ANALYTICAL GUARANTEES FOR REDUCED PRECISION FIXED-POINT MARGIN HYPERPLANE CLASSIFIERS

BY

CHARBEL SAKR

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Adviser:

Professor Naresh Shanbhag

ABSTRACT

Margin hyperplane classifiers such as support vector machines are strong predictive models having gained considerable success in various classification tasks. Their conceptual simplicity makes them suitable candidates for the design of embedded machine learning systems. Their accuracy and resource utilization can effectively be traded off each other through precision. We analytically capture this trade-off by means of bounds on the precision requirements of general margin hyperplane classifiers. In addition, we propose a principled precision reduction scheme based on the trade-off between input and weight precisions. Our analysis is supported by simulation results illustrating the gains of our approach in terms of reducing resource utilization. For instance, we show that a linear margin classifier with precision assignment dictated by our approach and applied to the 'two vs. four' task of the MNIST dataset is $\sim 2\times$ more accurate than a standard 8 bit low-precision implementation in spite of using $\sim 2 \times 10^4$ fewer 1 bit full adders and $\sim 2 \times 10^3$ fewer bits for data and weight representation.

To my parents, for their love and support.

ACKNOWLEDGMENTS

First and foremost, I would like to express my most sincere gratitude to my advisor, Professor Naresh Shanbhag. This work would have not been possible without his great guidance and the useful discussions we have had since I arrived at UIUC. Most importantly, I am very grateful to Prof. Shanbhag for encouraging me to explore the new and exciting area of machine learning in silicon.

I feel very lucky to have been mentored in my first year of graduate school by my friend Dr. Sai Zhang, from whom I have learned a lot. I am also very thankful to my friend Dr. Peter Kairouz, who helped me adapt to life at UIUC when I first joined.

None of the presented work would have been possible if it was not for the discussions and collaborations with my lab mates in the Shanbhag research group: Sujan Gonugondla, Ameya Patil, Dr. Yingyan Lin, Dr. Mingu Kang, Sungmin Lim, and Dr. Yongjune Kim.

Finally, I would like to express my deepest gratitude to my parents, Toufic and Léna. I would never have achieved anything if it was not for all the sacrifices they have made for me. This thesis is dedicated to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CPU	Central Processing Unit
DP	Dot Product
FA	Full Adder
FL Sim	Floating-point Simulation
FX Sim	Fixed-point Simulation
GLB	Geometric Lower Bound
GPU	Graphics Processing Unit
LMS	Least Mean Square
MAC	Multiply and Accumulate
MNIST	Modified National Institute of Standards and Technology Database
MVM	Matrix Vector Multiplication
NLIM	Non-Linear Input Mapping
NLOM	Non-Linear Output Mapping
PUB	Probabilistic Upper Bound
RBF	Radial Basis Function
SGD	Stochastic Gradient Descent
SQNR	Signal-to-Quantization Noise Ratio
SVM	Support Vector Machine
TP	Tensor Product
TPU	Tensor Processing Unit

CHAPTER 1

INTRODUCTION

1.1 Motivation

Because of their high computational and storage complexity, today, machine learning systems are deployed in the cloud and on large-scale general-purpose computing platforms such as CPU and GPU-based clusters [1]. A key challenge today is to incorporate inference capabilities into untethered (embedded) platforms such as cell phones, autonomous unmanned vehicles, and wearables. Such platforms, however, have stringent limits on available energy, computation, and storage resources. Enabling such *on-device intelligence* necessitates a fresh look at the design of resource-constrained learning algorithms and architectures.

Precision of data, weight vector, and internal signal representations in a machine learning implementation have a profound impact on its overall complexity. Not surprisingly, recent works have empirically studied the effect of moderate [2] and heavy [3–6] quantization on the performance of learning systems. Signal-to-quantization noise ratio (SQNR) has also been used as a metric to understand precision-accuracy trade-offs for the forward path [7] and training [8] of deep neural networks. These works raise the following questions: Is there a systematic way of choosing the minimum precision of data, weights, and internal signal representations? Are these precisions interdependent? How should one choose the precision of the training algorithm? Our work addresses these questions for the general case of margin hyperplane classifiers.

In fact, the questions listed above have been answered for the popular least mean-squared (LMS) adaptive filter [9–15] in the context of digital signal processing and communications systems. It turns out that there is a trade-off between data and coefficient precision in order to achieve a desired

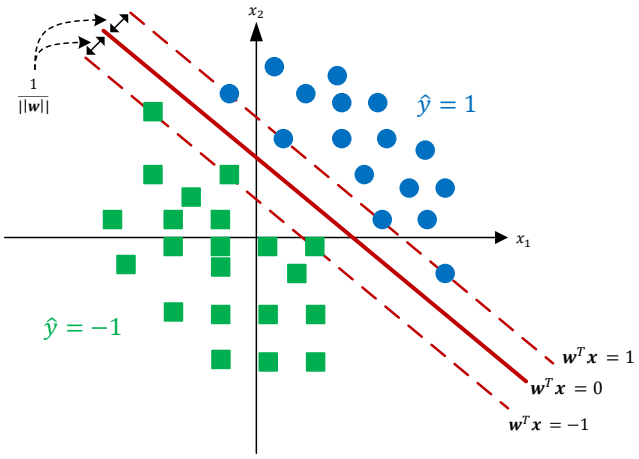

Figure 1.1: Illustration of the geometry of a margin hyperplane classifier.

SQNR at the output. Furthermore, the precision of the LMS weight update block needs to be greater than that of the coefficient in the filter to avoid a premature termination of the convergence process. This thesis leverages these insights in the design of fixed-point machine learning algorithms.

1.2 Background

1.2.1 The Classification Problem

Given a feature vector $\mathbf{x}$ of dimension D, with a corresponding unknown true label $y \in \{\pm 1\}$, it is desired to predict the class it belongs to. There are several approaches to solving the problem, amongst which are hyperplane classifiers. These separate the feature space by means of a hyperplane and assign a predicted label $\hat{y}$ based on the relative position of the feature vector with respect to the hyperplane. For generalizability, it is often desired to determine a maximum margin separating hyperplane in the feature space (see Fig. 1.1) as is the case for SVMs [16]. The classifier is said to have a soft margin when some of the feature vectors are allowed to lie within the margin and hence may be misclassified.

The simplest, yet effective, such classifier is a linear classifier which predicts

the label as follows:

$$\mathbf{w}^T\mathbf{x} + b \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} 0 \tag{1.1}$$

where $\mathbf{w}$ is the weight vector and b is the bias term. For notational convenience, we reformulate (1.1) into an equivalent *affine* form:

$$\mathbf{w}^T\mathbf{x} \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} 0 \tag{1.2}$$

by absorbing the bias term b into the weight vector and extending the feature space by one: $\mathbf{w} \leftarrow \begin{bmatrix} b & \mathbf{w}^T \end{bmatrix}^T$ and $\mathbf{x} \leftarrow \begin{bmatrix} 1 & \mathbf{x}^T \end{bmatrix}^T$. Often, data statistics make the classification problem linearly non-separable in the input feature space. To circumvent this issue, one method is to map the input feature space to a higher dimension, i.e., $\mathbf{x} \rightarrow \phi(\mathbf{x})$ such that it becomes linearly separable, i.e.:

$$\mathbf{w}^T\phi(\mathbf{x}) \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} 0 \tag{1.3}$$

We refer to this method as non-linear input mapping (NLIM). The non-linear map $\phi(\mathbf{x})$ typically lifts the data to a much higher dimension. Consequently, the dimension of the weight vector $\mathbf{w}$ in (1.3) is greater than that in (1.2).

Sometimes, it is impractical to map the input into a space where the data is linearly separable. The reason is that the corresponding dimension may be too large or even infinite. A remedy to this problem is the popular kernel trick which we refer to as non-linear output mapping (NLOM). The idea is to perform the similarity operation (projection, distance, etc.) in the original (lower dimensional) feature space and then apply a kernel to the result, as shown below:

$$\sum_{i=1}^{N_s} \alpha_i K(\mathbf{s}_i, \mathbf{x}) + b \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} 0 \tag{1.4}$$

where $K(\cdot,\cdot)$ is the kernel, b is a bias term, $\mathbf{s}_i$'s are called support vectors, and α_i's are the N_s constants associated with the N_s support vectors. The support vectors found during training characterize the margin of the separating region. Their number depends on the size of the training set and the dimensionality of the input and output spaces. Note that since the support vectors are obtained through training, it is not possible to absorb the bias term into the kernel in a similar fashion as in (1.2).

For the dot product kernel $K(\mathbf{s}_i, \mathbf{x}) = \mathbf{s}_i^T\mathbf{x}$, it can be seen that (1.2) and (1.4) are identical by letting $\mathbf{w} = \sum_{i=1}^{N_s} \alpha_i \mathbf{s}_i$ and extending the feature space by one. A slightly more sophisticated kernel is the polynomial kernel: $K(\mathbf{s_i}, \mathbf{x}) = (\mathbf{s}_i^T\mathbf{x})^d$ where d is the order of the polynomial. Another popular kernel is the radial basis function (RBF): $K(\mathbf{s}_i, \mathbf{x}) = \exp(-\frac{1}{2}\left\|\mathbf{s}_i - \mathbf{x}\right\|^2)$. This kernel maps the data into an infinite-dimensional space.

In [17], a reformulation for the second order polynomial kernel SVM was proposed. Essentially, using the fact that $(\mathbf{s_i}^T\mathbf{x})^2 = \mathbf{x}^T(\mathbf{s}_i\mathbf{s}_i^T)\mathbf{x}$, one can reformulate (1.4) as:

$$\mathbf{x}^T\mathbf{K}\mathbf{x} \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} 0 \tag{1.5}$$

where $\mathbf{K} = \sum_{i=1}^{N_s} \alpha_i \mathbf{s}_i\mathbf{s}_i^T$. Note that extending the feature space by one allowed us to absorb the bias term into the matrix $\mathbf{K}$. This reformulation is attractive for of two reasons: (1) the computational cost reduces from N_s inner products in $\mathcal{R}^D$ to $D+1$ inner products in $\mathcal{R}^D$ (typically $N_s \gg D$); (2) there is no need to store any support vectors.

1.2.2 Learning Classifier Parameters

It is possible to train hyperplane margin classifiers on the fly using the stochastic gradient descent (SGD) algorithm. For a linear classifier, the instantaneous loss function is:

$$Q(y_n, \mathbf{x}_n, \mathbf{w}) = \lambda \left\|\mathbf{w}\right\|^2 + \max\{0, 1 - y_n\mathbf{w}^T\mathbf{x}_n\} \tag{1.6}$$

where y_n is the true label corresponding to the n^{th} sample $\mathbf{x}_n$. The hinge loss term in (1.6) contributes to the margin and λ is a regularizer. The optimum weight vector $\mathbf{w}$ that minimizes this loss function can be determined using SGD via the following updates [18]:

$$\mathbf{w}_{n+1} = (1 - \gamma\ \lambda)\mathbf{w}_n + \begin{cases} 0 & \text{if } \ y_n\mathbf{w}_n^T\mathbf{x}_n > 1, \\ \gamma\ y_n\ \mathbf{x}_n & \text{otherwise} \end{cases} \tag{1.7}$$

For NLIM classifiers, the weights can be trained in a similar manner as follows:

$$\mathbf{w}_{n+1} = (1 - \gamma\ \lambda)\mathbf{w}_n + \begin{cases} 0 & \text{if } \ y_n \mathbf{w}_n^T \phi(\mathbf{x}_n) > 1, \\ \gamma\ y_n\ \phi(\mathbf{x}_n) & \text{otherwise} \end{cases} \tag{1.8}$$

For NLOM classifiers, we will consider only the decision directed mode. This is because NLOM requires knowledge of the support vectors that arise during training, and hence obtaining an SGD procedure to train such a system is difficult.

Finally, the quadratic form of (1.5) suggests a straightforward SGD training method. Indeed, as the gradient $\nabla_{\mathbf{K}} \left(\mathbf{x}^T \mathbf{K} \mathbf{x} \right) = \mathbf{x}\mathbf{x}^T$, the update equation is given by:

$$\mathbf{K}_{n+1} = (1 - \gamma\ \lambda)\mathbf{K}_n + \begin{cases} 0 & \text{if } \ y_n \mathbf{x}_n^T \mathbf{K} \mathbf{x}_n > 1, \\ \gamma\ y_n\ \mathbf{x}_n \mathbf{x}_n^T & \text{otherwise} \end{cases} \tag{1.9}$$

where L_2 regularization is applied to $\mathbf{K}$.

Figure 1.2 depicts the architectures of the various classifiers considered with online training weight update blocks shown for linear, NLIM, and quadratic form classifiers. The multiplications and additions are element-wise. The precision assignment per dimension for each signal considered in the upcoming analysis is highlighted. These are input precision B_X, weight precision B_F, and weight update precision B_W. In this thesis, we obtain bounds on these precisions in order to achieve a desired level of accuracy.

1.3 Reduced Precision Machine Learning

Traditional high-precision computations implemented in modern CPUs employ 64 bit floating-point representations and arithmetic. In the past decade, GPUs have gained increased popularity as machine learning accelerators thanks to their parallelization capabilities. Most GPUs employ 32 bit floating-point representations and arithmetic. Such high precision severely impacts the costs of computation, storage, and communications associated with the implementations of machine learning algorithms. Understandably, there has been a wave of research in the recent past that considers other models of

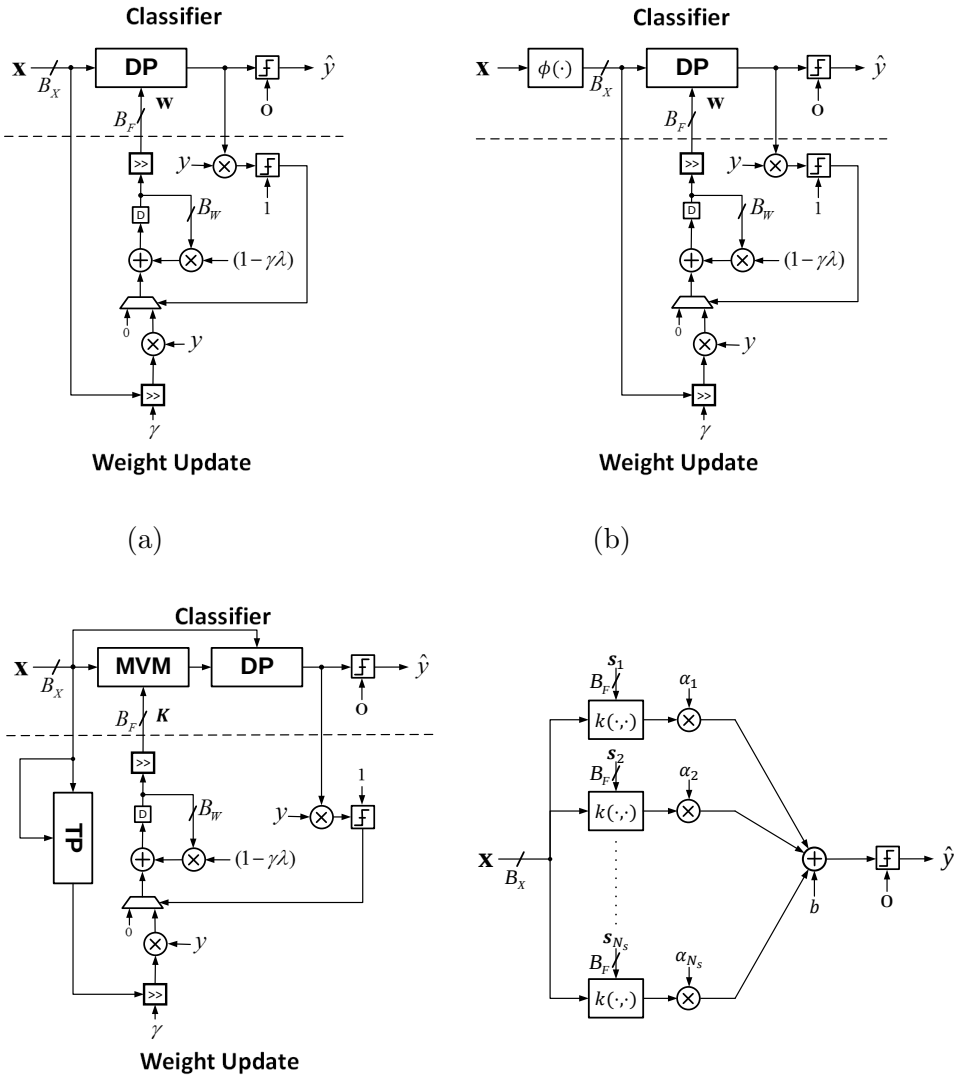

Figure 1.2: Architectures of various margin hyperplane classifiers showing the precision assignments per dimension, and the classifier and weight update blocks for: (a) linear, (b) NLIM, (c) quadratic form, and (d) the NLOM classifier. DP denotes a dot product block, MVM denotes a matrix vector multiplier, and TP denotes a tensor product block.

computations and representations at much lower precisions.

A common approach to representing and computing in reduced precision is to employ fixed-point quantization [19]. This approach was used in some popular machine learning accelerators such as Eyeriss [20], the Diannao family [21], and Google's tensor processing unit (TPU) [22]. In fixed-point,

quantization levels are scattered uniformly across the quantization domain. Fixed-point implementations are attractive because they are efficient. Furthermore, one advantage of fixed-point quantization is the extensive work on quantization noise analysis [9, 10, 23] generally allowing for an intuitive understanding of the precision requirements of machine learning algorithms [19] as will be shown in the remainder of this thesis.

One possibility is to consider reduced precision floating-point representations. While it has not been thoroughly investigated as of today, floating-point quantization seems promising due to its better dynamic range, making it more robust at the algorithm level. The main difficulty remains in the multiplicative noise model traditionally considered [23, 24]. Such a model is both pessimistic and mathematically less tractable than that of fixed-point quantization noise. Furthermore, the costs of implementing reduced precision floating-point algorithms are comparable to fixed-point implementations, giving the former no clear advantage.

Recently, some empirical works have demonstrated the possibility of using extreme measures in reduced precision. Those include log quantization or power of 2 quantization [25], binarization [5], and ternarization [26]. Such approaches seem to reduce the costs of implementation by a great deal; nevertheless, it is observed that the deployment of such quantization schemes is only made possible through algorithmic extensions rendering the overall complexity worse than a fixed-point counterpart [27]. In addition, the effects of those quantization methodologies are still not well understood analytically. Hence, in the remainder of this thesis we shall focus on reduced fixed-point precision in the context of the design and implementation of hyperplane classifiers.

1.4 Contributions

In this thesis, we propose an analytical framework to predict precision vs. accuracy trade-offs in the design of fixed-point learning algorithms thereby eliminating the need for trial-and-error. A simplified version of this work, addressing the issue of fixed-point linear support vector machines (SVM), can be found in [19]. Those results are extended in this thesis to obtain a complete and rigorous set of bounds for general margin hyperplane classifiers.

Specifically, we consider classifiers using non-linear input mapping (NLIM), non-linear output mapping (NLOM), also known as kernels, and quadratic forms. We analyze the trade-off between data and weight precisions. In addition, we propose a principled approach to reduce the precision of various algorithmic parameters while maintaining fidelity to the ideal (floating-point) accuracy. We quantify the benefits of this precision reduction in terms of computational (# of 1 b full adders) and representational (# of bits) costs. We test and validate all our results through simulations on the Breast Cancer Dataset from the UCI Machine Learning Repository [28] and the MNIST Dataset for handwritten character recognition [29].

CHAPTER 2

PRECISION ANALYSIS OF FIXED-POINT HYPERPLANE CLASSIFIERS

2.1 Classifier Precision

In this section, we assume that the classifier has been pretrained in floating-point and we analytically study how much can it be quantized and how its accuracy varies with its precision. In our analysis, we assume without loss of generality that all quantities of interest lie between ± 1. This can be achieved for data via scaling and for the weights by forcing saturation upon each iteration. Finally, unless otherwise stated, we allow the precision of internal signals to grow arbitrarily. That is, we do not incorporate intermediate round-offs in our analysis. The bit growth phenomenon only adds a logarithmic term to the computational complexity. The upcoming analysis can be extended by considering round-off noise terms but would become much less tractable. Figure 1.1 shows the geometric configuration of a margin hyperplane classifier. This illustration reveals interesting insights upon which we build our analysis.

2.1.1 Geometric Lower Bounds

The first result exploits the *geometry* of the classifier to provide a lower bound on the precision which guarantees that the quantized feature vectors lying outside the margin are classified correctly. The geometric lower bounds (GLB) are conservative in the sense that they are sufficient conditions for fixed-point classifiers to have an average accuracy close to their trained floating-point counterparts.

Finite precision computation modifies (1.2) to:

$$(\mathbf{w}+\mathbf{q}_w)^T(\mathbf{x}+\mathbf{q}_x) \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} 0 \tag{2.1}$$

where $\mathbf{q}_x \in \mathcal{R}^D$ and $\mathbf{q}_w \in \mathcal{R}^D$ are the quantization noise terms in $\mathbf{x}$ and $\mathbf{w}$ respectively. Each element of $\mathbf{q}_x$, except the first one, is a random variable uniformly distributed with support $[-\frac{\Delta_X}{2}, \frac{\Delta_X}{2}]$, where $\Delta_X = 2^{-(B_X-1)}$ is the input quantization step. The first term in $\mathbf{q}_x$ is zero. Similarly, each element of $\mathbf{q}_w$ is a random variable uniformly distributed with support $[-\frac{\Delta_F}{2}, \frac{\Delta_F}{2}]$, where $\Delta_F = 2^{-(B_F-1)}$ is the coefficient quantization step. This uniform assumption is standard [10], has been found to be accurate in signal processing and communications systems, and is validated by the experimental results in our thesis. Note that quantization perturbs both the feature vector $\mathbf{x}$ and the separating hyperplane defined by $\mathbf{w}$.

In what follows, for a feature vector $\mathbf{x}$, let us denote by $\hat{y}_{fl}(\mathbf{x})$ the label predicted by the floating-point classifier and by $\hat{y}_{fx}(\mathbf{x})$ the one predicted by the corresponding fixed-point classifier. The notation $\mathbf{a}_{\_}$ denotes a vector $\mathbf{a}$ without the first element. We start with the linear classifier and consider the GLB on B_X.

Theorem 1 (Geometric Lower Bound on B_X for a Linear Classifier)**.** *Given B_F, and $\mathbf{w}$, $\forall \mathbf{x} \in \mathcal{R}^D$ such that $|\mathbf{w}^T\mathbf{x}| > 1$, $\hat{y}_{fx}(\mathbf{x}) = \hat{y}_{fl}(\mathbf{x})$ if*

$$B_X > \log_2\left(\frac{||\mathbf{w}_{\_}||\sqrt{D-1}}{1-2^{-B_F}||\mathbf{x}||\sqrt{D}}\right) \tag{2.2}$$

where $B_F > \log_2\left(||\mathbf{x}||\sqrt{D}\right)$.

Proof. See this chapter's addendum (Section 2.3.1). □

The GLB for a linear classifier reveals the following insights: (1) larger margin (i.e., smaller $||\mathbf{w}||$ in Fig. 1.1) allows a greater reduction of B_X, (2) there is a trade-off between B_X and B_F, and (3) input precision B_X increases with dimension D and $||\mathbf{x}||$. Figure 2.1 shows the trade-off between B_X and B_F for several values of the dimension D. In each case, the starting value of B_F corresponds to the condition of Theorem 1.

Note that Theorem 1 is specific to a single feature vector $\mathbf{x}$ and can be extended to a dataset.

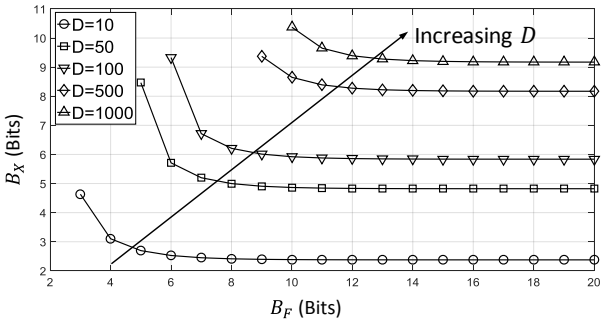

Figure 2.1: Illustration of the GLB for a linear classifier showing trade-off between input (B_X) and weight (B_F) precisions, and dependence on the dimension (D). In this example, the values of $||\mathbf{w}_{\_}||$ and $||\mathbf{x}||$ are taken to be the mean norm of the corresponding vectors assuming each entry is random and uniformly distributed between -1 and 1.

Corollary 1.1 (Geometric Lower Bound on B_X for a Linear Classifier on a Dataset)**.**
Given B_F, and $\mathbf{w}$, $\forall \mathbf{x} \in \mathcal{R}^D$ such that $|\mathbf{w}^T\mathbf{x}| > 1$, $\hat{y}_{fx}(\mathbf{x}) = \hat{y}_{fl}(\mathbf{x})$ if

$$B_X > \log_2 \left(\frac{||\mathbf{w}_{\_}||\sqrt{D-1}}{1 - 2^{-B_F} \max\limits_{\mathbf{x}\in\mathcal{X}} ||\mathbf{x}||\sqrt{D}} \right) \tag{2.3}$$

where $B_F > \log_2 \left(\max\limits_{\mathbf{x}\in\mathcal{X}} ||\mathbf{x}||\sqrt{D} \right)$.

In Table 2.1, we list the GLB for the different classifiers. These lower bounds on precision guarantee that any feature vector lying outside the margin of a floating-point classifier is identically classified by the fixed-point counterpart. The GLB is a guideline on the safe exploitation of the margin. Note that we may derive a GLB on B_X given B_F or on B_F given B_X. Each of these GLBs can be extended for a classifier operating on a dataset as in Corollary 1.1. Please refer to this chapter's addendum (Section 2.3.1) for the proofs.

Figure 2.2 shows a comparison across linear, NLIM, and quadratic form classifiers. It appears that the NLIM classifier has the highest input precision requirement, followed by the quadratic form and linear classifiers, in that order. The quadratic form classifier seems to have the highest weight precision requirement.

Table 2.1: Table of GLBs for linear, NLIM, NLOM, and quadratic form classifiers. The classifier parameters, for each case, are as defined in Section 1.2. The precision assignments are as shown in Fig. 1.2. Under these conditions we have $\hat{y}_{fx}(\mathbf{x}) = \hat{y}_{fl}(\mathbf{x})$. The first row identical to Theorem 1. The proofs of these bounds can be found in this chapter's addendum (Section 2.3.1).

Linear classifier	
GLB on B_X given B_F	$B_X > \log_2\left(\frac{\|\|\mathbf{w}_{\text{-}}\|\|\sqrt{D-1}}{1-2^{-B_F}\|\|\mathbf{x}\|\|\sqrt{D}}\right)$, where $B_F > \log_2\left(\|\|\mathbf{x}\|\|\sqrt{D}\right)$.
GLB on B_F given B_X	$B_F > \log_2\left(\frac{\|\|\mathbf{x}\|\|\sqrt{D}}{1-2^{-B_X}\|\|\mathbf{w}_{\text{-}}\|\|\sqrt{D-1}}\right)$, where $B_X > \log_2\left(\|\|\mathbf{w}_{\text{-}}\|\|\sqrt{D-1}\right)$.
Remarks	$\mathbf{x} \in \mathcal{R}^D$, $\|\mathbf{w}^T\mathbf{x}\| > 1$

NLIM classifier	
GLB on B_X given B_F	$B_X > \log_2\left(\frac{\|\|\mathbf{w}_{\text{-}}\|\|\sqrt{D_\phi-1}}{1-2^{-B_F}\|\|\phi(\mathbf{x})\|\|\sqrt{D_\phi}}\right)$, where $B_F > \log_2\left(\|\|\phi(\mathbf{x})\|\|\sqrt{D_\phi}\right)$.
GLB on B_F given B_X	$B_F > \log_2\left(\frac{\|\|\phi(\mathbf{x})\|\|\sqrt{D_\phi}}{1-2^{-B_X}\|\|\mathbf{w}_{\text{-}}\|\|\sqrt{D_\phi-1}}\right)$, where $B_X > \log_2\left(\|\|\mathbf{w}_{\text{-}}\|\|\sqrt{D_\phi-1}\right)$.
Remarks	$\phi \in \mathcal{R}^{D_\phi}$, $\|\mathbf{w}^T\phi(\mathbf{x})\| > 1$

NLOM classifier	
GLB on B_X given B_F	$B_X > \log_2\left(\frac{\sqrt{D}\left\|\sum_{i=1}^{N_s}\alpha_i\nabla_{\mathbf{x}}K(\mathbf{s}_i,\mathbf{x})\right\|}{1-2^{-B_F}\sqrt{D}\sum_{i=1}^{N_s}\left\|\alpha_i\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{x})\right\|}\right)$, where $B_F > \log_2\left(\sqrt{D}\sum_{i=1}^{N_s}\left\|\alpha_i\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{x})\right\|\right)$.
GLB on B_F given B_X	$B_F > \log_2\left(\frac{\sqrt{D}\sum_{i=1}^{N_s}\left\|\alpha_i\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{x})\right\|}{1-2^{-B_X}\sqrt{D}\left\|\sum_{i=1}^{N_s}\alpha_i\nabla_{\mathbf{x}}K(\mathbf{s}_i,\mathbf{x})\right\|}\right)$, where $B_X > \log_2\left(\sqrt{D}\left\|\sum_{i=1}^{N_s}\alpha_i\nabla_{\mathbf{x}}K(\mathbf{s}_i,\mathbf{x})\right\|\right)$.
Remarks	$\mathbf{x} \in \mathcal{R}^D$, $\left\|\sum_{i=1}^{N_s}\alpha_i K(\mathbf{s}_i,\mathbf{x}),\ +b\right\| > 1$

Quadratic form classifier	
GLB on B_X given B_F	$B_X > \log_2\left(\frac{2\|(\mathbf{Kx})_{\text{-}}\|\sqrt{D-1}}{1-2^{-B_F}D\|\mathbf{x}\|^2}\right)$, where $B_F > \log_2\left(D\|\mathbf{x}\|^2\right)$.
GLB on B_F given B_X	$B_F > \log_2\left(\frac{D\|\mathbf{x}\|^2}{1-2^{-(B_X-1)}\|(\mathbf{Kx})_{\text{-}}\|\sqrt{D-1}}\right)$, where $B_X > \log_2\left(2\|(\mathbf{Kx})_{\text{-}}\|\sqrt{D-1}\right)$.
Remarks	$\mathbf{x} \in \mathcal{R}^D$, $\|\mathbf{x}^T\mathbf{Kx}\| > 1$

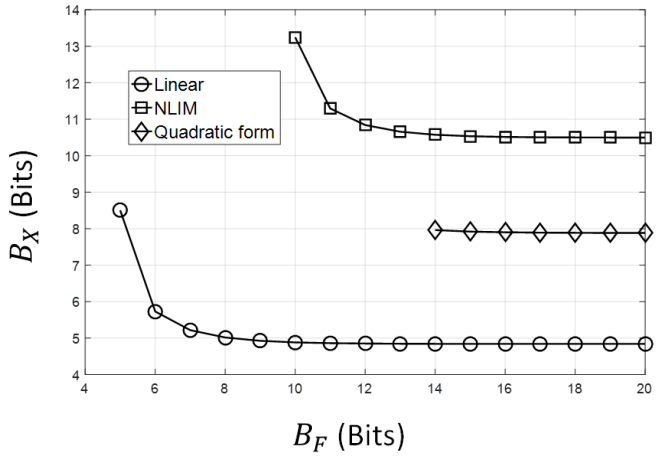

Figure 2.2: Comparison of the GLB across classifier types. The input dimension is arbitrarily chosen to be $D = 50$. For the NLIM classifier, the map considered is the second order polynomial one ($D_\phi = D^2$). For each case, inputs and weights are random, uniformly distributed between -1 and 1, per dimension. To obtain average bounds, 1000 datasets are generated. The GLBs plotted are the mean bounds in each case.

2.1.2 Probabilistic Upper Bounds

The GLB provides a lower bound on the precision requirements to ensure that fixed-point decisions are identical to floating-point for feature vectors lying outside the margin of the classifier. However, it does not provide quantitative guarantees on the fixed-point accuracy. Probabilistic upper bounds (PUBs), introduced next, upper bound the worst-case fixed-point accuracy for a given precision.

In what follows, we employ capital letters for random variables. We define probability of mismatch p_m between the decisions made by the floating-point and fixed-point algorithms as $p_m = \Pr\{\hat{Y}_{fx} \neq \hat{Y}_{fl}\}$, where $\hat{Y}_{fx}$ is the output of the fixed-point classifier and $\hat{Y}_{fl}$ is the output of the floating-point classifier. A small mismatch probability indicates that the classification accuracy of the fixed-point algorithm is very close to that of the floating-point algorithm. Indeed, if we know the accuracy of the floating-point system quantified by its probability of error $p_{e,fl} = \Pr\{\hat{Y}_{fl} \neq Y\}$ (Y is the true output), then we can obtain an upper bound on the fixed-point probability of error $p_{e,fx} = \Pr\{\hat{Y}_{fx} \neq Y\}$:

Table 2.2: List of E_1 and E_2 appearing in the PUB (Theorem 2) for linear, NLIM, NLOM, and quadratic form classifiers. For each case, the classifier parameters, as defined in Section 1.2, are pre-trained in floating-point. The expectations are taken over random inputs. The precision assignments are as shown in Figure 1.2.

Classifier type	E_1	E_2
Linear	$\mathbb{E}\left[\frac{\|\mathbf{w}_-\|^2}{\|\mathbf{w}^T\mathbf{X}\|^2}\right]$	$\mathbb{E}\left[\frac{\|\mathbf{X}\|^2}{\|\mathbf{w}^T\mathbf{X}\|^2}\right]$
NLIM	$\mathbb{E}\left[\frac{\|\mathbf{w}_-\|^2}{\|\mathbf{w}^T\phi(\mathbf{X})\|^2}\right]$	$\mathbb{E}\left[\frac{\|\phi(\mathbf{X})\|^2}{\|\mathbf{w}^T\phi(\mathbf{X})\|^2}\right]$
NLOM	$\mathbb{E}\left[\frac{\left\|\sum_{i=1}^{N_s}\alpha_i\nabla_{\mathbf{X}}K(\mathbf{s}_i,\mathbf{X})\right\|^2}{\left\|\sum_{i=1}^{N_s}\alpha_iK(\mathbf{s}_i,\mathbf{X})+b\right\|^2}\right]$	$\mathbb{E}\left[\frac{\sum_{i=1}^{N_s}\left\|\alpha_i\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{X})\right\|^2}{\left\|\sum_{i=1}^{N_s}\alpha_iK(\mathbf{s}_i,\mathbf{X})+b\right\|^2}\right]$
Quadratic form	$4\mathbb{E}\left[\frac{\|(\mathbf{KX})_-\|^2}{\|\mathbf{X}^T\mathbf{KX}\|^2}\right]$	$\mathbb{E}\left[\frac{\|\mathbf{X}\|^4}{\|\mathbf{X}^T\mathbf{KX}\|^2}\right]$

Proposition 1. *The fixed-point probability of error is upper bounded as follows:*

$$p_{e,fx} \leq p_{e,fl} + p_m \tag{2.4}$$

The right-hand side represents the union bound of two events: (1) misclassification, and (2) incorrect classification due to quantization.

Note that $p_{e,fx}$ is the quantity of interest as it characterizes the accuracy of the fixed-point system. While $p_{e,fl}$ is a metric of the algorithm itself and does not depend on quantization, p_m is the term that captures the impact of quantization on accuracy and we hence use it as a proxy for $p_{e,fx}$ in order to evaluate the accuracy of a fixed-point system. In what follows, we obtain analytical upper bounds on p_m as a function of the precision of the fixed-point system.

It turns out that, for all the classifiers considered in Fig. 1.2, the mismatch probability p_m is upper bounded as follows:

Theorem 2 (Probabilistic Upper Bound on p_m)**.** *Given B_X and B_F, the upper bound on the mismatch probability of a hyperplane classifier is given by:*

$$p_m \leq \frac{1}{24}\left(\Delta_X^2 E_1 + \Delta_F^2 E_2\right) \tag{2.5}$$

where, for each case, the values of E_1 and E_2 are listed in Table 2.2.

Proof. See this chapter's addendum (Section 2.3.2). □

In practice, the statistics (i.e., the expected values E_1 and E_2 in (2.5)) are calculated empirically. Note that the mismatch probability bound is increasing in Δ_X and Δ_F, which is expected as higher quantization noise variance leads to increased mismatch between fixed and floating-point algorithms.

Equation (2.5) reveals an interesting trade-off between input precision B_X and weight precision B_F. Indeed, in (2.5), the first term, $\Delta_X^2 E_1$, is the input quantization noise power gain while the second, $\Delta_F^2 E_2$, is the weight quantization noise power gain. Depending on the values taken by E_1 and E_2, it might be that one of the two terms dominates the sum. If so, then it would imply that B_F or B_X is unnecessarily large and hence can be reduced without increasing p_m. An intuitive first step to obtain a tighter upper bound is to make the two terms of comparable order, i.e., set $\Delta_X^2 E_1 = \Delta_F^2 E_2$ by constraining B_X and B_F as follows:

$$B_X - B_F = round\left(\log_2 \sqrt{\frac{E_1}{E_2}}\right) \tag{2.6}$$

where $round()$ denotes the rounding operation. This is an effective way to handle one of the two degrees of freedom introduced by (2.5).

One way to employ (2.6) is to consider minimizing the upper bound in (2.5) subject to the constraint $B_X + B_F = c$ for some constant c. Indeed, it can be shown that (2.6) would be a necessary condition of the corresponding solution.

From the expressions of E_1 and E_2 in Table 2.2 we note two points. First, for each classifier, the denominator within the expectation operator represents the confidence in classification. This means that, the better the classifier separates the data, the smaller E_1 and E_2 are expected to be. Hence, better data separability implies better tolerance to quantization, which is to be expected. Second, the numerators represent functions of the magnitudes of weight and input vectors in the decision space. Such magnitudes are direct functions of dimensionality and margin. Consequently, we may infer that higher dimensionality, or smaller margin, increases the values of E_1 and E_2, and hence leads to an increase in the precision requirements of the classifier. This correlates well with our observations in the case of the GLB.

The results presented in this section provide useful insights to determine suitable precision allocations for inputs and weights. First, the GLBs offer

a condition under which the behavior of a fixed-point system is expected to be similar to that of the corresponding floating-point one. However, they do not provide analytical guarantees on the resulting accuracy. The PUBs do, and (2.6) captures an interesting trade-off between both precisions. In practice, it is possible to use both sets of bounds to efficiently determine minimal precisions for a fixed-point implementation. For example, (2.6) can first be used to determine the optimal difference between B_X and B_F, then the GLB can be used to find a suitable pair of (B_X, B_F). Finally, the PUB can be used to estimate the expected accuracy loss. We shall demonstrate this approach in Chapter 3.

2.2 Precision in Training

In the preceding discussion, all learning parameters were assumed to have been obtained after full-precision training. In practice, it is standard to first implement a floating-point algorithm with desirable convergence properties and then quantize so as to preserve the convergence behavior unaltered. In this section, we consider the problem of finding precision requirements on B_W in the updates when training is done in fixed-point with feedforward precisions set to B_X and B_F.

Consider a linear classifier. Note that the right-hand side of (1.7) includes an attenuation term $(1-\gamma\ \lambda)\mathbf{w}_n$ and an update term equal to 0 or $\gamma\ y_n\ \mathbf{x}_n = \pm\gamma\mathbf{x}_n$ where $x_i \in [-1, 1]$ for $i = 1 \ldots D$. Without loss of generality, we assume that floating point convergence is achieved for $\lambda = 1$ and some small value of γ. Therefore, the attenuation factor $(1 - \gamma\ \lambda)$ is less than or close to unity independent of B_W.

Our next result ensures that the non-zero update term in (1.7) is non-zero during training in spite of weight update quantization.

Theorem 3 (Weight Update Requirements for a Linear Classifier)**.**
The following lower bound on weight update precision B_W is a sufficient condition to ensure full convergence:

$$B_W \geq B_X - \log_2(\gamma) \tag{2.7}$$

when B_X and B_F are the input and coefficient precisions, respectively.

Proof. Each step in (1.7) has magnitude $|\gamma x|$ where x is a scalar value taken by the different components of $\mathbf{x}$. For any non-zero update to remain non-zero after quantization, we need $|\gamma x| > \frac{1}{2} b_{min}$ where $b_{min} = 2^{-(B_W - 1)}$ is the value of the least significant bit (LSB) in the weight update block. So we require $|\gamma||x| > \frac{1}{2} 2^{-(B_W - 1)} = 2^{-B_W}$. This has to be satisfied for any non-zero value of x, but the minimum non-zero value taken by $|x|$ is $2^{-(B_X - 1)}$. So we get $|\gamma| \cdot 2^{-(B_X - 1)} > 2^{-B_W}$, which can be written as:

$$B_W > B_X - 1 - \log_2(\gamma) \Leftrightarrow B_W \geq B_X - \log_2(\gamma)$$

□

Theorem 3 provides a sufficient condition for convergence. As the SGD approximates the true gradient at every step, fewer bits than specified in (2.7) may work occasionally. We refer to these as boundary cases and present a detailed analysis of the corresponding learning behavior in this chapter's addendum (Section 2.3.3).

For a NLIM classifier, the exact same discussion applies. This is because (1.7) and (1.8) are structurally equivalent. For a quadratic form classifier the results change a little bit. Indeed, every entry (i, j) in the matrix $\mathbf{K}$ has an update term equal to $|\gamma x_i x_j|$ in absolute value. The product of two scalars increases the precision by a factor of 2. Following the same argument as in Theorem 3, we have the following result:

Corollary 3.1 (Weight Update Requirements for a Quadratic Form Classifier)**.**
The following lower bound on weight update precision B_W is a sufficient condition to ensure full convergence:

$$B_W \geq 2B_X - \log_2(\gamma) \tag{2.8}$$

when B_X and B_F are the input and coefficient precisions, respectively.

While the required precision for full convergence has been slightly modified in this case, the discussion of the learning behavior for lower precisions is exactly the same as that following Theorem 3. This is because that discussion observes the inputs starting from the most significant bit (MSB) and towards the LSB. The set $[-1, 1]$ being closed under multiplication, the update terms

Table 2.3: Output quantization noise N_o and corresponding upper bounds on its magnitude needed to prove the GLB for each classifier type. Note that $\mathbf{q}_x$, $\mathbf{q}_w$, $\mathbf{q}_{\phi(x)}$, $\mathbf{q}_{s_i}$, $\mathbf{q}_K$ are the quantization noise terms of $\mathbf{x}$, $\mathbf{w}$, $\phi(\mathbf{x})$, $\mathbf{s}_i$, and $\mathbf{K}$, respectively.

Classifier type	Output quantization noise N_o	Upper bound on output quantization noise magnitude $\lvert N_o \rvert$
Linear	$\mathbf{q}_w^T\mathbf{x} + \mathbf{w}^T\mathbf{q}_x$	$2^{-B_F}\lvert\lvert\mathbf{x}\rvert\rvert\sqrt{D}$ $+2^{-B_X}\lvert\lvert\mathbf{w}_{\text{-}}\rvert\rvert\sqrt{D-1}$
NLIM	$\mathbf{q}_w^T\phi(\mathbf{x}) + \mathbf{w}^T\mathbf{q}_{\phi(x)}$	$2^{-B_F}\lvert\lvert\phi(\mathbf{x})\rvert\rvert\sqrt{D_\phi}$ $+2^{-B_X}\lvert\lvert\mathbf{w}_{\text{-}}\rvert\rvert\sqrt{D_\phi-1}$
NLOM	$\sum_{i=1}^{N_s}\alpha_i\mathbf{q}_{s_i}^T\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{x})$ $+\sum_{i=1}^{N_s}\alpha_i\mathbf{q}_x^T\nabla_{\mathbf{x}}K(\mathbf{s}_i,\mathbf{x})$	$2^{-B_F}\sqrt{D}\sum_{i=1}^{N_s}\lVert\alpha_i\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{x})\rVert$ $+2^{-B_X}\sqrt{D}\left\lVert\sum_{i=1}^{N_s}\alpha_i\nabla_{\mathbf{x}}K(\mathbf{s}_i,\mathbf{x})\right\rVert$
Quadratic form	$2\mathbf{q}_x^T\mathbf{K}\mathbf{x} + \mathbf{x}^T\mathbf{q}_K\mathbf{x}$	$2^{-(B_X-1)}\lVert(\mathbf{K}\mathbf{x})_{\text{-}}\rVert\sqrt{D-1}$ $+2^{-B_F}\lVert\mathbf{x}\rVert^2 D$

are just as significant as those in the case of a linear classifier when looking at the MSB and onwards.

2.3 Addendum: Proofs and Boundary Cases

2.3.1 Proofs of Geometric Lower Bounds

The proof of the GLB is done in three steps:

- Step 1: Determine the output quantization noise N_o at the output of the classifier. In this step, we neglect cross products of quantization noise terms as their contribution is very small. For the NLOM classifier, this is equivalent to a first order Taylor expansion on the kernel. For the linear classifier we have:

$$N_o = \mathbf{q}_w^T\mathbf{x} + \mathbf{w}^T\mathbf{q}_x$$

- Step 2: Upper bound the magnitude of N_o using the triangle and Cauchy-Schwarz inequalities. Input quantization noise terms are upper bounded by 2^{-B_X} and weight quantization noise terms by 2^{-B_F}. For

the linear classifier we have:

$$\begin{aligned} |N_o| &\leq |\mathbf{q}_w^T \mathbf{x}| + |\mathbf{w}^T \mathbf{q}_x| \\ &\leq \|\mathbf{q}_w\| \, \|\mathbf{x}\| + \|\mathbf{w}\| \, \|\mathbf{q}_x\| \\ &\leq 2^{-B_F} ||\mathbf{x}|| \sqrt{D} + 2^{-B_X} ||\mathbf{w}_{\text{-}}|| \sqrt{D-1}. \end{aligned}$$

- Step 3: Set the upper bound on $|N_o|$ to be less than the functional margin of the classifier which is equal to 1 (see Fig. 1.1).

Step 3, up to a rearrangement of terms, is equivalent to the GLB as described in Section 2.1. In Table 2.3, we list the output quantization noise and corresponding upper bound for each classifier type.

2.3.2 Proofs of Probabilistic Upper Bounds

The PUB is proved in 4 steps:

- Step 1: For a single input, obtain the total output quantization noise N_o. This is identical to the Step 1 in the proof of the GLB above. This output quantization noise is a sum of independent input and weight quantization noise terms.

- Step 2: Compute the variance ($\sigma_{N_o}^2$) of this output quantization noise. Because of independence, it is the sum of the input ($\sigma_{q_x \to o}^2$) and weight ($\sigma_{q_w \to o}^2$) quantization noise variances referred to the output. For the linear classifier we have:

$$\sigma_{N_o}^2 = \frac{\Delta_X^2}{12} ||\mathbf{w}_{\text{-}}||^2 + \frac{\Delta_F^2}{12} ||\mathbf{x}||^2$$

- Step 3: Use this computed variance and Chebyshev's inequality to determine the probability of the quantization noise being larger in magnitude than the floating-point soft output z_o of the classifier. Because the quantization noise has a symmetric distribution, this probability needs to be divided by 2 (the mismatch is only caused when quantization noise and output have opposing signs). The upper bound is hence

Table 2.4: Input ($\sigma^2_{q_x\to o}$) and weight ($\sigma^2_{q_w\to o}$) quantization noise variances referred to output and classifier floating-point output (z_o).

Class. type	Linear	NLIM	NLOM	Quadratic form
$\sigma^2_{q_x\to o}$	$\frac{\Delta_X^2}{12}\|\|\mathbf{w}_{\text{-}}\|\|^2$	$\frac{\Delta_X^2}{12}\|\|\mathbf{w}_{\text{-}}\|\|^2$	$\frac{\Delta_X^2}{12}\left\|\sum_{i=1}^{N_s}\alpha_i\nabla_{\mathbf{x}}K(\mathbf{s}_i,\mathbf{x})\right\|^2$	$\frac{\Delta_X^2}{3}\|\|(\mathbf{Kx})_{\text{-}}\|\|^2$
$\sigma^2_{q_w\to o}$	$\frac{\Delta_F^2}{12}\|\|\mathbf{x}\|\|^2$	$\frac{\Delta_F^2}{12}\|\|\phi(\mathbf{x})\|\|^2$	$\frac{\Delta_F^2}{12}\sum_{i=1}^{N_s}\|\alpha_i\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{x})\|^2$	$\frac{\Delta_F^2}{12}\|\|\mathbf{x}\|\|^4$
z_o	$\mathbf{w}^T\mathbf{x}$	$\mathbf{w}^T\phi(\mathbf{x})$	$\sum_{i=1}^{N_s}\alpha_iK(\mathbf{s}_i,\mathbf{x})+b$	$\mathbf{x}^T\mathbf{Kx}$

derived as follows:

$$p_m=\frac{1}{2}P(|N_o|>|z_o|)\leq\frac{\sigma^2_{N_o}}{2|z_o|^2}=\frac{\sigma^2_{q_x\to o}+\sigma^2_{q_w\to o}}{2|z_o|^2}$$

- Step 4: Use the law of total probability over the data to obtain the averaged upper bound on p_m. For the linear classifier we obtain:

$$p_m\leq\frac{\Delta_X^2}{24}\mathbb{E}\left[\frac{\|\mathbf{w}_{\text{-}}\|^2}{|\mathbf{w}^T\mathbf{X}|^2}\right]+\frac{\Delta_F^2}{24}\mathbb{E}\left[\frac{\|\mathbf{X}\|^2}{|\mathbf{w}^T\mathbf{X}|^2}\right]$$

For each classifier type, we list the values of $\sigma^2_{q_x\to o}$, $\sigma^2_{q_w\to o}$, and z_o in Table 2.4. The values of E_1 and E_2 in Table 2.2 are equal to $\frac{12}{\Delta_X^2}\mathbb{E}\left[\frac{\sigma^2_{q_x\to o}}{|Z_o|^2}\right]$ and $\frac{12}{\Delta_F^2}\mathbb{E}\left[\frac{\sigma^2_{q_w\to o}}{|Z_o|^2}\right]$ for each classifier type, respectively.

For the quadratic form case, we used the fact that, for a datapoint $\mathbf{x}$, $Var(\mathbf{x}^T\mathbf{q}_K\mathbf{x})=\frac{\Delta_F^2}{12}\|\mathbf{x}\|^4$. This result is proved as follows:

$$\begin{aligned}
&Var(\mathbf{x}^T\mathbf{q}_K\mathbf{x})=\mathbb{E}\left[(\mathbf{x}^T\mathbf{q}_K\mathbf{x})^2\right]=\mathbf{x}^T\mathbb{E}\left[\mathbf{q}_K\mathbf{x}\mathbf{x}^T\mathbf{q}_K^T\right]\mathbf{x}\\
&=\mathbf{x}^T\mathbb{E}\left[\begin{bmatrix}\mathbf{q}_{K,1}^T\mathbf{x}\\ \vdots\\ \mathbf{q}_{K,D}^T\mathbf{x}\end{bmatrix}\begin{bmatrix}\mathbf{q}_{K,1}^T\mathbf{x} & \dots & \mathbf{q}_{K,D}^T\mathbf{x}\end{bmatrix}\right]\mathbf{x}\\
&=\mathbf{x}^T\frac{\Delta_F^2}{12}\|\mathbf{x}\|^2\mathbf{I}_{D\times D}\mathbf{x}=\frac{\Delta_F^2}{12}\|\mathbf{x}\|^4
\end{aligned}$$

where $\mathbf{q}_{K,i}^T$ for $i=1\dots D$ are the row vectors of $\mathbf{q}_K$ and $\mathbf{I}_{D\times D}$ is the identity matrix of size $D\times D$. The fourth equality holds because the quantization terms are independent of each other making the off-diagonal elements of the matrix in the third equation a product of two zero-mean independent terms.

2.3.3 Precision of Training for Boundary Cases

We analyze the learning behavior when the precision is less than the one predicted by Theorem 3.

As $1-\gamma\lambda \approx 1$, we assume $(1-\gamma\lambda)w_{i,n} \approx w_{i,n}$ for $i = 1 \dots D$. Let $K = B_W$ and $M = B_X$. Let $\tilde{x}_{i,n} = y_n x_n$ be the update term per dimension. The following cases describe the corresponding two's complement arithmetic:

Case 1: $B_W = 1 - log_2(\gamma) \Leftrightarrow \gamma = 2^{-(B_W-1)}$ then:

$$\begin{array}{rccccc} \gamma = & 0 & .0 & \dots & 1 & \\ w_{i,n} = & b_{w_0} & .b_{w_1} & \dots & b_{w_{K-1}} & \\ \gamma\tilde{x}_{i,n} = & b_{x_0} & .b_{x_0} & \dots & b_{x_0} & b_{x_1} \dots b_{x_{M-1}} \end{array}$$

where $\{b_{w_i}\}_{i=0}^{K-1}$ and $\{b_{x_i}\}_{i=0}^{M-1}$ are the binary expansions of $w_{i,n}$ and $x_{i,n}$, respectively. Hence, if $\tilde{x}_{i,n} \geq 0 \rightarrow w_{i,n+1} = w_{i,n}$, and if $\tilde{x}_{i,n} < 0 \rightarrow w_{i,n+1} = w_{i,n} - \gamma$. We hence obtain a sign-SGD behavior only when $\tilde{x}_{i,n} < 0$. Therefore, in case I, there is no way to guarantee convergence.

Case 2: $B_W = 2 - log_2(\gamma) \Leftrightarrow \gamma = 2^{-(B_W-2)}$ then:

$$\begin{array}{rccccc} \gamma = & 0 & .0 & \dots & 1 & 0 \\ w_{i,n} = & b_{w_0} & .b_{w_1} & \dots & b_{w_{K-2}} & b_{w_{K-1}} \\ \gamma\tilde{x}_{i,n} = & b_{x_0} & .b_{x_0} & \dots & b_{x_0} & b_{x_1} \quad b_{x_2} \dots b_{x_{M-1}} \end{array}$$

Hence, $\tilde{x}_{i,n} \geq 0.5 \rightarrow w_{i,n+1} = w_{i,n} + 0.5\gamma$, $0 \leq \tilde{x}_{i,n} < 0.5 \rightarrow w_{i,n+1} = w_{i,n}$, $-0.5 \leq \tilde{x}_{i,n} < 0 \rightarrow w_{i,n+1} = w_{i,n} - 0.5\gamma$, and, $-1 \leq \tilde{x}_{i,n} < 0.5 \rightarrow w_{i,n+1} = w_{i,n} - \gamma$. We get a more precise but very noisy estimate of the gradient. We may observe inaccurate convergence.

Case 3: $B_W = -log_2(\gamma) \Leftrightarrow \gamma = 2^{-B_W}$ then:

$$\begin{array}{rccccc} \gamma = & 0 & .0 & \dots & 0 & 1 \\ w_{i,n} = & b_{w_0} & .b_{w_1} & \dots & b_{w_{K-1}} & \\ \gamma\tilde{x}_{i,n} = & b_{x_0} & .b_{x_0} & \dots & b_{x_0} & b_{x_0} \quad b_{x_1} \dots b_{x_{M-1}} \end{array}$$

Hence, if $\tilde{x}_{i,n} \geq 0 \rightarrow w_{i,n+1} = w_{i,n}$, and if $\tilde{x}_{i,n} < 0 \rightarrow w_{i,n+1} = w_{i,n} - 2\gamma$. We again get a sign-SGD behavior only if $\tilde{x}_{i,n} < 0$ but this time the step size has doubled. Again, there is no way to guarantee any sort of convergence in this

case. Things are made even worse because when updates do happen, they are two times greater than in the first case.

2.4 Summary

In this chapter, we have presented an analytical framework to determine the precision requirements of hyperplane classifiers. We have considered both inference and training blocks. For inference, we have presented two bounds on the input and weight precisions based on the geometry and statistics of the learning problem, respectively. For training, we have analyzed the conditions for weight update precision under which convergence in fixed-point is theoretically guaranteed provided it is also guaranteed in floating-point. In the next chapter we will validate these theoretical findings through numerical simulations on real datasets.

CHAPTER 3

SIMULATION RESULTS

In this chapter, we validate the analysis of Chapter 2 via simulation with the UCI Breast Cancer Dataset [28] and the 'two vs. four' task applied to the MNIST Dataset for handwritten character recognition [29].

3.1 Complexity in Fixed-Point

We identify two implementation costs associated with the implementation of signal processing and machine learning systems: the computational cost and the representational cost.

The computational cost is measured in numbers of 1 bit full adders ($\#FA$) to come up with one decision. Dot products are the predominant computing structures in machine learning systems. We assume they are realized in a multiply and accumulate (MAC) fashion where additions and multiplications are implemented using ripple carry adders and Baugh-Wooley multipliers, respectively. Consequently, the number of full adders used to compute the dot product between two D dimensional vectors with entries quantized to B_X bits and B_F bits respectively is

$$DB_XB_F + (D-1)(B_X + B_F + \lceil \log_2(D) \rceil - 1) \tag{3.1}$$

and the output is $(B_X + B_F + \lceil \log_2(D) \rceil)$ bits.

Equation (3.1) describes the computational cost of the linear classifier and the NLIM one (but with D replaced by D_ϕ). The quadratic form classifier will have a total computational cost equal to:

$$\begin{aligned} &D^2B_XB_F + D(D-1)(B_X + B_F + \lceil \log_2(D) \rceil - 1) \\ &+ DB_X(B_X + B_F + \lceil \log_2(D) \rceil) \\ &+ (D-1)(2B_X + B_F + 2\lceil \log_2(D) \rceil - 1) \end{aligned} \tag{3.2}$$

For NLOM, we consider the computational cost dependent on precision. For instance, the evaluation of the norm difference when using a RBF kernel has a total computational cost of:

$$N_s(DB_{X,F} + DB_{X,F}^2 + (D-1)(2B_{X,F} + \lceil \log_2(D) \rceil - 1)) \tag{3.3}$$

where $B_{X,F} = \max(B_X, B_F)$.

The representational cost is defined as the total number of bits needed to represent all parameters (inputs and weights). It is a measure of the complexity of storage and communications. Depending on the application, either one of the representational costs associated with weights or inputs may be more important.

3.2 Validation of Bounds

We consider two scenarios for each classifier type:

- Scenario A: $B_X = B_F$.
- Scenario B: $B_X - B_F$ defined by (2.6).

For each of the two scenarios:

1. We sweep the value of B_X (and the corresponding value of B_F) and perform fixed-point simulation (FX Sim) to obtain the associated true fixed-point classification error rate (p_e).
2. For each pair (B_X, B_F), we find the corresponding probabilistic upper bound on the classification error rate (PUB_e) based on Proposition 1 and the PUB (Theorem 2).
3. We find the smallest value of B_X (and hence B_F) that satisfies the GLB in Table 2.1.

Figure 3.1 shows our results for linear, NLIM (second order polynomial map), quadratic form, and NLOM (RBF kernel) classifiers for the Breast Cancer Dataset. The training and testing sets were obtained by independently sampling 500 random samples for each from the dataset. The classifiers were pretrained in floating-point using SGD with $\gamma = 2^{-10}$ and $\lambda = 1$

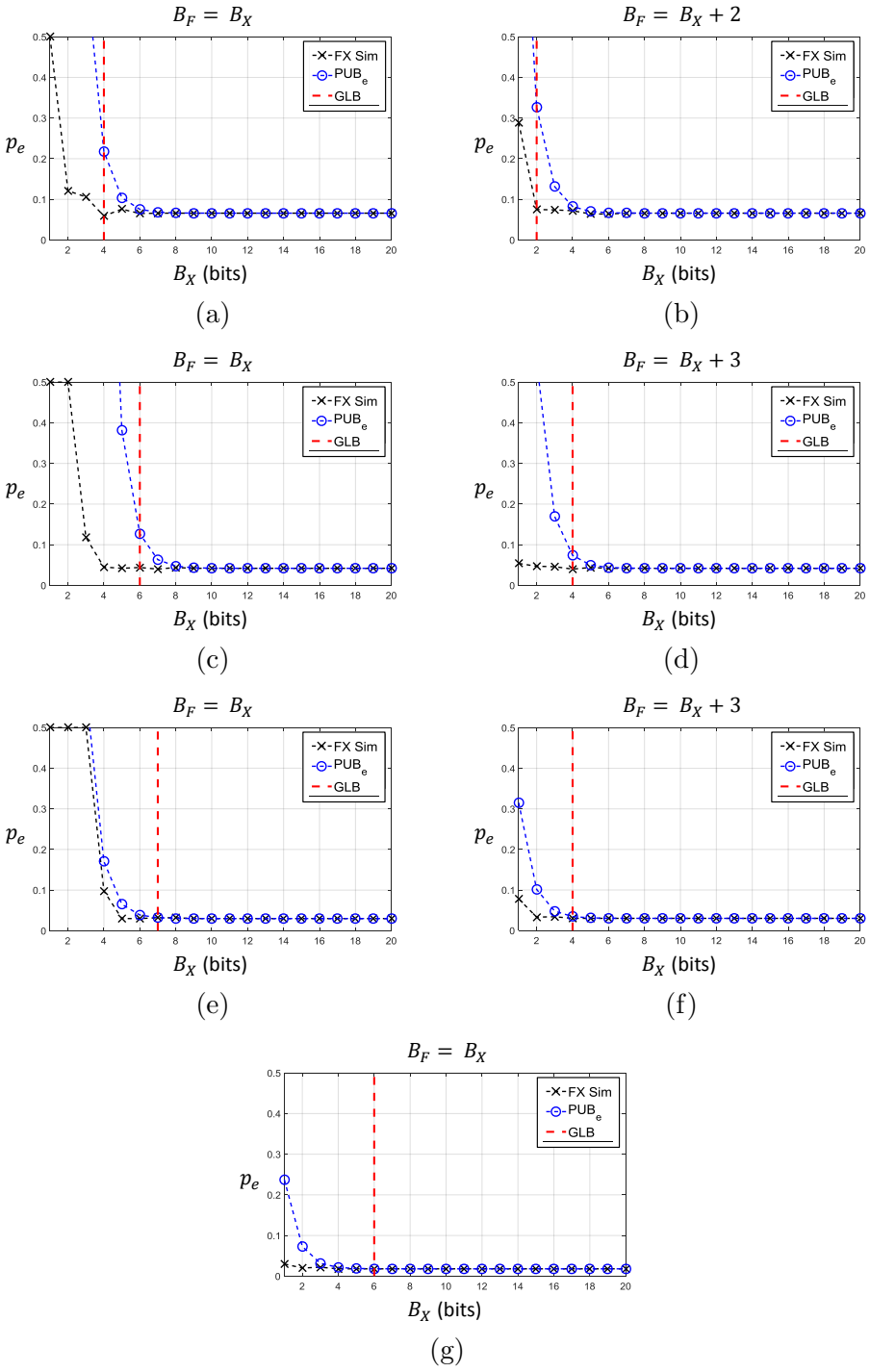

Figure 3.1: Results for classification on the Breast Cancer Dataset: classification error rate p_e in fixed-point simulations (FX Sim), analytical probabilistic upper bound (PUB$_e$), geometric lower bound (GLB) for $B_X = B_F$ and $B_X - B_F$ determined by (2.6) for linear (a,b), NLIM (second order polynomial) (c,d), quadratic form (e,f), and NLOM (RBF) (g) classifiers. For the NLOM classifier, (2.6) dictates $B_X = B_F$.

Table 3.1: Summary of Fig. 3.1 illustrating minimum precision requirements for hyperplane classifiers on the Breast Cancer Dataset dictated by FX Sim, GLB, and PUB_e when $B_X = B_F$ and $B_F - B_X$ determined by (2.6).

Classifier type	$B_X = B_F$			$B_F - B_X$ dictated by (2.6)		
	FX Sim	GLB	PUB_e	FX Sim	GLB	PUB_e
Linear	(4,4)	(4,4)	(6,6)	(2,4)	(2,4)	(4,6)
NLIM	(4,4)	(6,6)	(8,8)	(4,7)	(4,7)	(6,9)
NLOM	(2,2)	(6,6)	(3,3)	(2,2)	(6,6)	(3,3)
Quadratic form	(5,5)	(6,6)	(5,5)	(2,5)	(4,7)	(3,6)

except for the NLOM classifier which was trained using the commercial LIBSVM package [30].

For the linear classifier, Fig. 3.1(a) shows the validity of the GLB and the PUB_e for equal input and weight precisions. Indeed, fixed-point simulations for precisions larger than the GLB (4 bits) offer no significant accuracy gains while lower precisions seem to quickly degrade the accuracy. The PUB_e also successfully upper bounds the error obtained in fixed-point. Figure 3.1(b) shows the benefits of using (2.6) which dictates in this example that $B_F = B_X + 2$. Indeed, not only are the GLB and PUB_e still valid, but it is also possible to decrease B_X to 2 bits, as reflected by the GLB and supported by the fixed-point simulation, while maintaining good accuracy.

Similar trends are observed for NLIM (Fig. 3.1(c,d)) and quadratic form (Fig. 3.1(e,f)) classifiers. Indeed, the GLB reaches the precision value after which the fixed-point accuracy saturates to within 2 bits, and the PUB_e successfully upper bounds the fixed-point probability of error. Interestingly, the PUB_e is much tighter for the quadratic form classifier as compared to the NLIM classifier.

Figure 3.1(g) shows our results for the NLOM classifier. There are a total of 98 support vectors. In this case (2.6) results in $B_X = B_F$, and this is why we have only one plot. Observe that the PUB_e is much tighter than the GLB. Indeed, the PUB_e tracks the fixed-point simulations to precisions as low as 3 bits. The GLB predicts 6 bits.

A detailed breakdown of the minimum precision requirements determined by FX Sim, GLB, and PUB_e for all classifier types is presented in Table 3.1.

So far, all results were performed on the Breast Cancer Dataset. To show the generality of our results, we also conduct a similar experiment on the 'two

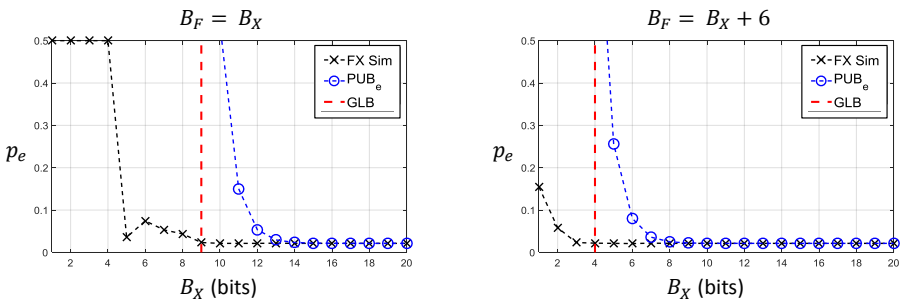

Figure 3.2: Results for linear classification on the 'two vs. four' task for the MNIST Dataset: classification error rate p_e in fixed-point simulations (FX Sim), analytical probabilistic upper bound (PUB$_e$), geometric lower bound (GLB) for (a) $B_X = B_F$ and (b) $B_X - B_F = 6$ as dictated by (2.6).

vs. four' task on the MNIST Dataset where we consider a linear classifier. Again, the classifier is first pretrained using SGD as discussed in Section 1.2.

The training and test sets were obtained by selecting the 'two' and 'four' instances from the original MNIST dataset. Overall, we had 11800 training and 2014 testing samples, respectively. The classifier was trained using SGD with $\gamma = 2^{-10}$ and $\lambda = 1$. The results are shown in Fig. 3.2. Once again, we find the numerical results to be consistent with the analysis of Section 2.1.

Interestingly, in the experiments on the Breast Cancer Dataset, (2.6) seemed to always yield a B_F larger than B_X by a few bits (2 or 3) except for the case of NLOM. For the MNIST experiment, this trend seems to continue and is even more pronounced as the difference $B_F - B_X = 6$ bits. In fact, this should not be surprising. Indeed, the feature vectors in the MNIST dataset are grayscale images whereas the weights define the separating hyperplane. It is hence reasonable to expect the precision requirements of weights to be higher as slight changes to the separating hyperplane are more detrimental to the classification accuracy.

3.3 Complexity vs. Accuracy Trade-offs

We compare costs and performance for the following setups:

1. minimum value of B_X specified by the GLB with $B_X = B_F$,

2. minimum value of B_X specified by the GLB with the difference between

B_X and B_F satisfying (2.6),

3. 8 bit quantization for all parameters,

4. an arbitrary precision assignment not satisfying the bounds presented in Section 2.1.

The first setup takes into account only one half of the theory proposed in Section 2.1, while the second shows the possibility and benefits of leveraging both GLB and PUB for more aggressive, yet principled quantization. We chose 8 bits in the third setup as it is a typical representative of low-precision/high-performance arithmetic [22]. The fourth setup is intended to highlight the drawbacks of aggressive and unprincipled quantization.

For the Breast Cancer Dataset, Table 3.2 shows how our principled quantization strategy makes it possible to operate at low cost while maintaining accuracy. Indeed, for the linear classifier, an unstructured precision assignment of 2 and 3 bits for inputs and weights, respectively, does reduce the computational and representational costs, but results in a relatively high test error. At the expense of only $\sim 1.2\times$ (178/146) computational cost and $\sim 1.2\times$ representational cost, $\sim 1.8\times$ classification error rate reduction is possible using a $(2, 4)$ quantization as dictated by (2.6) and the GB. This corresponds to $\sim 5\times$ reduction in computational cost and $\sim 2.6\times$ reduction in representational cost compared to the traditional 8 bits quantization. Similar trends are observed for the other classifiers.

Similarly, this comparison for the MNIST experiment is shown in Table 3.3. Interestingly, we observe here that the constraint $B_X = B_F$ is too harsh. Indeed, although the value $B_X = B_F = 9$ bits determined by the GLB yields a resulting accuracy $\sim 2\times$ better than the precision assignment of $B_X = B_F = 8$ bits, this should not be considered a satisfactory result. In fact, when taking into account the trade-off between B_X and B_F as described in (2.6), $B_F = B_X + 6$ here, we are able to reduce B_X down to only 4 bits. This corresponds to $\sim 1.7\times$ and $\sim 1.3\times$ decrease in the computational and representational costs with a negligible degradation in accuracy. Note that this quantization strategy results in a classifier $\sim 2\times$ more accurate than the one quantized to 8 bits in spite of using $\sim 2 \times 10^4$ fewer FAs and $\sim 2 \times 10^3$ fewer bits for data and weight representation. This highlights the importance of the analysis of the trade-off between input and weight precisions that was

Table 3.2: Results for the Breast Cancer Dataset. Comparison of computational cost, representational cost, and test error for linear, NLIM (second order polynomial), quadratic form, and NLOM (RBF) classifiers. The precision assignments considered are the standard low precision quantization (8,8), the minimum (B_X, B_F) satisfying the GLB when $B_X = B_F$, the minimum (B_X, B_F) satisfying the GLB when $B_F - B_X$ is dictated by (2.6), and one arbitrarily chosen assignment violating the bounds in Section 2.1. In the case of the NLOM classifier, the second and third precision assignments are identical. In each case, the use of the GLB and (2.6) makes it possible to reduce complexity but maintain accuracy.

Linear Classifier

(B_X, B_F)	Computational cost $(\#FA)$	Representational cost (bits)	Test error
$(8, 8)$	894	168	6.6%
$(4, 4)$	286	84	5.9%
$(2, 4)$	178	64	7.5%
$(2, 3)$	146	53	13.5%

NLIM (second order polynomial) Classifier

(B_X, B_F)	Computational cost $(\#FA)$	Representational cost (bits)	Test error
$(8, 8)$	5.9×10^3	1048	4.4%
$(6, 6)$	3.7×10^3	786	4.4%
$(4, 7)$	3.1×10^3	722	4.0%
$(3, 3)$	1.4×10^3	393	11.8%

Quadratic form Classifier

(B_X, B_F)	Computational cost $(\#FA)$	Representational cost (bits)	Test error
$(8, 8)$	11.9×10^3	1048	3.2%
$(7, 7)$	9.5×10^3	917	3.2%
$(4, 7)$	5.6×10^3	887	3.0%
$(4, 4)$	3.8×10^3	524	9.8%

NLOM (RBF) Classifier

(B_X, B_F)	Computational cost $(\#FA)$	Representational cost (bits)	Test error
$(8, 8)$	10×10^4	7920	1.8%
$(6, 6)$	6.6×10^4	5940	1.8%
$(1, 1)$	1.1×10^4	990	3.0%

Table 3.3: Results for linear classification on the 'two vs. four' task for the MNIST Dataset. Comparison of computational cost, representational cost, and test error. The precision assignments considered are the minimum (B_X, B_F) satisfying the GLB when $B_X = B_F$, the standard low precision quantization (8,8), the minimum (B_X, B_F) satisfying the GLB when $B_F - B_X = 6$ as dictated by (2.6), one arbitrarily chosen assignment of (3,6) violating the bounds in Section 2.1. The use of the GLB and (2.6) makes it possible to reduce complexity but maintain accuracy.

(B_X, B_F)	Computational cost $(\#FA)$	Representational cost (bits)	Test error
$(9, 9)$	85×10^3	14×10^3	2.3%
$(8, 8)$	70×10^3	13×10^3	4.4%
$(4, 10)$	49×10^3	11×10^3	2.2%
$(3, 6)$	28×10^3	7×10^3	8.5%

described in Section 2.1.

3.4 Training Behavior

Here we illustrate the impact of precision on training as evidence to support the discussion in Section 2.2. We start with the Breast Cancer Dataset. The feedforward precisions are set as follows: we choose $B_X = 6$ in order to separate the Theorem 3 scenario from the three boundary cases in Appendix 2.3.3, and we choose B_F as specified by the GLB. That way, the feedforward precisions are minimal in a geometric sense and the full convergence of the algorithm is determined by the weight update precision (Theorem 3).

We shall consider a linear and a quadratic form classifier. We consider two scenarios: training with a small step size ($\gamma = 2^{-10}$) and training with a large step size ($\gamma = 2^{-5}$). The dataset is once again split into 500 training and 500 testing samples. Each time we show the convergence behavior of the training loss function, which is the objective being minimized, and the test error rate, which is the target metric to be minimized. Each experiment is conducted over 30 independent runs and the curves shown hereafter are ensemble averages over these runs. For each run, the initial weights are set to zeros and λ is set to 1.

Figure 3.3 shows convergence curves for a linear classifier trained on the Breast Cancer Dataset. We show convergence curves for weight update pre-

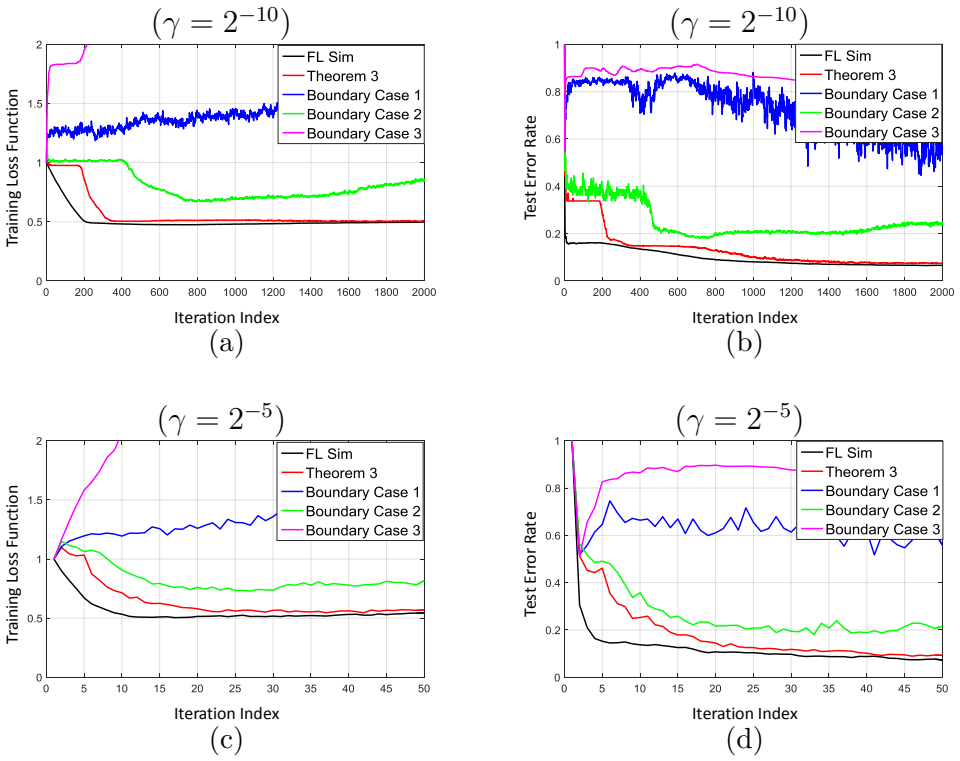

Figure 3.3: Results for linear classifier training on the Breast Cancer Dataset: (a) ensemble training loss function (1.6) and (b) ensemble test error rate for a small learning rate; (c) ensemble training loss function and (d) ensemble test error rate for a large learning rate. FL Sim denotes the floating-point simulation. In each case, a weight update precision of $B_X - \log_2(\gamma)$ (Theorem 3) is enough to mimic floating-point behavior in fixed-point. Accuracy degradation is observed for lower precisions.

cisions of: (a) $B_X - \log_2(\gamma)$ (Theorem 3), $1 - \log_2(\gamma)$ (Boundary Case 1), (b) $2 - \log_2(\gamma)$ (Boundary Case 2), and $-\log_2(\gamma)$ (Boundary Case 3). As shown, in both scenarios, a weight update precision of $B_X - \log_2(\gamma)$ is enough to mimic floating-point behavior in fixed-point. For weight update precisions of $1 - \log_2(\gamma)$ and $-\log_2(\gamma)$, the occasional sign-SGD updates are not enough for convergence as discussed in Sections 2.2 and 2.3.3. For a weight update precision of $2 - \log_2(\gamma)$, we do observe a decrease in the training loss function and the test error rate, but the accuracies obtained are not as good as those of the floating-point trials. This demonstrates the sufficiency property of Theorem 3.

As far as the training loss function is concerned, similar trends are observed for the quadratic form (Fig. 3.4) classifier. Interestingly, the test error rate

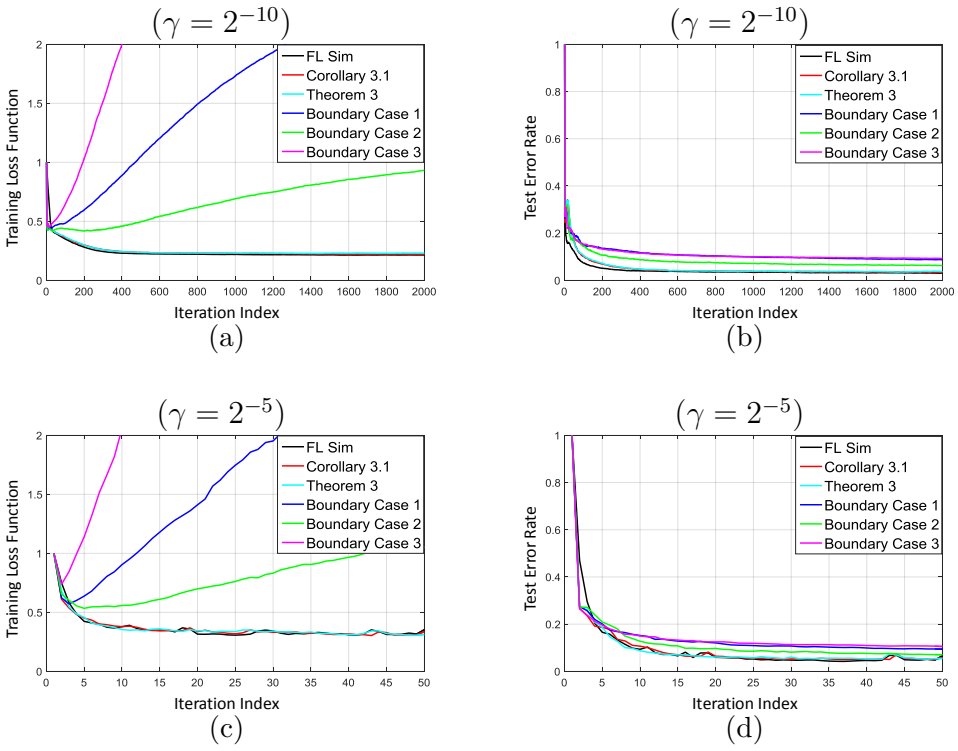

Figure 3.4: Results for quadratic form classifier training on the Breast Cancer Dataset: (a) ensemble training loss function ((1.6), matrix (Frobenius) norm replaces vector norm) and (b) ensemble test error rate for a small learning rate ($\gamma = 2^{-10}$); (c) ensemble training loss function and (d) ensemble test error rate for a large learning rate ($\gamma = 2^{-5}$). FL Sim denotes the floating-point simulation. The weight update precision of $2B_X - \log_2(\gamma)$ (Corollary 3.1) only marginally improves the accuracy over that of $B_X - \log_2(\gamma)$.

does go down in spite of imprecise updates. Nonetheless, we see that higher precision leads to greater overall accuracy. Note that we not only considered weight update precision of $B_X - \log_2(\gamma)$, but also $2B_X - \log_2(\gamma)$ as dictated by Corollary 3.1. Results show that this increased precision only marginally improves the convergence behavior.

Figure 3.5 illustrates these results for the 'two vs. four' classification task on the MNIST Dataset for linear classification with a learning rate of 2^{-10}. The feedforward precisions are again chosen to be minimal in the geometric sense. Those precisions were determined in the previous subsection to be 4 and 10 for the inputs and weights, respectively. The results here are very consistent with those observed for the experiments on the Breast Cancer Dataset. Indeed, it is clearly seen that a weight update precision of

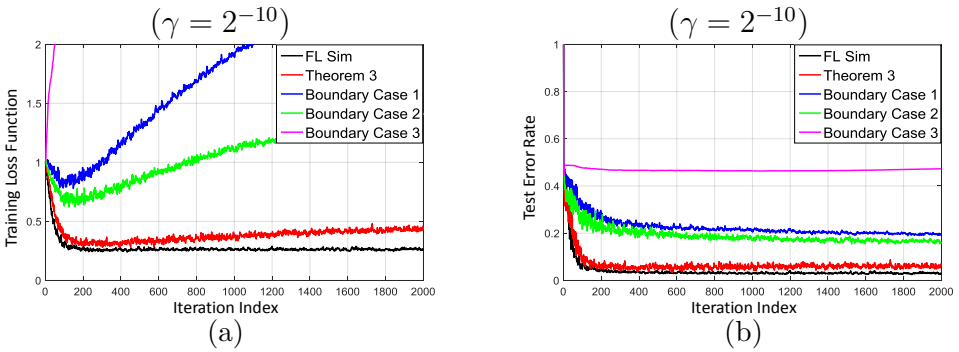

Figure 3.5: Results for linear classifier training on the 'two vs. four' task for the MNIST Dataset: (a) ensemble training loss function (1.6) and (b) ensemble test error rate for a learning rate $\gamma = 2^{-10}$. FL Sim denotes the floating-point simulation. A weight update precision of $B_X - \log_2(\gamma)$ (Theorem 3) is enough to mimic floating-point behavior in fixed-point. Accuracy degradation is observed for lower precisions.

$B_W = B_X - \log_2(\gamma)$ as dictated by Theorem 3 is of paramount importance for successful convergence. The results here are very consistent with those observed for the experiments on the Breast Cancer Dataset. Indeed, it is clearly seen that a weight update precision of $B_W = B_X - \log_2(\gamma)$ as dictated by Theorem 3 is of paramount importance for successful convergence.

3.5 Summary

In this chapter we have validated the theoretical results presented in Chapter 2. We have presented fixed-point simulations on the Breast Cancer and MNIST datasets. The numerical results show close agreement with the predicted precision requirements. Furthermore, the benefits of the analysis in terms of complexity reduction have been illustrated. New complexity metrics have been proposed in the computational and representational costs. These two meaningful costs, as well as the presented precision analysis, provide a natural way for designers to estimate complexity measures in fixed-point and to better explore the trade-offs of accuracy vs. precision and complexity.

CHAPTER 4

CONCLUSION

We have presented a theoretical analysis of the behavior of general fixed-point margin hyperplane classifiers. The results presented consist of bounds based on the geometry and statistics of the classification task and on the convergence conditions of the training task. By characterizing the trade-off between input and weight precisions, an efficient precision reduction scheme was presented. This framework eliminates the need for expensive trial-and-error. Furthermore, it presents guidelines for minimizing resource utilization. This utilization was captured by the computational and representational costs. Simulation results shown support the developed theory and highlight its benefits.

Several insights can be taken from our work. These include a trade-off between input and weight precision which is useful for minimizing the overall precision. Furthermore, it was observed from the GLB that precision increases logarithmically with the dimensionality of the problem. From the PUB, it was seen that the mismatch probability between fixed-point and floating-point classifier decays, at worst, exponentially with precision. It is in fact possible to derive a tighter bound on this mismatch probability using the Chernoff bound which marginally improves this dependence making it double exponential in precision. Finally, it was shown that the early stopping criterion applies to fixed-point training and provides a sufficient condition on the weight update precision for full convergence.

Future work includes a deeper dive into the topic of complexity vs. accuracy in machine learning. This includes a similar analysis for more complicated algorithms such as deep neural networks. The presented work takes a conservative approach in the model of quantization. It is possible to shape quantization noise statistics using dithering during training. Such an approach might lead to greater reductions in precision. Another line of work is to study the effects of floating-point quantization. While fixed-point im-

plementations are usually more efficient than floating-point ones, the latter benefit from a much wider dynamic range which could be beneficial to the robustness in classification. An orthogonal direction is to consider the structure of the algorithms themselves. It is well established that data-driven models inhibit large redundancies which can be exploited to trade off complexity with robustness. The above constitute the first steps in the important task of developing a unified and principled framework to understand complexity vs. accuracy in the design and implementation of machine learning systems.

REFERENCES

[1] A. Coates et al., "Deep learning with COTS HPC systems," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 1337–1345.

[2] S. Gupta et al., "Deep Learning with Limited Numerical Precision," in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 1737–1746.

[3] M. Kim et al., "Bitwise Neural Networks," *arXiv preprint arXiv:1601.06071*, 2016.

[4] M. Courbariaux et al., "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.

[5] M. Courbariaux et al., "Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[6] M. Rastegari et al., "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *arXiv preprint arXiv:1603.05279*, 2016.

[7] D. Lin et al., "Fixed Point Quantization of Deep Convolutional Networks," *arXiv preprint arXiv:1511.06393*, 2015.

[8] D. Lin et al., "Overcoming challenges in fixed point training of deep convolutional networks," *arXiv preprint arXiv:1607.02241*, 2016.

[9] M. Goel et al., "Finite-precision analysis of the pipelined strength-reduced adaptive filter," *Signal Processing, IEEE Transactions on*, vol. 46, no. 6, pp. 1763–1769, 1998.

[10] C. Caraiscos and B. Liu, "A roundoff error analysis of the lms adaptive algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 1, pp. 34–41, 1984.

[11] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Transactions on Signal Processing*, vol. 43, no. 12, pp. 3087–3090, 1995.

[12] R. Rocher et al., "Accuracy evaluation of fixed-point LMS algorithm," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, vol. 5. IEEE, 2004, pp. V–237.

[13] C. Shi and R. W. Brodersen, "An automated floating-point to fixed-point conversion methodology," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 2. IEEE, 2003, pp. II–529.

[14] R. Gupta and A. O. Hero, "Transient behavior of fixed point LMS adaptation," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 376–379.

[15] P. S. Diniz et al., "Analysis of LMS-Newton adaptive filtering algorithms with variable convergence factor," *IEEE Transactions on Signal Processing*, vol. 43, no. 3, pp. 617–627, 1995.

[16] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[17] K. H. Lee et al., "Low-energy formulations of support vector machine kernel functions for biomedical sensor applications," *Journal of Signal Processing Systems*, vol. 69, no. 3, pp. 339–349, 2012.

[18] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[19] C. Sakr et al., "Minimum precision requirements for the SVM-SGD learning algorithm," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017.

[20] Y. Chen et al., "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2016, pp. 262–263.

[21] D. Liu, T. Chen, S. Liu, J. Zhou, S. Zhou, O. Teman, X. Feng, X. Zhou, and Y. Chen, "Pudiannao: A polyvalent machine learning accelerator," in *ACM SIGARCH Computer Architecture News*, vol. 43, no. 1. ACM, 2015, pp. 369–381.

[22] M. Abadi et al., "Tensorflow: A system for large-scale machine learning," Google Brain, Tech. Rep., 2016, arXiv preprint. [Online]. Available: https://arxiv.org/abs/1605.08695

[23] B. Widrow and I. Kollár, *Quantization Noise*. Cambridge University Press, 2008, vol. 2.

[24] B. Liu and T. Kaneko, "Error analysis of digital filters realized with floating-point arithmetic," *Proceedings of the IEEE*, vol. 57, no. 10, pp. 1735–1747, 1969.

[25] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "Lognet: Energy-efficient neural networks using logarithmic computation," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5900–5904.

[26] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[27] C. Sakr et al., "Analytical Guarantees on Numerical Precision of Deep Neural Networks," in *International Conference on Machine Learning*, 2017, pp. 3007–3016.

[28] A. Asuncion et al., "UCI machine learning repository," 2007.

[29] Y. LeCun et al., "The mnist database of handwritten digits," 1998.

[30] C.-C. Chang et al., "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.