} \right)$

17. ;

18. $q_{t+1}^{s+1} \leftarrow q_t^{s+1} + (u_{t+1}^{s+1} - z)$

19. $\tilde{w}^{s+1} = \sum_{i=0}^m p_i w_i^{s+1};$

**Output**: $w_a$ chosen uniformly randomly from $\{w_{t+1}^{s+1}_{m-1} \}_{t=0}^S$ $s=0$.

### 3.2.1 Hybrid Adaptive Learning Rate and Variance Control

Selecting the learning rate is important because it can dramatically effect the performance and convergence rate of optimization. In SVRG a single fixed learning rate is specified. If $\ell_1$-regularization is combined with non-convex SVRG the single fixed learning steps cannot control the reduced variance in SVRG to achieve fast convergence. Another criticism against variance reduced schemes is that the ability to escape local minimum and saddle points in non-convex optimization problems is reduced. Research has shown that adaptive learning rates can be applied with reduced variance to provide faster convergence rates on non-convex optimization (Goodfellow et al., 2016; Reddi et al., 2016a). Furthermore, a single fixed learning rate may not be able to balance the gradient terms associated with the loss and regularizer.
In the standard SVRG algorithm (Alg. 3.1) the balance of the gradient update between the full batch and stochastic estimate is fixed. In my approach, I introduce a hyper-parameter to allow the algorithm to span a range of behaviours from stochastic, through reduced variance, to batch gradient descent. Consequently with careful adoption of this hyper-parameter the algorithm is able to gain benefits from the stochastic element to speed up computation and escape local minima/saddle points, and reduced variance to improve convergence rate.

Following the work from (Reddi et al., 2016a) which developed an approach to determine an adaptive learning rate $\eta$, I extend it to show how the hyper-parameter $\lambda$ can be estimated and adapted to control the variance. I show that the hybrid adaptive learning rate in each epoch defines a range of $\lambda$ values. The argument shows that an adaptive learning rate could be derived as a maximum of two terms. I adopt the same approach for Alg. 3.2 and let $\bar{t} = ts$ from 1 to $T$ where $T = mS$ is the total number of iterations and $m = \lfloor n(1 - \mu_1) \rfloor / 3\mu_0\mu_1$, $p_m = 1$ and $p_i = 0$ for $0 \leq i < m$, giving the learning rate as

$$\eta_t = \max\{\eta_0 / \sqrt{\bar{t}}, \mu_1 / Ln^{2/3}(1 - \mu_0)^2\},$$  \hspace{0.5cm} (3.4)$$

where $(0 < \mu_0, \mu_1 < 1)$, $L$ is constant value. The first term is dependent on $\bar{t}$ and hence will decrease with increasing iterations, whereas the second term is independent of $\bar{t}$ but dependent upon the data size. $\lambda$ can be used to adjust the weighting of the stochastic element with the full batch gradient to control the variance of the optimization approach and this can be done at each iteration. The first term of the learning rate expression is larger than second term when $\bar{t}$ is small. In the first case, the variance will be reduced to accelerate the convergence rate in early iterations. On the other hand, when the second term becomes larger later in optimization, higher variance will help gradients escape from local minimum. Consequently, my hypothesis is that when the first term in Equation 3.4 is maximum a reduced variance approach $(0.5 < \lambda \leq 1)$ will be better, and in the second case a more stochastic gradient approach $(0 \leq \lambda < 0.5)$ will provide better optimization.

### 3.2.2 Learning Rate for Cumulative $\ell_1$ regularization.

I set the learning rate $\omega_t$ for cumulative $\ell_1$-regularization following (Tsuruoka et al., 2009) as,

$$\omega_t = \omega_0\alpha^{t/n},$$  \hspace{0.5cm} (3.5)$$

where the $\alpha$ is a decay constant. The scheduling is practical because exponential decay ensures that the learning rate does not drop too fast at the beginning and too slowly at the end.
3.2.3 Bias-based Pruning

To further reduce the number of weights, I add a bias-based pruning \( \hat{b} \) after the \( \ell_1 \)-regularization in each iteration. The pruning rule is based on the following heuristic (Fonseca and Fleming, 1995): connections (weights) in each layer will be removed if their value is smaller than the network’s minimal bias. If the absolute value of weight connections are smaller than the absolute value of the smallest bias of the entire network in each batch, these connections have least contribution to the node, which can be removed. In practice, bias-based pruning has no effect on train and test loss.

3.3 Experiments

In order to estimate and compare the effect of my compression method on different topologies, I select deep neural networks (DNNs) and convolutional neural networks (CNNs). The DNN chosen is LeNet-300-100 which has two fully connected layers as hidden layers with 300 and 100 neurons respectively. The CNN chosen is LeNet-5 which has two convolutional layers and two fully connected layers. I evaluate the performance of my new compression method using MNIST, and CIFAR as benchmarks. my compression method was implemented using Caffe[1].

3.3.1 Experimental Settings

1. Learning rate \( \eta \) and \( \omega \): my experiments determined the two learning rates \( \eta_t \) and \( \omega_t \) according to Equation 3.4 and Equation 3.5 respectively. The coefficients \( \eta_0, \mu_0, \mu_1 \) and \( L \) are empirically chosen so that SSVRG gives the best performance on the training loss. The value of \( \eta_0/\sqrt{t} \) will be reduced by the increasing iteration number, \( t \), but the value of \( \mu_1/Ln^{2/3}(1 - \mu_0)^2 \) is fixed for a given data-set. A value of \( \mu_0 = \frac{1}{5} \) was determined by experiment and \( L = 4 \) was taken from (Johnson and Zhang, 2013; Li et al., 2016). For the learning rate \( \omega, \alpha = 0.8 \) was chosen by experiment to give good performance.

2. Initialization: Weight initialization is important when training neural networks. Following the approach by (Glorot and Bengio, 2010), I use normalized initialization to scale the value of initial weights in each layer by uniform \( r = [-\sqrt{6/n_{in} + n_{out}}, \sqrt{6/n_{in} + n_{out}}] \), where \( n_{in} \) is the number of neurons in the input layer and \( n_{out} \) is the number of neurons in the output layer.

3. Loss function: In my experiments the quality is evaluated using cross entropy by the softmax log-loss function as the result of training/test error. I also use mean squared error (MSE) to estimate the effectiveness of the training/test loss.

[1]Caffe is a DL framework. Source code can be download: http://caffe.berkeleyvision.org
Figure 3.1: Effects of $\lambda$ on the performance and convergence rate when $\eta$ is decayed.

### 3.3.2 Evaluating the Range of $\lambda$

I experimentally estimate the appropriate range of the hyper-parameter $\lambda$ via calculating test error and training loss of LeNet-5 model on MNIST, CIFAR-10 and CIFAR-100 datasets shown in Figure 3.1 and Figure 3.2. Based on the two cases of the learning rate $\eta$, I found that when the value of learning rate $\eta$ depends on number of iterations, it can achieve good performance and fast convergence on LeNet-5 applied for three datasets when $0.5 < \lambda \leq 1$ as shown in Figure 3.1. Particularly, the optimal value of $\lambda$ in the first case is close to 0.8 because if $\lambda$ is larger than 0.8 (e.g. $\lambda = 0.95$) the performance deteriorates. Otherwise in the second case, when the value of learning rate is fixed which
depends on the number of training samples \( \eta = \mu_1/Ln^{2/3}(1 - \mu_0)^2 \), good performance and fast convergence can be obtained if \( \lambda \) is close to zero as shown in Figure 3.2.

### 3.3.3 Evaluation of the Trade-off Between Memory Requirements and Performance

The compression rates of the SSVRG approach are controlled by \( \gamma \), providing a range of performance-compression operating points. Figure 3.3 shows how the test error and weight sparsity vary as the regularization parameter \( \gamma \) is adjusted. In Figure 3.3, it has three experiments correspondingly with three datasets. LeNet-300-100 is used for MNIST and LeNet-5 for the two CIFAR datasets. The comparison is shown for
three compression methods, SSVRG(with/without bias pruning) and MSVRG-C-L1 that combines MSVRG with cumulative $\ell_1$-regularization. SSVRG is shown to outperform MSVRG optimization ([Reddi et al., 2016a]) with cumulative $\ell_1$-regularization (MSVRG-C-L1) to provide superior models over the pareto front. Bias pruning is shown to offer some benefit but the benefit is limited and not universal.

To highlight the results, I show the weight statistics for each layer of a good operating point of the model for MNIST in Table 3.1 and 3.2. The results show that SSVRG can significantly remove many weights in the fully connected layers. For LeNet-300-100 models the number of weights in the first fully connected layers (ip1) contains about 88% of the total number of weights and this can be compressed by 97%. Furthermore, SSVRG is effective to reduce the number of weights in both of the convolutional layers (conv1 and conv2) and the two fully connected layers (ip1 and ip2) in the LeNet-5 model. SSVRG reduces the number of weights to 4% with a 1.58% test error rate on the LeNet-300-100 model and 0.6% of the weights with a 0.72% test error on the LeNet-5 model.

## Table 3.1: Comparison of five compression methods on the MNIST data-set.

| Model         | Method          | Error | $\frac{|w|}{|w| \neq 0}$ | Iterations (#) |
|---------------|-----------------|-------|------------------------|----------------|
| LeNet-300-100 | Original        | 1.64% | 1×                      | -              |
|               | Group Lasso (GL)| 1.53% | 3×                      | -              |
|               | Pruning         | 1.58% | 12×                     | ≥ 12,000       |
|               | DNS             | 1.99% | 56×                     | 10,000         |
|               | Sparse VD       | 1.92% | 68×                     | -              |
|               | SSVRG           | 1.92% | 103×                    | 4,000          |
| LeNet-5       | Original        | 0.8%  | 1×                      | -              |
|               | Group Lasso (GL)| 1%    | 5×                      | -              |
|               | Pruning         | 0.77% | 12×                     | ≥ 20,000       |
|               | DNS             | 0.91% | 108×                    | 10,000         |
|               | Sparse VD       | 0.75% | 280×                    | -              |
|               | SSVRG           | 0.79% | 295×                    | 5,000          |

Figure 3.3: The performance of SSVRG evaluation with other two baseline methods.
A further experiment compares the convergence rate of SSVRG with MSVRG without regularization via estimating the training/test loss and the number of iterations when the value of gradients being converged to zero. In this experiment, I choose the optimal value of $\lambda$ for my method which has significantly faster convergence than MSVRG on all
Table 3.2: SSVRG compression results by layer on two DL models for MNIST data-set.

(a) MNIST data-set with LeNet-300-100 model.

| Layer | Original network #Weights | SSVRG #Weights | Compress rate $|w| \neq 0$ $\frac{|w|}{|w|}$ |
|-------|---------------------------|----------------|-----------------|
| ip1   | 235K(940KB)               | 8.0K           | 3%              |
| ip2   | 30K(120KB)                | 2.5K           | 8.3%            |
| ip3   | 1K(4KB)                   | 0.3K           | 30%             |
| Total | 266K(1070KB)              | 10.8K          | 4%(25×)         |
| Classification Error | 1.64% | 1.58% | - |

(b) MNIST data-set with LeNet-5 model.

| Layer | Original network #Weights | SSVRG #Weights | Compress rate $|w| \neq 0$ $\frac{|w|}{|w|}$ |
|-------|---------------------------|----------------|-----------------|
| conv1 | 0.5K                      | 0.33K          | 66%             |
| conv2 | 25K                       | 0.5K           | 2%              |
| ip1   | 400K                      | 1.2K           | 0.3%            |
| ip2   | 5K                        | 0.6K           | 12%             |
| Total | 431K                      | 2.63K          | 0.61%(163×)     |
| Classification Error | 0.80% | 0.72% | - |

three different datasets shown in Figure 3.4.

3.3.4 Comparison with Leading Results

In Table 3.1, I choose three other competitive compression methods with leading results for comparison. Table 3.1 shows that the compression rate of my method on LeNet-5 can be up to 300× with lower test error than the original model. Consequently, after convergence the network parameters of LeNet-300-100 and LeNet-5 are reduced by 103× and 295×, respectively, which results in less than 3% of the network connections being maintained while the prediction accuracy are slightly better than the original models.

3.4 Discussion

In this chapter, I proposed the SSVRG method that can efficiently reduce the number of weights in neural networks without sacrificing accuracy by employing a hybrid learning rate and hyper-parameter to control the variance reduction. The appropriate value of $\lambda$ can balance the different part of the gradients in SVRG and work with a flexible hybrid learning rate to accelerate the convergence and improve the performance of the model on sparse learning representations. Experiments show that the method can significantly reduce the number of weights by up to 300× without accuracy loss enhancing the ability to deploy deep neural networks on embedded devices.
Part II

Improving Optimization
Algorithms for Model Learning
Chapter 4

Background of Model Optimization for Training

In Part I it was shown that a good numerical optimization can significantly boost the performance of DL algorithms, and hence is a central problem in machine learning (ML). Many traditional ML models (e.g. logistic regression, support vector machines) can be trained with a convex objective for classification and prediction. Optimization methods on such convex models can find a global optimal solution. Comparatively, modern ML model (e.g. Deep Learning (DL) models, graphical models and sparse learning) can achieve better performance on more complex tasks. However, these methods are more difficult to train and estimate a global optima since they are non-convex problems. As a result, it motivates us to design, analyse, and extend non-convex optimization to enhance its performance for DL training. By improving the performance of the optimization algorithm it is possible to reduce computation and power requirements for training a DL model, irrespective of the training platform. Hence, the work here has application to enabling training on embedded platforms as well as increased efficiency on more powerful hardware. The focus is on developing fast and efficient mathematical optimization methods to obtain the optimal solution for non-convex ML models, particularly DL models. This chapter begins by introducing the concepts of convex and non-convex optimization, which is followed by a survey of existing optimization techniques.

4.1 Problem Setting

The primary optimization problem of my interest is given by

$$\min_{x \in \mathbb{R}^d} f(x), \quad \text{where} \quad f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x),$$

(4.1)
where each function $f_i : \mathbb{R}^d \to \mathbb{R}$ is smooth (possibly non-convex) for all samples $i \in \{1, ..., n\}$. I use empirical risk minimization (ERM) and finite-sum problem interchangeably to describe this problem setting.

I use smoothness assumptions on objective functions by Lipschitz conditions on the functions throughout this thesis. Such an assumption is common in the analysis of first-order methods. I defined the condition as following,

**Definition 4.1.1.** \((L\text{-smooth functions})\) I assume the a functions $f : \mathbb{R}^d \to \mathbb{R}$ is $L$-smooth if there is a constant $L$ such that

$$\| \nabla f_i(x) - \nabla f_i(y) \| \leq L \| x - y \|, \forall x, y \in \mathbb{R}^d. \quad (4.2)$$

### 4.2 Convex and Non-convex Optimization

Problems are said to be convex if the objective function $f(x)$ and each $f_i(x)$ in Equation 4.1 are convex for all $n$. Convex problems have two types including strongly and non-strongly convex problems. [Reddi et al., 2016a] defined strongly and non-strongly convex problem via a parameter $\varsigma$. A function $f$ is called $\varsigma$-**strongly convex** if there is $\varsigma > 0$ such that

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{\varsigma}{2} ||x - y||^2, \forall x, y \in \mathbb{R}^d. \quad (4.3)$$

On the other hand, I say $f$ is non-strongly convex when $f$ is 0-strongly convex where $\varsigma = 0$. In convex problems, optimization algorithms are guaranteed to find a global minima in finite time [Allen-Zhu, 2018; Tsuruoka et al., 2009]. The primary reasons for targeting convex problems is their ubiquitoustness in applications and their relative ease of solving them [Defazio et al., 2014]. There are many machine learning algorithms which can be trained using a convex objective, such as Support Vector Machines, Logistic Regression, Least-Squares and Tree-weighted Belief Propagation, which have seen real world application [Defazio et al., 2014]. However, with the advancement of deep neural network approaches and their enhanced performance focus has shifted to their corresponding non-convex optimisation. A non-convex problem given by Equation 4.1 has $f$ or individual $f_i(i \in [n])$ being non-convex. Unlike convex problems, there are many local minima in non-convex problem, which results in an NP hard problem to estimate a global minima in finite time [Jain and Kar, 2017; Du et al., 2017]. Considering the NP-hardness for a general set of relevant problems, it is necessary to use additional assumptions to guarantee efficient global minima in deep learning. As a result, recent theoretical studies have proven global minima in deep learning by using linear assumptions, e.g. $L$-smooth conditions on functions [Ge et al., 2016; Du et al., 2017].
Both convex and non-convex problems can be solved by numerical optimization algorithms typically involving the use of gradient information. The gradient approaches can be divided into two types including first-order methods (i.e. based on gradients of function), and second-order methods (i.e. based on the Hessian matrix of function). The first-order methods for optimization are particularly favoured in ML community due to their scalable nature and reduced computation with respect to second-order methods. Ideas developed in convex optimization can be extended to address non-convex problems. For example, Stochastic Variance Reduced Gradient (SVRG) is a first-order optimization approach, proposed by (Johnson and Zhang, 2013) which accelerates the rates of convergence of the stochastic gradient descent (SGD) method in convex problems by reducing variance of the gradients. Further research (Reddi et al., 2016a) provided both theoretical and experimental results of the SVRG method on non-convex problems. Since DL tasks on large-scale datasets are computationally demanding, it is a big challenge of non-convex optimization to estimate the global minima or a good local minima that is close to global minima. Meanwhile it also requires the optimization to have the ability to escape from weak local minima.

4.3 Performance Evaluation

To evaluate the performance of optimization algorithms, there are measures of convergence criteria and black box oracles.

4.3.1 Convergence Criteria

Convergence criteria are an important aspect of optimization algorithms which measure the rate at which algorithms converge to a solution. Commonly employed convergence criteria in the convex domain are $|f(x) - f(x^*)|$ or $|x - x^*|$ where $x^*$ is the global optima. However, such convergence criteria cannot be used in non-convex problems as it is hard to find the global optima. As a result, I follow the concept from (Nesterov, 2014) and use a gradient-based measure, $||\nabla f(x)||^2$, as the convergence criterion for non-convex problems to determine how the iterates converge. I use the following definition for my analysis on non-convex problems.

**Definition 4.3.1.** (Nesterov, 2014) A point $x$ is called $\epsilon$-accurate if $||\nabla f(x)||^2 \leq \epsilon$. A stochastic iterative algorithm can achieve $\epsilon$-accuracy within $t$ iterations if $E[||\nabla f(x^t)||^2] \leq \epsilon$, where the expectation is over the stochastic algorithm.
4.3.2 Black Box Oracles

A black box oracle is a useful method for providing a clear exposition of the problem and its structure (Reddi et al., 2016a), and can be used to compare the performance of different optimization methods in a simple manner. Typically, there are four types of black box oracles, including Incremental First-order Oracle (IFO), Incremental Second-order Oracle (ISO), Proximal Oracle (PO) and Linear Oracle (LO). IFO based complexity analysis was introduced to study lower bounds for finite-sum problems to calculate the computational complexity of an optimization algorithm (Bottou et al., 2018; Johnson and Zhang, 2013). IFO is popular for analysing first-order optimizations since it requires first-order information at each iteration (Reddi et al., 2016a; Agarwal and Bottou, 2015a). In contrast, the ISO oracle uses second-order information of the function. PO and LO oracles are useful for non-smooth, non-convex problem domains. As a result, all algorithms proposed in this thesis use first-order information of the function, and thus simply query the IFO oracle to compare the convergence of competing optimization algorithms where it is defined as

Definition 4.3.2. (Reddi et al., 2016a; Agarwal and Bottou, 2015a) I denote \( F_n \) for all functions of the form 4.1. For \( f \in F_n \), an IFO takes an index \( i \in \{1, 2, 3, ..., n\} \) and a point \( x \in \mathbb{R}^d \), and returns the pair \((f_i(x), \nabla f_i(x))\).

4.4 Optimization Techniques

Popular first-order methods such as Gradient Descent, Stochastic Gradient (SG)-based algorithms, Momentum (Polyak, 1964), Adagrad (Duchi et al., 2011), and randomized Coordinate Descent (Golub and Van Loan, 1996; Ramdas, 2014), can be used in different domains (e.g. convex, non-convex but linear (smooth), and even non-convex non-linear) by exploring the structure of the Problem 4.1. SG-based algorithms have some advantages over other methods, such as reduced memory and computation requirements due to a single or batched data sample being processed and its stochastic nature can prevent the estimated point becoming stuck in a local minima. As a result, this thesis will focus on analysing the SG-based optimization techniques. In the following sections I introduce three typical first-order optimization algorithms that inspire us to propose my new algorithms.

4.4.1 Gradient Descent

A standard (or ’full batch’) Gradient Descent (GD) is used to minimize an objective function by iteratively moving in the direction of steepest descent that is defined by the negative of gradient. In each iteration the magnitude of this step must be controlled to
ensure stability. However, the process is expensive in computation, which is amplified in large-scale datasets. Given the objective function in Equation 4.1, the algorithm updates the iterates according to
\[ x^{t+1} = x^t - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x^t). \]  

### 4.4.2 Stochastic Gradient Descent

A standard SGD is an approach which estimates the gradient from a single random sample of the data in every iteration. Compared with the GD method the advantage of SGD is the reduction in the number of gradient calculations. The update rule of SGD is given by

\[ x^{t+1} = x^t - \eta \nabla f_i(x^t) \]  

where \( i \) is randomly chosen from \( \{1, ..., n\} \). However, stochastically choosing a single sample in each iteration, SGD suffers from a higher variance of the gradient than GD, which results in less robust convergence. Mini-batch gradient descent techniques are a variation of the gradient descent algorithm that splits the training datasets into small batches and randomly chooses small batches of data in each iteration. This technique can be applied in SGD and is referred to as mini-batch SGD, which has higher efficiency than SGD. As a result, I applied the mini-batch technique into my optimization algorithm throughout this thesis.

### 4.4.3 Stochastic Variance Reduced Gradient

The Stochastic Variance Reduced Gradient (SVRG) method was briefly introduced and used in the previous two chapters. We revisit the SVRG method in this chapter in greater detail in order to better understand the advantages and disadvantages of this method for optimization.

SVRG is a variant of Variance Reduced (VR)-based SGD optimization that improves the SGD method by reducing the variance of SGD, which combines the reduced computation with improved convergence to a stationary point. The SVRG update rule has three items including the randomly selected single/batched gradient of the point in the current iteration, and the randomly selected single/batched gradient of the point plus the full gradient from a previous epoch, which is formulated as

\[ x^{t+1} = x^t - \eta \left( \nabla f_{i_t}(x^t) - \nabla f_{i_t}(\tilde{x}) + \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\tilde{x}) \right), \]  

where \( i_t \) is random number from the range \( \{1, 2, 3, ..., n\} \) and \( \tilde{x} \) is the point several SGD iterations previously. However, SVRG has three drawbacks. Firstly, SVRG has no
ability to control the reduced variance. If over reducing variance, the estimator is able to converge quickly to a stationary point but this point may be a poor minima resulting in low accuracy of training model. Alternatively, when the variance is higher the estimator has greater difficulty settling down to a local minima resulting in higher computational complexity to obtain a solution. Secondly, SVRG has high computations to calculate the average value of full gradients in every epoch since the training data samples $n$ is usually very large. Thirdly, to compensate for the variance in the gradient the learning could be reduced to improve convergence to a minima.

4.5 Performance of Non-convex Optimization Techniques

There are three key factors that can impact the performance of convergence in optimization methods.

4.5.1 Unbiased and Biased Estimations

There are two types of estimator for the gradient which are termed the unbiased and biased estimators. In Chapter 3 I used variance controlled SVRG combined with $\ell_1$-regularization for reducing the number of weights in DL models. I provided a hyper-parameter $\lambda$ to control the variance of SVRG optimization and in doing so introduced a biased estimator for the gradient. The bias of an estimator measures the difference between the estimated point and the true target. If an estimator has zero bias, it is called an unbiased estimator. Using an unbiased estimator is preferable since it can guarantee to converge to the true target. However, in practice a biased estimator is more often used than unbiased (Chen and Luss 2018; Tadić and Doucet 2011), for a number of reasons, e.g. using unbiased estimator in some applications is difficult to compute; in some cases, it is not necessary to apply an unbiased estimator because the biased version can find an approximate point which is very close to the real target in a more efficient manner. As a result, I will later introduce a variance controller, $\lambda$, that can balance the trade-off of biased/unbiased estimators.

4.5.2 Batch and Mini-batch Size

The computational complexity of VR-based SGD is sensitive to the number of samples $n$ since they require $O(n)$ IFO orders to calculate the average value of whole gradients in each epoch. As a result, smaller $n$ plays an important rule in dramatically reducing IFO oracles (Lei et al. 2017a; Babanezhad et al. 2015). Batch methods provide a subset of whole training samples to bound the learning problem. The size of the subset is represented by $B$ where $1 \leq B \leq n$. Thus, recent research (Lei et al. 2017a; Fang
et al., 2018; Zhou et al., 2018) has focused on applying batch methods to VR-based SGD optimization so as to reduce the IFO oracle, which achieved leading results of IFO oracles. However, in practice, $B$ has a trade-off between computational complexity and the performance of optimization. Large $B$ maintains the original learning problem but is inefficient in reducing computational complexity. In contrast, small $B$ can provide a small number of IFO orders, but the optimization loses information from all of the original training samples. As a result, determining an optimal batch size to balance this trade-off is an important problem. Compared to SVRG in Alg. 3.1 that randomly selects a single training sample in each inner iteration, a further refinement modifies the algorithm to choose a small group of training samples in each inner iteration which is referred to as ‘mini-batch’ and can save computational cost. The mini-batch size is represented by $b$ where $1 \leq b \leq B \leq n$, and will have an effect on the variance of gradients. A small mini-batch size boosts the stochastic nature of the gradients, for instance a standard SGD having mini-batch size $b = 1$ has high variance. Conversely, a large mini-batch size can reduce the variance of gradients and may improve the effectiveness of the optimization. An extreme case is GD optimization that has a variance of zero, where the mini-batch size $b = n$. Consequently choosing an appropriate mini-batch size to control the variance of gradients in optimization is another key problem.

4.5.3 Learning Rates

Learning rates ($\eta$) also play an important role in optimization. It controls the searching area being considered in next iterations, which impacts on the performance of optimization. Two common approaches to the learning rate including a constant or a decayed learning rate. However, these two versions have a trade-off between fast convergence and escaping from bad local minima. A decayed learning rate can lead to fast convergence but can limit the ability to escape from bad local minima; conversely a constant learning rate can address this problem, but may be more difficult to converge to a stationary point. As a result, determining a suitable learning rate schedule for optimization is a further problem.
Chapter 5

A Stochastic Gradient Method with Hybrid Estimation

A number of optimization approaches have been proposed for optimizing non-convex objectives (e.g. DL models), such as batch gradient descent, stochastic gradient descent and stochastic variance reduced gradient descent. Theory shows these optimization methods can converge by using an unbiased gradient estimator. However, in practice biased estimation of gradient can allow more efficient convergence to the vicinity since an unbiased approach is computationally more expensive. To produce fast convergence there are two trade-offs of these optimization strategies which are between stochastic/batch, and between biased/unbiased. This chapter proposes an integrated approach which can control the nature of the stochastic element in the optimizer and can balance the trade-off of estimator between the biased and unbiased by using a hyper-parameter. It is shown theoretically and experimentally that this hyper-parameter can be configured to provide an effective balance to improve the convergence rate.

5.1 Introduction

In this chapter, I pay an attention to stochastic algorithms for non-convex finite-sum problems of the form that has been showed in last chapter Equation 5.1. Here I recall this objective function:

$$\min_{x \in \mathbb{R}^d} f(x), \quad f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x),$$  \hspace{1cm} (5.1)$$

where individual $f_i(i \in [n])$ and $f$ are non-convex but Lipschitz smooth ($\mathcal{L}$-smooth) (Stron-gin and Sergeyev 2000, Reddi et al. 2016a). I use $\mathcal{F}_n$ to denote all functions of Equation 5.1 and analyse my optimization method using the IFO framework (Reddi et al. 2016a).
Based on complexity analysis, IFO evaluates lower bounds for finite-sum problems \cite{Bottou2018}. The underlying training algorithms for non-convex problems are still stochastic gradient descent (SGD) and its heuristic variants to solve Equation 5.1 \cite{Allen-Zhu2018}. One of the variants is Variance Reduced (VR) based stochastic optimization approaches (e.g. stochastic variance reduced gradient (SVRG)) which has been shown to accelerate the convergence rate of SGD by reducing the noise of gradients on non-convex problems \cite{Johnson2013}.

VR-based stochastic algorithms have three problems. Firstly, VR schemes reduce the ability to escape local minima in later iterations due to a diminishing variance. The challenge in VR-based stochastic optimization is therefore to control the reduction in variance. Secondly, SVRG is an unbiased estimation, which can increase computation over biased estimation \cite{Liang2009}. Thirdly, the learning rate in such an algorithm is fixed and relatively large, which has the advantage of encouraging initial points out of local minima in early iterations but can hinder optimization convergence to a local optima.

To address these three problems, I propose my method Integrated SVRG (ISVRG\textsuperscript{+}) which can control the variance reduction and choose the biased or unbiased estimator in each iteration to accelerate the convergence rate of non-convex optimization. I summarize and list my main contributions:

1. I introduce ISVRG\textsuperscript{+}, a well-balanced VR method for SGD. I provide a theoretical analysis of my algorithm on non-convex problems.

2. ISVRG\textsuperscript{+} balances the trade-off between biased and unbiased estimation, which can provide a fast convergence rate.

3. Compared with SGD and SVRG-based optimization, my method can achieve comparable or faster rates of convergence. To the best of my knowledge, I provide the first analysis about controlling the variance reduction to balance the gradient of SVRG and balance the nature of the estimator between biased and unbiased to obtain provably superior performance to SGD and its variants on non-convex problems. Table 5.1 compares the theoretical rates of convergence of four methods, which shows that ISVRG\textsuperscript{+} has the fastest rate of convergence. Generally, the learning rate schedule in SGD is decayed by increasing number of iteration. In SVRG (Alg. 3.1), the learning rate is fixed. MSVRG \cite{Reddi2016} is a modified SVRG method where the learning rate in each iteration is chosen from the maximum of two terms, including a term which decays with increasing iteration and a term which is related to the number of training samples. ISVRG\textsuperscript{+} adopts an adaptive learning rate similar to MSVRG.
Table 5.1: Comparison of the IFO complexity of different algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>IFO calls on non-convex</th>
<th>The schedule of Learning rate $\eta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>$O(1/\epsilon^2)$</td>
<td>Decayed</td>
</tr>
<tr>
<td>SVRG</td>
<td>$O(n + (n^{2/3}/\epsilon))$</td>
<td>Fixed</td>
</tr>
<tr>
<td>MSVRG</td>
<td>$O(\min(1/\epsilon^2, n^{2/3}/\epsilon))$</td>
<td>$\max{\text{Decayed, Fixed}}$</td>
</tr>
<tr>
<td>ISVRG$^+$</td>
<td>$O(\min(1/\epsilon^2, n^{1/4}/\epsilon))$</td>
<td>$\max{\text{Decayed, Fixed}}$</td>
</tr>
</tbody>
</table>

4. I show empirically that ISVRG$^+$ has faster rates of convergence than SGD, SVRG and MSVRG (Reddi et al., 2016a) which is a modified SVRG using an adaptive learning schedule with standard SVRG.

5.2 Preliminaries

For my analysis, I follow the definitions from other researchers including $\mathcal{L}$-smooth (Lipschitz continuous gradients), $\epsilon$-accuracy (Ghadimi and Lan, 2013) and computational cost IFO (Agarwal and Bottou, 2015b) which were defined in Definition 4.1.1, Definition 4.3.1 and Definition 4.3.2 respectively. Following Reddi et al. (2016a) $\sigma$-bounded gradients are defined.

**Definition 5.2.1.** (Reddi et al., 2016a) $f \in \mathcal{F}_n$ has $\sigma$-bounded gradient if $\| \nabla f_i(x) \| \leq \sigma$ for all $i \in [n]$ and $x \in \mathbb{R}$.

It is necessary to use two theorems from Reddi et al. (2016a) that provide upper bounds of expectation of SVRG corresponding to two cases of the learning rate, but in order to provide a consistent comparison with my algorithm I introduce two changes: a) re-scaling the gradient of SVRG $v_i^{t+1}$ by multiplying 0.5 in Alg 3.1 in order to allow Alg 3.1 to become the same with my new SVRG algorithm when $\lambda = 0.5$; b) modifying the learning rate to include $\eta$ and $\beta$ in their theorem to provide a simplified parametrization for the learning schedule. In the first case of the learning rate depending on iteration number $T$, Reddi et al. (2016a) use $\sigma$-bounded gradients to achieve the upper bound. After my modifications, the modified upper bound is shown in Theorem 5.2.1.

**Theorem 5.2.1.** Suppose $f$ has $\sigma$-bounded gradients. Let $\eta_{SVRG} = \frac{C_{SVRG}}{\sqrt{T}}$, where $C_{SVRG} = \frac{\sqrt{f(x^0) - f(x^*)}}{2L\sigma^2}$, and $x^*$ is an optimal solution to Equation 5.1. Alg. 3.1 satisfies

$$\min_{0 \leq t \leq T-1} \mathbb{E}[\| \nabla f(x^t) \|^2] \leq \sqrt{2} \sqrt{\frac{2(f(x^0) - f(x^*))L}{T} \sigma.}$$
In the second case of learning rate depending on the training sample size $n$, after my modification, a upper bound without $\sigma$-bounded proposed by (Reddi et al., 2016a) is shown in following theorem.

**Theorem 5.2.2.** Let $f \in F_n$, let $c_{m_{SVRG}} = 0$, $\eta > 0$, $\beta_t = \beta > 0$ and $c_{SVRG} = c_{t+1}(1 + \eta \beta + 2\eta^2 L^2) + L^3 \eta^2$, so the intermediate result $\Omega_{SVRG} = (\eta - \frac{c_{t+1}\eta}{\beta_t} - L\eta^2 - 2c_{t+1}\eta^2) > 0$, for $t$ from 0 to $m - 1$. Define the minimum value of $\gamma_{n_{SVRG}} = \min_t \Omega_{SVRG}$. Further let $p_i = 0$ where $0 \leq i < m$, $p_m = 1$, and $T$ is a multiple of $m$. Defining the output of Alg. 3.1 as $x_a$ I have the following upper bound:

$$E[\|\nabla f(x_a)\|^2] \leq \frac{f(x^0) - f(x^*)}{T\gamma_{n_{SVRG}}}$$

where $x^*$ is an optimal solution to Equation 5.1.

Further, to achieve an explicit upper bound in Theorem 5.2.2 it is necessary to define the relationship between $\gamma_{n_{SVRG}}$ and $n$. Reddi et al. (2016a,b) specified $\eta$ and $\beta$, resulting in the following theorem.

**Theorem 5.2.3.** Suppose $f \in F_n$, $\eta = \frac{1}{3Ln^\alpha}$ ($0 < \mu_0 < 1$ and $0 < \alpha \leq 1$), $\beta = L/n^{\alpha/2}$, $m_{SVRG} = \lfloor \frac{9n^{\alpha/2}}{5} \rfloor$, $T$ is the total number of iterations which is a multiple of $m_{SVRG}$, and $\nu_{SVRG} > 0$. So I have $\gamma_{n_{SVRG}} \geq \frac{\nu_{SVRG}}{18Ln^\alpha}$ in Theorem 5.2.2. The output $x_a$ of Alg. 3.1 satisfies

$$E[\|\nabla f(x_a)\|^2] \leq \frac{18Ln^\alpha[f(x^0) - f(x^*)]}{Tp_{SVRG}}$$

where $x^*$ is an optimal solution to Equation 5.1.

The proof detail of above three theorems please to see in (Reddi et al., 2016a,b).

### 5.3 Related works

Many SGD-based methods have been applied to optimize functions in different domains. One approach is Stochastic Average Gradient (SAG), which uses a memory of previous gradient values to achieve a linear convergence rate, which can be used to optimize a finite set of smooth functions in a strongly-convex domain (Schmidt et al., 2013). Further, inspired from SAG and SVRG, Stochastic Average Gradient Ascent (SAGA) is an incremental gradient algorithm with fast linear convergence rate that can be used in three different domains, including non-strongly convex problems (Defazio et al., 2014), non-convex but linear problems (Reddi et al., 2016). For non-smooth non-convex finite-sum functions, proximal operators to handle nonsmoothness in the convex problem can cooperate with non-convex optimization as ProxSGD, ProxGD, ProxSAG, ProSVRG and ProxSAGA (Reddi et al., 2016b; Sra, 2012).
Many real-world learning scenarios such as graph DL models require expensive computation of the sample gradient and an unbiased estimator is usually computationally expensive or unavailable [Chen and Luss 2018]. As a result, many works have proposed asymptotically biased optimizations with biased gradient estimators as an economic alternative to an unbiased version that does not converge to the minima, but to their vicinity [Chen et al. 1987, Chen and Gao 1989, Chen and Luss 2018, Tadič and Doucet 2017, Chen et al. 2018]. These methods provide a good insight into the biased gradient search, however they hold under restrictive conditions which are very hard to verify for complex stochastic gradient algorithms. In this chapter, I analyse the nature of biased/unbiased estimators in the optimization process on non-convex problems, and propose a method combining the benefits of both biased and unbiased estimator to achieve a fast convergence rate.

### 5.4 Integrated SVRG with biased estimation

For the first challenge of SVRG, the balance of the gradient update between the full batch and stochastic estimators is fixed. I introduce a hyper-parameter $\lambda$ to balance the weighting of the stochastic element with the full batch gradient to allow the algorithm to choose appropriate behaviours from stochastic, through reduced variance, to batch gradient descent. As a result, the adoption of $\lambda$ in the first-order iterative algorithm can gain benefits from the stochastic estimator to speed-up computation and escape the local minimum, and reduce variance to accelerate the rates of convergence. To address the second challenge associated with the trade-off between biased/unbiased estimator, I use this hyper-parameter to choose the appropriate estimator from biased to unbiased during the whole optimization. In terms of the third challenge of SVRG associated with fixed learning rates, some research has shown that adaptive learning rates can be applied with reduced variance to provide faster convergence rates on non-convex optimization [Reddi et al. 2016, Goodfellow et al. 2016]. I follow the work of MSVRG [Reddi et al. 2016], with an adaptive learning schedule chosen to maximize between two cases of learning schedules which are based on increasing number of iterations, $t$, and the number of samples, $n$. Thus, the learning rate can be decayed by increasing the number of iterations but is also lower bounded by the data size to prevent the adaptive learning rate from decreasing too quickly.

#### 5.4.1 Weighted unbiased estimator analysis

To control the nature behaviours of estimator from stochastic to batch in Alg 3.1 I now introduce a hyper-parameter $\lambda$ as a variance controller that generates a weighted unbiased or biased version of estimator in SVRG. In this section, I focus on a weighted unbiased version that is shown in Alg 5.1. Under appropriate conditions, I can achieve...
Algorithm 5.1: \texttt{SVRG}_{unbiased}(\mathbf{x}^0, \{\eta_t\}_{t=0}^T, \{p_i\}_{i=0}^m, m, S)

\begin{algorithm}
\textbf{Input}: Epoch length \(m\), learning rate \(\eta\), number of epochs \(S = T/m\) where \(T\) is total number of iterations, discrete probability distribution \(\{p_i\}_{i=0}^m\);
1 Initialize \(\mathbf{x}^0 = x_0^0 = \mathbf{x}^0\);
2 for \(s = 0\) to \(S - 1\) do
3 \hspace{1em} \(s^{s+1} = x_s^m\);
4 \hspace{1em} \(g^{s+1} = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\tilde{x}^s)\)
5 for \(t = 0\) to \(m - 1\) do
6 \hspace{2em} Randomly select \(i_t\) from \(\{1, ..., n\}\)
7 \hspace{2em} \(v_t^{s+1} = (1 - \lambda)\nabla f_{i_t}(x_t^{s+1}) - \lambda (\nabla f_{i_t}(\tilde{x}^s) - g^{s+1})\)
8 \hspace{2em} \(x_{t+1}^{s+1} = x_t^{s+1} - \eta_t \Delta v_t^{s+1}\)
9 \hspace{1.5em} \(\tilde{x}^{s+1} = \sum_{i=0}^m p_i x_i^{s+1}\)
\end{algorithm}

\textbf{Output}: \(\tilde{x}^S\)

the following theorems for Alg. 5.1

\textbf{Theorem 5.4.1.} Suppose \(f \in \mathcal{F}_n\) have \(\sigma\)-bounded gradient. Let \(\eta_t = \eta \Delta_{unbiased} = \frac{c_{unbiased}}{\sqrt{\Delta + 1}}\) for \(0 \leq \Delta \leq T - 1\) where \(c_{unbiased} = \sqrt{\frac{f(x_0) - f(x^*)}{(2\lambda^2 - 2\lambda + 1)L\sigma^2}}\) and let \(T\) be a multiple of \(m\). Further let \(p_m = 1\), and \(p_i = 0\) for \(0 \leq i < m\). Then the output \(x_a\) of Alg. 5.1 has a bounded gradient given by

\[\mathbb{E}[\|\nabla f(x_a)^2\|] \leq \sqrt{\frac{2(\lambda^2 - 2\lambda + 1)}{(1 - \lambda)}} \sqrt{\frac{2(f(x_0) - f(x^*))L}{T}} \sigma\]

where \(x^*\) is the optimal solution to Equation 5.1

\textbf{Proof Sketch:} To obtain the upper bound of the gradient in output \(x_a\), one can use the \(\mathcal{L}\)-smooth function in Eq. 5.3 to bound the gradient using \(\sigma\) defined in Definition 5.2.1 which results in Eq. 5.2

In Theorem 5.4.1 I schedule a decayed learning rate \(\eta_{\Delta_{unbiased}} \propto 1/\sqrt{\Delta + 1}\) [Reddi et al. 2016a], which can avoid knowing the total number of inner iterations across all epochs \(T\) in Alg. 5.1 in advance. Compared with the upper bound in Theorem 5.2.1 I can achieve a lower upper bound in Theorem 5.4.1 if \(0 < \lambda < \frac{1}{2}\) and the optimal value of \(\lambda^* \to 0\).

In the second case where the learning rate \(\eta_t\) is fixed depending upon data size \(n\), I can achieve the following result.

\textbf{Theorem 5.4.2.} Let \(f \in \mathcal{F}_n\), let \(c_m = 0\), \(\eta_t = \eta > 0\), \(\beta_t = \beta > 0\), \(c_{unbiased} = c_t + (1 + (1 - \lambda)\eta)\beta_t + 2(1 - \lambda)^2\eta^2 L^2 + L^3\eta^2\), so the intermediate result \(\Omega_{\Delta_{unbiased}} = (\eta_t - (1 - \lambda)^2\eta^2 L^2 - 2(1 - \lambda)^4 L^2\eta^2) > 0\), for \(0 \leq t \leq m - 1\). Define the minimum value of \(\gamma_{\Delta_{unbiased}} := \min_t \Omega_{\Delta_{unbiased}}\). Further let \(p_i = 0\) for \(0 \leq i < m\) and
$p_m = 1$, and $T$ is a multiple of $m$. Then the output $x_a$ of Alg. 5.1 has a bounded gradient given by
\[ E[\| \nabla f(x_a) \|^2] \leq \frac{f(x^0) - f(x^*)}{T \gamma_{n\text{unbiased}}}, \]
where $x^*$ is the optimal solution to Equation 5.1.

**Proof Sketch:** Using Lemma 5.7.1 one can obtain an upper bound of the output gradient $\nabla f(x_a)$ since $x_0^{s+1} = \tilde{x}^s$ and $x_m^{s+1} = \tilde{x}^{s+1}$. For the total number of iterations, $T$, the upper bound can be updated between the starting point $x^0$ to the optimal point $x^*$ in Theorem 5.4.2.

Theorem 5.4.2 provides a general form of upper bound of output gradients. To calculate the IFO complexity, a more specified version of upper bound should be further observed.

Now, I use the same schedule of parameters $\eta$ and $\beta$ as in Theorem 5.2.3 to determine $\gamma_{n\text{unbiased}}$ in the following theorem.

**Theorem 5.4.3.** Suppose $f \in F_n$, let $\eta = \frac{1}{3Ln0a}$ ($0 \leq a \leq 1$, and $0 < \alpha \leq 1$), $\beta = \frac{L}{n^\alpha}$ ($b > 0$), $m_{\text{unbiased}} = \left\lfloor \frac{3n(3a+b)\alpha}{1-\lambda} \right\rfloor$ and $T$ is the total number of iterations. Then, I can obtain the lower bound $\gamma_{n\text{unbiased}} \geq \frac{(1-\lambda)\nu}{9n(2a-b)\alpha L}$ in Theorem 5.4.2. For the output $x_a$ of Alg. 5.1 I have
\[ E[\| \nabla f(x_a) \|^2] \leq \frac{9n(2a-b)\alpha L[f(x^0) - f(x^*)]}{(1-\lambda)T\nu}, \]
where $x^*$ is the optimal solution to Equation 5.1.

where the $\lambda \neq 0$ since the gradient in this case is not $\sigma$-bounded.

**Proof Sketch:** Using Equation 5.15 one can obtain an upper bound for $c_0$ from Equation 5.20, resulting in Equation 5.22. The lower bound of $\gamma_{n\text{unbiased}}$ can be obtained from Equation 5.23. Finally, applying the lower bound of $\gamma_{n\text{unbiased}}$ and modified definitions for $\eta$ and $\beta$, the upper bound of the output in Theorem 5.4.2 can be expressed by the result in Theorem 5.4.3.

As a result, compared with Theorem 5.2.3 I can achieve a lower upper bound in above theorem when $0 < \lambda < 1 - \frac{n(2a-b)\alpha}{2} < 1$ and the optimal value of $\lambda^* \to 0$.

**5.4.2 Biased estimator analysis**

In this section I theoretically analyse the performance of a biased SVRG using the same learning rate schedule with a weighted unbiased version in Alg. 5.2.
Theorem 5.4.4. Suppose $f \in F_n$ has $\sigma$-bounded gradient. Let

$$\eta_{\text{biased}} = \eta_{\Delta} = \frac{c_{\text{biased}}}{\sqrt{\Delta + 1}}$$

for $0 \leq \Delta \leq T - 1$ where $c_{\text{biased}} = \sqrt{\frac{f(x_0) - f(x^*)}{2\lambda L \sigma^2}}$ and let $T$ be a multiple of $m$. Further let $p_m = 1$, and $p_i = 0$ for $0 \leq i < m$. Then the output $x_n$ of Alg. 5.2 has a bounded gradient given by

$$E[\|\nabla f(x_n)^2\|] \leq \frac{2(1 - \lambda)}{\sqrt{\lambda}} \sqrt{\frac{2(f(x_0) - f(x^*))L}{T \sigma}}$$

where $x^*$ is the optimal solution to Equation 5.1.

Proof Sketch: To obtain the upper bound of the gradient in output $x_n$, one can use the $L$-smooth function in Eq. 5.26 to bound the gradient using $\sigma$ defined in Definition 5.2.1 which results in Eq. 5.29. Since in this case the variance of the biased estimator is required to be bounded by an unbiased version, $\lambda$ has a condition that $0 \leq \lambda \leq \frac{2}{3}$ which is proven in Equation 5.24 and Equation 5.25.

I can achieve a lower upper bound of expectation in Theorem 5.4.4 than scaled SVRG in Theorem 5.2.1 when $\lambda$ satisfies the two conditions simultaneously that $0 \leq \lambda \leq \frac{2}{3}$ and $\frac{1}{2} < \lambda \leq 1$. Consequently, the range of $\lambda$ is $\frac{1}{2} < \lambda \leq \frac{2}{3}$, and the optimal value of $\lambda = \lambda^* = \frac{2}{3}$.

For the second case of the learning rate, the biased version of SVRG can obtain its upper bound of expectation according to the following theorem.

Theorem 5.4.5. Let $f \in F_n$, let $c_m = 0$, $\eta_t = \eta > 0$, $\beta_t = \beta > 0$, $c_{t+1} = c_{t+1}(1 + \eta \beta + 2(1 - \lambda)^2 \eta^2 L^2) + L^2 \eta^2 (1 - \lambda)^2$, so the intermediate result $\Omega_{t_{\text{biased}}} = (\eta_t - \frac{c_{t+1} \eta_t}{\beta_t} - \lambda^2 L \eta_t^2 -$
\[ 2\lambda^2 c_{i+1} \eta^2_t > 0, \text{ for } 0 \leq t \leq m - 1. \] Define the minimum value of \( \gamma_n := \min_t \Omega_{\text{biased}}. \) Further let \( p_i = 0 \) for \( 0 \leq i < m \) and \( p_m = 1, \) and \( T \) is a multiple of \( m. \) Then the output \( x_a \) of Alg. 5.3 has a bounded gradient given by
\[
\mathbb{E}[\| \nabla f(x_a) \|^2] \leq \frac{f(x^0) - f(x^*)}{T \gamma_{\text{biased}}},
\]
where \( x^* \) is the optimal solution to Equation 5.1.

**Proof Sketch:** Using Lemma [5.7.2] one can obtain an upper bound of the output gradient \( \nabla f(x_a) \) since \( x_{0}^{s+1} = \tilde{x}^s \) and \( x_{m}^{s+1} = \tilde{x}^{s+1}. \) For the total number of iterations, \( T, \) the upper bound can be updated between the starting point \( x^0 \) to the optimal point \( x^* \) in Theorem 5.4.

Similarly, Theorem [5.4.5] provides a general form of upper bound of output gradients. To calculate the IFO complexity, a more specified version of upper bound should be further observed. Now, I use same schedule of \( \eta \) and \( \beta \) from Theorem 5.2.3 and 5.4.2 to determine \( \gamma_{\text{biased}}, \) and achieve the following result.

**Theorem 5.4.6.** Suppose \( f \in F_n, \) let \( \eta = \frac{1}{3Ln^{a\alpha}} \) (0 \leq a \leq 1 and 0 < a \leq 1), \( \beta = \frac{L}{n^{b\alpha}} \) (b > 0), \( m_{\text{biased}} = \frac{3n^{2a}}{2(1 - \lambda)} \) and \( T \) is the total number of iterations. Then, I can obtain the lower bound \( \gamma_{\text{biased}} \geq \frac{(1 - \lambda)\lambda \nu_1}{9Ln^{2a - b}\alpha} \) in Theorem 5.4.5 Then the output \( x_a \) of Alg. 5.2 has a bounded gradient given by
\[
\mathbb{E}[\| \nabla f(x_a) \|^2] \leq \frac{9Ln^{(2a - b)\alpha} [f(x^0) - f(x^*)]}{(1 - \lambda)T \nu_1},
\]
where \( x^* \) is the optimal solution to Equation 5.1 and \( \lambda \) should satisfy in the range as 0 < \( \lambda < \frac{2}{3} \) where will be detailed in Proofs.

**Proof Sketch:** Using Equation 5.38 one can obtain an upper bound for \( c_0 \) from Equation 5.43 resulting in Equation 5.45. The lower bound of \( \gamma_{\text{biased}} \) can be obtained from Equation 5.46. Finally, applying the lower bound of \( \gamma_{\text{biased}} \) and modified definitions for \( \eta \) and \( \beta, \) the upper bound of the output in Theorem 5.4.5 can be expressed by the result in Theorem 5.4.6

In comparison to the upper bound of expectation in Theorem 5.2.3 I can achieve a lower upper bound when \( \lambda \) satisfies \( 0 < \frac{1 - \sqrt{1 - 4n^{(2a - b - 1)\alpha}}}{2} < \lambda < \frac{1 + \sqrt{1 - 4n^{(2a - b - 1)\alpha}}}{2} < \frac{2}{3}, \) and hence the optimal value of \( \lambda^* = 0.5. \)
5.4.3 Variance control and combined biased and unbiased estimation

To estimate the performance of unbiased and biased estimators, I investigate the value of the upper bound for two cases of learning rate. In the first case when the learning rate decayed by iteration number, the upper bound of the biased estimator is minimized when $\lambda^*=2/3$ in Theorem 5.4.1 and for the unbiased version is minimized when $\lambda^*=0$ in Theorem 5.4.4. This result shows that the biased and weighted SVRG estimator can provide a tighter upper bound than unbiased SGD when the learning rate is decayed. In the second case the learning rate is fixed, the upper bound of the unbiased version is minimized when $\lambda^* \to 0$ in Theorem 5.4.2, which is lower than the biased minimum of $\lambda^*=0.5$ in Theorem 5.4.5. When the learning rate is fixed standard SVRG is better than SGD. Therefore, it shows that unbiased and weighted SVRG is better than biased SVRG and SGD.

Consequently, these results give rise to a new optimization method, ISVRG$^+$ (Alg. 5.3), which can combine biased and unbiased SVRG which controls the reduced variance so as to improve the rates of convergence.

**Algorithm 5.3: ISVRG$^+$** $(x^0_0, \{\eta_i\}_{i=0}^{T}, \{p_i\}_{i=0}^{m}, m, S)$

**Input**: Epoch length $m$, learning rate $\eta_n = \max\left\{\frac{\sqrt{c_{\text{biased}}}}{ms}, \frac{1}{3L\nu a^2}\right\}$, number of epochs $S = T/m$ where $T$ is total number of iterations, discrete probability distribution $\{p_i\}_{i=0}^{m}$, and $\lambda \to 0$.

1. Initialize $x^0_0 = x^0$.
2. for $s=0$ to $S-1$ do
3.   \[ x^s_0 = x^s_0; \]
4.   \[ g^{s+1} = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\tilde{x}^s); \]
5. for $t = 0$ to $m - 1$ do
6.   Randomly select $i_t$ from $\{1, ..., n\}$
7.   if $\eta_t = \frac{\sqrt{c_{\text{biased}}}}{ms}$ then
8.     \[ v_t^{s+1} = \frac{1}{s} \left( \nabla f_i(x_t^{s+1}) - \nabla f_i(\tilde{x}^s) \right) + \frac{2}{3} g^{s+1}; \]
9.   else if $\eta_t = \eta = \frac{1}{3L\nu a}$ then
10.    \[ v_t^{s+1} = (1 - \lambda) \nabla f_i(x_t^{s+1}) - \lambda (\nabla f_i(\tilde{x}^s) - g^{s+1}); \]
11.    \[ x_{t+1} = x_t^{s+1} - \eta_\Delta v_t^{s+1}; \]
12.    \[ \tilde{x}^{s+1} = \sum_{i=0}^{m} p_i x_{i+1}^{s+1}; \]

The general result for ISVRG$^+$ is given in the following theorem.

**Theorem 5.4.7.** Let $f \in F_n$ have $\sigma$-bounded gradients. Let
\[ \eta_t = \eta_\Delta = \max\left\{\frac{c}{\sqrt{\Delta} + 1}, \frac{1}{3L\nu a^2}\right\} \text{ for } \Delta \text{ from } 0 \text{ to } T - 1, \] and $c = \sqrt{\frac{3f(x^0) - f(x^*)}{4L\sigma^2}}$. Further let $T$ is a multiple of $m$, $p_m = 1$ and $p_i = 0$ for $0 \leq i < m$. Then, the output $x_a$ of Alg. 5.3 satisfies
\[ \mathbb{E}[\|\nabla f(x_a)\|^2] \leq \tilde{v} \min\left\{\sqrt{3}\frac{(f(x^0) - f(x^*))L}{T}, \frac{9n(3a-b)\nu L[f(x^0) - f(x^*)]}{Tv_2}\right\}, \]
where $x^*$ is an optimal solution to Equation 5.1. $0 \leq a \leq 1$, $0 < \alpha \leq 1$ and $b > 0$. $\tilde{\nu}, \nu_2$ are universal constants.

Based on the result from Theorem 5.4.7 and rewriting the above result in terms IFO calls, I should specify the optimal value of parameters including $a, b$ and $\alpha$ from Theorem 5.4.7. The optimal value of $a, b$ and $\alpha$ can be found by two conditions: (a) the upper bound of $\text{ISVRG}^*$ is lower than that of scaled standard SVRG in Theorem 5.2.3 on magnitude level as $n^0 < n^{(2a-b)\alpha} \leq n^\alpha$, if $\lambda = \lambda^* = 0$. (2) $0 < \alpha = 1/(3a + b) \leq 1$ when $0 \leq a \leq 1$ and $b > 0$. Thus, I can achieve lowest upper bound when $\frac{a}{b} < 1$ and $\alpha \leq \frac{1}{4}$. Such parameters can give rise to the following key result of the chapter, which are showed in Corollary 5.4.1. In this corollary, the IFO complexity of $\text{ISVRG}^*$ is the minima between $1/\epsilon^2$ which is equal to the IFO complexity of SGD method [Reddi et al., 2016a; Ghadimi and Lan, 2016] and $n^{1/4}/\epsilon$. Now I calculate the IFO calls in a general form, which is showed in Corollary 5.4.1.

**Corollary 5.4.1.** Suppose $f \in \mathcal{F}_n$, the IFO complexity of Alg. 5.3 (with parameters from Theorem 5.4.7) achieves an $\epsilon$-accurate solution that is $\mathcal{O}(\min\{1/\epsilon^2, n^{1/4}/\epsilon\})$, where the number of IFO calls is minimized when $\frac{a}{b} < 1$ and $\alpha \leq \frac{1}{4}$.

**Proof:** The result follows from Theorem 5.4.7 and the fact that $m = \lfloor 3n^{(3a+b)\alpha} \rfloor$. Suppose $\alpha < 1/4$, then $m = o(n)$. However, $n$ IFO calls are invested in calculating the average gradient at the end of each epoch. On the other hand, when $\alpha \geq 1/4$, the total number of IFO calls made by Alg. refa5.4 in each epoch is $\Omega(n)$ since $m = \lfloor 3n^{4a} \rfloor$. Using this relationship, the orcal calls required for calculating the average gradient (per epoch) is of lower order, leading to $\mathcal{O}(n + (n^\alpha/\epsilon))$ IFO calls.

### 5.4.4 Comparison of the convergence rates

In this section, I compare the convergence rates of ISVRG with three other optimization methods including SGD, standard SVRG and Gradient Descent (GD) provided by [Reddi et al. 2016a], and discuss how the hyper-parameter $\lambda$ works with the hybrid adaptive learning rate to control the variance and to balance the trade-off between bias/unbiased estimation. All results in comparison are based on IFO complexity to achieve an $\epsilon$-accurate solution.

1. Dependence on $\epsilon$: One dependence of convergence rate of the algorithms is on iteration $T$ (or $\epsilon$). Non-convex SGD as $\mathcal{O}(1/\epsilon^2)$ depends on $\epsilon$. Standard SVRG, ISVRG and GD methods on non-convex problems are $\mathcal{O}(1/\epsilon)$. As a result, the speed-up in convergence over SGD is especially significant when high accuracy solutions are required and hence $\epsilon$ is small.
2. Dependence on \( n \): SGD is independent of \( n \) whereas SVRG and GD depend on \( n \). Hence, the IFO complexity of SVRG and GD are sensitive to \( n \). Reddi et al. (2016a) show the dependence for GD is \( n \) for both convex and non-convex, and the dependence for SVRG is reduced to \( n^{1/2} \) for convex and \( n^{2/3} \) for non-convex. However, they pose an open question whether the difference of dependence on \( n \) between convex and non-convex is due to the non-convexity or their special choice of analysis. To answer this open question, I use a general form to set up parameters in my analysis, including learning rates \( \eta \), \( \beta \) and hyper-parameter \( \lambda \) to define the optimal value of \( m \) step size and the final result of upper bound so as to avoid an arbitrary analysis. Compared with their results, ISVRG has the dependence of \( n \) reduced from \( n^{2/3} \) to \( n^{1/2} \) for non-convex problems, which the result is same with that for convex problems. As a result, I analyse the open question that the difference in dependence on \( n \) is not only a factor that affects non-convexity, but also other factors include the learning rate \( \eta \) and \( \lambda \).

3. Learning rates: In my analysis, there are two cases of learning rates schedules of SVRG, including the learning rate which reduces with iterate \( t \), and the second case where the learning rate is fixed by the training sample size \( n \). Following the range of \( \lambda \) obtained by previous theorems and corollaries, I now discuss how the hyper-parameter \( \lambda \) works with the hybrid adaptive learning rate to control the variance. For the case where the first term of the learning rate is larger than second term when \( t \) is small, larger \( \lambda \) can reduce the variance to accelerate the convergence in early iterations. On the other hand, when the second term becomes larger later in optimization, the value of \( \lambda \) depends on the size of training samples. Higher variance will help gradients escape from local minimal. So the values of \( \lambda \) in large-scale application can make the algorithm behave more stochastically. As a result, when the first term is maximum a more reduced variance approach will provide better optimization, and in the second case a stochastic gradient approach will be better. To achieve the local optima in Equation 5.1 I theoretically calculate two optimal values of \( \lambda \) as \( \lambda^* \), including \( \lambda^* = \frac{2}{3} \) if \( \eta_\Delta = c/\sqrt{\Delta + 1} \) and the \( \lambda^* \rightarrow 0 \) if \( \eta = \frac{1}{3Ln^2} \).

Consequently, when the learning rate is decayed by increasing number of iterations, a biased estimation with more reduced variance will fast converge to a point, and when the learning rate is fixed depending on the number of data samples \( n \), a unbiased estimation with stochastic gradient will be better. All theoretical proof details are in appendix.

5.5 Application

To experimentally confirm the theoretical results and insights, I train three common DL topologies, including LeNets incuding LeNet-300-100 which has two fully connected
layers as hidden layers with 300 and 100 neurons respectively, and LeNet-5 which has two convolutional layers and two fully connected layers, and VGG-16 (Simonyan and Zisserman, 2014b) on three datasets including MNIST, CIFAR-10 and tiny ImageNet that is a subset of ImageNet challenge (2012 ILSVRC (Russakovsky et al., 2015)), which contains 500 categories. Each category has 600 training images, 200 validation images and 200 test images, each image is re-sized to 96×96 pixels. My method was implemented using Caffe that is a DL framework. I use SVRG as my baseline and I choose MSVRG method to compare with ISVRG+. MSVRG is a leading VR scheme based on stochastic methods which can perform better than SGD and GD for non-convex optimization (Reddi et al., 2016a; J. Reddi et al., 2016). In my experiments the maximal value of learning rate was chosen as $\eta_t = \max\{\eta_0/\Delta, 1/(3Ln^{1/5})\}$, where the $\Delta$ from 1 to $Sm$ and $L = 1, 10, 100$. To evaluate the performance of ISVRG+, I choose the cross entropy using the softmax log loss function as the result of test error and mean squared error (MSE) to evaluate the quality of the neural networks.

In Figure 5.1, I compared the performance of SGD, MSVRG and ISVRG+. For SGD I set-up $\lambda = 0$ and $\eta = \eta_0/(\Delta)$ in Alg 5.1 and for MSVRG $\lambda = 0.5$ and $\eta_t = \max\{\eta_0/\Delta, 1/(3Ln^{1/5})\}$ in Alg 5.1. In Figure 5.1 the blue lines show the SGD as baseline. The red lines show the test loss and error of ISVRG+ which are all lower than the other methods. Particularly, both test error and loss of ISVRG+ drop down dramatically in later epochs since the adaptive learning rate starts to depend on the data size $n$ rather than $\Delta$, and correspondingly $\lambda$ changes from 2/3 to 0 meaning that the variance of gradients becomes higher with the unbiased estimator and can help points escape from local minima. Consequently, compared with SGD and MSVRG, I can see that ISVRG+ can converge faster in early epochs in both test error/loss on all three problems, and the results significantly decrease after several epochs when the learning rate changes from decaying to fixed. Thus ISVRG+ has the lowest test error/loss, which can efficiently accelerate the rates of convergence.

5.6 Discussion

In this chapter, I proposed a VR-based optimization ISVRG+ for non-convex problems. I theoretically determined that a hyper-parameter $\lambda$ working with a adaptive learning rate in each iteration can control the reduced variance of SVRG and balance the trade-off between biased/unbiased estimator. Moreover, to verify my theoretical results, I experiment on different datasets on DL models to estimate the range of $\lambda$ and compare these with other leading results. Both theoretical and experimental results show that ISVRG+ can efficiently accelerate rates of convergence and is faster than SVRG and SGD for non-convex optimization.
Figure 5.1: Comparison of rates of convergence in three approaches, including SGD, MSVRG and $ISVRG^+$ via test loss/error.

5.7 Proof details

I now provide the proof of my theorems in the chapter.

Proof of Theorem 5.4.1

Theorem. Suppose $f \in \mathcal{F}_n$ have $\sigma$-bounded gradient. Let $\eta_t = \eta_{\text{unbiased}} = c_{\text{unbiased}} \sqrt{\Delta + 1}$ for $0 \leq \Delta \leq T - 1$ where $c_{\text{unbiased}} = \sqrt{f(x_0) - f(x^*) / (2\lambda^2 - 2\lambda + 1) L \sigma^2}$ and let $T$ be a multiple of $m$. Further let $p_m = 1$, and $p_i = 0$ for $0 \leq i < m$. Then the output $x_a$ of Alg. 5.1 I have

$$E[\|\nabla f(x_a)\|^2] \leq \sqrt{\frac{2(\lambda^2 - 2\lambda + 1)}{(1 - \lambda) T}} \sqrt{\frac{2(f(x^0) - f(x^*))L}{T}} \sigma$$
Proof. As the learning rate decay from 1 to $T$, I use Definition 5.2 to bound gradients $v_t^{s+1}$ as following:

\[
\mathbb{E}[||v_t^{s+1}||^2] = \mathbb{E}[|| (1 - \lambda) \nabla f_t(x_t^{s+1}) - \lambda (\nabla f_t(\tilde{x}^s) - \nabla f(\tilde{x}^s)) ||^2) \\
\leq 2(\mathbb{E}[|| (1 - \lambda) \nabla f_t(x_t^{s+1}) ||^2 + || \lambda (\nabla f_t(\tilde{x}^s) - \nabla f(\tilde{x}^s)) ||^2]) \\
\leq 2((1 - \lambda)^2 \mathbb{E}[|| \nabla f_t(x_t^{s+1}) ||^2] + \lambda^2 \mathbb{E}[|| \nabla f_t(\tilde{x}^s) ||^2]) \\
\leq (4\lambda^2 - 4\lambda + 2)\sigma^2,
\]

(5.2)

where the first inequality I followed Lemma 5.7.3 when $r=2$. The second inequality I followed (a) $\sigma$-bounded gradient property of $f$ and (b) the fact that for a random variable $\zeta$ followed $\mathbb{E}[|| \zeta - \mathbb{E}[\zeta] ||^2] \leq \mathbb{E}[|| \zeta ||^2]$.

Since $f$ is $\mathcal{L}$-smooth, I have

\[
\mathbb{E}[f(x_t^{s+1})] \leq \mathbb{E}[f(x_t^{s+1}) + \langle \nabla f(x_t^{s+1}), x_t^{s+1} - x_t \rangle] + \frac{L}{2} || x_t^{s+1} - x_t ||^2.
\]

(5.3)

Using Alg. 5.1 to update and since $\mathbb{E}[\nabla f(x_t^{s+1})] = \nabla f(x_t^{s+1})$ (unbiasedness of the stochastic gradients), Ineq. 5.3 would be updated as:

\[
\mathbb{E}[f(x_t^{s+1})] \leq \mathbb{E}[f(x_t^{s+1})] - \eta\Delta(1 - \lambda) || \nabla f(x_t^{s+1}) ||^2 + \frac{L\eta^2}{2} || v_t^{s+1} ||^2.
\]

(5.4)

Adding the bound of $v_t^{s+1}$ from Ineq. 5.2 to Ineq. 5.4 I can obtain that:

\[
\mathbb{E}[f(x_t^{s+1})] \leq \mathbb{E}[f(x_t^{s+1})] - \eta\Delta(1 - \lambda) \mathbb{E}[|| \nabla f(x_t^{s+1}) ||^2] + \frac{L\eta^2}{2} \mathbb{E}[|| v_t^{s+1} ||^2].
\]

(5.5)

Thus the Ineq. 5.5 can be alternated as

\[
\mathbb{E}[|| \nabla f(x_t^{s+1}) ||^2] \leq \frac{1}{\eta\Delta(1 - \lambda)} \mathbb{E}[f(x_t^{s+1}) - f(x_{t+1})] + \frac{L\eta(2\lambda^2 - 2\lambda + 1)}{(1 - \lambda)}\sigma^2,
\]

(5.6)

where $t \in \{0, ..., m - 1\}$, $s \in \{0, ..., S - 1\}$, $\Delta \in \{0, ..., T - 1\}$, and $T = mS$.

The minimum upper bound in Ineq. 5.7 can be achieved when $t = m - 1$ and $s = S - 1$, and use the constant $\eta$ I can obtain:

\[
\min_{t,s} \mathbb{E}[|| \nabla f(x_t^{s+1}) ||^2] \leq \frac{1}{T} \sum_{s=0}^{S-1} \sum_{t=0}^{m-1} \mathbb{E}[|| f(x_t^{s+1}) ||^2] + \frac{L\eta(2\lambda^2 - 2\lambda + 1)}{(1 - \lambda)}\sigma^2
\]

(5.7)

\[
\leq \frac{1}{T} \left( \frac{1}{\eta(1 - \lambda)} \mathbb{E}[f(x^0) - f(x_T)] + \frac{L\eta(2\lambda^2 - 2\lambda + 1)}{(1 - \lambda)}\sigma^2 \right)
\]

\[
\leq \frac{1}{T\eta(1 - \lambda)} (f(x^0) - f(x^*)) + \frac{L\eta(2\lambda^2 - 2\lambda + 1)}{(1 - \lambda)}\sigma^2
\]
The first inequality can hold due to the minimum is less than average. The second inequality is achieved from Equation 5.6, and the third one follows the fact that $f(x^*) \leq f(x^T)$. To calculate learning rate $\eta$, I take the derivative of the last inequality in Ineq. 5.7 as

$$\frac{\partial}{\partial \eta} \left( \frac{1}{T(1-\lambda)} (f(x^0) - f(x^*)) + \frac{L\eta(2\lambda^2 - 2\lambda + 1)}{(1-\lambda)} \sigma^2 \right) = 0 \quad (5.8)$$

Thus, $\eta_{\text{unbiased}} = \eta = c/\sqrt{\Delta + 1}$, where $c_{\text{unbiased}} = \sqrt{\frac{f(x^0) - f(x^*)}{(2\lambda^2 - 2\lambda + 1)L\sigma^2}}$. Bring the result of $\eta_{\text{unbiased}} = c_{\text{unbiased}}/\sqrt{\Delta + 1}$ to Equation 5.7, I can achieve the upper bound of expectation as

$$\min_{t,s} \mathbb{E}[\|\nabla f(x_{s+1}^t)\|^2] \leq \frac{1}{T(1-\lambda)} \left( \frac{\sqrt{T}(f(x^0) - f(x^*))}{c_{\text{unbiased}}} + \frac{Lc_{\text{unbiased}}\sigma^2}{(1-\lambda)} \right) \leq \frac{1}{\sqrt{T}(1-\lambda)} \left( \frac{1}{c_{\text{unbiased}}} (f(x^0) - f(x^*)) + Lc_{\text{unbiased}}\sigma^2 \right). \quad (5.9)$$

For the case that the learning rate depends on the data size $n$, I provide one useful lemma in Lemma 5.7.1 firstly that can be used for proofing my Theorems.

**Lemma 5.7.1.** For $c_{\text{unbiased}}, c_{t+1}, \beta_t > 0$, I have

$$c_{t_{\text{unbiased}}} = c_{t+1}(1 + \eta_t \beta_t (1 - \lambda) + 2(1 - \lambda)^2 \eta_t^2 L^2) + L^2 \eta_t^2.$$  

Let $\eta_t, \beta_t$ and $c_{t+1}$ is given so that the $\Omega_{\text{unbiased}} > 0$ can be showed as

$$\Omega_{\text{unbiased}} = \eta_t - \frac{c_{t+1} + 1}{\beta_t} (1 - \lambda) - (1 - \lambda)^2 L\eta_t^2 - 2(1 - \lambda)^4 c_{t+1} \eta_t^2.$$  

Thus, the iterates in Alg. 5.1 satisfy the bound:

$$\mathbb{E}[\|\nabla f(x_s^{t+1})\|^2] \leq \frac{R_{s+1}^t - R_{t+1}^{s+1}}{\Omega_{\text{unbiased}}}$$

where $R_{s+1}^t := \mathbb{E}[f(x_{s+1}^t) + c_{\text{unbiased}} (1 - \lambda) x_{s+1}^t - \lambda x_s^t \|^2]$ for $0 \leq s \leq S - 1.$
Chapter 5 A Stochastic Gradient Method with Hybrid Estimation

Proof. To further bound the result in Ineq. 5.27 since $f$ is $L$-smooth, I require to bound the intermediate iterates $v^{s+1}_t$, which is showed following inequalities:

$$
\mathbb{E}[\|v^{s+1}_t\|^2] = \mathbb{E}[\| (1 - \lambda)(\nabla f_t(x^{s+1}_t) - \lambda(\nabla f_t(x^s) - \nabla f(x^s)) \|^2)]
= \mathbb{E}[\| \zeta^{s+1}_t + \lambda \nabla f(x^s) - (1 - \lambda)\nabla f(x^{s+1}_t) + (1 - \lambda)\nabla f(x^{s+1}_t) \|^2]
\leq 2\mathbb{E}[\| (1 - \lambda)\nabla f(x^{s+1}_t) \|^2] + 2\mathbb{E}[\| \zeta^{s+1}_t - \mathbb{E}[\zeta^{s+1}_t] \|^2]
\leq 2(1 - \lambda)^2\mathbb{E}[\| \nabla f(x^{s+1}_t) \|^2] + 2\mathbb{E}[\| (1 - \lambda)\nabla f_t(x^{s+1}_t) - \lambda \nabla f_t(x^s) \|^2]
\leq 2(1 - \lambda)^2\mathbb{E}[\| \nabla f(x^{s+1}_t) \|^2] + 2L^2\mathbb{E}[\| (1 - \lambda)x^{s+1}_t - \lambda x^s \|^2],
$$

(5.10)

where $0 \leq \lambda \leq 1$. In the first inequality, the variable $\zeta$ is showed as

$$
\zeta^{s+1}_t = \frac{1}{|I_t|} \sum_{i_t \in I_t} ((1 - \lambda)\nabla f_{i_t}(x^{s+1}_t) - \lambda \nabla f_{i_t}(x^s)),
$$

(5.11)

since $\mathbb{E}[\zeta^{s+1}_t] = (1 - \lambda)\nabla f(x^{s+1}_t) - \lambda \nabla f(x^s)$. The second inequality is obtain from Ineq. 5.10. And the last inequality, I followed the Equation 5.3 and $L$-smooth function: $||\nabla f(x) - \nabla f(y)|| \leq L||x - y||$.

Consider now the Lyapinov function:

$$
R^{s+1}_t := \mathbb{E}[f(x^{s+1}_t) + c_t \| (1 - \lambda)x^{s+1}_t - \lambda x^s \|^2].
$$

(5.12)

To bound Equation 5.12 I require the bound of $\mathbb{E}[\| (1 - \lambda)x^{s+1}_{t+1} - \lambda x^s \|^2]$ as following:

$$
\mathbb{E}[\| (1 - \lambda)x^{s+1}_{t+1} - \lambda x^s \|^2] = \mathbb{E}[\| (1 - \lambda)(x^{s+1}_{t+1} - x^{s+1}_t) + (1 - \lambda)x^{s+1}_t - \lambda x^s \|^2]
\leq \mathbb{E}[\| (1 - \lambda)x^{s+1}_{t+1} - x^{s+1}_t \|^2] + \| (1 - \lambda)x^{s+1}_t - \lambda x^s \|^2 + 2\mathbb{E}[\| (1 - \lambda)(x^{s+1}_{t+1} - x^{s+1}_t), ((1 - \lambda)x^{s+1}_t - \lambda x^s)\|]
\leq \mathbb{E}[\|v^{s+1}_t\|^2] + \| (1 - \lambda)x^{s+1}_t - \lambda x^s \|^2]
\leq 2\eta(1 - \lambda)\mathbb{E}[\| \nabla f(x^{s+1}_t), (1 - \lambda)x^{s+1}_t - \lambda x^s \|]
\frac{1}{2}\beta_t \| (1 - \lambda)x^{s+1}_t - \lambda x^s \|^2]
\leq 2\eta(1 - \lambda)\mathbb{E}[\| v^{s+1}_t \|^2 + \| \nabla f(x^{s+1}_t) \|^2]
\leq 2\eta T \mathbb{E}[\| \nabla f(x^{s+1}_t) \|^2 + \| \nabla f(x^{s+1}_t) \|^2 + \frac{1}{2}\beta_t \| (1 - \lambda)x^{s+1}_t - \lambda x^s \|^2]
$$

(5.13)

The second equality follows from the unbiasedness of the update of Alg 5.1. The last inequality follows from application of Cauchy-Schwarz and Young’s inequality. Combining Equation 5.10, Equation 5.12 and Equation 5.13, I can achieve the bound of
\[ R_{t+1}^{\text{unbiased}} := \mathbb{E}[f(x^t_{t+1}) + c_{t+1} \parallel (1 - \lambda)x^t_{t+1} - \lambda \tilde{x}^s \parallel^2] \] as

\[ R_{t+1}^{\text{unbiased}} \leq \mathbb{E}[f(x^t_{t+1}) - \eta_t \parallel \nabla f(x^t_{t+1}) \parallel^2 + \frac{L\eta_t^2}{2} \parallel v^t_{t+1} \parallel^2] + \mathbb{E}[c_{t+1}\eta_t^2(1 - \lambda)^2 \parallel v^t_{t+1} \parallel^2 + c_{t+1} \parallel (1 - \lambda)x^t_{t+1} - \lambda \tilde{x}^s \parallel^2] - 2c_{t+1}(1 - \lambda)\eta_t \mathbb{E}[\frac{1}{2}\beta_t \parallel \nabla f(x^t_{t+1}) \parallel^2 + \frac{1}{2}\beta_t \parallel (1 - \lambda)x^t_{t+1} - \lambda \tilde{x}^s \parallel^2] \]

\[ \leq \mathbb{E}[f(x^t_{t+1}) - (\eta_t + \frac{c_{t+1}\eta_t(1 - \lambda)}{\beta_t}) \parallel \nabla f(x^t_{t+1}) \parallel^2] \]

\[ + \left(\frac{L\eta_t^2}{2} + c_{t+1}\eta_t^2(1 - \lambda)^2\right) \mathbb{E}[\parallel v^t_{t+1} \parallel^2] \]

\[ + (c_{t+1} + c_{t+1}\eta_t\beta_t(1 - \lambda)) \mathbb{E}[\parallel (1 - \lambda)x^t_{t+1} - \lambda \tilde{x}^s \parallel^2] \]

\[ = \mathbb{E}[f(x^t_{t+1})] - (\eta_t - \frac{c_{t+1}\eta_t(1 - \lambda)}{\beta_t}) - (1 - \lambda)^2L\eta_t^2 - 2(1 - \lambda)^4c_{t+1}\eta_t^2) \mathbb{E}[\parallel \nabla f(x^t_{t+1}) \parallel^2] \]

\[ + (c_{t+1}(1 + \eta_t\beta_t(1 - \lambda) + 2(1 - \lambda)^2\eta_t^2L^2) + L^3\eta_t^2) \mathbb{E}[\parallel (1 - \lambda)x^t_{t+1} - \lambda \tilde{x}^s \parallel^2] \]

\[ \leq R^t_{t+1} - (\eta_t - \frac{c_{t+1}\eta_t(1 - \lambda)}{\beta_t}) - (1 - \lambda)^2L\eta_t^2 - 2(1 - \lambda)^4c_{t+1}\eta_t^2) \mathbb{E}[\parallel \nabla f(x^t_{t+1}) \parallel^2]. \quad (5.14) \]

The last inequality follows \( R^t_{t+1} := \mathbb{E}[f(x^t_{t+1}) + c_t \parallel (1 - \lambda)x^t_{t+1} - \lambda \tilde{x}^s \parallel^2] \) where

\[ c_{t, \text{unbiased}} = c_{t+1}(1 + \eta_t\beta_t(1 - \lambda) + 2(1 - \lambda)^2\eta_t^2L^2) + L^3\eta_t^2. \quad (5.15) \]

Thus, the Ineq. (5.14) can be alternated as

\[ \mathbb{E}[\parallel \nabla f(x^t_{t+1}) \parallel^2] \leq \frac{R_{t+1}^t - R_{t+1}^{\text{unbiased}}}{\Omega_{\text{unbiased}}}, \quad (5.16) \]

where \( \Omega_{\text{unbiased}} = \eta_t - \frac{c_{t+1}\eta_t(1 - \lambda)}{\beta_t} - (1 - \lambda)^2L\eta_t^2 - 2(1 - \lambda)^4c_{t+1}\eta_t^2 \)

\[
\text{Proof of Theorem } 5.4.2
\]

**Theorem.** Let \( f \in \mathcal{F}_n \), let \( c_m = 0 \), \( \eta_t = \eta > 0 \), \( \beta_t = \beta > 0 \), \( c_{t, \text{unbiased}} = c_{t+1}(1 + (1 - \lambda)\eta\beta + 2(1 - \lambda)^2\eta^2L^2) + L^3\eta^2 \), so the intermediate result \( \Omega_{\text{unbiased}} = (\eta_t - (1 - \lambda)d_{t+1}\eta_t) - (1 - \lambda)^2L\eta_t^2 - 2(1 - \lambda)^4c_{t+1}\eta_t^2 > 0 \), for \( 0 \leq t \leq m - 1 \). Define the minimum value of \( \gamma_{\text{unbiased}} := \min \Omega_{\text{unbiased}}. \) Further let \( p_i = 0 \) for \( 0 \leq i < m \) and \( p_m = 1 \), and \( T \) is a multiple of \( m \). So the output \( x_a \) of Alg. [5.1] I have

\[ \mathbb{E}[\parallel \nabla f(x_a) \parallel^2] \leq \frac{f(x^0) - f(x^*)}{T\gamma_{\text{unbiased}}}, \]

where \( x^* \) is the optimal solution to Problem [5.4].
Proof. Using the result from Lemma 5.7.2 and \( \eta_t = \eta \) when \( t \in \{0, \ldots, m-1\} \), I can achieve the following bound:

\[
\sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leq \frac{R_0^{s+1} - R_m^{s+1}}{\gamma_{\text{unbiased}}}, \tag{5.17}
\]

Thus, the bound in Ineq. 5.17 can updated as

\[
\sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leq \mathbb{E}[f(\tilde{x}^s) - f(\tilde{x}^{s+1})], \tag{5.18}
\]

where \( R_0^{s+1} = \mathbb{E}[f(\tilde{x}^s)] \) since \( x_0^{s+1} = \tilde{x}^s \) and \( R_m^{s+1} = \mathbb{E}[f(\tilde{x}^{s+1})] \) since \( x_m^{s+1} = \tilde{x}^{s+1} \), which I use the condition that \( c_m = 0, p_m = 1, \) and \( p_i = 0 \) for \( i < m \). For the total number of iterations \( T = Sm \), I further sum up iteration \( s \) as

\[
\frac{1}{T} \sum_{s=0}^{S-1} \sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leq \frac{f(x^0) - f(x^*)}{T\gamma_{\text{unbiased}}}, \tag{5.19}
\]

where the \( \tilde{x}^0 = x^0 \) and \( \tilde{x}^* = x^* \). Thus, I can obtain my final result. \( \square \)

**Proof of Theorem 5.4.3**

**Theorem.** Suppose \( f \in \mathcal{F}_n \), let \( \eta = \frac{1}{3Ln^a} \) (0 ≤ \( a \leq 1 \), and 0 < \( \alpha \leq 1 \)), \( \beta = \frac{L}{n^b\alpha} \) (\( b > 0 \)), \( m_{\text{unbiased}} = \lfloor \frac{3n(3a+b)\alpha}{(1-\lambda)} \rfloor \) and \( T \) is the total number of iterations. Then, I can obtain the lower bound \( \gamma_{\text{unbiased}} \geq \frac{(1-\lambda)T\nu}{9n^2(2a-b)\alpha L} \) in Theorem 5.4.2. For the output \( x_a \) of Alg. 5.7 I have

\[
\mathbb{E}[\|\nabla f(x_a)\|^2] \leq \frac{9n^2(2a-b)\alpha L[f(x^0) - f(x^*)]}{(1-\lambda)T\nu},
\]

where \( x^* \) is an optimal solution to Equation 5.1.

**Proof.** Using the relation in Equation 5.15 and \( c_m = 0 \), I estimated the upper bound of \( c_0 \) as

\[
c_0 = L^3\eta^2(1 + \theta_{\text{unbiased}})^m - 1 \quad \frac{\theta_{\text{unbiased}}}{\theta_{\text{unbiased}}}, \tag{5.20}
\]

where \( \theta_{\text{unbiased}} = 2(1-\lambda)^2L^2\eta^2 + \eta\beta(1-\lambda) \). Let \( \eta = \frac{1}{3Ln^a} \) and \( \beta = \frac{L}{n^b\alpha} \), the \( \theta \) can be alternated as:

\[
\theta_{\text{unbiased}} = 2(1-\lambda)^2L^2\eta^2 + \eta\beta(1-\lambda)
= \frac{(1-\lambda)(2(1-\lambda)^2)}{3n(3a+b)\alpha} + \frac{2(1-\lambda)^2}{9n^2(2a\alpha)}
\leq \frac{1-\lambda}{3n(3a+b)\alpha}, \tag{5.21}
\]
Using the above bound $\theta$, I can get the further bound of $c_0$ as

$$
c_0 = \frac{L^3[(1 + \theta_{unbiased})^m - 1]}{9L^2n^{2\alpha}} \left( \frac{1 - \lambda}{3n^{(a-b)\alpha}} + \frac{2(1 - \lambda)^2}{9n^{2\alpha}} \right)
$$

$$
\leq \frac{L^3[(1 + \theta_{unbiased})^m - 1]}{3L^2(1 - \lambda)n^{(a-b)\alpha} + 2L^2(1 - \lambda)^2}
$$

$$
\leq \frac{L^3(e - 1)}{3L^2n^{(a-b)\alpha}},
$$

(5.22)

In the first inequality, due to the value of $(1 + \theta_{unbiased})^m$ is increasing when $m_{unbiased} = \lfloor \frac{1}{\theta_{unbiased}} \rfloor > 0$, I can use $\lim_{l \to \infty}(1 + \frac{1}{l})^l = e$ (the $e$ is Euler's number) to calculate upper bound of $(1 + \theta)^m$. Next, the lower bound of $\gamma_{unbiased}$ is given as:

$$
\gamma_{unbiased} = \min_{\ell} (\eta - \frac{c_{\ell + 1}\eta}{\beta}(1 - \lambda) - (1 - \lambda)^22(1 - \lambda)^4c_{\ell + 1}\eta^2)
$$

$$
\geq (\eta - \frac{c_0\eta}{\beta}(1 - \lambda) - (1 - \lambda)^22(1 - \lambda)^4c_0\eta^2)
$$

(5.23)

where $\nu$ is independent of $n$. According to Theorem 5.4.2, I can achieve my result.

To guarantee theoretical asymptotic convergence for the biased estimator in the first case of the learning rate, the rate should be close to 0 at the end of iteration $T \to \infty$.

**Proof of Theorem 5.4.4**

**Theorem.** Suppose $f \in \mathcal{F}_n$ have $\sigma$-bounded gradient. Let $\eta_{biased} = \eta_{\Delta} = c_{biased}/\sqrt{\Delta + 1}$ for $0 \leq \Delta \leq T - 1$ where $c_{biased} = \sqrt{\frac{f(x_0) - f(x^*)}{2\lambda\sigma^2}}$ and let $T$ be a multiple of $m$. Further let $p_m = 1$, and $p_i = 0$ for $0 \leq i < m$. Then the output $x_o$ of Alg. 5.2 I have

$$
\mathbb{E}[\| \nabla f(x_o) \|] \leq \frac{2(1 - \lambda)}{\sqrt{\lambda}} \frac{\sqrt{2(f(x_0) - f(x^*))L\sigma}}{T}
$$

**Proof.** As the learning rate decay from 1 to $T$, I use Definition 5.2.1 to bound gradients $v_t^{t+1}$ as following:

$$
\mathbb{E}[\| v_t^{t+1} \|^2] = \mathbb{E}[\| (1 - \lambda)(\nabla f_t(x_t^{t+1}) - \nabla f_t(x^*)) + \lambda\nabla f(x^*) \|^2]
$$

$$
= \mathbb{E}[\| (1 - \lambda)\nabla f_t(x_t^{t+1}) - (1 - \lambda)\nabla f_t(x^*) + \lambda\nabla f(x^*) \|^2]
$$

$$
\leq 2(\mathbb{E}[\| (1 - \lambda)\nabla f_t(x_t^{t+1}) \|^2 + \| (1 - \lambda)\nabla f_t(x^*) - \lambda\nabla f(x^*) \|^2])
$$

$$
\leq 2((1 - \lambda)^2\mathbb{E}[\| \nabla f_t(x_t^{t+1}) \|^2] + (1 - \lambda)^2\mathbb{E}[\| \nabla f_t(x^*) \|^2])
$$

$$
\leq 4(1 - \lambda)^2\sigma^2,
$$

(5.24)
where the first inequality I followed Lemma 5.7.3 when $r=2$. The second inequality I followed (a) $\sigma$-bounded gradient property of $f$ and (b) the fact that for a random variable $\zeta$ which has a upper bounding as

$$
\mathbb{E}[\| (1 - \lambda)\zeta - \lambda \mathbb{E}[\zeta] \|^2] = \mathbb{E}[(1 - \lambda)^2 \| \zeta \|^2 - 2(1 - \lambda)\lambda \zeta \mathbb{E}[\zeta] + \lambda^2 \mathbb{E}^2[\zeta] \\
= (1 - \lambda)^2 \mathbb{E}[\| \zeta \|^2] - (2\lambda - 3\lambda^2)\mathbb{E}^2[\zeta] \\
\leq (1 - \lambda)^2 \mathbb{E}[\| \zeta \|^2],
$$

(5.25)

where the inequality should satisfy a condition that $0 \leq \lambda \leq \frac{2}{3}$. Thus, when the $\lambda$ in the range $0 \leq \lambda \leq \frac{2}{3}$, which can prove my method that can be converged in finite-sum training iterations by theoretical way.

Since $f$ is $\mathcal{L}$-smooth, I have

$$
\mathbb{E}[f(x_{t+1}^s)] \leq \mathbb{E}[f(x_{t}^s)] + \langle \nabla f(x_{t}^s), x_{t+1}^s - x_{t}^s \rangle + \frac{L}{2} \| x_{t+1}^s - x_{t}^s \|^2.
$$

(5.26)

Using Alg. 5.2 to update and since $\mathbb{E}[\langle \nabla f(x_{t}^s), x_{t+1}^s - x_{t}^s \rangle] = \mathbb{E}[(\lambda - 2)\| \nabla f(x_{t}^s) \|^2]$ (unbiasedness of the stochastic gradients when $t \to \infty$), Ineq. (5.26) would be updated as:

$$
\mathbb{E}[f(x_{t+1}^s)] \leq \mathbb{E}[f(x_{t}^s)] - \lambda \eta_{\Delta} \| \nabla f(x_{t}^s) \|^2 + \frac{L\eta_{\Delta}^2}{2} \| v_{t}^s \|^2.
$$

(5.27)

Adding the bound of $v_{t}^s$ from Ineq. (5.24) to Ineq. (5.27) I can obtain that:

$$
\mathbb{E}[f(x_{t+1}^s)] \leq \mathbb{E}[f(x_{t}^s)] - \lambda \eta_{\Delta} \mathbb{E}[\| \nabla f(x_{t}^s) \|^2] + \frac{L\eta_{\Delta}^2}{2} (4(1 - \lambda)^2)\sigma^2
$$

(5.28)

Thus the Ineq. (5.28) can be alternated as

$$
\mathbb{E}[\| \nabla f(x_{t}^s) \|^2] \leq \frac{1}{\eta_{\Delta} \lambda} \mathbb{E}[f(x_{t}^s) - f(x_{t+1}^s)] + \frac{L\eta_{\Delta}^2}{\lambda} (2(1 - \lambda)^2)\sigma^2,
$$

(5.29)

where $t \in \{0, \ldots, m - 1\}$, $s \in \{0, \ldots, S - 1\}$, $\Delta \in \{0, \ldots, T - 1\}$, and $T = mS$.

The minimum upper bound in Ineq. (5.30) can be achieved when $t = m - 1$ and $s = S - 1$, then I can obtain:

$$
\min_{t,s} \mathbb{E}[\| \nabla f(x_{t}^s) \|^2] \leq \frac{1}{T} \sum_{s=0}^{S-1} \sum_{t=0}^{m-1} \mathbb{E}[\| f(x_{t}^s) \|^2] + \frac{L\eta_{\Delta}^2}{\lambda} (2(1 - \lambda)^2)\sigma^2
\\ \leq \frac{1}{T \eta_{\lambda}} \mathbb{E}[f(x^0) - f(x^T)] + \frac{L\eta(2(1 - \lambda)^2)}{\lambda} \sigma^2
\\ \leq \frac{1}{T \eta_{\lambda}} (f(x^0) - f(x^*)) + \frac{L\eta(2(1 - \lambda)^2)}{\lambda} \sigma^2
$$

(5.30)

The first inequality can hold due to the minimum is less than average. The second inequality is achieved from Equation (5.29) and the third one is followed the fact that $f(x^*) \leq f(x^T)$. To calculate learning rate $\eta_{\Delta} = \eta$, I take the derivative of the last
Thus, the iterates in Alg. 5.2 satisfy the bound:

\[ \frac{1}{T} \left( f(x^0) - f(x^*) \right) + \frac{L\eta}{\lambda} (2(1-\lambda)^2 \sigma^2) \] 

Thus, \( \eta_\Delta = \eta = c/\sqrt{\Delta + 1} \), where \( c = \sqrt{f(x^0) - f(x^*) / 2\lambda L \sigma^2} \). Bring the result of \( \eta_\Delta = \eta = c/\sqrt{\Delta + 1} \) to Equation 5.30 I can achieve the upper bound of expectation as

\[ \min_{t,s} E[\| \nabla f(x^{s+1}_t) \|^2] \leq \frac{1}{\sqrt{T}} \left( \frac{1}{c\lambda} (f(x^0) - f(x^*)) + 2Lc\sigma^2 \right). \]

For the case that the learning rate depends on the data size \( n \), I provide one useful lemma in Lemma 5.7.2 firstly that can be used for proofing my Theorems.

**Lemma 5.7.2.** For \( c_t, c_{t+1}, \beta_t > 0 \), I have

\[ c_{\text{biased}} = c_{t+1} + \eta_t \beta_t + 2(1-\lambda)^2 \lambda_t^2 L^2 + L^2 \lambda_t^2 (1-\lambda)^2. \]

Let \( \eta_t, \beta_t \) and \( c_{t+1} \) is given so that the \( \Omega_t > 0 \) can be showed as

\[ \Omega_{\text{biased}} = \eta_t - \frac{c_{t+1} \eta_t}{\beta_t} - \lambda^2 L \eta_t^2 - 2\lambda^2 c_{t+1} \eta_t^2 \]

Thus, the iterates in Alg. 5.2 satisfy the bound:

\[ E[\| \nabla f(x^{s+1}_t) \|^2] \leq \frac{R_{t+1}^{s+1} - R_{t+1}^{s+1}}{\Omega_{\text{biased}}} \]

where \( R_{t+1}^{s+1} := E[f(x^{s+1}_t) + c_{\text{biased}} \| x^{s+1}_t - \tilde{x}^s \|^2] \) for \( 0 \leq s \leq S - 1 \).

**Proof.** To further bound the result in Ineq. 5.27 since \( f \) is \( L \)-smooth, I require to bound the intermediate iterate \( v^{s+1}_t \), which is showed following inequalities:

\[ E[\| v^{s+1}_t \|^2] \]

\[ = E[\| (1-\lambda)(\nabla f_t(x^{s+1}_t) - \nabla f_{t+1}(\tilde{x}^t)) + \lambda \nabla f(\tilde{x}^s) \|^2] \]

\[ = E[\| (1-\lambda)\zeta^{s+1}_t + \lambda \nabla f(\tilde{x}^s) - \lambda \nabla f(x^{s+1}_t) + \lambda \nabla f(x^{s+1}_t) \|^2] \]

\[ \leq 2E[\| \lambda \nabla f(x^{s+1}_t) \|^2] + 2E[\| (1-\lambda)\zeta^{s+1}_t - \lambda E[\zeta^{s+1}_t] \|^2] \]

\[ = 2\lambda^2 E[\| \nabla f(x^{s+1}_t) \|^2] + 2E \left[ (1-\lambda)^2 E[\zeta^{s+1}_t]^2 - (2\lambda - 3\lambda^2) E[\zeta^{s+1}_t]^2 \right] \]

\[ = 2\lambda^2 E[\| \nabla f(x^{s+1}_t) \|^2] + 2(1-\lambda)^2 E \left[ (\zeta^{s+1}_t)^2 - \frac{2\lambda - 3\lambda^2}{(1-\lambda)^2} E[\zeta^{s+1}_t]^2 \right] \]

\[ \leq 2\lambda^2 E[\| \nabla f(x^{s+1}_t) \|^2] + 2(1-\lambda)^2 E[\zeta^{s+1}_t]^2 \]

\[ \leq 2\lambda^2 E[\| \nabla f(x^{s+1}_t) \|^2] + 2(1-\lambda)^2 L^2 E[\| x^{s+1}_t - \tilde{x}^s \|^2], \]
In the first inequality, the variable $\zeta$ is showed as

$$
\zeta_{t+1}^{s+1} = \frac{1}{|I_t|} \sum_{i \in I_t} (\nabla f_i(x_t^{s+1}) - \nabla f_i(\tilde{x}^s)),
$$

(5.34)

since $E[\zeta_{t+1}^{s+1}] = \nabla f(x_t^{s+1}) - \nabla f(\tilde{x}^s)$. I can achieve the third equation based on the fact that $E[(1 - \lambda)Z - \lambda E[Z]^2] = E[Z][(1 - \lambda)Z - (2\lambda - 3\lambda^2)E[Z]^2]$. The second inequality is obtain from Ineq. 5.25.

Consider now the Lyapunov function:

$$
R_{t+1}^{s+1} := E[f(x_t^{s+1}) + c_t \cdot ||x_t^{s+1} - \tilde{x}^s||^2].
$$

To bound Equation 5.35 I require the bound of $E[||x_t^{s+1} - \tilde{x}^s||^2]$ as following:

$$
E[||x_t^{s+1} - \tilde{x}^s||^2]
= E[||x_t^{s+1} - x_t^{s+1} + x_t^{s+1} - \tilde{x}^s||^2]
= E[||x_t^{s+1} - x_t^{s+1}||^2 + 2||x_t^{s+1} - \tilde{x}^s||^2 + 2\langle x_t^{s+1} - x_t^{s+1}, x_t^{s+1} - \tilde{x}^s \rangle]
= E[\eta_t^2 ||v_t^{s+1}||^2 + 2||x_t^{s+1} - \tilde{x}^s||^2 - 2\eta_t E[\nabla f(x_t^{s+1}), x_t^{s+1} - \tilde{x}^s]]
\leq E[\eta_t^2 ||v_t^{s+1}||^2 + 2||x_t^{s+1} - \tilde{x}^s||^2 + 2\eta_t E \left[ \frac{1}{2\beta_t} \right] \nabla f(x_t^{s+1}) ||^2 + \frac{1}{2} \beta_t \cdot ||x_t^{s+1} - \tilde{x}^s||^2]
$$

(5.36)

The second equality follows from the unbiasedness of the update of Alg 5.2. The last inequality follows from application of Cauchy-Schwarz and Young’s inequality.

Combing Equation 5.33, Equation 5.35 and Equation 5.36, I can achieve the bound of $R_{t+1}^{s+1} := E[f(x_t^{s+1}) + c_{t+1} \cdot ||x_t^{s+1} - \tilde{x}^s||^2]$ as

$$
R_{t+1}^{s+1} \leq E[f(x_t^{s+1}) - \eta_t \cdot \nabla f(x_t^{s+1}) ||^2 + \frac{L\eta_t^2}{2} ||v_t^{s+1}||^2]
+ E[c_{t+1} \cdot \eta_t^2 ||v_t^{s+1}||^2 + c_{t+1} ||x_t^{s+1} - \tilde{x}^s||^2]
+ 2c_{t+1} \cdot \eta_t \cdot E \left[ \frac{1}{2\beta_t} \right] \nabla f(x_t^{s+1}) ||^2 + \frac{1}{2} \beta_t \cdot ||x_t^{s+1} - \tilde{x}^s||^2
\leq E[f(x_t^{s+1}) - (\eta_t - \frac{c_{t+1} \cdot \eta_t}{\beta_t}) \cdot \nabla f(x_t^{s+1}) ||^2]
+ \left(\frac{L\eta_t^2}{2} + c_{t+1} \cdot \eta_t^2 \right) E[||v_t^{s+1}||^2] + (c_{t+1} + c_{t+1} \cdot \eta_t) E[||x_t^{s+1} - \tilde{x}^s||^2]
= E[f(x_t^{s+1}) - (\eta_t - \frac{c_{t+1} \cdot \eta_t}{\beta_t} - \lambda^2 L\eta_t^2 - 2\lambda^2 c_{t+1} \cdot \eta_t^2) E[||\nabla f(x_t^{s+1})||^2] + (c_{t+1} + 1 + \eta_t \cdot \beta_t + (1 - \lambda)^2 \eta_t^2 L^2) + (1 - \lambda)^2 L^3 \eta_t^2) E[||x_t^{s+1} - \tilde{x}^s||^2]
\leq R_{t+1}^{s+1} - (\eta_t - \frac{c_{t+1} \cdot \eta_t}{\beta_t} - \lambda^2 L\eta_t^2 - 2\lambda^2 c_{t+1} \cdot \eta_t^2) E[||\nabla f(x_t^{s+1})||^2].
$$

(5.37)

The last inequality follows $R_{t+1}^{s+1} := E[f(x_t^{s+1}) + c_t \cdot ||x_t^{s+1} - \tilde{x}^s||^2]$ where

$$
c_{t+1} = c_{t+1} + 1 + \eta_t \cdot \beta_t + 2(1 - \lambda)^2 \eta_t^2 L^2 + (1 - \lambda)^2 L^3 \eta_t^2.
$$

(5.38)
Thus the Ineq. 5.37 can be alternated as
\[ E[\|\nabla f(x^t_{s+1})\|^2] \leq \frac{R_{t+1}^{s+1} - R_s^{s+1}}{\Omega_{t+1}^{biased}}, \] (5.39)

where \( \Omega_{t+1}^{biased} = \eta_t - \frac{c_t+1\eta_t}{\beta_t} - \lambda^2 L\eta_t^2 - 2\lambda^2 c_t+1\eta_t^2 \).

**Proof of Theorem 5.4.5**

**Theorem.** Let \( f \in F_n \), let \( c_m = 0, \eta_t = \eta > 0, \beta_t = \beta > 0, c_{t+1}^{biased} = c_t+1(1+\eta\beta + 2(1-\lambda)\eta^2 L^2) + L^2\eta^2(1-\lambda)^2 \), so the intermediate result \( \Omega_{t+1}^{biased} = (\eta_t - \frac{c_t+1\eta_t}{\beta_t} - \lambda^2 L\eta_t^2 - 2\lambda^2 c_t+1\eta_t^2) > 0 \), for \( 0 \leq t \leq m-1 \). Define the minimum value of \( \gamma_{t+1}^{biased} = \min_t \Omega_{t+1}^{biased} \). Further let \( p_i = 0 \) for \( 0 \leq i < m \) and \( p_m = 1 \), and \( T \) is a multiple of \( m \). So the output \( x_a \) of Alg. 5.2 I have
\[ E[\|\nabla f(x_a)\|^2] \leq \frac{f(x^0) - f(x^*)}{T\gamma_m^{biased}}, \]
where \( x^* \) is the optimal solution to Problem 5.1.

**Proof.** Using the result from Lemma 5.7.2 and \( \eta_t = \eta \) when \( t \in \{0, \ldots, m-1\} \), I can achieve the following bound:
\[ \sum_{t=0}^{m-1} E[\|\nabla f(x^t_{s+1})\|^2] \leq \frac{R_0^{s+1} - R_m^{s+1}}{\gamma_m^{biased}}, \] (5.40)

Thus, the bound in Ineq. 5.40 can updated as
\[ \sum_{t=0}^{m-1} E[\|\nabla f(x^t_{s+1})\|^2] \leq \frac{E[f(\tilde{x}^s) - f(\tilde{x}^{s+1})]}{\gamma_m^{biased}}, \] (5.41)

where \( R_0^{s+1} = E[f(\tilde{x}^s)] \) since \( x_0^{s+1} = \tilde{x}^s \) and \( R_m^{s+1} = E[f(\tilde{x}^{s+1})] \) since \( x_m^{s+1} = \tilde{x}^{s+1} \), which I use the condition that \( c_m = 0, p_m = 1, \) and \( p_i = 0 \) for \( i < m \). For the total number of iterations \( T = Sm \), I further sum up iteration \( s \) as
\[ \frac{1}{T} \sum_{s=0}^{S-1} \sum_{t=0}^{m-1} E[\|\nabla f(x^t_{s+1})\|^2] \leq \frac{f(x^0) - f(x^*)}{T\gamma_m^{biased}}, \] (5.42)

where the \( \tilde{x}^0 = x^0 \) and \( \tilde{x}^s = x^s \). Thus, I can obtain my final result. \( \square \)
Chapter 5 A Stochastic Gradient Method with Hybrid Estimation

Proof of Theorem 5.4.6

Theorem. Suppose \( f \in F_n \), let \( \eta = \frac{1}{3L_n^{2\alpha}} \) (0 \( \leq a \leq 1 \) and 0 \( < \alpha \leq 1 \)), \( \beta = \frac{L}{n^{2\alpha}} \) (b > 0), \( m_{biased} = \lfloor \frac{3n^{2\alpha}}{2(1-\lambda)} \rfloor \) and \( T \) is the total number of iterations. Then, I can obtain the lower bound \( \gamma_{n_{biased}} \geq \frac{(1-\lambda)\lambda \nu_1}{\lambda(1-\lambda)T\nu_1} \) in Theorem 5.4.5. For the output \( x_a \) of Alg. 5.2 I have

\[
\mathbb{E}[\|\nabla f(x_a)\|^2] \leq \frac{9Ln^{(2a-b)\alpha}f(x^0) - f(x^*)}{\lambda(1-\lambda)T\nu_1},
\]

where \( x_* \) is an optimal solution to Equation 5.1.

Proof. Using the relation in Equation 5.38 and \( c_m = 0 \), I estimated the upper bound of \( c_0 \) as

\[
c_0 = L^3 \eta^2 (1-\lambda)^2 \frac{(1+\theta_{biased})^m - 1}{\theta_{biased}}, \tag{5.43}
\]

where \( \theta_{biased} = 2(1-\lambda)^2L^2\eta^2 + \eta\beta \). Let \( \eta = \frac{1}{3L_n^{2\alpha}} \) and \( \beta = \frac{L}{n^{2\alpha}} \), the \( \theta_{biased} \) can be alternated as:

\[
\theta_{biased} = 2(1-\lambda)^2L^2\eta^2 + \eta\beta = \frac{2(1-\lambda)^2}{9n^{2\alpha}} + \frac{1}{3n^{(a+b)\alpha}} \tag{5.44}
\leq \frac{2(1-\lambda)}{3n^{2\alpha}}.
\]

Using the above bound \( \theta \), I can get the further bound of \( c_0 \) as

\[
c_0 = \frac{(1-\lambda)^2L[(1+\theta_{biased})^m - 1]}{2(1-\lambda)^2 + \frac{3}{n^{(b-a)\alpha}}} \leq \frac{L(1-\lambda)^2(e-1)}{3n^{(a-b)\alpha}}, \tag{5.45}
\]

where \( 0 \leq \mu_0 \leq 1 \) and \( n \geq 1 \). In the first inequality, due to the value of \( (1+\theta_{biased})^{m_{biased}} \) is increasing when \( m_{biased} = \lfloor \frac{1}{\theta} \rfloor > 0 \), I can use \( \lim_{t\to\infty}(1 + \frac{1}{t})^t = e \) (the \( e \) is Euler’s number) to calculate upper bound of \( (1+\theta_{biased})^{m_{biased}} \). Next, the lower bound of \( \gamma_{n_{biased}} \) is given as:

\[
\gamma_{n_{biased}} = \min_t (\eta - \frac{c_0 + 1}{\beta} - \lambda^2 L\eta^2 - 2\lambda^2 c_0 \eta^2) \geq (\eta - \frac{c_0 \eta}{\beta} - \lambda^2 L\eta^2 - 2\lambda^2 c_0 \eta^2) \geq \frac{(1-\lambda)\lambda \nu_1}{\lambda L n^{(2a-2b)\alpha}}, \tag{5.46}
\]

where \( \nu_1 \) is independent of \( n \). According to Theorem 5.4.5 I can achieve my result. \( \square \)
Lemma 5.7.3. (Reddi et al. [2016a]). For random variables $z_1, \ldots, z_r$, I have

$$
\mathbb{E} [ \| z_1 + \ldots + z_r \|^2 ] \leq r \mathbb{E} [ \| z_1 \|^2 + \ldots + \| z_r \|^2 ].
$$

(5.47)
Chapter 6

A Variance Controlled Stochastic Method with Biased Estimation

In this chapter, I proposed a new technique, variance controlled stochastic gradient (VCSG) to further improve the performance of ISVRG$^+$ algorithm. Particularly, apart from the $\lambda$ that has been introduced in last chapter for controlling the reduced variance of SVRG and balancing the trade-off between unbiased and biased estimations, I focus on minimising the number of full gradient calculations in SVRG, a variance-bounded batch that will be introduced to reduce the number of gradient calculations required in each iteration. For smooth non-convex functions, the proposed algorithm converges to an approximate first-order stationary point (i.e. $E \| \nabla f(x) \|^2 \leq \epsilon$) within $O(\min\{1/\epsilon^{5/4}, n^{1/8}/\epsilon\})$ number of stochastic gradient evaluations, which improves the best known gradient complexity of stochastic gradient based method SPIDER $O(\min\{1/\epsilon^{3/2}, n^{1/2}/\epsilon\})$. It is shown theoretically and experimentally that VCSG can be deployed to improve convergence.

6.1 Introduction

I still study smooth non-convex optimization problems which is shown in Equation 5.1. Many earlier works have focused on the asymptotic performance of algorithms (Gaivoronski 1994; Bertsekas 1997; Tseng 1998) and non-asymptotic complexity bounds have emerged (Lei et al. 2017b). To my knowledge, the first non-asymptotic convergence for stochastic gradient descent (SGD) was proposed by (Ghadimi and Lan 2016) with $O(1/\epsilon^2)$. Full batch gradient decent (GD) is known to ensure convergence with $O(n/\epsilon)$. Compared with SGD, the rate of GD has better dependence on $\epsilon$ but worse dependence on $n$ due to the requirement of computing a full gradient. Variance reduced (VR) methods based on SGD, e.g. Stochastic Variance Reduced Gradient (SVRG) (Johnson and Zhang 2013), SAGA (Defazio et al. 2014) have been shown to achieve better dependence on $n$ than GD on non-convex problems with $O(n + (n^{2/3}/\epsilon))$ (Reddi et al. 2016; Reddi 2019).
However, compared with SGD, the rate of VR based methods still have worse dependence on $\epsilon$ unless $\epsilon \ll n^{-2/3}$. Recently, Lei et al. (2017b) proposed a method called SCSG combining the benefits of SGD and SVRG, which is the first algorithm that achieves a better rate than SGD and is no worse than SVRG with $O(1/\epsilon^{5/3} \wedge n^{2/3}/\epsilon)$ ($(a \wedge b)$ means $\min(a, b)$). SNVRG proposed by Zhou et al. (2018) uses nested variance reduction to reduce the result of SCSG to $O((1/\epsilon^{3/2}) \log(\epsilon^{-1}) \wedge (n^{1/2}/\epsilon) \log(n))$ that outperforms both SGD, GD and SVRG. The rate of SNVRG can be further reduced to $O(1/\epsilon^{3/2} \wedge n^{1/2}/\epsilon)$ by a method called SPIDER which is proposed by Fang et al. (2018). To the best of my knowledge, this is a leading result of gradient complexity for smooth non-convex optimization. Their work motivates the research question about whether an algorithm based on SGD and VR-based methods can further reduce the rate of SPIDER when it depends on $\epsilon$ in the regime of modest target accuracy, and depends on $n$ in the regime of high target accuracy.

For SGD and VR-based stochastic algorithms, there are three challenges. Firstly, they do not require a full gradient computation as in SVRG method. As a result, SCSG, SNVRG, SPIDER reduce the full batch-size from $O(n)$ to its subset as $O(B)$ where $B < n$, which can significantly reduce the computational cost. However, there exists a challenge of how to appropriately scale the subset of samples in each stage of optimization to accelerate the convergence and also achieve the same accuracy with full samples. Secondly, the variance of SGD is reduced by VR methods since the gradient of SGD is often too noisy to converge. However, VR schemes reduce the ability to escape local minima in later iterations due to a diminishing variance (Bi and Gunn, 2019). The challenge of SGD and VR methods is therefore to control the variance of gradients. Lastly, there exists a trade-off between biased/unbiased estimation in VR-based algorithms. SVRG is an unbiased estimation that can guarantee to converge, but is not efficient to be used in real-world applications. Biased estimation can give a lower upper bound of the mean squared error (MSE) loss function (Liang et al., 2009), and many works have proposed asymptotically biased optimization with biased gradient estimators as an economic alternative to an unbiased version. These do not converge to the minima, but to their vicinity (Chen et al., 1987; Chen and Luss, 2018; Chen et al., 2018; Chen and Gao, 1989). These methods provide a good insight into the biased gradient search, however they hold under restrictive conditions which are very hard to verify for complex stochastic gradient algorithms. Thus, the last challenge is how to balance the unbiased and biased estimator in different stages of the optimization process on non-convex problems.

To address these three challenges, I propose my method Variance Controlled Stochastic Gradient (VCSG) which can control the reduced variance of the subset of gradients and choose the biased or unbiased estimator in each iteration to accelerate the convergence rate of non-convex optimization.

I summarize and list my main contributions:
• I provide a new method VCSG, a well-balanced VR method for SGD to achieve a competitive convergence rate. I also provide a theoretical analysis of my algorithm on non-convex problems. To the best of my knowledge, I provide the first analysis that the controlled variance reduction can achieve comparable or faster convergence than gradient-based optimization. Table 6.1 compares the theoretical rates of convergence of seven methods, which shows that VCSG has the fastest rate of convergence. I show empirically that VCSG has faster rates of convergence than GD, SGD, SVRG and SCSG. In this table, the IFO calls of SNVRG use $\tilde{O}(\cdot)$ to hide logarithmic factors, which equals $\tilde{O}((1/\epsilon^{3/2}) \log(\epsilon^{-1}) \land (n^{1/2}/\epsilon) \log(n))$

• VCSG provides an appropriate size of sample in each iteration by the controlled variance reduction, which can significantly save computational cost.

• VCSG balances the trade-off between biased and unbiased estimation, which can provide a fast convergence rate.

Table 6.1: Comparison of results on IFO calls (defined in Definition 4.3.2) of gradient methods for smooth non-convex problems.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>IFO calls on Non-convex</th>
<th>Batch size $B$</th>
<th>Learning rate $\eta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$O(n/\epsilon)$</td>
<td>$n$</td>
<td>$O(L^{-1})$</td>
</tr>
<tr>
<td>SGD</td>
<td>$O(1/\epsilon^2)$</td>
<td>$n$</td>
<td>$O(L^{-1})$</td>
</tr>
<tr>
<td>SVRG</td>
<td>$O(n + (n^{2/3}/\epsilon))$</td>
<td>$n$</td>
<td>$O(L^{-1}n^{-2/3})$</td>
</tr>
<tr>
<td>SCSG</td>
<td>$O(1/\epsilon^{5/3} \land n^{2/3}/\epsilon)\land B(B &lt; n)$</td>
<td>$B(B &lt; n)$</td>
<td>$O(L^{-1}(n^{-2/3} \land \epsilon^{4/3}))$</td>
</tr>
<tr>
<td>SNVRG</td>
<td>$O((1/\epsilon^{3/2}) \land (n^{1/2}/\epsilon))\land B(B &lt; n)$</td>
<td>$B(B &lt; n)$</td>
<td>$O(L^{-1})$</td>
</tr>
<tr>
<td>SPIDER</td>
<td>$O(1/\epsilon^{3/2} \land n^{1/2}/\epsilon)\land B(B &lt; n)$</td>
<td>$B(B &lt; n)$</td>
<td>$O(\epsilon L^{-1})$</td>
</tr>
<tr>
<td>VCSG</td>
<td>$O(1/\epsilon^{5/4} \land n^{1/8}/\epsilon)\land B(B &lt; n)$</td>
<td>$B(B &lt; n)$</td>
<td>$O(L^{-1} \land L^{-1}B^{-1/4})$</td>
</tr>
</tbody>
</table>

6.2 Preliminaries

I follow part of the work in SCSG. Based on their algorithm settings, I denote that a random variable $N$ has a geometric distribution $N \sim Geom(\gamma)$ if $N$ is supported on the non-negative integrates, which their elementary calculation has been shown as

$$E_{N \sim Geom(\gamma)} = \frac{\gamma}{1 - \gamma},$$

(6.1)

To show results in my method, I define the lower bound of complexity as:

$$f^* = \inf_x f(x), \quad \Delta_f = f(\tilde{x}_0) - f^*.$$  

(6.2)

An upper bound of the variance of the stochastic gradients can be defined as:

$$S^* = \sup_x \frac{1}{n} \sum_{i=1}^{n} \| \nabla f_i(x) - \nabla f(x) \|^2.$$  

(6.3)
For my analysis, the background that are required to introduce definitions for $L$-smooth, $\epsilon$-accuracy and computational cost IFO which have been defined in Definition 4.1.1 Definition 4.3.1 and Definition 4.3.2 respectively.

6.3 Variance controlled SVRG with a combined unbiased/biased estimation

To resolve the first challenge of SG-based optimization, I provide an adjustable schedule of batch size $B < n$, which scales the sample size for optimization. In the second challenge, the balance of the gradient update between the full batch and stochastic estimators is fixed. One method [Bi and Gunn, 2019] balanced the gradient of SVRG in terms of the stochastic element and its variance to allow the algorithm to choose appropriate behaviours of gradient from stochastic, through reduced variance, to batch gradient descent by introducing a hyper-parameter $\lambda$. Based on this method, I focus on the $\lambda$ for the subset of full gradients. Towards the last challenge associated with the trade-off between biased and unbiased estimator, I analyse the nature of biased and unbiased estimators in different stages of the non-convex optimization, and propose a method combining the benefits of both biased and unbiased estimator to achieve a fast convergence rate. Firstly, I show a generic form of batched SVRG in Alg 6.1 which is proposed by [Lei et al., 2017b]. Compared with SVRG algorithm showed in Alg. 3.1 the batched SVRG algorithm has a mini-batch procedure in the inner loop and outputs a random sample instead of an average of the iterates. As seen in the Alg 6.1 batched SVRG method consists of multiple epochs, the batch-size $B_j$ is randomly chosen from the whole samples $n$ in $j$-th epoch and work with mini-batch $b_j$ to generate the total number of updates for inner $k$-th epoch by a geometric distribution with mean equal to the batch size. Finally it outputs a random sample from $\{\tilde{x}_j\}_{j=1}^T$. This is a standard way also proposed by [Nemirovski et al., 2009], which can save additional overhead by calculating the minimum value of output as $\arg\min_{j \leq T} \|\nabla f(\tilde{x}_j)\|$. By Eq. 6.1 the average total cost provided by [Lei et al., 2017b] is

$$
\sum_{j=1}^{T} (B_j + b_j E N_j) = \sum_{j=1}^{T} (B_j + b_j E N_j) = 2 \sum_{j=1}^{T} B_j.
$$

(6.4)

Define $T(\epsilon)$ is the minimum number of epochs, then the all outputs are $\epsilon$-accurate which is defined in 4.3.2 if

$$
T(\epsilon) = \min T : E \|\nabla f(\tilde{x}_{T'})^2\| \leq \epsilon \text{ for all } T' \geq T.
$$

(6.5)
We then use $C_{\text{comp}}(\epsilon)$ to present IFO cost, and IFO complexity to reach an $\epsilon$-accurate solution for Alg. 6.1 is

$$\mathbb{E}C_{\text{comp}}(\epsilon) \leq 2 \sum_{j=1}^{T(\epsilon)} B_j.$$  \hfill (6.6)

**Algorithm 6.1: Batching SVRG**

**input:** Number of epochs $T$, step-size $(\eta_j)_{j=1}^T$, batch size $(B_j)_{j=1}^T$, mini-batch sizes $(b_j)_{j=1}^T$

1. for $j = 0$ to $T$
   2. Uniformly sample a batch $I_j \subset \{1, \ldots, n\}$ with $|I_j| = B_j$;
   3. $g_j \leftarrow \nabla f_{I_j}(\tilde{x}_{j-1})$;
   4. $\tilde{x}^{(j)}_0 \leftarrow \tilde{x}_{j-1}$
   5. Generate $N \sim \text{Geom}(B_j/(B_j + b_j))$;
   6. for $k = 1$ to $N_j$
      7. Randomly select $I_{k-1} \subset \{1, \ldots, n\}$ with $|I_{k-1}| = b_j$;
      8. $v_{k-1}^{(j)} = \nabla f_{I_{k-1}}(x^{(j)}_{k-1}) - \nabla f_{\tilde{I}_{k-1}}(x^{(j)}_0) + g_j$; $x_k^{(j)} = x_{k-1}^{(j)} - \eta_j v_{k-1}^{(j)}$;
   9. $\tilde{x}_j \leftarrow x^{(j)}_{N_j}$;

**output:** Sample $\tilde{x}_T^*$ from $\{\tilde{x}_j\}_{j=1}^T$ with $P(\tilde{x}_T^* = \tilde{x}_j) \propto \eta_j B_j / b_j$

For the cases of unbiased/biased estimations for batched SVRG, I provide upper bounds on their convergence for their gradients in the following two sub-sections. Proof details are presented in the appendix.

### 6.3.1 Weighted unbiased estimator with one-epoch analysis

In the first case, I introduce a hyper-parameter $\lambda$ that is applied in a weighted unbiased version of batched SVRG, and is shown in Alg 6.2. Since my method based on SVRG, the $\lambda$ should be within the range $0 < \lambda < 1$ in both unbiased and biased cases. For a single epoch, $j$, I define the weighted unbiased variance as

$$e_j = \lambda \left( \nabla f_{I_j}(\tilde{x}_{j-1}) - \nabla f(\tilde{x}_{j-1}) \right).$$  \hfill (6.7)

Thus, the gradients in Alg 6.2 can be updated within the $j$-th epoch as

$$\mathbb{E}_{I_k} v_k^{(j)} = (1 - \lambda) \nabla f(x_k^{(j)}) + e_j.$$  \hfill (6.8)

This reveals the key difference between batched SVRG and variance controlled batched SVRG on both unbiased and biased estimators. Most of the novelty in my analysis lies in dealing with the extra term $e_j$. 
Algorithm 6.2: Batching SVRG

**input**: Number of epochs $T$, step-size $(\eta_j)_{j=1}^T$, batch size $(B_j)_{j=1}^T$, mini-batch sizes $(b_j)_{j=1}^T$

1. for $j = 0$ to $T$ do
   2. Uniformly sample a batch $\mathcal{I}_j \subset \{1, \ldots, n\}$ with $|\mathcal{I}_j| = B_j$;
   3. $g_j \leftarrow \nabla f_{\mathcal{I}_j}(\tilde{x}_{j-1})$;
   4. $\tilde{x}_0(j) \leftarrow \tilde{x}_{j-1}$
   5. : Generate $N \sim \text{Geom}(B_j/(B_j + b_j))$;
   6. for $k = 1$ to $N_j$ do
      7. Randomly select $\mathcal{I}_k - 1 \subset \{1, \ldots, n\}$ with $|\tilde{\mathcal{I}}_k| = b_j$;
      8. $v_{k-1}^{(j)} = (1 - \lambda)\nabla f_{\tilde{\mathcal{I}}_{k-1}}(x_{k-1}) - \lambda (\nabla f_{\tilde{\mathcal{I}}_{k-1}}(x_0) - g_j)$;
      9. $x_k^{(j)} = x_{k-1}^{(j)} - \eta_j v_{k-1}^{(j)}$
   10. $\tilde{x}_j \leftarrow x_{N_j}^{(j)}$

**output**: Sample $\tilde{x}_T^*$ from $\{\tilde{x}_j\}_{j=1}^T$ with $P(\tilde{x}_T^* = \tilde{x}_j) \propto \eta_j B_j/b_j$

By bounding the term $e_j$, we can achieve a lower bound of batch-size. Following (Babanezhad et al., 2015), the variance of the norms of gradients $\mathcal{K}^2$ is given as

$$\frac{1}{n - 1} \sum_{i=1}^{n} \left[ \| \nabla f_i(\tilde{x}_{j-1}) \|^2 - \| \nabla f(\tilde{x}_{j-1}) \|^2 \right] \leq \mathcal{K}^2,$$

then we have that (L. Lohr, 2000)

$$\mathbb{E}_{\mathcal{I}_j} \| e_j \|^2 \leq \lambda^2 \frac{n - B_j}{n B_j} \mathcal{K}^2.$$

If the upper bound of $\mathbb{E}_{\mathcal{I}_j} \| e_j \|^2 \leq \sigma \rho^{2j}$ (Babanezhad et al., 2015), where $\sigma \geq 0$ is a constant for some $\rho < 1$, the $B_j$ should satisfy

$$B_j \geq \frac{n \mathcal{K}^2}{\mathcal{K}^2 + n \lambda^2 \sigma \rho^{2j}}.$$

Based on the Samuelson inequality (Niezgoda, 2007) that $\mathcal{K}^2 \leq \frac{n}{\sqrt{n - 1}} S^*$, Inq 6.11 can be further bounded in following theorem.

**Theorem 6.3.1.** If $\mathbb{E}_{\mathcal{I}_j} \| e_j \|^2 \leq \sigma \rho^{2j}$ in Alg 6.2 ($\sigma \geq 0$ is a constant for some $\rho < 1$), the lower bound of batch-size $B_j$ can be achieved as following,

$$B_j \geq \frac{n S^*}{S^* + \lambda^2 \sigma \rho^{2j}}.$$

Apart from the batch size, the schedules of the learning rate and mini-batch size also effect the performance. I use two general formats of schedule in terms of $\eta_j$ and $b_j$ to calculate the best schedules in each stage of optimization for unbiased and biased
estimators. Under such parameter settings, I estimated the upper bound of expectation of gradients in a single epoch and achieved the result in Theorem 6.3.2.

**Theorem 6.3.2.** Let \( \eta_j L = \gamma \left( \frac{b_j}{B_j} \right)^\alpha \) \((0 \leq \alpha \leq 1)\) and \( \gamma \geq 0 \). Suppose \( B_j \geq b_j \geq B_j^\beta \) \((0 \leq \beta \leq 1)\) for all \( j \), then under Definition 4.1.1, the output \( \tilde{x}_j \) of Alg 6.2 we have

\[
\mathbb{E}\|\nabla f(\tilde{x}_j)\|^2 \leq \frac{2L}{\gamma} \left( \frac{b_j}{B_j} \right)^{1-\alpha} \mathbb{E}(f(\tilde{x}_{j-1}) - f(\tilde{x}_j)) + 2\lambda I(B_j < n) \frac{S^*}{B_j^{1-\alpha}},
\]

where \( 0 < \lambda < 1 \) and \( 2(1-\lambda) - (2\gamma B_j^{\alpha\beta-\alpha} + 2B_j^{\beta-1})(1-\lambda)^2 - 1.16(1-\lambda)^2 \) is positive when \( B_j \geq 3 \) and \( 0 \leq \gamma \leq \frac{13}{50} \).

**Proof Sketch:** Combine two equations in Lemma 6.6.5 and Lemma 6.6.6, we can achieve a upper bound of unbiased version gradient in single epoch. And further use Lemma 6.6.4, the final result of Theorem 6.3.2 can be achieved.

Since, Lei et al. (2017a); Lei and Jordan (2017) determined the learning rate \( \eta \leq \frac{1}{3L} \) that \( \gamma \leq \frac{1}{3} \) which can guarantees the convergence in non-convex case. Thus \( \gamma \leq \frac{1}{3} \) is a upper bound that is considered in my two biased and unbiased cases. In unbiased cases, \( \gamma \leq \frac{13}{50} \leq \frac{1}{3} \).

### 6.3.2 Biased estimator on one-epoch analysis

In this section I theoretically analyze the performance of the biased estimator, which is shown in Alg 6.3. In this case I define the variance of the gradient as

\[
e_j = \lambda \nabla f_{\ell_j}(\tilde{x}_{j-1}) - (1-\lambda)\nabla f(\tilde{x}_{j-1}).
\]

Using the same approach adopted in the unbiased version, I achieve the bound of batch-size for the biased case, which is shown in the following theorem and corollary.

**Theorem 6.3.3.** If \( \mathbb{E}_{\ell_j} \|e_j\|^2 \leq \sigma \rho^2 j \) in Alg 6.3 \((\sigma \geq 0 \text{ is a constant for some } \rho < 1)\), the lower bound of batch-size \( B_j \) can be achieved as,

\[
B_j \geq \frac{nS^*}{S^* + (1-\lambda)^2n\frac{1}{2}\sigma \rho^2 j}
\]

Applying the same schedule of \( \eta_j \) and \( b_j \) that are used in the unbiased case, we can achieve the result of this case, which is shown in Theorem 6.3.4.
Algorithm 6.3: Batching SVRG\textit{biased}

\textbf{input}: Number of epochs $T$, step-size ($\eta_j$)\textsubscript{T} for $j = 1$, batch size ($B_j$)\textsubscript{T} for $j = 1$, mini-batch sizes ($b_j$)\textsubscript{T} for $j = 1$.

1. for $j = 0$ to $T$ do
   2. Uniformly sample a batch $I_j \subset \{1, ..., n\}$ with $|I_j| = B_j$;
   3. $g_j \leftarrow \nabla f_{I_j}(\tilde{x}_{j-1})$;
   4. $\tilde{x}_0^{(j)} \leftarrow \tilde{x}_{j-1}$
   5. Generate $N \sim \text{Geom}(B_j/(B_j + b_j))$;
   6. for $k = 1$ to $N_j$ do
      7. Randomly select $I_k^{(j-1)} \subset \{1, ..., n\}$ with $|I_k^{(j-1)}| = b_j$;
      8. $v_k^{(j)} = (1 - \lambda) \left( \nabla f_{I_k^{(j-1)}}(x_k^{(j-1)} - \nabla f_{x_k^{(j-1)}}(x_0^{(j)})) + \lambda g_j \right)$;
      9. $x_k^{(j)} = x_k^{(j-1)} - \eta_j v_k^{(j)}$;
   10. $\tilde{x}_j \leftarrow x_N^{(j)}$;

\textbf{output}: Sample $\tilde{x}_T$ from $\{\tilde{x}_j\}_{j=1}^T$ with $P(\tilde{x}_T = \tilde{x}_j) \propto \eta_j B_j / b_j$

\textbf{Theorem 6.3.4.} let $\eta_j L = \gamma (b_j / B_j)^\alpha$ ($0 \leq \alpha \leq 1$) and $\gamma \leq \frac{1}{3}$. Suppose $B_j \geq b_j \geq B_j^\beta$ ($0 \leq \beta \leq 1$) for all $j$, then under Definition 4.1.1 the output $\tilde{x}_j$ of Alg 6.3 we have,

$$\mathbb{E} \| \nabla f(\tilde{x}_j) \|^2 \leq \frac{2L}{\gamma} (\frac{b_j}{B_j})^{1-\alpha} \mathbb{E}(f(\tilde{x}_{j-1}) - f(\tilde{x}_j)) + 2(1 - \lambda)^2 \frac{I(B_j \leq n)}{B_j^{1-4\alpha}} S^*$$

$$\frac{2(1 - \lambda) - (2\gamma B_j^{\alpha - \alpha} + 2B_j^{3-1} - 4LB_j^{2\alpha - 2})(1 - \lambda)^2 - 1.16(1 - \lambda)^2}{2},$$

where $0 < \lambda < 1$.

\textbf{Proof Sketch:} Combine two equations in Lemma 6.6.10 and Lemma 6.6.11 we can achieve a upper bound of biased version gradient in single epoch. And further use Lemma 6.6.9 the final result of Theorem 6.3.2 can be achieved.

6.3.3 Convergence analysis for smooth non-convex objectives

Over all epochs $T$, the output $\tilde{x}_T^*$ that is randomly selected from $\{\tilde{x}_j\}_{j=1}^T$ should be non-convex and $L$-smooth. Combining two cases of upper bound for unbiased/biased estimation in Theorem 6.3.2 and Theorem 6.3.4 respectively, we can achieve the following results.

\textbf{Theorem 6.3.5.} Under Definition 4.1.1 and Theorem 6.3.2, Theorem 6.3.4 the output $\tilde{x}_T^*$ can achieve two results based on two versions of the estimator.
• If using the unbiased estimator, Alg. 6.2, the output $\tilde{x}^*_T$ can be bounded using,

$$\mathbb{E} \| \nabla f(\tilde{x}^*_T) \|^2 \leq \frac{(2L_\gamma)\triangle f}{\theta \sum_{j=1}^T b_j^{\alpha-1} B_j^{1-\alpha}} + \frac{2\lambda^4 I(B_j < n)S^*}{\theta B_j^{1-4\alpha}},$$

• If using the biased estimator, Alg. 6.3, the output $\tilde{x}^*_T$ can be bounded using,

$$\mathbb{E} \| \nabla f(\tilde{x}_j) \|^2 \leq \frac{(2L_\gamma)\triangle f}{\theta_{\text{biased}} \sum_{j=1}^T b_j^{\alpha-1} B_j^{1-\alpha}} + \frac{2(1-\lambda)^2 I(B_j < n)S^*}{\theta_{\text{biased}} B_j^{1-4\alpha}}.$$ 

Where $\theta = 2(1-\lambda) - (2\gamma B_j^{\alpha-\alpha} + 2B_j^{\beta-1})(1-\lambda)^2 - 1.16(1-\lambda)^2$ and $\theta_{\text{biased}} = 2(1-\lambda) - (2\gamma B_j^{\alpha-\alpha} + 2B_j^{\beta-1} - 4LB_j^{2\alpha-2})(1-\lambda)^2 - 1.16(1-\lambda)^2$. 

To achieve the best upper bounds of the two estimators in Theorem 6.3.5, it is possible to specify the optimal value of $\lambda$ as $\lambda^*$ with three schedules including step-size $\eta_j$, mini-batch size $b_j$ and batch size $B_j$. Since $\lambda^*$ is hard to obtain by optimizing both terms on the right side of equations in Theorem 6.3.5, this problem can be addressed after specifying these three schedules. As a result, I now determine these three schedules.

When $1 \leq B_j = B \leq n$ and $1 \leq b_j = B_j \leq B_j$, Theorem 6.3.5 implies that

$$\mathbb{E} \| \nabla f(\tilde{x}^*_T) \|^2 = \mathcal{O} \left( \frac{L\triangle f}{TB_j^{1+\alpha\beta-\alpha-\beta}} + \frac{S^*}{B_j^{1-4\alpha}} \right), \quad (6.13)$$

which covers two cases of complexity bounds due to the batch-size having two different dependencies.

1. Dependence on $\epsilon$: If $b_j = B_j^\beta$ ($0 < \beta \leq 1$) and $B_j = B$ for some $1 < B < n$, the second term of Equation 6.13 can be made $\mathcal{O}(\epsilon)$ by setting $B_j^{1-4\alpha} \geq \mathcal{O} \left( \frac{S^*}{\epsilon} \right)$ where $\alpha$ should satisfy the range $0 \leq \alpha < 1/4$. Under such a setting that $B_j$ depends on $\epsilon$, $T(\epsilon) = \left( \frac{L\triangle f}{\epsilon B_j^{1+\alpha\beta-\alpha-\beta}} \right)$ resulting in the complexity rate $\mathbb{E}_{\text{comp}}(\epsilon) = \mathcal{O} \left( \frac{L\triangle f B_j^{\beta+\alpha(1-\beta)}}{\epsilon} \right)$. Due to the range of $\alpha$, this rate has a range from $\mathcal{O} \left( \frac{L\triangle f S^*}{\epsilon^{1+\beta}} \right)$ to $\mathcal{O} \left( \frac{L\triangle f S^*}{\epsilon^2} \right)$, which can be equal or better than SGD as shown in Table 6.1. To achieve an optimal rate in this case, I now focus on the optimal value of $\beta$ which can effect the step size and mini-batch size. In the extreme case when $b_j = 1$ and $B_j = n$, the optimal schedule of step size provided by (Reddi et al., 2016a; Reddi et al., 2016; Allen Zhu and Hazan, 2016; Lei et al., 2017b) is similar in form but with different $\alpha = 2/3$. For my case $b_j = B_j^\beta > 1$ and $B_j < n,$
their step-size should be re-scaled as

$$\eta_j L = \gamma B_j^{2(\beta-1)}$$  \hspace{1cm} (6.14)

The step size showed in Equation (6.14) can be bounded by my range of step-size, which is shown in Equation (6.15). Thus, I can estimate the range of $\beta$ as $0 \leq \beta \leq 5/8$.

$$\frac{\gamma}{L} \geq \frac{\gamma}{L} \left( \frac{1}{B_j} \right)^{2(1-\beta)} \geq \frac{\gamma}{L} \left( \frac{1}{B_j} \right)^{1/2}$$  \hspace{1cm} (6.15)

2. Dependence on $n$: If $b_j = 1$ (when $\beta = 0$) and $B_j = B$ for some $1 < B < n$, Equation (6.13) implies that $E \| \nabla f(\tilde{x}_T^*) \|^2 = \mathcal{O} \left( \frac{L \Delta f}{TB^{1-\alpha}} + \frac{S^*}{B^{1-4\alpha}} \right)$. Since the value of $B_j$ depends on $n$, $T(\epsilon)$ can be made as $\mathcal{O} \left( 1 + \frac{L \Delta f}{\epsilon B^{1-\alpha}} \right) \leq \mathcal{O} \left( 1 + \frac{L \Delta_f}{\epsilon B^2} \right)$ where $0 \leq \alpha \leq 1/4$, which yields the complexity bound $E_{\text{comp}}(\epsilon) = \mathcal{O} \left( B + \frac{B^3 \Delta f}{\epsilon} \right)$. This upper bound of rate can guarantee to be better than SVRG, as shown in Table 6.1.

Based on the above analysis, I provide an adjusted batch-size that is the minimum value between two terms depending on $n$ and $\epsilon$. When $B_j$ depends on $\epsilon$, e.g. $B_j = S^*/\epsilon$, I use the schedule of $b_j = B_j^{5/8}$, $\eta_j \leq 1/(3L)$. Otherwise when $B_j$ depends on $n$, e.g. a schedule of batch size provided by Theorem 6.3.1 and Theorem 6.3.3, the best choice is $b_j = 1$ and $\eta_j = \frac{1}{3L} \left( \frac{1}{B_j} \right)^{1/2}$.

Now that I have determined the three schedules including $B_j$, $\eta_j$ and $b_j$, I can estimate the optimal value of $\lambda^*$. For the case of $B_j$ with the first term depending on $\epsilon$, $b_j = B_j^{5/8}$, $\eta_j = \frac{1}{3L}$. Equation (6.13) will be specified as $E \| \nabla f(\tilde{x}_T^*) \|^2 = \mathcal{O} \left( \frac{L \Delta f e^{3/8}}{TS^{3/8}} + \epsilon \right)$. In this case the norm of variance $S^*$ and $\Delta_f$ are relatively small, so the value of the second term in the above equation becomes more important than first term. Thus, I optimize the coefficient of the second term of $E \| \nabla f(\tilde{x}_T^*) \|^2$ in Theorem 6.3.5 in both unbiased/biased cases and select the optimal value of $\lambda^* = \frac{1}{16} (15 - \sqrt{97}) \approx 0.32$ in the unbiased estimation case. Otherwise, $B_j$ equals the second term depending on $n$, $b_j = 1$ and $\eta_j = \frac{1}{3L} \left( \frac{1}{B_j} \right)^{1/2}$,

Equation (6.13) is specified as $E \| \nabla f(\tilde{x}_T^*) \|^2 = \mathcal{O} \left( \frac{L \Delta_f}{TB^{3/4}} + S^* \right)$. In this case, $S^*$ is relatively large resulting in $\Delta_f$ becoming very large. Thus, the value of the first term is more important to minimize. As a result, I optimize the coefficient of first term of $E \| \nabla f(\tilde{x}_T^*) \|^2$ in Theorem 6.3.5 and achieve $\lambda^* = \frac{5}{8}$ in biased estimation.
6.3.4 Best of two worlds

We have seen in the previous section that the variance controlled SVRG combines the benefits of both SVRG and SGD. I now show these benefits can be made more pronounced by $\lambda^*$ and the three schedules in different stages of optimization. I introduce my algorithm VCSG shown in Alg 6.4. Compared with ISVRG$^+$ introduced in last chapter, we do not assume the variance of VCSG to be bounded by unbiased version since my algorithm can guarantee that VCSG uses unbiased estimator to converge the point in a real target by reduced and stable variance in the end. Following Alg 6.4, we can achieve a general

Algorithm 6.4: (Mini-Batch)VCSG

<table>
<thead>
<tr>
<th>input</th>
<th>Number of epochs $T$, step-size $(\eta_j)_j^T$, batch size $B_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$(B_j)_j^T = {S^<em>/\epsilon \wedge S^</em> + 0.08 \cdot n^{1/2} \sigma^j}$ where $\sigma \geq 0$, $\rho &lt; 1$, and mini-batch sizes $(b_j)_j^T$ for $j = 1$ to $T$ do</td>
</tr>
<tr>
<td></td>
<td>1 Uniformly sample a batch $I_j \subset {1, ..., n}$ with $</td>
</tr>
<tr>
<td></td>
<td>2 $g_j \leftarrow \nabla f_{I_j}(\tilde{x}_{j-1})$;</td>
</tr>
<tr>
<td></td>
<td>3 $\tilde{x}<em>j(0) \leftarrow \tilde{x}</em>{j-1}$;</td>
</tr>
<tr>
<td></td>
<td>4 Generate $N_j \sim \text{Geom}(B_j/(B_j + b_j))$;</td>
</tr>
<tr>
<td></td>
<td>5 for $k = 1$ to $N_j$ do</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>Sample $\tilde{x}_T$ from $(\tilde{x}_j)_j^T$ with $P(\tilde{x}_T = \tilde{x}_j) \propto \eta_j B_j/b_j$</td>
</tr>
</tbody>
</table>

result for VCSG in the following theorem.

**Theorem 6.3.6.** Suppose $\gamma \leq 1/3$, let $B_j = \min \left\{ S^*/\epsilon, S^* + 0.1 \cdot n^{1/2} \sigma^j \right\}$, under Definition 4.1.1 the output $\tilde{x}_T$ in Alg 6.4 satisfies one of two bounds.
• if \( B_j = \frac{S^*}{\epsilon} \), \( b_j = B_j^2 \), \( \eta_j = \frac{\gamma}{L} \) and \( \lambda^* = \frac{1}{16}(15 - \sqrt{97}) \) with an unbiased estimator, then
\[
E \| \nabla f(\tilde{x}_T) \|^2 \leq \frac{15.7L}{\epsilon} \frac{\Delta f}{T} \sum_{j=1}^{T} B_j^2 + \frac{0.17(I(B_j < n))S^*}{B_j},
\]

• if \( B_j = \frac{nS^*}{S^* + 0.1 \cdot n^\frac{1}{2}\sigma_j^2} \), \( b_j = 1 \), \( \eta_j = \frac{\gamma}{L}(\frac{1}{B_j})^{\frac{1}{4}} \) and \( \lambda^* = \frac{5}{8} \) with a biased estimator, then
\[
E \| \nabla f(\tilde{x}_T) \|^2 < \frac{5.3L}{\epsilon} \frac{\Delta f}{T} \sum_{j=1}^{T} B_j^2 + 0.75S^*.
\]

Now we discuss how these parameters including \( \lambda \), step-size, batch-size and mini-batch size work together to control the variance of gradients from stochastic to batch and balance the trade-off between bias/unbiased estimation in batched optimization. In very early iterations, the first term of \( B_j \) might be chosen due to the low variance. In this case, the small \( \lambda \) can increase the variance resulting in the batch-size being increased and the gradient being more stochastic to provide directions to explore other regions in the search space. For the second case the second term of \( B_j \) would be smaller than the first term due to the high variance \( S^* \), large \( \lambda \) and relatively small step-size; working with the biased estimator can fast reduce variance, which can accelerate the convergence.

On the other hand, when the variance is reduced too much by the first case, the first term of \( B_j \) becomes smaller, VCSG would change to the first case that the small value of \( \lambda \) allows the gradient to behave more stochastically. Higher variance will help points escape from local minima. When the variance becomes small and stable, the unbiased estimator can guarantee to find the objective point in the end.

Compared two upper bounds from Theorem 6.3.6 I found the unbiased bound is weaker than biased version if batch-size is chosen the second term. And also in the end of training, the algorithm choose the first term of batch-size that guarantees to use unbiased estimator to converge. As a result, I use unbiased upper bound from Theorem 6.3.6 to calculate the computational complexity of VCSG. As shown in Corollary 6.3.1 below, the complexity of VCSG (IFO) surpasses GD, SGD, SVRG, SCSG [Lei et al., 2017b] and SPIDER [Fang et al., 2018].

**Corollary 6.3.1.** **Under all parameters setting in Theorem 6.3.6 and \( B_j \equiv B = \min \left\{ \frac{S^*}{\epsilon}, \frac{nS^*}{S^* + 0.1 \cdot n^\frac{1}{2}\sigma^2_j} \right\} \) then it holds that**

\[
E_{comp}(\epsilon) = O \left( B + \frac{L \Delta f}{\epsilon} \cdot B^2 \right)
\]
\[
= O \left( \frac{S^*}{\epsilon} \wedge \frac{nS^*}{S^* + 0.1 \cdot n^\frac{1}{2}\sigma^2_j} \right) + \frac{L \Delta f}{\epsilon} \cdot \left( \frac{S^*}{\epsilon} \wedge \frac{nS^*}{S^* + 0.1 \cdot n^\frac{1}{2}\sigma^2_j} \right)^\frac{1}{2}.
\]
Assume that $L \Delta f, S^*, \sigma \rho^j = \mathcal{O}(1)$, the above bound can be simplified to

$$E_{\text{comp}}(\epsilon) = \mathcal{O}\left(\frac{1}{\epsilon} \wedge n^{\frac{3}{2}}\right) = \mathcal{O}\left(\frac{1}{\epsilon^2} \wedge \frac{n^{\frac{3}{2}}}{\epsilon}\right).$$

\[\text{(a)}\]

\[\text{(b)}\]

\[\text{(c)}\]

Figure 6.1: Comparison of rates of convergence in four approaches, including SGD, SVRG, SCSG and VCSG.
6.4 Application

To experimentally verify my theoretical results and insights, I evaluate VCSG compared with SVRG, SGD, and SCSG on three common DL topologies, including LeNet (LeNet-300-100 which has two fully connected layers as hidden layers with 300 and 100 neurons respectively, and LeNet-5 which has two convolutional layers and two fully connected layers) and VGG-16 (Simonyan and Zisserman, 2014b) using three datasets including MNIST, CIFAR-10 and tiny ImageNet. Tiny ImageNet contains 200 classes for training each with 500 images and the test set contains 10,000 images. Each image is re-sized to 64 × 64 pixels (Russakovsky et al., 2015). I initialize $B_j = B_0 = \frac{n}{2}$ (Babanezhad et al., 2015). Correspondingly the rest of key parameters can be specified as $b_j = b_0 = (\frac{n}{2})^{5/8}$, $\eta = \frac{1}{3L(B_0)^{\frac{1}{4}}}$, and $\lambda = \frac{5}{8}$ via using biased estimator in the first iteration. And I consider a practical setting $b_j = B_j^{5/8}$ in both cases during the whole training process, resulting in the value of $N_j$ becoming stable as $B_j^{3/8}$. Meanwhile, I choose a scaled SGD as my baseline when $\lambda = 0$ with decayed learning rate $\eta_j = \eta_0 / (j)$, and SVRG when $\lambda = 0.5$ with fixed learning rate $\eta_j = 1/(3Ln^{\frac{1}{4}})$ in Alg 6.1. The reason I choose SCSG is that my algorithm is inspired from SCSG which is a leading batched SVRG. Figure 6.1 compares the performance of four methods, including SGD, SVRG, SCSG and VCSG, via test error, training loss and training time usage. It has two baselines in all sub-figures, including the performance of SVRG and SGD. The performance of SCSG test error and training loss are smaller than SGD on MNIST and CIFAR-10 data sets, which is consistent with the experimental results shown in (Lei et al., 2017b). However, in the ImageNet data set which is a relatively larger scale application than the previous two data sets, the performance of SCSG becomes worse than SVRG and SGD, which showed a weak robustness in my experiments. By contrast, VCSG shown as the green color in all three datasets has the lowest test error and training loss among all methods. In the ImageNet data set both the test error of VCSG is initially higher than SVRG and SGD, but VCSG

![Figure 6.2: Visualization of test error of four approaches, including SGD, SVRG and SCSG and VCSG against time consumption.](image-url)
has the ability to reduce the test error and loss dramatically after around 75 epochs. One possible explanation is that the algorithm changes the batch size to the first term resulting in an escape from a local minima by increasing the variance so as to find a better solution. The right hand column of Figure 6.1 presents the time usage, and it can be seen that SVRG and SGD are similar having higher training time than the other two methods in all three data sets. By using a batch of samples SCSG and VCSG can significantly reduce the training time which is more clearly shown in Figure 6.2.

In Figure 6.2 I use a more visualized format to show the time usage in Figure 6.1. We can see in three sub-figures VCSG can achieve a lowest test error over a shorter period of time. To achieve the 0.025 top-1 test error in MNIST data set, VCSG only takes 16 seconds which is around 2× faster than SCSG, 3× faster than SVRG and 4× faster than SGD. In CIFAR-10 to achieve 0.3 top-1 test error, VCSG is around 6× faster than SVRG, 4× faster than SCSG and 13× faster than SGD. In the ImageNet data set, to achieve 0.55 top-5 test error VCSG can be faster than other methods by up to 5×.

6.5 Discussion

In this chapter, I proposed a VR-based optimization VCSG for non-convex problems. I theoretically determined that a hyper-parameter \( \lambda \) in each iteration can control the reduced variance of SVRG and balance the trade-off between a biased and an unbiased estimator. Meanwhile, an adjustable batch bounded by controlled reduced variance can work with \( \lambda \), step size and mini-batch to choose an appropriate estimator to converge faster to a stationary point on non-convex problems. Moreover, to verify my theoretical results, my experiments use three datasets on three DL models to present the performance of VCSG via test error/loss and elapsed time, and compare these with other leading results. Both theoretical and experimental results show VCSG can efficiently accelerate convergence. I believe that my algorithm is worthy of further study for non-convex optimization, particular in the training of deep neural networks in large-scale applications.

6.6 Proof details

I provide proof details of my theorems in the chapter.

Technique lemmas

The first two lemmas I will used in my theorems are from Lemma A.1 and Lemma A.2 in Lei et al. (2017b).
Lemma 6.6.1. Let \( x_1, \ldots, x_M \in \mathbb{R}^d \) be an arbitrary population of \( N \) vectors with
\[
\sum_{j=1}^{M} x_j = 0.
\]
Further let \( J \) be a uniform random subset of \( \{1, \ldots, M\} \) with size \( m \). Then
\[
\mathbb{E} \left\| \frac{1}{m} \sum_{j \in J} x_j \right\|^2 = \frac{M - m}{(M - 1)m} \mathbb{E} \left\| x_j \right\|^2 \leq \frac{1}{M} \mathbb{E} \left\| x_j \right\|^2.
\]
The geometric random variable \( N_j \) has the key properties below.

Lemma 6.6.2. Let \( N \overset{\text{Geom}}{\sim} \gamma \) for some \( \gamma > 0 \). Then for any sequence \( D_0, D_1, \ldots, D_N \) with \( \mathbb{E} |D_N| < \infty \),
\[
\mathbb{E}(D_N - D_{N+1}) = \left( 1 - \gamma \right)(D_0 - \mathbb{E}D_N).
\]

Unbiased Estimator Version

One-Epoch Analysis

My algorithm is based on the SVRG method, thus the hyper-parameter \( \lambda \) should be within the range as \( 0 < \lambda < 1 \) in both unbiased and biased cases. We start by bounding the gradient \( \mathbb{E}_k \| v^{(j)}_k \|^2 \) in Lemma 6.6.3 and the variance \( \mathbb{E}_k \| e_j \|^2 \) in Lemma 6.6.4.

Lemma 6.6.3. Under Definition 4.1.1,
\[
\mathbb{E}_k \| v^{(j)}_k \|^2 \leq \frac{L^2}{b_j} \| (1 - \lambda) x^{(j)}_k - \lambda x^{(j)}_0 \|^2 + 2(1 - \lambda)^2 \| \nabla f(x^{(j)}_k) \|^2 + 2 \lambda^2 \| e_j \|^2.
\]

Proof. Using the fact that for a random variable \( Z \) \( \mathbb{E} \| Z \|^2 = \mathbb{E} \| Z - EZ \|^2 + \| \mathbb{E} Z \|^2 \), we have
\[
\mathbb{E}_k \| v^{(j)}_k \|^2 = \mathbb{E}_k \| v^{(j)}_k - \mathbb{E}_k v^{(j)}_k \|^2 + \| \mathbb{E}_k v^{(j)}_k \|^2
\]
\[
= \mathbb{E}_k \| (1 - \lambda) \nabla f_k(x^{(j)}_k) - \lambda \nabla f_k(x^{(j)}_0) - ((1 - \lambda) \nabla f(x^{(j)}_k) - \lambda \nabla f(x^{(j)}_0)) \|^2
\]
\[
+ \| (1 - \lambda) \nabla f(x^{(j)}_k) + \lambda e_j \|^2
\]
\[
\leq \mathbb{E}_k \| (1 - \lambda) \nabla f_k(x^{(j)}_k) - \lambda \nabla f_k(x^{(j)}_0) - ((1 - \lambda) \nabla f(x^{(j)}_k) - \lambda \nabla f(x^{(j)}_0)) \|^2
\]
\[
+ 2 \| (1 - \lambda) \nabla f(x^{(j)}_k) \|^2 + 2 \| \lambda e_j \|^2.
\]

(6.16)
By Lemma 6.6.3

\[ \mathbb{E}_{\mathcal{I}_k} \| (1 - \lambda) \nabla f_{\mathcal{I}_k}(x_k^{(j)}) - \lambda \nabla f_{\mathcal{I}_k}(x_0^{(j)}) - ((1 - \lambda) \nabla f(x_k) - \lambda \nabla f(x_0)) \|^2 \]

\[ \leq \frac{1}{b_j} \cdot \frac{1}{n} \sum_{i=1}^{n} \| (1 - \lambda) \nabla f_i(x_k^{(j)}) - \lambda \nabla f_i(x_0^{(j)}) - ((1 - \lambda) \nabla f(x_k) - \lambda \nabla f(x_0)) \|^2 \]

\[ = \frac{1}{b_j} \cdot \frac{1}{n} \sum_{i=1}^{n} \| (1 - \lambda) \nabla f_i(x_k^{(j)}) - \lambda \nabla f_i(x_0^{(j)}) \|^2 \]

\[ \leq \frac{1}{b_j} \cdot \frac{1}{n} \sum_{i=1}^{n} \| (1 - \lambda) \nabla f_i(x_k^{(j)}) - \lambda \nabla f_i(x_0^{(j)}) \|^2 \]

\[ \leq \frac{1}{b_j} \cdot L^2 \| (1 - \lambda)x_k^{(j)} - \lambda x_0^{(j)} \|^2 \]

(6.17)

where the last line is based on Definition 4.1.1 then the bound of the gradient can be alternatively written as,

\[ \mathbb{E}_{\mathcal{I}_k} \| e_k^{(j)} \|^2 \leq \frac{L^2}{b_j} \| (1 - \lambda)x_k^{(j)} - \lambda x_0^{(j)} \|^2 + 2(1 - \lambda)^2 \| \nabla f(x_k^{(j)}) \|^2 + 2\lambda^2 \| e_j \|^2 . \]

(6.18)

**Lemma 6.6.4.**

\[ \mathbb{E}_{\mathcal{I}_j} \| e_j \|^2 \leq \lambda^2 \frac{I(B_j < n)}{B_j} \cdot S^*. \]

**Proof.** Based on Lemma 6.6.3 and the observation that \( \tilde{x}_{j-1} \) is independent of \( \mathcal{I}_j \), the bound of variance \( e_j \) can be expressed as

\[ \mathbb{E}_{\mathcal{I}_j} \| e_j \|^2 = \frac{n - B_j}{(n - 1)B_j} \cdot \frac{\lambda^2}{n} \sum_{i=1}^{n} \| \nabla f_i(\tilde{x}_{j-1}) - \nabla f(\tilde{x}_{j-1}) \|^2 \]

\[ \leq \lambda^2 \frac{n - B_j}{(n - 1)B_j} \cdot S^* \leq \lambda^2 \frac{I(B_j < n)}{B_j} S^* \]

(6.19)

where the upper bound of the variance of the stochastic gradients

\[ S^* = \frac{1}{n} \sum_{i=1}^{n} \| \nabla f_i(\tilde{x}_{j-1}) - \nabla f(\tilde{x}_{j-1}) \|^2 . \]

Theorem 6.3.1 below defines the bound of batch-size, \( B_j \), for the unbiased estimator case.

**Proof of Theorem 6.3.1**

**Theorem.** If the expectation of the variance \( \mathbb{E}_{\mathcal{I}_j} \| e_j \|^2 \leq \sigma \rho^2 \) in Alg 6.2 (\( \sigma \geq 0 \) is a constant for some \( \rho < 1 \)), the lower bound of the batch-size, \( B_j \), can be expressed as,

\[ B_j \geq \frac{nS^*}{S^* + \lambda^2 n \frac{1}{2} \sigma \rho^2}. \]
Proof. To define the bound of the batch-size, $B_j$, for the biased estimator case, I estimate the lower and upper bounds of the variance to control the size of the batch. Based on the result from Lemma 6.6.4 and using the result that the norms of the gradients are bounded by $K^2$ for all $x_j$ (Babanezhad et al. 2015), we have

$$
\frac{1}{n-1} \sum_{i=1}^{n} [\| \nabla f_i(\tilde{x}_{j-1}) \|^2 - \| \nabla f(\tilde{x}_{j-1}) \|^2] \leq K^2,
$$

(6.20)

and using the inequality from (L. Lohr 2000) we have

$$
E_{I_j} \| e_j \|^2 \leq \lambda^2 \frac{n - B_j}{nB_j} K^2.
$$

(6.21)

If we want $E_{I_j} \| e_j \|^2 \leq \sigma \rho^2 j$, for a constant value $\sigma \geq 0$ and for some $\rho^2 j < 1$, we need

$$
B_j \geq \frac{nK^2}{K^2 + n\lambda^2 \sigma \rho^2 j}
$$

(6.22)

Using the Samuelson inequality (Niezgoda 2007), $K^2$ satisfies

$$
\sqrt{\frac{(n-1)}{n-1}} \sum_{i=1}^{n} [\| \nabla f_i(\tilde{x}_{j-1}) \|^2 - \| \nabla f(\tilde{x}_{j-1}) \|^2] \geq n \cdot (\nabla f_i(\tilde{x}_{j-1}) - \nabla f(\tilde{x}_{j-1})).
$$

(6.23)

Inq. 6.23 can alternatively be written using Lemma 6.6.4 as

$$
\sqrt{n-1} E[\| \nabla f_i(\tilde{x}_{j-1}) \|^2 - \| \nabla f(\tilde{x}_{j-1}) \|^2] \geq nE[\nabla f_i(\tilde{x}_{j-1}) - \nabla f(\tilde{x}_{j-1})]^2.
$$

(6.24)

Inq. 6.24 can be substituted by upper bounds $K$ and $S^*$ giving

$$
\sqrt{n-1} \cdot K^2 \geq n \cdot S^*.
$$

(6.25)

Thus, the result from Inq. 6.22 can be written as

$$
B_j \geq \frac{nK^2}{K^2 + n\lambda^2 \sigma \rho^2 j}
$$

$$
\geq \frac{n}{\sqrt{n-1}} \frac{n}{S^* + n\lambda^2 \sigma \rho^2 j}.
$$

(6.26)
Lemma 6.6.5. Suppose \( \eta_j L < 1 \), then under Definition 4.1.1

\[
(1 - \lambda) \eta_j (1 - (1 - \lambda) L \eta_j) B_j \mathbb{E} \| \nabla f(\tilde{x}_j) \|^2 + \lambda \eta_j B_j \mathbb{E} < e_j, \nabla f(\tilde{x}_j) > \\
\leq b_j \mathbb{E}(f(\tilde{x}_{j-1}) - f(\tilde{x}_j)) + \frac{\eta_j^2 B_j L^3}{2b_j} \mathbb{E} \| \tilde{x}_j - \tilde{x}_{j-1} \|^2 + \lambda^2 \eta_j^2 B_j \mathbb{E} \| e_j \|^2.
\]

where \( \mathbb{E} \) denotes the expectation with respect to all randomness.

Proof. By Definition 4.1.1 we have

\[
\mathbb{E}_{\tilde{x}_k}[f(x_{k+1}^{(j)})] \leq f(x_k^{(j)}) - \eta_j < \mathbb{E}_{\tilde{x}_k} v_k, \nabla f(x_k^{(j)}) > + \frac{L \eta_j^2}{2} \mathbb{E}_{\tilde{x}_k} \| v_k \|^2
\]

\[
= f(x_k^{(j)}) - \eta_j < ((1 - \lambda) \nabla f(x_k^{(j)}) + \lambda e_j), \nabla f(x_k^{(j)}) > + \frac{L \eta_j^2}{2} \mathbb{E}_{\tilde{x}_k} \| v_k \|^2
\]

\[
\leq f(x_k^{(j)}) - \eta_j (1 - \lambda) \| \nabla f(x_k^{(j)}) \|^2 - \eta_j < \lambda e_j, \nabla f(x_k^{(j)}) > \\
+ \frac{L^3 \eta_j^2}{2b_j} \| (1 - \lambda)x_k^{(j)} - \lambda x_0^{(j)} \|^2
\]

\[
+ L \eta_j^2 (1 - \lambda)^2 \| \nabla f(x_k^{(j)}) \|^2 + L \eta_j^2 \lambda^2 \| e_j \|^2
\]

(6.27)

\[
= f(x_k^{(j)}) - (\eta_j (1 - \lambda) - L \eta_j^2 (1 - \lambda)^2) \| \nabla f(x_k^{(j)}) \|^2 - \eta_j < e_j, \nabla f(x_k^{(j)}) > \\
+ \frac{L^3 \eta_j^2}{2b_j} \| (1 - \lambda)x_k^{(j)} - \lambda x_0^{(j)} \|^2 + L \eta_j^2 \lambda^2 \| e_j \|^2
\]

\[
\leq f(x_k^{(j)}) - (\eta_j (1 - \lambda) - L \eta_j^2 (1 - \lambda)^2) \| \nabla f(x_k^{(j)}) \|^2 - \eta_j < e_j, \nabla f(x_k^{(j)}) > \\
+ \frac{L^3 \eta_j^2}{2b_j} \| x_k^{(j)} - x_0^{(j)} \|^2 + L \eta_j^2 \lambda^2 \| e_j \|^2
\]

Let \( \mathbb{E}_j \) denote the expectation \( \tilde{x}_0, \tilde{x}_1, ..., \) given \( \tilde{N}_j \) since \( \tilde{N}_j \) is independent of them and let \( k = N_j \) in Inq. 6.27. As \( \tilde{x}_{k+1}, \tilde{x}_{k+2}, ... \) are independent of \( x_k^{(j)} \) and taking the expectation with respect to \( N_j \) and using Fubini’s theorem, Inq. 6.27 implies that

\[
\eta_j (1 - \lambda)(1 - (1 - \lambda) L \eta_j) \mathbb{E}_{N_j} \mathbb{E}_j[\| \nabla f(x_{N_j}^{(j)}) \|^2] + \lambda \eta_j \mathbb{E}_{N_j} \mathbb{E}_j < e_j, \nabla f(x_{N_j}^{(j)}) > \\
\leq \mathbb{E}_{N_j}[(\mathbb{E}_j[f(x_{N_j}^{(j)})] - \mathbb{E}_j[f(x_{N_j}^{(j)}, +)]) + \frac{L^3 \eta_j^2}{2b_j} \mathbb{E}_{N_j} \mathbb{E}_j \| (1 - \lambda)x_{N_j}^{(j)} - \lambda x_0^{(j)} \|^2]
\]

\[
+ L \lambda^2 \eta_j^2 \| e_j \|^2
\]

\[
= \frac{b_j}{B_j} (f(x_0^{(j)}) - \mathbb{E}_j \mathbb{E}_{N_j}[f_{N_j}^{(j)}]) + \frac{L^3 \eta_j^2}{2b_j} \mathbb{E}_{N_j} \mathbb{E}_j \| (1 - \lambda)x_{N_j}^{(j)} - \lambda x_0^{(j)} \|^2] + L \lambda^2 \eta_j^2 \| e_j \|^2
\]

(6.28)

where the last equation in Inq. 6.28 follows from Lemma 6.6.2. The lemma substitutes \( x_{N_j}^{(j)}(x_0^{(j)}) \) by \( \tilde{x}_j(\tilde{x}_{j-1}) \).

\( \square \)
Lemma 6.6.6. Suppose $\eta_j^2 L^2 B_j < b_j^2$, then under Definition 4.1.1,

$$
(b_j - \frac{\eta_j^2 L^2 B_j}{b_j}) \mathbb{E}[\| \tilde{x}_j - \tilde{x}_{j-1} \|^2] + 2\lambda \eta_j B_j \mathbb{E} < e_j, (\tilde{x}_j - \tilde{x}_{j-1}) > \\
\leq -2\eta_j (1 - \lambda) B_j \mathbb{E} < \nabla f(\tilde{x}_j), (\tilde{x}_j - \tilde{x}_{j-1}) > + 2(1 - \lambda)^2 \eta_j^2 B_j \mathbb{E}[\| \nabla f(\tilde{x}_j) \|^2] \\
+ 2\lambda^2 \eta_j^2 B_j \mathbb{E}[\| e_j \|^2]
$$

Proof. Since $x_{k+1}^{(j)} = x_k^{(j)} - \eta_j v_k^{(j)}$, we have

$$
\mathbb{E}_x [\| x_{k+1}^{(j)} - x_0^{(j)} \|^2] \\
= \| x_k^{(j)} - x_0^{(j)} \|^2 - 2\eta_j < \mathbb{E}_x v_k^{(j)}, (x_k^{(j)} - x_0^{(j)}) > + \eta_j^2 \mathbb{E}_x [\| v_k^{(j)} \|^2] \\
= \| x_k^{(j)} - x_0^{(j)} \|^2 - 2(1 - \lambda) \eta_j < \nabla f(x_k^{(j)}), (x_k^{(j)} - x_0^{(j)}) > - 2\eta_j < e_j, (x_k^{(j)} - x_0^{(j)}) > \\
+ \eta_j^2 \mathbb{E}_x [\| v_k^{(j)} \|^2] \\
\leq (1 + \frac{\eta_j^2 L^2}{b_j}) \| x_k^{(j)} - x_0^{(j)} \|^2 - 2\eta_j (1 - \lambda) < \nabla f(x_k^{(j)}), x_k^{(j)} - x_0^{(j)} > \\
- 2\lambda \eta_j < e_j, (x_k^{(j)} - x_0^{(j)}) > + 2(1 - \lambda)^2 \eta_j^2 \| \nabla f(x_k^{(j)}) \|^2 + 2\lambda^2 \eta_j^2 \| e_j \|^2.
$$

(6.29)

where the last inequality follows from Lemma 6.6.3. Using the same notation $\mathbb{E}_j$ from Theorem 6.3.1 we have,

$$
2\eta_j (1 - \lambda) \mathbb{E}_j < \nabla f(x_k^{(j)}), (x_k^{(j)} - x_0^{(j)}) > + 2\lambda \eta_j \mathbb{E}_j < e_j, (x_k^{(j)} - x_0^{(j)}) > \\
\leq (1 + \frac{\eta_j^2 L^2}{b_j}) \mathbb{E}_j [\| x_k^{(j)} - x_0^{(j)} \|^2 - \mathbb{E}_j [\| x_k^{(j)} - x_0^{(j)} \|^2] + 2(1 - \lambda)^2 \eta_j^2 \| \nabla f(x_k^{(j)}) \|^2 \\
+ 2\lambda^2 \eta_j^2 \| e_j \|^2.
$$

(6.30)

Let $k = N_j$, and using Fubini’s theorem, we have,

$$
2(1 - \lambda) \eta_j \mathbb{E}_{N_j} \mathbb{E}_j < \nabla f(x_{N_j}^{(j)}), (x_{N_j}^{(j)} - x_0^{(j)}) > + 2\lambda \eta_j \mathbb{E}_{N_j} \mathbb{E}_j < e_j, (x_{N_j}^{(j)} - x_0^{(j)}) > \\
\leq (1 + \frac{\eta_j^2 L^2}{b_j}) \mathbb{E}_{N_j} \mathbb{E}_j [\| x_{N_j}^{(j)} - x_0^{(j)} \|^2 - \mathbb{E}_{N_j} \mathbb{E}_j [\| x_{N_j}^{(j)} - x_0^{(j)} \|^2] + 2(1 - \lambda)^2 \eta_j^2 \| \nabla f(x_{N_j}^{(j)}) \|^2 \\
+ 2\lambda^2 \eta_j^2 \| e_j \|^2.
$$

(6.31)

The lemma is then proved by substituting $x_{N_j}^{(j)}(x_0^{(j)})$ by $\tilde{x}_j(\tilde{x}_{j-1})$.

Lemma 6.6.7.

$$
b_j \mathbb{E} < e_j, (\tilde{x}_j - \tilde{x}_{j-1}) > = -\eta_j (1 - \lambda) B_j \mathbb{E} < e_j, \nabla f(\tilde{x}_j) > - \lambda^2 \eta_j B_j \mathbb{E} \| e_j \|^2
$$
Chapter 6 A Variance Controlled Stochastic Method with Biased Estimation

**Proof.** Let $M_k^{(j)} = < e_j, (x_k^{(j)} - x_0^{(j)}) >$, then we have

$$\mathbb{E}_{N_j} < e_j, (\hat{x}_j - \tilde{x}_{j-1}) > = \mathbb{E}_{N_j} M_j^{(j)}. \quad (6.32)$$

Since $N_j$ is independent of $(x_0^{(j)}, e_j)$, it has

$$\mathbb{E} < e_j, (\hat{x}_j - \tilde{x}_{j-1}) > = \mathbb{E} M_j^{(j)}. \quad (6.33)$$

Also $M_0^{(j)} = 0$, then we have

$$\mathbb{E} \tau_k (M_{k+1}^{(j)} - M_k^{(j)})$$

$$= \mathbb{E} \tau_k < e_j, (x_{k+1}^{(j)} - x_k^{(j)}) >$$

$$= -\eta_j < e_j, \mathbb{E} \tau_k [v_k^{(j)}] >. \quad (6.34)$$

Using the same notation $\tau_j$ in Lemma [6.6.5] and Lemma [6.6.6] we have

$$\mathbb{E} \tau_j (M_{k+1}^{(j)} - M_k^{(j)}) = -\eta_j (1 - \lambda) < e_j, \mathbb{E} \tau_j \nabla f(x_k^{(j)}) > -\lambda^2 \eta_j \| e_j \|^2. \quad (6.35)$$

Let $k = N_j$ in Equation (6.33). Using Fubini’s theorem and Lemma [6.6.4] we have,

$$\frac{b_j}{B_j} \mathbb{E}_{N_j} M_j^{(j)} = -\eta_j (1 - \lambda) < e_j, \mathbb{E}_{N_j} \mathbb{E}_j \nabla f(x_k^{(j)}) > -\eta_j \| e_j \|^2. \quad (6.36)$$

The lemma is then proved by substituting $x_{N_j}^{(j)}(x_0^{(j)})$ by $\tilde{x}_j(\tilde{x}_{j-1})$. \qed

**Proof of Theorem [6.3.2]**

**Theorem.** Let $\eta L = \gamma (\frac{b_j}{B_j})^\alpha$ where $0 \leq \alpha \leq 1$ and $\gamma \geq 0$. Suppose $B_j \geq b_j \geq B_j^\beta$ ($0 \leq \beta \leq 1$) for all $j$, then under Definition [4.1.1] the output $\tilde{x}_j$ of Alg [6.2] satisfies

$$\mathbb{E} \| \nabla f(\tilde{x}_j) \|^2 \leq \frac{2 L}{\gamma} (\frac{b_j}{B_j})^{1-\alpha} \mathbb{E} (f(\tilde{x}_{j-1}) - f(\tilde{x}_j)) + 2 \lambda^2 \frac{I(B_j < n)}{B_j^{1-\alpha}} S^*$$

where $0 < \lambda < 1$ and $2(1 - \lambda) - 2 \gamma B_j^{\alpha - \alpha} + 2 B_j^{\beta - 1} (1 - \lambda)^2 - 1.16(1 - \lambda)^2$ is positive when $B_j \leq 3$, $0 \leq \gamma \leq \frac{13}{50}$ and $0 < \lambda < 1$. 

Proof. Multiplying the function in Lemma 6.6.5 by 2 and the function in Lemma 6.6.6 by $\frac{b_j}{\eta_j B_j}$ and summing them, then we have,

\[
2\eta_j B_j(1-\lambda)(1-(1-\lambda)\eta_j) - \frac{(1-\lambda)b_j}{B_j} E \| \nabla f(\tilde{x}_j) \|^2 \\
+ \frac{b_j^3 - \eta_j^2 L\eta_j^2 b_j B_j}{b_j \eta_j B_j} E \| \tilde{x}_j - \tilde{x}_{j-1} \|^2 \\
+ 2\lambda_2 \eta_j B_j E < e_j, \nabla f(\tilde{x}_j) > + 2\lambda b_j E < e_j, (\tilde{x}_j - \tilde{x}_{j-1}) > \\
= 2\eta_j B_j(1-\lambda)(1-(1-\lambda)\eta_j) - \frac{(1-\lambda)b_j}{B_j} E \| \nabla f(\tilde{x}_j) \|^2 \\
+ \frac{\frac{3 - (1-\lambda)^2 \eta_j^2 L^2 b_j B_j}{(1-\lambda)\eta_j B_j}}{E \| \tilde{x}_j - \tilde{x}_{j-1} \|^2} \\
- \tilde{x}_{j-1} \|^2 - 2\frac{\lambda^3}{(1-\lambda)} \eta_j B_j E \| e_j \|^2 \quad \text{(Lemma 6.6.7)}
\]

Using the fact that $2 < q, p \leq \beta \| q \|^2 + \frac{1}{\beta} \| p \|^2$ for any $\beta > 0$, $-2(1-\lambda)b_j E < \nabla f(\tilde{x}_j), (\tilde{x}_j - \tilde{x}_{j-1}) >$ in Inq. 6.37 can be bounded as

\[
\leq -2(1-\lambda)b_j E < \nabla f(\tilde{x}_j), (\tilde{x}_j - \tilde{x}_{j-1}) > \\
\leq (1-\lambda)\left( \frac{b_j^3 - (1-\lambda)^2 \eta_j^2 L^2 b_j B_j}{(1-\lambda)\eta_j B_j} \| \tilde{x}_j - \tilde{x}_{j-1} \|^2 \right) \\
+ \frac{\frac{\frac{3 - (1-\lambda)^2 \eta_j^2 L^2 b_j B_j}{(1-\lambda)\eta_j B_j}}{E \| \tilde{x}_j - \tilde{x}_{j-1} \|^2}}{E \| \tilde{x}_j - \tilde{x}_{j-1} \|^2} \\
+ \frac{\eta_j B_j(2(1-\lambda) - 2(1-\lambda)^2 \eta_j b_j}{(1-\lambda)\eta_j B_j} \\
- \frac{b_j^3 - (1-\lambda)^2 \eta_j^2 L^2 b_j B_j}{(1-\lambda)\eta_j B_j} \| \nabla f(\tilde{x}_j) \|^2 \quad \text{(6.38)}
\]

Then Inq. 6.37 can be expressed as

\[
\frac{\eta_j B_j}{b_j}(2(1-\lambda) - 2(1-\lambda)^2 \eta_j b_j}{b_j} \\
- \frac{\frac{\frac{3 - (1-\lambda)^2 \eta_j^2 L^2 b_j B_j}{(1-\lambda)\eta_j B_j}}{E \| \tilde{x}_j - \tilde{x}_{j-1} \|^2}}{E \| \tilde{x}_j - \tilde{x}_{j-1} \|^2} \\
\leq 2E(f(\tilde{x}_{j-1}) - f(\tilde{x}_j)) + \frac{\eta_j B_j \lambda^2}{b_j}(\frac{\frac{\frac{3 - (1-\lambda)^2 \eta_j^2 L^2 b_j B_j}{(1-\lambda)\eta_j B_j}}{E \| \tilde{x}_j - \tilde{x}_{j-1} \|^2} + \eta_j L + \frac{b_j}{B_j} E \| e_j \|^2.
\]

Since $\eta_j L = \gamma (\frac{b_j}{B_j})^\alpha$, $b_j \geq 1$ and $B_j \geq b_j \geq B_j^\beta$ where $\alpha > 0$ and $\beta \geq 0$ by Theorem 6.3.1, a one part in left hand side of above inequality can be simplified and positive as following:

\[
b_j^3 - (1-\lambda)^2 \eta_j^2 L^2 b_j B_j - (1-\lambda)^2 \eta_j^2 L^3 B_j^2 \\
= b_j^3(1-\lambda)^2 \eta_j^2 \gamma^2 B_j^{2\alpha - 2} - (1-\lambda)^2 \gamma^3 \frac{b_j^{3\alpha - 3}}{B_j^{3\alpha - 2}}) \\
\geq b_j^3(1-\lambda)^2 \gamma^2 B_j^{-1} - (1-\lambda)^2 \gamma^3 B_j^{-1}) \geq 0.86b_j^3
\]
By Eq. 6.40 the left side of Inq. 6.39 can be simplified since the factor of geometry distribution $\gamma \geq 0$ as

$$\frac{\eta B_j}{b_j} (2(1 - \lambda) - 2(1 - \lambda)^2 L \eta j - 2(1 - \lambda)^2 \frac{B_j}{b_j} (1 - \lambda)^2 \eta^2 L^2 b_j B_j - (1 - \lambda)^2 \eta^2 L^3 B_j^2) \mathbb{E} \| \nabla f(\tilde{x}_j) \|^2$$

$$\geq \frac{\gamma}{L} B_j^{\alpha - \beta + 1} \left( 2(1 - \lambda) - (2 B_j^{\alpha - \beta} + 2 \frac{b_j}{B_j})(1 - \lambda)^2 - 1.16(1 - \lambda)^2 \right) \mathbb{E} \| \nabla f(\tilde{x}_j) \|^2$$

Equation 6.41 is positive when $0 \leq \gamma \leq \frac{13}{50}$ and $B_j \geq 3$. Moreover, Lei et al. (2017a); Lei and Jordan (2017) determined the learning rate $\eta = \frac{\gamma b_j}{L B_j} \leq \frac{1}{3L}$ that $\gamma \leq \frac{1}{3}$ which can guarantees the convergence in non-convex case. In my case, $\gamma \leq \frac{13}{50}$ satisfies within the range $\gamma \leq \frac{1}{3}$. Then Equation 6.39 can be simplified by Equation 6.41 as

$$\mathbb{E} \| \nabla f(\tilde{x}_j) \|^2 \leq \frac{2 \gamma}{L} B_j^{\alpha - \beta + 1} \left( 2(1 - \lambda) - (2 B_j^{\alpha - \beta} + 2 B_j^{-1})(1 - \lambda)^2 - 1.16(1 - \lambda)^2 \right)$$

Then, using Lemma 6.6.4 Inq. 6.42 can be rewritten as

$$\mathbb{E} \| \nabla f(\tilde{x}_j) \|^2 \leq \frac{2 \gamma}{L} B_j^{\alpha - \beta + 1} \left( 2(1 - \lambda) - (2 B_j^{\alpha - \beta} + 2 B_j^{-1})(1 - \lambda)^2 - 1.16(1 - \lambda)^2 \right).$$

**Biased Estimator Version**

**One-Epoch Analysis**

For the biased estimation version, we still start by bounding the gradient $\mathbb{E}_{\tilde{x}_k} \| v^{(j)}_k \|^2$ in Lemma 6.6.8 and the variance $\mathbb{E}_{\tilde{x}_j} \| e_j \|^2$ in Lemma 6.6.9.
Lemma 6.6.8. Under Definition 4.1.1

\[ \mathbb{E}_{\tilde{I}_k} \| v_k^{(j)} \|^2 \leq \frac{(1 - \lambda)^2 L^2}{b_j} \| x_k^{(j)} - x_0^{(j)} \|^2 + 2(1 - \lambda)^2 \| \nabla f(x_k^{(j)}) \|^2 + 2 \| e_j \|^2. \]

Proof. Using the fact that for a random variable \( Z \), \( \mathbb{E} \| Z \|^2 = \| Z - \mathbb{E} Z \|^2 + \| \mathbb{E} Z \|^2 \), we have

\[ \begin{align*}
\mathbb{E}_{\tilde{I}_k} \| v_k^{(j)} \|^2 &= \mathbb{E}_{\tilde{I}_k} \| v_k^{(j)} - \mathbb{E}_{\tilde{I}_k} v_k^{(j)} \|^2 + \| \mathbb{E}_{\tilde{I}_k} v_k^{(j)} \|^2 \\
&= \mathbb{E}_{\tilde{I}_k} \| (1 - \lambda)(\nabla f_{\tilde{I}_k}(x_k^{(j)}) - \nabla f_{\tilde{I}_k}(x_0^{(j)})) - (1 - \lambda)(\nabla f(x_k^{(j)}) - \nabla f(x_0^{(j)})) \|^2 \\
&\quad + \| (1 - \lambda)\nabla f(x_k^{(j)}) + e_j \|^2.
\end{align*} \]

By Lemma 6.6.1, the first part of inequality in Equation 6.44 can be rewritten as,

\[ \begin{align*}
(1 - \lambda)^2 \mathbb{E}_{\tilde{I}_k} \| \nabla f_{\tilde{I}_k}(x_k^{(j)}) - \nabla f_{\tilde{I}_k}(x_0^{(j)}) - (\nabla f(x_k^{(j)}) - \nabla f(x_0^{(j)})) \|^2 \\
&\leq \frac{(1 - \lambda)^2}{b_j} \cdot \frac{1}{n} \sum_{i=1}^n \| \nabla f_i(x_k^{(j)}) - \nabla f_i(x_0^{(j)}) - (\nabla f(x_k^{(j)}) - \nabla f(x_0^{(j)})) \|^2 \\
&= \frac{(1 - \lambda)^2}{b_j} \cdot \frac{1}{n} \sum_{i=1}^n \| \nabla f_i(x_k^{(j)}) - \nabla f_i(x_0^{(j)}) \|^2 - \| (\nabla f(x_k^{(j)}) - \nabla f(x_0^{(j)})) \|^2 \\
&\leq \frac{(1 - \lambda)^2}{b_j} \cdot \frac{1}{n} \sum_{i=1}^n \| \nabla f_i(x_k^{(j)}) - \nabla f_i(x_0^{(j)}) \|^2 \\
&\leq \frac{(1 - \lambda)^2}{b_j} \cdot L^2 \| x_k^{(j)} - x_0^{(j)} \|^2.
\end{align*} \]

where the last line is based on Definition 4.1.1 then the bound of the gradient can be written as,

\[ \mathbb{E}_{\tilde{I}_k} \| v_k^{(j)} \|^2 \leq \frac{(1 - \lambda)^2 L^2}{b_j} \| x_k^{(j)} - x_0^{(j)} \|^2 + 2(1 - \lambda)^2 \| \nabla f(x_k^{(j)}) \|^2 + 2 \| e_j \|^2. \]

(6.46)

Lemma 6.6.9.

\[ \mathbb{E}_{\tilde{I}_j} \| e_j \|^2 \leq (1 - \lambda)\frac{I(B_j < n)}{B_j} \mathbb{S}^* + (1 - 2\lambda)^2 \mathbb{E}_{\tilde{I}_j} \| \nabla f_i(\tilde{x}_{j-1}) \|^2 \\
= \mathbb{E}_{\tilde{I}_j} \| \tilde{e}_j \|^2 + (1 - 2\lambda)^2 \mathbb{E}_{\tilde{I}_j} \| \nabla f_i(\tilde{x}_{j-1}) \|^2 \\
\] where \((1 - \lambda)^2 \frac{I(B_j < n)}{B_j} \mathbb{S}^* = \mathbb{E}_{\tilde{I}_j} \| \tilde{e}_j \|^2 \) and \(0 < \lambda < 1\).
Chapter 6 A Variance Controlled Stochastic Method with Biased Estimation

93

expressed as,

Theorem. If the expectation of the variance \( E[I_j \| e_j \|^2] \leq \sigma \rho^2 j \) in Alg 6.3 (\( \sigma \geq 0 \) is a constant for some \( \rho < 1 \)) and \( 0 < \lambda < 1 \), the lower bound of the batch-size, \( B_j \), can be expressed as,

\[
B_j \geq \frac{n \mathcal{S}^*}{\mathcal{S}^* + (1 - \lambda)^2 n \Sigma \sigma \rho^2 j}
\]

Proof. To define the bound of the batch-size, \( B_j \), for the biased estimator case, we estimate the lower and upper bounds of the variance to control the size of the batch.
Based on the result from Lemma [6.6.9] and using the result that the norms of the gradients are bounded by $K^2$ for all $x_j$ [Babanezhad et al., 2015], we have

$$\frac{1}{n-1} \sum_{i=1}^{n} [(1-\lambda)^2 \| \nabla f_i(\tilde{x}_{j-1}) \|^2 - \lambda^2 \| \nabla f(\tilde{x}_{j-1}) \|^2]$$

$$\leq (1-\lambda)^2 \frac{1}{n-1} \sum_{i=1}^{n} [\| \nabla f_i(\tilde{x}_{j-1}) \|^2 - \| \nabla f(\tilde{x}_{j-1}) \|^2] + (1-2\lambda)^2 \mathbb{E}_{I_j}[\nabla f_i(\tilde{x}_{j-1})]^2$$

$$\leq (1-\lambda)^2 K^2 + (1-2\lambda)^2 \mathbb{E}_{I_j}[\nabla f_i(\tilde{x}_{j-1})]^2,$$

and we use the same approach we applied in the unbiased case which is shown from Inq. [6.21] to [6.25] to achieve a bound of the batch size when $0 < \lambda < 1$. The batch size can be bounded as,

$$B_j \geq \frac{nK^2}{K^2 + (1-\lambda)^2 n \sigma^2} \geq \frac{n - \frac{n}{\sqrt{n-1}} S^*}{\sqrt{n-1} S^* + (1-\lambda)^2 n \sigma^2} \geq \frac{n^2 S^*}{nS^* + (1-\lambda)^2 n \frac{1}{2} \sigma^2}.$$

(6.49)

**Lemma 6.6.10.** Suppose $\eta_j L < 1$, then under Definition [4.1.1]

$$(1-\lambda)(1-(1-\lambda)L \eta_j) \eta_j B_j \mathbb{E} \| \nabla f(\tilde{x}_j) \|^2 + \eta_j B_j \mathbb{E} < e_j, \nabla f(\tilde{x}_j) >$$

$$\leq b_j \mathbb{E}(f(\tilde{x}_{j-1}) - f(\tilde{x}_j)) + \frac{(1-\lambda)^2 \eta_j^2 B_j L^3}{2b_j} \mathbb{E} \| \tilde{x}_j - \tilde{x}_{j-1} \|^2 + L \eta_j^2 B_j \mathbb{E} \| e_j \|^2.$$

where $\mathbb{E}$ denotes the expectation with respect to all randomness.

**Proof.** By Definition [4.1.1] we have

$$\mathbb{E}_{I_k}[f(x^{(j)}_{k+1})] \leq f(x^{(j)}_{k}) - \eta_j < \mathbb{E}_{I_k} v_k, \nabla f(x^{(j)}_{k}) > + \frac{L \eta_j^2}{2} \mathbb{E}_{I_k} \| v_k \|^2$$

$$= f(x^{(j)}_{k}) - \eta_j < ((1-\lambda) \nabla f(x^{(j)}_{k}) + e_j), \nabla f(x^{(j)}_{k}) > + \frac{L \eta_j^2}{2} \mathbb{E}_{I_k} \| v_k \|^2$$

$$\leq f(x^{(j)}_{k}) - \eta_j < (1-\lambda) \| \nabla f(x^{(j)}_{k}) \|^2 - \eta_j < e_j, \nabla f(x^{(j)}_{k}) >$$

$$+ \frac{L^3 \eta_j^2 (1-\lambda)^2}{2b_j} \| x^{(j)}_{k} - x^{(j)}_{0} \|^2 + L \eta_j^2 \| \nabla f(x^{(j)}_{k}) \|^2 + L \eta_j^2 \| e_j \|^2$$

(6.50)

$$= f(x^{(j)}_{k}) - (\eta_j (1-\lambda) - L \eta_j^2 (1-\lambda)^2) \| \nabla f(x^{(j)}_{k}) \|^2$$

$$- \eta_j < e_j, \nabla f(x^{(j)}_{k}) > + \frac{L^3 \eta_j^2 (1-\lambda)^2}{2b_j} \| x^{(j)}_{k} - x^{(j)}_{0} \|^2 + L \eta_j^2 \| e_j \|^2$$

Let $\mathbb{E}_{j}$ denote the expectation $\tilde{X}_0, \tilde{X}_1, ..., $ given $\tilde{N}_j$ since $\tilde{N}_j$ is independent of them and let $k=\tilde{N}_j$ in Inq [6.50] As $\tilde{X}_{k+1}, \tilde{X}_{k+2}, ...$ are independent of $x^{(j)}_{k}$ and taking the expectation
with respect to $N_j$ and using Fubini’s theorem, Inq. 6.50 implies that

$$\eta_j(1 - \lambda)(1 - (1 - \lambda)L\eta_j)\mathbb{E}_{N_j}\mathbb{E}_j[\| \nabla f(x_{N_j}^{(j)}) \|^2] + \eta_j\mathbb{E}_{N_j}\mathbb{E}_j < e_j, \nabla f(x_{N_j}^{(j)}) >$$

$$\leq \mathbb{E}_{N_j}[\mathbb{E}_j[f(x_{N_j}^{(j)})] - \mathbb{E}_j[f(x_{N_j+1}^{(j)})]]$$

$$+ \frac{L^3\eta_j^2(1 - \lambda)^2}{2b_j}\mathbb{E}_{N_j}\mathbb{E}_j[\| x_{N_j}^{(j)} - x_0^{(j)} \|^2] + L\eta_j^2 \| e_j \|^2$$

$$= \frac{b_j}{B_j}(f(x_0^{(j)}) - \mathbb{E}_j[\mathbb{E}_j[f(x_{N_j}^{(j)})]]) + \frac{L^3\eta_j^2(1 - \lambda)^2}{2b_j}\mathbb{E}_{N_j}\mathbb{E}_j[\| x_{N_j}^{(j)} - x_0^{(j)} \|^2] + L\eta_j^2 \| e_j \|^2$$

(6.51)

where the last equation in Inq. 6.51 follows from Lemma 6.6. The lemma substitutes $x_{N_j}^{(j)}(x_0^{(j)})$ by $\tilde{x}_j(\tilde{x}_{j-1})$.

\textbf{Lemma 6.6.11.} Suppose $\eta_j^2L^2B_j < b_j$, then under Definition \textit{smooth1},

$$(b_j - \frac{(1 - \lambda)^2\eta_j^2L^2B_j}{b_j})\mathbb{E}_j[\| \tilde{x}_j - \tilde{x}_{j-1} \|^2] + 2\eta_jB_j\mathbb{E}_j < e_j, (\tilde{x}_j - \tilde{x}_{j-1}) >$$

$$\leq -2(1 - \lambda)\eta_jB_j\mathbb{E}_j < \nabla f(\tilde{x}_j), (\tilde{x}_j - \tilde{x}_{j-1}) >$$

$$+ 2(1 - \lambda)^2\eta_j^2B_j\mathbb{E}_j[\| \nabla f(\tilde{x}_j) \|^2] + 2\eta_j^2B_j\mathbb{E}_j[\| e_j \|^2]$$

\textbf{Proof.} Since $x_{k+1}^{(j)} = x_k^{(j)} - \eta_jv_k^{(j)}$, we have

$$\mathbb{E}_{\tilde{x}_k}[\| x_{k+1}^{(j)} - x_0^{(j)} \|^2]$$

$$= \| x_k^{(j)} - x_0^{(j)} \|^2 - 2\eta_j < \mathbb{E}_{\tilde{x}_k}v_k^{(j)}, (x_k^{(j)} - x_0^{(j)}) > + \eta_j^2\mathbb{E}_{\tilde{x}_k}[\| v_k^{(j)} \|^2]$$

$$= \| x_k^{(j)} - x_0^{(j)} \|^2 - 2\eta_j(1 - \lambda) < \nabla f(x_k^{(j)}), (x_k^{(j)} - x_0^{(j)}) > - 2\eta_j < e_j, (x_k^{(j)} - x_0^{(j)}) >$$

$$+ \eta_j^2\mathbb{E}_{\tilde{x}_k}[\| v_k^{(j)} \|^2]$$

$$\leq (1 + \frac{(1 - \lambda)^2\eta_j^2L^2}{b_j}) \| x_k^{(j)} - x_0^{(j)} \|^2 - 2\eta_j(1 - \lambda) < \nabla f(x_k^{(j)}), x_k^{(j)} - x_0^{(j)} >$$

$$- 2\eta_j < e_j, (x_k^{(j)} - x_0^{(j)}) >$$

$$+ 2(1 - \lambda)^2\eta_j^2 \| \nabla f(x_k^{(j)}) \|^2 + 2\eta_j^2 \| e_j \|^2.$$  

(6.52)

where the last inequality is based on Lemma 6.6.8 Using the same notation $\mathbb{E}_j$ in Theorem 6.3.1 we have

$$2\eta_j(1 - \lambda)\mathbb{E}_j < \nabla f(x_k^{(j)}), (x_k^{(j)} - x_0^{(j)}) > + 2\eta_j\mathbb{E}_j < e_j, (x_k^{(j)} - x_0^{(j)}) >$$

$$\leq (1 + \frac{(1 - \lambda)^2\eta_j^2L^2}{b_j})\mathbb{E}_j \| x_k^{(j)} - x_0^{(j)} \|^2 - \mathbb{E}_j \| x_k^{(j)} - x_0^{(j)} \|^2$$

$$+ 2(1 - \lambda)^2\eta_j^2 \| \nabla f(x_k^{(j)}) \|^2 + 2\eta_j^2 \| e_j \|^2.$$  

(6.53)
Let $k = N_j$, and using Fubini’s theorem, we have,
\[2\eta_j(1 - \lambda)\mathbb{E}_{N_j} \mathbb{E}_j < \nabla f(x_{N_j}^{(j)}), (x_{N_j}^{(j)} - x_0^{(j)}) + 2\eta_j \mathbb{E}_{N_j} \mathbb{E}_j < e_j, (x_{N_j}^{(j)} - x_0^{(j)}) > \]
\[\leq (1 + \frac{(1 - \lambda)^2\eta_jL^2}{b_j})\mathbb{E}_{N_j} \mathbb{E}_j \| x_{N_j}^{(j)} - x_0^{(j)} \|^2 - \mathbb{E}_{N_j} \mathbb{E}_j \| x_{N_j}^{(j)} - x_0^{(j)} \|^2 + 2(1 - \lambda)^2\eta_j^2 \mathbb{E}_{N_j} \| \nabla f(x_{N_j}^{(j)}) \|^2 + 2\eta_j^2 \| e_j \|^2 \]
\[= \left(\frac{-b_j}{B_j} + \frac{(1 - \lambda)^2\eta_jL^2}{b_j}\right)\mathbb{E}_{N_j} \mathbb{E}_j \| x_{N_j}^{(j)} - x_0^{(j)} \|^2 + 2(1 - \lambda)^2\eta_j^2 \mathbb{E}_{N_j} \| \nabla f(x_{N_j}^{(j)}) \|^2 + 2\eta_j^2 \| e_j \|^2 \cdot \tag{6.54}\]

The lemma is then proved by substituting $x_{N_j}^{(j)}(x_0^{(j)})$ by $\tilde{x}_j(\hat{x}_j^{-1})$.

**Lemma 6.6.12.**

\[b_j \mathbb{E} < e_j, (\tilde{x}_j - \hat{x}_j^{-1}) >= -\eta_j(1 - \lambda)B_j \mathbb{E} < e_j, \nabla f(\tilde{x}_j) >= -\eta_j \mathbb{E}_j \| e_j \|^2 \]

**Proof.** Let $M_k^{(j)} = e_j, (x_{k}^{(j)} - x_0^{(j)}) >$, then we have
\[\mathbb{E}_{N_j} < e_j, (\tilde{x}_j - \hat{x}_j^{-1}) >= \mathbb{E}_{N_j} M_{N_j}^{(j)} \cdot \]

Since $N_j$ is independent of $(x_0^{(j)}, e_j)$, it has
\[\mathbb{E} < e_j, (\tilde{x}_j - \hat{x}_j^{-1}) >= \mathbb{E} M_{N_j}^{(j)}. \tag{6.55}\]

Also $M_0^{(j)} = 0$, then we have
\[\mathbb{E}_{\tilde{x}_k} (M_{k+1}^{(j)} - M_k^{(j)})\]
\[= \mathbb{E}_{\tilde{x}_k} < e_j, (x_{k+1}^{(j)} - x_k^{(j)}) >= -\eta_j < e_j, \mathbb{E}_{\tilde{x}_k} \nu_k^{(j)} > \]
\[=-\eta_j(1 - \lambda) < e_j, \nabla f(x_k^{(j)}) > -\eta_j \| e_j \|^2 . \tag{6.56}\]

Using the same notation $\mathbb{E}_j$ in Theorem 6.3.1 we have
\[\mathbb{E}_j (M_{k+1}^{(j)} - M_k^{(j)}) = -\eta_j(1 - \lambda) < e_j, \mathbb{E}_j f(x_k^{(j)}) > -\eta_j \| e_j \|^2 . \tag{6.57}\]

Let $k = N_j$ in Equation 6.57. Using Fubini’s theorem and Lemma 6.6.2 we have,
\[\frac{b_j}{B_j} \mathbb{E}_{N_j} M_{N_j}^{(j)} = -\eta_j(1 - \lambda) < e_j, \mathbb{E}_{N_j} \mathbb{E}_j f(x_k^{(j)}) > -\eta_j \| e_j \|^2 . \tag{6.58}\]

The lemma is then proved by substituting $x_{N_j}^{(j)}(x_0^{(j)})$ by $\tilde{x}_j(\hat{x}_j^{-1})$.  \qed
Proof of Theorem 6.3.4

Theorem. Let \( \eta L = \gamma \left( \frac{b_j}{B_j} \right)^\alpha \) (0 < \( \alpha < 1 \)) and \( \gamma \leq \frac{1}{3} \). Suppose \( \gamma \leq \frac{1}{3} \) and \( B_j \geq b_j \geq B_j^\beta \) (0 \( \leq \beta < 1 \)) for all \( j \), then under Definition 4.1.1, the output \( \tilde{x}_j \) of Alg 6.2 we have,

\[
\mathbb{E} \left\| \nabla f(\tilde{x}_j) \right\|^2 \\
\leq \frac{2\mathbb{E}[f(\tilde{x}_{j-1}) - f(\tilde{x}_j)] + 2(1-\lambda)^2 \frac{\gamma}{L} B_j^{\alpha+3\alpha-\beta} I(B_j < n) S^*}{B_j^{1-\alpha+\alpha\beta-\beta} \left( 2(1-\lambda) - (2\Gamma B_j^{\alpha-\alpha} + 2B_j^{3-1} - 4LB_j^{2\alpha-2})(1-\lambda)^2 - 1.16(1-\lambda)^2 \right)},
\]

where \( 0 < \lambda < 1 \).

Proof. Multiplying Equation 6.6.10 by 2 and Equation 6.6.11 by \( b_j \eta_j B_j \) and summing them, then we have,

\[
2\eta_j B_j (1-\lambda)(1-(1-\lambda)L\eta_j - \frac{(1-\lambda)b_j}{B_j}) \mathbb{E} \left\| \nabla f(\tilde{x}_j) \right\|^2 \\
+ \frac{b_j^3 - (1-\lambda)^2 \eta_j^2 L^2 b_j B_j - (1-\lambda)^2 \eta_j^3 L^3 B_j^2}{b_j \eta_j B_j} \mathbb{E} \left\| \tilde{x}_j - \tilde{x}_{j-1} \right\|^2 \\
+ 2\eta_j B_j \mathbb{E} < e_j, \nabla f(\tilde{x}_j) > + 2b_j \mathbb{E} < e_j, (\tilde{x}_j - \tilde{x}_{j-1}) > \\
= 2\eta_j B_j (1-\lambda)(1-(1-\lambda)L\eta_j - \frac{(1-\lambda)b_j}{B_j} + \frac{(2\lambda - 1)^2}{2\eta_j B_j (1-\lambda)}) \mathbb{E} \left\| \nabla f(\tilde{x}_j) \right\|^2 \\
+ \frac{b_j^3 - (1-\lambda)^2 \eta_j^2 L^2 b_j B_j - (1-\lambda)^2 \eta_j^3 L^3 B_j^2}{b_j \eta_j B_j} \mathbb{E} \left\| \tilde{x}_j - \tilde{x}_{j-1} \right\|^2 \\
- 2\eta_j B_j \mathbb{E} \left\| \tilde{e}_j \right\|^2 \text{ (Lemma 6.6.12)} \\
\leq -2(1-\lambda)b_j \mathbb{E} < \nabla f(\tilde{x}_j), (\tilde{x}_j - \tilde{x}_{j-1}) > \\
+ 2b_j \mathbb{E} (f(\tilde{x}_{j-1}) - f(\tilde{x}_j)) + (2L\eta_j^2 B_j + 2\eta_j b_j) \mathbb{E} \left\| \tilde{e}_j \right\|^2
\]

Using the fact that \( 2 < q, p \geq \beta \parallel q \parallel^2 + \frac{1}{\beta} \parallel p \parallel^2 \) for any \( \beta > 0 \), \(-2b_j \mathbb{E} < \nabla f(\tilde{x}_j), (\tilde{x}_j - \tilde{x}_{j-1}) > \) in Inq. 6.59 can be bounded as

\[
(1-\lambda)b_j \eta_j B_j \mathbb{E} \left\| \nabla f(\tilde{x}_j) \right\|^2 \\
\leq \frac{(1-\lambda)b_j \eta_j B_j}{b_j^3 - (1-\lambda)^2 \eta_j^2 L^2 b_j B_j - (1-\lambda)^2 \eta_j^3 L^3 B_j^2} \mathbb{E} \left\| \nabla f(\tilde{x}_j) \right\|^2 \\
+ \frac{b_j^3 - (1-\lambda)^2 \eta_j^2 L^2 b_j B_j - (1-\lambda)^2 \eta_j^3 L^3 B_j^2}{(1-\lambda)b_j \eta_j B_j} \mathbb{E} \left\| \tilde{x}_j - \tilde{x}_{j-1} \right\|^2
\]
Then Inq. 6.59 can be rewritten as

\[
\eta_j \frac{B_j}{b_j} (2(1-\lambda) - 2(1-\lambda)^2 L \eta_j - 2(1-\lambda)^2 \frac{b_j}{B_j} + \frac{(2\lambda - 1)^2}{\eta_j B_j} - \frac{(1-\lambda)^2 b_j^3}{B_j} - (1-\lambda)^2 \eta_j^2 L^2 b_j B_j - (1-\lambda)^2 \eta_j^3 L^3 B_j^2) ||\nabla f(\bar{x}_j)||^2
\leq 2 \mathbb{E} (f(\bar{x}_j) - f(\bar{x})) + \frac{2 \eta_j B_j}{b_j} (1 + \eta_j L + \frac{b_j}{B_j}) \mathbb{E} ||\epsilon_j||^2.
\]

(6.61)

Since \( \eta_j L = \gamma \frac{b_j}{B_j} \gamma \), \( b_j \geq 1 \) and \( B_j \geq b_j \geq B_j^\beta \) where \( 0 < \alpha \leq 1, 0 \leq \beta \leq 1 \), we have

\[
b_j^3 - (1-\lambda)^2 \eta_j^2 L^2 b_j B_j - (1-\lambda)^2 \eta_j^3 L^3 B_j^2
= b_j^3 (1 - (1-\lambda)^2 \gamma^2 \frac{b_j^{2\alpha-2}}{B_j^{2\alpha-1}} - (1-\lambda)^2 \gamma^3 \frac{b_j^{3\beta-3}}{B_j^{3\beta-2}})
= b_j^3 (1 - (1-\lambda)^2 \gamma^2 B_j^{-1} - (1-\lambda)^2 \gamma^3 B_j^{-1}) \geq 0.86 b_j^3
\]

By Eq. 6.62 the left side of Inq. 6.61 can be simplified as

\[
\mathbb{E} ||\nabla f(\bar{x}_j)||^2
= \gamma \frac{B_j^{1-\alpha+\beta-\beta}}{L}
\left(2(1-\lambda) - (2\gamma B_j^{\beta-\alpha} + 2B_j^{\beta-1})(1-\lambda) + \frac{(2\lambda - 1)^2}{\gamma B_j^{2\alpha-2} - 1.16(1-\lambda)^2}ight)
\mathbb{E} ||\nabla f(\bar{x}_j)||^2
\geq \gamma \frac{B_j^{\alpha\beta-\alpha-\beta+1}}{L}
\left(2(1-\lambda) - (2\gamma B_j^{-1} + 2B_j^{-1} - 4)(1-\lambda)^2 - 1.16(1-\lambda)^2\right)
\mathbb{E} ||\nabla f(\bar{x}_j)||^2.
\]

(6.63)

Equation 6.63 is positive when \( 0 \leq \gamma \leq 2.42 B_j - 1 \) and \( B_j \geq 1 \). Moreover, Lei et al. (2017a) and Lei and Jordan (2017) determined the learning rate \( \eta = \frac{\gamma b_j}{L B_j} \leq \frac{1}{3 L} \) that \( \gamma \leq \frac{1}{3} \), which guarantees the convergence in non-convex case. In my case, \( \gamma \) should satisfy the range \( 0 \leq \gamma \leq \frac{1}{3} \leq 2.42 B_j - 1 \), thus \( \gamma \leq \frac{1}{3} \).
Then Equation 6.61 can be simplified by Equation 6.63 as

\[
\mathbb{E} \| \nabla f(\tilde{x}_j) \|^2 \leq \frac{\gamma}{L} B_j^{1-\alpha+\beta-\beta} \left( 2(1 - \lambda) - (2\gamma B_j^{\beta-\alpha} + 2B_j^{3-1} - 4LB_j^{2\alpha-2})(1 - \lambda)^2 - 1.16(1 - \lambda)^2 \right)
\]

Then, using Lemma 6.6.2, Inq. 6.64 can be expressed as

\[
\mathbb{E} \| \nabla f(\tilde{x}_j) \|^2 \leq \frac{\gamma}{L} B_j^{1-\alpha+\beta-\beta} \left( 2(1 - \lambda) - (2\gamma B_j^{\beta-\alpha} + 2B_j^{3-1} - 4LB_j^{2\alpha-2})(1 - \lambda)^2 - 1.16(1 - \lambda)^2 \right)
\]

Convergence Analysis for \( L \)-smooth Objectives

Proof of Theorem 6.3.5

**Theorem.** Under the specifications of Theorem 6.3.2, Theorem 6.3.4, and Definition 4.1.1, the output \( \tilde{x}_T \) can achieve its upper bound of gradients depending on two estimators.

- For the unbiased estimator (Alg. 6.2), \( 0 < \lambda < 1 \). The upper bound is given by,

\[
\mathbb{E} \| \nabla f(\tilde{x}_j) \|^2 \leq \frac{2L}{\gamma} \frac{(\Delta_f)}{\theta \sum_{j=1}^T b_j^{\alpha-1} B_j^{1-\alpha}} + \frac{2\lambda^4 I(B_j < n) S^*}{\theta B_j^{1-4\alpha}},
\]

- For the biased estimator (Alg. 6.3), \( 0 < \lambda < 1 \). The upper bound is shown as,

\[
\mathbb{E} \| \nabla f(\tilde{x}_j) \|^2 \leq \frac{2L}{\gamma} \frac{(\Delta_f)}{\theta_{biased} \sum_{j=1}^T b_j^{\alpha-1} B_j^{1-\alpha}} + \frac{2(1 - \lambda)^2 I(B_j < n) S^*}{\theta_{biased} B_j^{1-4\alpha}},
\]

where \( \theta = 2(1 - \lambda) - (2\gamma B_j^{\beta-\alpha} + 2B_j^{3-1})(1 - \lambda)^2 - 1.16(1 - \lambda)^2 > 0 \), and \( \theta_{biased} = 2(1 - \lambda) - (2\gamma B_j^{\beta-\alpha} + 2B_j^{3-1} - 4LB_j^{2\alpha-2})(1 - \lambda)^2 - 1.16(1 - \lambda)^2 \).
Proof. Since $\hat{x}_T^*$ is a random element from $(\hat{x}_j)_{j=1}^T$ with

$$P(\hat{x}_T^* = \hat{x}_j) \propto \frac{\eta_j B_j}{b_j} \propto \left(\frac{B_j}{b_j}\right)^\alpha,$$

(6.66)

Inq. 6.43 and 6.65 will be re-scaled as Inq. 6.67 and 6.68 respectively.

- For the unbiased estimator (Alg. 6.2), the upper bound is shown as,

$$E \| \nabla f(\hat{x}_T^*) \|^2 \leq \frac{\left(\frac{2L}{\gamma}\right) \Delta f}{\theta \sum_{j=1}^T b_j^{\alpha-1} B_j^{1-\alpha}} + \frac{2\lambda^4 I(B_j < n) S^*}{\theta B_j^{1-4\alpha}},$$

(6.67)

where $\theta = 2(1 - \lambda) - (2\gamma B_j^{\alpha-\alpha_2} + 2B_j^{3-1})(1 - \lambda)^2 - 1.16 \lambda^2$.

- For the biased estimator (Alg. 6.3), the upper bound is shown as,

$$E \| \nabla f(\hat{x}_j) \|^2 \leq \frac{\left(\frac{2L}{\gamma}\right) \Delta f}{\theta_{biased} \sum_{j=1}^T b_j^{\alpha-1} B_j^{1-\alpha}} + \frac{(1 - \lambda)^2 I(B_j < n) S^*}{\theta_{biased} B_j^{1-4\alpha}},$$

(6.68)

where $\theta_{biased} = 2(1 - \lambda) - (2\gamma B_j^{\alpha-\alpha_2} + 2B_j^{3-1} - 4LB_j^{2\alpha-2})(1 - \lambda)^2 - 1.16(1 - \lambda)^2$. 

$\square$
Chapter 7
Conclusions and Further Work

Deep learning techniques have triggered the AI revolution, developing a wide range of AI applications that are changing our lives. There is increasing demand for these powerful AI capabilities to be deployed more ubiquitously on more resource constrained devices. However, due to the memory-intensive and computational nature, deep learning techniques have had limited application on embedded systems. To address this problem, I have proposed algorithms for improving the efficiency and sparsity of embedded deep learning.

The **efficiency** and **sparsity** of DL models are two main goals to enable their deployment on embedded systems. To achieve the these goals **algorithmic** and **hardware** aspects must be considered. This thesis focuses on the algorithmic aspect and has achieved the two goals by improving **optimization** and **model compression** algorithms. Nevertheless consideration of the hardware aspects, such as the capabilities offered by specialist architectures (e.g. FPGA, ASIC), has potential to further enhance the application of the algorithms developed in this thesis. The relationship between these concepts is shown in Figure 7.1

**Compression** To save memory footprints of deep learning models, I have proposed the method **Sparse SVRG** (SSVRG) that can efficiently reduce the storage and energy required by large neural networks when running inference, which was introduced in Chapter 3. With careful removal of unnecessary model parameters, SSVRG is able to effectively compress a neural network without sacrificing prediction accuracy. SSVRG has two steps including optimization and compression which is achieved through regularization. In the optimization process, I improved the SVRG optimization method by using a variance controller $\lambda$, and also provided a flexible hybrid learning rate to cooperate with $\lambda$ for fast training. Then in the compression process, my optimization method can efficiently work with cumulative $\ell_1$-regularizer, which can improve the performance of the model on sparse learning representations.
Chapter 7 Conclusions and Further Work

My method SSVRG is verified on both convolutional neural networks and deep neural networks. Compared with other compression method, the key benefit of my method is that compact, sparse and efficient neural networks can be automatically produced by SSVRG at training time, which has no need to retrain the whole networks resulting in reduced training iterations over comparable methods. The second benefit is that my compression technique combines the variance controlled optimization with the regularization process that not only can adjust weights to achieve high accuracy but also can boost the network generalization (Neyshabur et al., 2017; Kim et al., 2016; Zhao et al., 2017).

**Optimization** To reduce the computational cost of deep learning models in their training phase, I proposed two Variance Reduction (VR)-based optimization algorithms, which can efficiently accelerate the rate of convergence. In my algorithms, I considered improvements to standard VR-based optimization (e.g. SVRG) in deep learning models by four aspects: controlling variance reduction, balancing unbiased/biased estimations, scheduling an appropriate learning rate, and estimating an appropriate batch/mini-batch size. Focusing on the first three aspects, I proposed my first optimization algorithm *Integrated SVRG*\(^+\) (*ISVRG*\(^+\)), which was introduced in Chapter 5. I theoretically determined that a hyper-parameter \(\lambda\) working with an adaptive learning rate in each iteration can control the reduced variance of SVRG and balance the trade-off between biased/unbiased estimator. As the IFO computational complexity of standard SVRG as \(n^{2/3}/\epsilon\), *ISVRG*\(^+\) can reduce the result to \(O(\min(1/\epsilon^2, n^{1/4}/\epsilon))\). To further reduce the rate of convergence, I improved *ISVRG*\(^+\) by the last aspect that use batching method to save computations on training samples in each iteration. This idea inspired us to propose a second optimization algorithm *Variance Controlled Stochastic Gradients*
(VCSG) that was introduced in Chapter 6. To analyse VCSG, I theoretically determined that an adjustable batch size bounded by controlled reduced variance can effectively work with both $\lambda$, step size and mini-batch so as to converge faster to a stationary point on non-convex problems. Finally, VCSG can achieve IFO result as $\mathcal{O}(\min(1/\epsilon^{5/4}, n^{1/8}/\epsilon))$, which to my knowledge improves the current best IFO result of SPIDER algorithm (Fang et al., 2018; Wang et al., 2018) by $\mathcal{O}(\min(1/\epsilon^{1/4}, n^{3/8}/\epsilon))$.

Whilst my methods reduce the size and increase the efficiency of deep learning models there are still several areas of future work to investigate that could further improve these aspects. My compression method has focused on the reduction of the number of weights, but has not considered how one might be able to also reduce the bit representation of the weights and signals in the network. This has potential to further reduce the memory and computational requirements and it is worth exploring whether this could be done without compromising performance. Some work has already been undertaken in this area. Al-Hami et al. (2018); Lin et al. (2016) has shown that a floating point representation can more efficiently represent weights rather than fixed point, but the activation parameters can be better represented by fixed points. Another factor to consider is to allow the representation for weights to vary by layer and to determine a suitable way to calculate these deviations. For example, convolutional layers may require higher computing precision. On the other hand, fixed points may be a more suitable representation for weights in fully-connected layers. However, due to its higher power usage, floating point may be unavailable in some embedded systems and fixed point may be the only option. In an alternative scenario such as re-configurable computing there will be design choices between realising fixed or floating computing blocks and their inherent cost (Fifield et al., 2016).

A more ambitious goal is to consider the possibility of providing embedded systems with algorithms that can also train the networks. This is a worthy goal and would enable applications to adapt to local data without the need for network connectivity. Furthermore some applications with privacy concerns would also benefit from such an approach where the application needs to adapt to local user data (Han et al., 2015). In such scenarios privacy issues may govern that the data is retained on the device to maintain privacy such as wearable medical devices monitoring users with different health situations.
Bibliography


