Training Fixed-Point Classifiers for On-Chip Low-Power Implementation

HASSAN ALBALAWI, YUANNING LI, and XIN LI, Carnegie Mellon University

In this article, we develop several novel algorithms to train classifiers that can be implemented on chip with low-power fixed-point arithmetic with extremely small word length. These algorithms are based on Linear Discriminant Analysis (LDA), Support Vector Machine (SVM), and Logistic Regression (LR), and are referred to as LDA-FP, SVM-FP, and LR-FP, respectively. They incorporate the nonidealities (i.e., rounding and overflow) associated with fixed-point arithmetic into the offline training process so that the resulting classifiers are robust to these nonidealities. Mathematically, LDA-FP, SVM-FP, and LR-FP are formulated as mixed integer programming problems that can be robustly solved by the branch-and-bound methods described in this article. Our numerical experiments demonstrate that LDA-FP, SVM-FP, and LR-FP substantially outperform the conventional approaches for the emerging biomedical applications of brain decoding.

CCS Concepts: • **Hardware → Electronic design automation**

Additional Key Words and Phrases: Machine learning, fixed-point arithmetic, low power

ACM Reference Format:
Hassan Albalawi, Yuanning Li, and Xin Li. 2017. Training fixed-point classifier for on-chip low-power implementation. ACM Trans. Des. Autom. Electron. Syst. 22, 4, Article 69 (June 2017), 18 pages.
DOI: http://dx.doi.org/10.1145/3057275

1. INTRODUCTION

Machine learning is an important technique that has been continuously growing during the past several decades [Bishop 2006; Mitchell 1997]. It has now been applied to a variety of big data for numerous commercial applications, including healthcare, social network, etc. Recently, a number of emerging applications, such as portable/implantable biomedical devices for electrocardiography (ECG) and electroencephalography (EEG) data processing [Kim et al. 2011; Roh et al. 2012; Shoaib et al. 2012; Winokur et al. 2013; Yoo et al. 2013] have posed a strong need to implement machine learning algorithms with Application-Specific Integrated Circuits (ASICs) due to the following reasons.

Small latency: The response of a machine learning engine must be sufficiently fast for many real-time applications such as vital sign monitoring [Clifton et al. 2012] and deep brain stimulation [Kringelbach et al. 2007]. In these cases, an on-chip machine learning engine must be implemented to locally process the data to ensure fast response time, instead of transmitting the data to an external device (e.g., a cloud server) for remote processing.

This work is supported in part by the National Science Foundation under grant CCF-1314876 and by the CMU-SYSU Collaborative Innovation Research Center at Carnegie Mellon University.
Authors' address: H. Albalawi, Y. Li, and X. Li, Electrical and Computer Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213; emails: {halbalaw, ynli, xinli}@cmu.edu.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2017 ACM 1084-4309/2017/06-ART69 $15.00
DOI: http://dx.doi.org/10.1145/3057275

Low power: To facilitate a portable/implantable device to continuously operate over a long time without recharging the battery, its power consumption must be minimized. Especially for the applications where power consumption is highly constrained (e.g., less than 10μW), it is necessary to design an ASIC circuit, instead of relying on a general-purpose microprocessor, to meet the tight power budget.

To maximally reduce the power consumption of on-chip machine learning for portable/implantable applications, fixed-point arithmetic, instead of floating-point arithmetic, must be adopted to implement machine learning algorithms and the word length for fixed-point computing must be aggressively minimized. While fixed-point arithmetic has been extensively studied for digital signal processing [Constantinide et al. 2003; Kinsman and Nicolici 2011; Mallik et al. 2007; Padgett and Anderson 2009], it is rarely explored for emerging machine learning tasks. Historically, most machine learning algorithms are only developed and validated by their software implementations (e.g., MATLAB, C++, etc.) based on double-precision floating-point arithmetic. It remains an open question how to appropriately map these algorithms to a low-power ASIC circuit implemented with fixed-point arithmetic.

In this article, we consider a case study of three well-known machine learning algorithms for binary classification: Linear Discriminant Analysis (LDA), Support Vector Machine (SVM), and Logistic Regression (LR) [Bishop 2006]. We will demonstrate that rounding error incurred from fixed-point arithmetic can significantly distort the classification output, if they are not appropriately modeled and incorporated into the classifier training process. In other words, the conventional classification algorithms developed for double-precsion floating-point operation must be "redesigned" in order to be made "robust" to rounding error. Similar "numerical" issues have been extensively studied in many other application domains. For instance, pivoting is an important technique for Gaussian elimination that is needed to mitigate the numerical error of a linear solver [Golub and Van Loan 2012]. The fundamental question here is how we can improve the robustness of LDA, SVM, and LR to make them suitable for on-chip low-power implementation.

Toward this goal, we propose to reformulate the training process of LDA, SVM, and LR as Mixed-Integer Programming (MIP) problems with consideration of the nonidealities (i.e., rounding and overflow) posed by fixed-point arithmetic. Our redesigned LDA algorithm is referred to as LDA-FP in this article. A novel branch-and-bound method is developed to find the globally optimal classification boundary of LDA-FP. With our redesigned training algorithm, LDA-FP is made maximally robust to the nonidealities associated with fixed-point arithmetic and, hence, can be efficiently implemented with extremely small word length for on-chip low-power operation.

On the other hand, we refer to the redesigned SVM and LR algorithms as SVM-FP and LR-FP, respectively. Similar to LDA-FP, the branch-and-bound method is applied to solve SVM-FP and LR-FP and find their optimal classification boundaries. In addition, several efficient heuristics (e.g., piecewise linear approximation [Hsiung et al. 2008]) are adopted to make the branch-and-bound solver computationally efficient.

Our proposed algorithms (i.e., LDA-FP, SVM-FP, and LR-FP) are validated for two practical applications, including (i) movement decoding for brain computer interface [Nicolelis 2001; Wang et al. 2013] and (ii) EEG-based drowsiness detection [Correa et al. 2014; Yeoa et al. 2009]. As will be demonstrated by our experimental results in Section 6, the proposed approach is able to reduce the word length by up to $1.67\times$ compared to the conventional techniques, without surrendering any classification accuracy.

The remainder of this article is organized as follows. In Section 2, we review the background on LDA, SVM, and LR for binary classification. Next, the mathematical formulations and numerical solvers are described for LDA-FP, SVM-FP, and LR-FP in

Sections 3–5. The efficacy of our proposed approach is demonstrated by two experimental examples in Section 6. Finally, we conclude in Section 7.

2. BACKGROUND

2.1. Linear Discriminant Analysis

LDA is a machine learning algorithm that has been extensively applied to a large number of binary classification problems [Bishop 2006]. To illustrate the basic idea of LDA, we consider two sets of training data $\{\mathbf{x}_A^{(n)}; n = 1, 2, \ldots, N_A\}$ and $\{\mathbf{x}_B^{(n)}; n = 1, 2, \ldots, N_B\}$ corresponding to two classes, where $\mathbf{x}_A^{(n)} = [x_{A,1}^{(n)} x_{A,2}^{(n)} \ldots x_{A,M}^{(n)}]^T$ and $\mathbf{x}_B^{(n)} = [x_{B,1}^{(n)} x_{B,2}^{(n)} \ldots x_{B,M}^{(n)}]^T$ are the feature vectors of the nth sampling point from the class A and B, respectively, M stands for the number of features, and N_A and N_B represent the numbers of sampling points for these two classes, respectively. LDA aims to find an optimal projection direction $\mathbf{w} \in \Re^M$ so that the two classes are maximally separated after projecting the feature vector $\mathbf{x}$ along the direction $\mathbf{w}$ [Bishop 2006].

Toward this goal, we first quantitatively calculate the between-class scatter matrix $\mathbf{S}_{B,\mathbf{x}} \in \Re^{M\times M}$ and the within-class scatter matrix $\mathbf{S}_{W,\mathbf{x}} \in \Re^{M\times M}$ for the feature vector $\mathbf{x}$ [Bishop 2006]:

$$\mathbf{S}_{B,\mathbf{x}} = (\boldsymbol{\mu}_A - \boldsymbol{\mu}_B) \cdot (\boldsymbol{\mu}_A - \boldsymbol{\mu}_B)^T, \tag{1}$$

$$\mathbf{S}_{W,\mathbf{x}} = \frac{1}{2} \cdot (\boldsymbol{\Sigma}_A + \boldsymbol{\Sigma}_B), \tag{2}$$

where $\boldsymbol{\mu}_A \in \Re^M$ and $\boldsymbol{\mu}_B \in \Re^M$ stand for the mean vectors:

$$\boldsymbol{\mu}_A = \frac{1}{N_A} \cdot \sum_{n=1}^{N_A} \mathbf{x}_A^{(n)}, \tag{3}$$

$$\boldsymbol{\mu}_B = \frac{1}{N_B} \cdot \sum_{n=1}^{N_B} \mathbf{x}_B^{(n)}, \tag{4}$$

and $\boldsymbol{\Sigma}_A \in \Re^{M\times M}$ and $\boldsymbol{\Sigma}_B \in \Re^{M\times M}$ represent the covariance matrices:

$$\boldsymbol{\Sigma}_A = \frac{1}{N_A} \cdot \sum_{n=1}^{N_A} \left(\mathbf{x}_A^{(n)} - \boldsymbol{\mu}_A\right) \cdot \left(\mathbf{x}_A^{(n)} - \boldsymbol{\mu}_A\right)^T, \tag{5}$$

$$\boldsymbol{\Sigma}_B = \frac{1}{N_B} \cdot \sum_{n=1}^{N_B} \left(\mathbf{x}_B^{(n)} - \boldsymbol{\mu}_B\right) \cdot \left(\mathbf{x}_B^{(n)} - \boldsymbol{\mu}_B\right)^T. \tag{6}$$

Projecting the feature vector $\mathbf{x}$ along the direction $\mathbf{w}$ yields

$$y = \mathbf{w}^T \cdot \mathbf{x}. \tag{7}$$

Since the projection result y is equal to a linear combination of all features weighted by $\mathbf{w}$, the projection direction $\mathbf{w}$ is often referred to as the *weight vector* for LDA in the literature [Bishop 2006].

It is straightforward to verify that the between-class scatter $S_{B,y} \in \Re$ and the within-class scatter $S_{W,y} \in \Re$ for the projection result y can be written as [Bishop 2006]

$$S_{B,y} = \mathbf{w}^T \cdot \mathbf{S}_{B,\mathbf{x}} \cdot \mathbf{w}, \tag{8}$$

$$S_{W,y} = \mathbf{w}^T \cdot \mathbf{S}_{W,\mathbf{x}} \cdot \mathbf{w}. \tag{9}$$

In order to maximally separate the two classes, the between-class scatter $S_{B,y}$ should be maximized, while the within-class scatter $S_{W,y}$ should be minimized. Hence, we can formulate the following optimization to minimize the ratio between $S_{W,y}$ and $S_{B,y}$ [Bishop 2006]:

$$\min_{\mathbf{w}} \frac{S_{W,y}}{S_{B,y}} = \frac{\mathbf{w}^T \cdot \mathbf{S}_{W,\mathbf{x}} \cdot \mathbf{w}}{\mathbf{w}^T \cdot \mathbf{S}_{B,\mathbf{x}} \cdot \mathbf{w}} = \frac{\mathbf{w}^T \cdot \mathbf{S}_{W,\mathbf{x}} \cdot \mathbf{w}}{[(\boldsymbol{\mu}_A - \boldsymbol{\mu}_B)^T \cdot \mathbf{w}]^2}. \tag{10}$$

There are two important clarifications that should be made regarding the optimization formulation in Equation (10). First, it is easy to verify that the optimal solution $\mathbf{w}$ of Equation (10) is not unique, since the cost function in Equation (10) is dependent on the direction of $\mathbf{w}$ only and it is independent of the length of $\mathbf{w}$. Namely, if $\mathbf{w}$ is an optimal solution of Equation (10), multiplying $\mathbf{w}$ by any nonzero scalar λ (i.e., $\lambda \cdot \mathbf{w}$) does not change the cost function and, hence, is also an optimal solution of Equation (10). When LDA is implemented, an additional constraint is often added to specify the length of the vector $\mathbf{w}$ (e.g., $\|\mathbf{w}\|_2 = 1$ where $\|\cdot\|_2$ denotes the L_2-norm of a vector) so that the optimal solution becomes unique.

Second, even though the optimization formulation in Equation (10) is not convex, it can be mapped to a generalized eigenvalue problem and solved both efficiently (i.e., with low computational cost) and robustly (i.e., with guaranteed global optimum) [Bishop 2006]. Assuming that the within-class scatter matrix $\mathbf{S}_{W,\mathbf{x}}$ is full-rank, the optimal $\mathbf{w}$ of Equation (10) can be proven to be [Bishop 2006]

$$\mathbf{w} \propto \mathbf{S}_{W,\mathbf{x}}^{-1} \cdot (\boldsymbol{\mu}_A - \boldsymbol{\mu}_B). \tag{11}$$

The weight vector $\mathbf{w}$ in Equation (11) can be normalized by its length if we want to keep the final solution with unit length. Once $\mathbf{w}$ is determined, the following linear decision boundary can be constructed for binary classification:

$$\mathbf{w}^T \cdot \mathbf{x} - \mathbf{w}^T \cdot \frac{\boldsymbol{\mu}_A + \boldsymbol{\mu}_B}{2} = \begin{cases} \geq 0 & (Class\ A) \\ < 0 & (Class\ B) \end{cases}, \tag{12}$$

where $\mathbf{x}$ denotes the feature vector of a sampling point that should be classified.

2.2. SVM

SVM is another popular machine learning algorithm that has been widely used for pattern recognition and data classification [Vapnik et al. 1998]. It aims to find the optimal classification boundary that maximizes the margin. Similar to LDA, we consider the training data $\{\mathbf{x}_A^{(n)}; n = 1, 2, \ldots, N_A\}$ and $\{\mathbf{x}_B^{(n)}; n = 1, 2, \ldots, N_B\}$ for two classes. When a linear SVM is applied, the classification boundary is represented by $\mathbf{w}^T \cdot \mathbf{x} + b = 0$, where $\mathbf{w}$ and b should be determined from the training data. The optimal $\mathbf{w}$ and b can be solved from the following optimization problem to maximize the classification margin [Bishop 2006]:

$$\begin{aligned} \min_{\mathbf{w},b,\xi_A^{(n)},\xi_B^{(n)}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + \lambda \cdot \sum_{n=1}^{N_A} \xi_A^{(n)} + \lambda \cdot \sum_{n=1}^{N_B} \xi_B^{(n)} & \\ \text{s.t.} \quad & \mathbf{w}^T \cdot \mathbf{x}_A^{(n)} + b \geq 1 - \xi_A^{(n)} & (n = 1, \ldots, N_A) \\ & \xi_A^{(n)} \geq 0 & (n = 1, \ldots, N_A) \\ & -\mathbf{w}^T \cdot \mathbf{x}_B^{(n)} - b \geq 1 - \xi_B^{(n)} & (n = 1, \ldots, N_B) \\ & \xi_B^{(n)} \geq 0 & (n = 1, \ldots, N_B) \end{aligned}, \tag{13}$$

where $\{\xi_A^{(n)}; n = 1, 2, \ldots, N_A\}$ and $\{\xi_B^{(n)}; n = 1, 2, \ldots, N_B\}$ represent the slack variables for measuring classification error, and λ is a parameter that can be determined by cross-validation [Bishop 2006].

2.3. LR

LR has been used for classification problems in many practical applications. Considering the training data $\{\mathbf{x}_A^{(n)}; n = 1, 2, \ldots, N_A\}$ and $\{\mathbf{x}_B^{(n)}; n = 1, 2, \ldots, N_B\}$ for two classes, we define the following model to describe the probability for each sampling point $\mathbf{x}$ to belong to a given class [Bishop 2006]:

$$p(A|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \cdot \mathbf{X} - b)}, \tag{14}$$

$$p(B|\mathbf{x}) = 1 - \frac{1}{1 + \exp(-\mathbf{w}^T \cdot \mathbf{x} - b)} = \frac{\exp(-\mathbf{w}^T \cdot \mathbf{x} - b)}{1 + \exp(-\mathbf{w}^T \cdot \mathbf{x} - b)}, \tag{15}$$

where $\mathbf{w}$ and b define the classifier. The optimal values of $\mathbf{w}$ and b can be found by maximum likelihood estimation based on the training data:

$$\min_{\mathbf{w},b} -\sum_{n=1}^{N_A} \log\left[\frac{1}{1 + \exp\left(-\mathbf{w}^T \cdot \mathbf{x}_A^{(n)} - b\right)}\right] - \sum_{n=1}^{N_B} \log\left[\frac{\exp\left(-\mathbf{w}^T \cdot \mathbf{x}_B^{(n)} - b\right)}{1 + \exp\left(-\mathbf{w}^T \cdot \mathbf{x}_B^{(n)} - b\right)}\right] + \rho \cdot \mathbf{w}^T \mathbf{w}, \tag{16}$$

where $\rho \cdot \mathbf{w}^T \mathbf{w}$ is the additional regularization term to avoid overfitting and the optimal value of ρ should be determined by cross-validation.

2.4. Fixed-Point Classifier

The conventional classification algorithms are able to find the optimal decision boundary, assuming that all computations can be performed without rounding error. In practice, once the classifier is implemented with fixed-point arithmetic, rounding error can substantially bias the decision boundary and, hence, distort the classification output. In this case, the classification boundary determined by a conventional machine learning algorithm is no longer optimal.

To understand the reason, we consider a simple 2D example for LDA. The objective is to determine an optimal linear boundary to separate Class A and Class B. By applying LDA, we find the optimal boundary $P_N^{(LDA)}$ shown in Figure 1(a). In addition to this nominal boundary $P_N^{(LDA)}$, Figure 1(a) further plots two perturbed boundaries, $P_L^{(LDA)}$ and $P_U^{(LDA)}$, due to rounding error. Note that if $P_N^{(LDA)}$ is rounded to $P_L^{(LDA)}$, a large classification error is expected. On the other hand, Figure 1(b) shows a robust boundary $P_N^{(Robust)}$ that is less sensitive to rounding error than $P_N^{(LDA)}$. Even if $P_N^{(Robust)}$ is perturbed to $P_L^{(Robust)}$ or $P_U^{(Robust)}$, the classification error remains negligible.

The aforementioned discussion reveals an important observation that the conventional machine learning algorithms may result in large classification error, since they do not explicitly consider the rounding error during classifier training. It, in turn, motivates us to redesign these conventional algorithms in order to generate a robust classifier that is least sensitive to rounding error.

3. FIXED-POINT LINEAR DISCRIMINANT ANALYSIS

3.1. Problem Formulation

The nonidealities (i.e., rounding and overflow) posed by fixed-point arithmetic can be broadly classified into two different categories: (i) the nonidealities associated with the feature vector $\mathbf{x}$ and (ii) the nonidealities associated with the weight vector $\mathbf{w}$. For the feature vector $\mathbf{x}$, all features in $\mathbf{x}$ can be carefully scaled to avoid overflow.

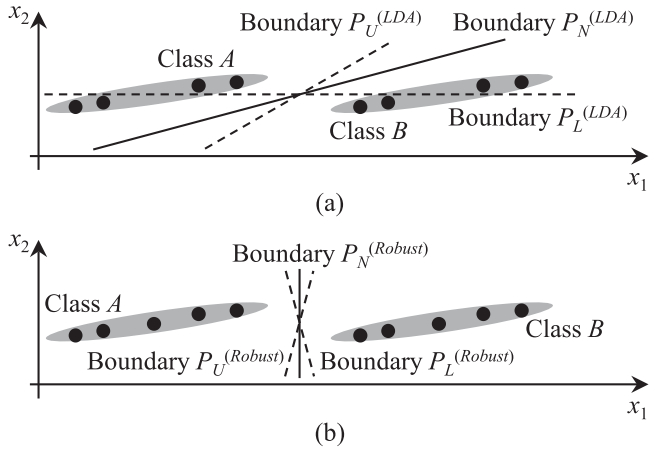

Fig. 1. (a) The optimal boundary $P_N^{(LDA)}$ determined by LDA is highly sensitive to rounding error. (b) A robust boundary $P_N^{(Robust)}$ can be found and it is insensitive to rounding error.

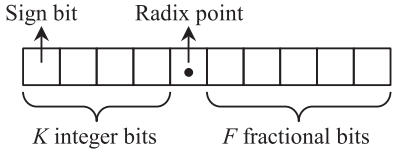

Fig. 2. An example is shown for the fixed-point format of $QK.F$ based on two's complement.

In addition, the feature vector $\mathbf{x}$ should be rounded to its fixed-point representation, before the training data is used to learn the classifier. As such, the training algorithm can easily take into account the rounding error associated with $\mathbf{x}$. In other words, the conventional LDA algorithm is able to address the nonidealities for the feature vector $\mathbf{x}$ appropriately.

On the other hand, mitigating the nonidealities of the weight vector $\mathbf{w}$ is not trivial. If we simply follow the conventional LDA algorithm, $\mathbf{w}$ is determined by Equation (11) and then normalized and rounded to its fixed-point representation. In this case, large classification error may occur, as is demonstrated by the simple example in Figure 1. Hence, our focus of this section is to derive a new LDA-FP algorithm to solve the optimal $\mathbf{w}$ that is suitable for fixed-point implementation.

Without loss of generality, we assume that a fixed-point number is represented in the format of $QK.F$ based on two's complement [Padgett and Anderson 2009, Chapter 4]. It has K integer bits (including the sign bit) and F fractional bits, as shown in Figure 2. The total word length is $K + F$. In this article, we further assume that all fixed-point operations in the classifier are implemented with the same format $QK.F$. In practice, it is possible to further optimize the word length for each individual operation. For instance, different elements $\{w_m;\ m = 1, 2, \ldots, M\}$ of the weight vector $\mathbf{w}$ can be assigned with different word lengths. However, since we mainly focus on classifier training in this article, the problem of word length optimization will be considered as a separate topic for our future research.

Given the fixed-point representation $QK.F$ shown in Figure 2, we must consider a number of important constraints when deriving our proposed optimization formulation for LDA-FP.

Rounding error: Since the weight vector $\mathbf{w}$ is represented in the format of $QK.F$, each element of $\mathbf{w}$, w_m where $m \in \{1, 2, \ldots, M\}$ can only take a finite number of possible

values:

$$w_m \in \Omega(m = 1, 2, \ldots, M), \tag{17}$$

where Ω stands for the set of all possible values that can be represented by $QK.F$.

Overflow: While the overflow of the feature vector $\mathbf{x}$ can be easily prevented by appropriately scaling $\mathbf{x}$ during a preprocessing step, we must carefully constrain the weight vector $\mathbf{w}$ so that calculating the projection $y = \mathbf{w}^T \cdot \mathbf{x}$ in Equation (7) does not result in an overflow. To this end, we statistically model the feature vector $\mathbf{x}$ as a multivariate Gaussian distribution:

$$\mathbf{x} \sim \begin{cases} Gauss\left(\boldsymbol{\mu}_A, \boldsymbol{\Sigma}_A\right) \ (Class\ A) \\ Gauss\left(\boldsymbol{\mu}_B, \boldsymbol{\Sigma}_B\right) \ (Class\ B) \end{cases}. \tag{18}$$

Note that the probability distribution of $\mathbf{x}$ depends on the class that $\mathbf{x}$ belongs to. Such a Gaussian model has been widely applied in the literature by the machine-learning community [Bishop 2006].

Based on Equation (18), each multiplication $w_m \cdot x_m$, where $m \in \{1,2, \ldots ,M\}$, yields a Gaussian distribution:

$$w_m \cdot x_m \sim \begin{cases} Gauss\left(w_m \cdot \mu_{A,m}, w_m^2 \cdot \sigma_{A,m}^2\right) \ (Class\ A) \\ Gauss\left(w_m \cdot \mu_{B,m}, w_m^2 \cdot \sigma_{B,m}^2\right) \ (Class\ B) \end{cases}, \tag{19}$$

where $\mu_{A,m}$ and $\mu_{B,m}$ are the mth elements of $\boldsymbol{\mu}_A$ and $\boldsymbol{\mu}_B$, respectively, and $\sigma_{A,m}^2$ and $\sigma_{B,m}^2$ are the mth diagonal elements of $\boldsymbol{\Sigma}_A$ and $\boldsymbol{\Sigma}_B$, respectively. With the Gaussian distributions in Equation (19), we can statistically define a confidence interval for $w_m \cdot x_m$:

$$\beta = \Phi^{-1}(0.5 + 0.5 \cdot \rho) \tag{20}$$

$$\begin{aligned} &[w_m\mu_{A,m} - \beta \cdot |w_m| \cdot \sigma_{A,m}, w_m\mu_{A,m} + \beta \cdot |w_m| \cdot \sigma_{A,m}] \ (Class\ A), \\ &[w_m\mu_{B,m} - \beta \cdot |w_m| \cdot \sigma_{B,m}, w_m\mu_{B,m} + \beta \cdot |w_m| \cdot \sigma_{B,m}] \ (Class\ B), \end{aligned} \tag{21}$$

where ρ denotes the confidence level and $\Phi^{-1}(\cdot)$ stands for the inverse cumulative distribution function of a standard Gaussian distribution. The confidence level ρ measures the probability for $w_m \cdot x_m$ to take a value within the confidence interval. If ρ is sufficiently large, the confidence interval in Equation (21) "almost" covers the range of $w_m \cdot x_m$. Hence, the confidence interval in Equation (21) must be within the range of the fixed-point representation $QK.F$ to avoid overflow:

$$\begin{aligned} w_m \cdot \mu_{A,m} - \beta \cdot |w_m| \cdot \sigma_{A,m} &\geq -2^{K-1}, \\ w_m \cdot \mu_{B,m} - \beta \cdot |w_m| \cdot \sigma_{B,m} &\geq -2^{K-1}, \\ w_m \cdot \mu_{A,m} + \beta \cdot |w_m| \cdot \sigma_{A,m} &\leq 2^{K-1} - 2^{-F}, \\ w_m \cdot \mu_{B,m} + \beta \cdot |w_m| \cdot \sigma_{B,m} &\leq 2^{K-1} - 2^{-F}. \end{aligned} \tag{22}$$

In addition to the multiplication $w_m \cdot x_m$, the projection result $y = \mathbf{w}^T \cdot \mathbf{x}$ is a linear combination of all features with Gaussian distributions and, hence, remains a Gaussian distribution:

$$\mathbf{w}^T \cdot \mathbf{x} \sim \begin{cases} Gauss\left(\mathbf{w}^T \cdot \boldsymbol{\mu}_A, \mathbf{w}^T \cdot \boldsymbol{\Sigma}_A \cdot \mathbf{w}\right) \ (Class\ A) \\ Gauss\left(\mathbf{w}^T \cdot \boldsymbol{\mu}_B, \mathbf{w}^T \cdot \boldsymbol{\Sigma}_B \cdot \mathbf{w}\right) \ (Class\ B). \end{cases} \tag{23}$$

Similar to Equations (19)–(22), we can statistically define the confidence interval for y and then set up the following constraints to prevent y from overflowing:

$$\begin{aligned} \mathbf{w}^T \boldsymbol{\mu}_A - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_A \mathbf{w}} &\geq -2^{K-1}, \\ \mathbf{w}^T \boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_B \mathbf{w}} &\geq -2^{K-1}, \\ \mathbf{w}^T \boldsymbol{\mu}_A + \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_A \mathbf{w}} &\leq 2^{K-1} - 2^{-F}, \\ \mathbf{w}^T \boldsymbol{\mu}_B + \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_B \mathbf{w}} &\leq 2^{K-1} - 2^{-F}. \end{aligned} \tag{24}$$

Two important clarifications must be made here. First, we do not need to explicitly set up any overflow constraint for the intermediate sum when calculating the weighted sum $y = \mathbf{w}^T \cdot \mathbf{x}$. As long as the final projection result y does not overflow and the fixed-point numbers are implemented with two's complement based on wrapping, the overflow of the intermediate sum should not introduce any error on y. To intuitively illustrate this important property, we consider a simple example of calculating the summation $y = 3 + 3 - 4$ using $Q3.0$. Since the range of $Q3.0$ is $[-4, 3]$, calculating the intermediate sum $011 + 011 = 110$ results in an overflow. After calculating the final result, however, we get $110 + 100 = 010$. Compared to the correct value of $y = 3 + 3 - 4 = 2$, the final result represented by $Q3.0$ remains correct.

Second, rounding and overflow are closely coupled and must be simultaneously considered in our proposed training process. On the one hand, if we focus on rounding without overflow, the classifier may carry extremely large weight values, represented in the format of $QK.F$, to minimize the impact of rounding. Such a classifier, however, is likely to overflow due to its large weight values. On the other hand, if we consider overflow without rounding, the classifier may use extremely small weight values to minimize the probability to overflow. However, these small weight values can be substantially distorted after rounding.

Classification accuracy: To maximize classification accuracy of LDA, the weight vector $\mathbf{w}$ must be optimized to maximally separate the two classes. Toward this goal, we borrow the cost function in Equation (10) to minimize the ratio between the within-class scatter $S_{W,y}$ and the between-class scatter $S_{B,y}$, resulting in the following optimization:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{\mathbf{w}^T \cdot \mathbf{S}_{W,\mathbf{x}} \cdot \mathbf{w}}{\left[(\boldsymbol{\mu}_A - \boldsymbol{\mu}_B)^T \cdot \mathbf{w}\right]^2} \\ \text{s.t.} \quad & w_m \cdot \mu_{A,m} - \beta \cdot |w_m| \cdot \sigma_{A,m} \geq -2^{K-1} && (m = 1, 2, \ldots, M) \\ & w_m \cdot \mu_{B,m} - \beta \cdot |w_m| \cdot \sigma_{B,m} \geq -2^{K-1} && (m = 1, 2, \ldots, M) \\ & w_m \cdot \mu_{A,m} + \beta \cdot |w_m| \cdot \sigma_{A,m} \leq 2^{K-1} - 2^{-F} && (m = 1, 2, \ldots, M) \\ & w_m \cdot \mu_{B,m} + \beta \cdot |w_m| \cdot \sigma_{B,m} \leq 2^{K-1} - 2^{-F} && (m = 1, 2, \ldots, M) \\ & \mathbf{w}^T \boldsymbol{\mu}_A - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_A \mathbf{w}} \geq -2^{K-1} \\ & \mathbf{w}^T \boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_B \mathbf{w}} \geq -2^{K-1} \\ & \mathbf{w}^T \boldsymbol{\mu}_A + \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_A \mathbf{w}} \leq 2^{K-1} - 2^{-F} \\ & \mathbf{w}^T \boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_B \mathbf{w}} \leq 2^{K-1} - 2^{-F} \\ & w_m \in \Omega && (m = 1, 2, \ldots, M) \end{aligned} \tag{25}$$

Note that the optimization formulation in Equation (25) represents a mixed integer programming problem [Boyd and Vandenberghe 2004; Wolsey 1998] that is difficult to solve due to the following two reasons. First, the solution $\mathbf{w}$ is constrained to a discrete set, instead of a continuous set. Second, the cost function is represented as the ratio of two quadratic functions and, hence, is not convex. To address these challenges, we will further propose a novel branch-and-bound method with several efficient heuristics

to solve Equation (25) and find the global optimum. The details of our proposed solver will be discussed in the next subsection.

3.2. Branch-and-Bound Solver

The branch-and-bound method has been widely used to solve mixed integer programming problems [Wolsey 1998]. The key idea is to iteratively partition the optimization variables into subintervals. For each subinterval, the lower and upper bounds of the cost function are estimated in order to efficiently remove the irrelevant subintervals that do not contain the optimal solution from the search space. However, when the branch-and-bound method is applied to solve Equation (25), it is not trivial to efficiently estimate the lower and upper bounds for a given subinterval of $\mathbf{w}$, because the cost function in Equation (25) is the ratio of two quadratic functions and, hence, not convex [Boyd and Vandenberghe 2004]. In what follows, we will propose several novel ideas to address this technical challenge so that the optimization problem in Equation (25) can be solved by the branch-and-bound method robustly (i.e., with guaranteed global optimum).

Lower bound estimation: We introduce an additional variable:

$$t = (\boldsymbol{\mu}_A - \boldsymbol{\mu}_B)^T \cdot \mathbf{w} \tag{26}$$

and rewrite the cost function in Equation (25) as

$$\min_{\mathbf{w},t} \frac{\mathbf{w}^T \cdot \mathbf{S}_{W,\mathbf{x}} \cdot \mathbf{w}}{t^2} \tag{27}$$

Note that both $\mathbf{w}$ and t are now considered as optimization variables and they should be partitioned into subintervals when searching for the optimal solution by the branch-and-bound method.

With a given subinterval:

$$\begin{gathered} l_{wm} \le w_m \le u_{wm} \quad (m = 1, 2, \ldots, M) \\ l_t \le t \le u_t, \end{gathered} \tag{28}$$

where l_{wm} and l_t represent the lower bounds of w_m and t, respectively, and u_{wm} and u_t stand for the upper bounds of w_m and t, respectively. We can now estimate the lower bound of the cost function by solving the following optimization:

$$\begin{aligned} \min_{\mathbf{w},t} \ & \frac{1}{\eta} \cdot \mathbf{w}^T \cdot \mathbf{S}_{W,\mathbf{x}} \cdot \mathbf{w} \\ \text{s.t.} \ & w_m \cdot \mu_{A,m} - \beta \cdot |w_m| \cdot \sigma_{A,m} \ge -2^{K-1} && (m = 1, 2, \ldots, M) \\ & w_m \cdot \mu_{B,m} - \beta \cdot |w_m| \cdot \sigma_{B,m} \ge -2^{K-1} && (m = 1, 2, \ldots, M) \\ & w_m \cdot \mu_{A,m} + \beta \cdot |w_m| \cdot \sigma_{A,m} \le 2^{K-1} - 2^{-F} && (m = 1, 2, \ldots, M) \\ & w_m \cdot \mu_{B,m} + \beta \cdot |w_m| \cdot \sigma_{B,m} \le 2^{K-1} - 2^{-F} && (m = 1, 2, \ldots, M) \\ & \mathbf{w}^T \boldsymbol{\mu}_A - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_A \mathbf{w}} \ge -2^{K-1} \\ & \mathbf{w}^T \boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_B \mathbf{w}} \ge -2^{K-1} \\ & \mathbf{w}^T \boldsymbol{\mu}_A + \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_A \mathbf{w}} \le 2^{K-1} - 2^{-F} \\ & \mathbf{w}^T \boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_B \mathbf{w}} \le 2^{K-1} - 2^{-F} \\ & t = (\boldsymbol{\mu}_A - \boldsymbol{\mu}_B)^T \cdot \mathbf{w} \\ & l_{wm} \le w_m \le u_{wm} && (m = 1, 2, \ldots, M) \\ & l_t \le t \le u_t \end{aligned} \tag{29}$$

where η is a constant defined by

$$\eta = \sup_{l_t \le t \le u_t} t^2. \tag{30}$$

In Equation (30), sup(·) denotes the supremum (i.e., the least upper bound) of a set. Since t^2 is simply a quadratic function, it is easy to calculate the supremum over the interval $l_t \le t \le u_t$ and determine the value of η.

The optimization in Equation (29) is a convex second-order cone programming problem [Boyd and Vandenberghe 2004] and, hence, can be solved efficiently. Compared to Equation (25), the formulation in Equation (29) is "relaxed" in two different ways. First, the denominator of the cost function is relaxed to the maximum possible value η over the interval $l_t \le t \le u_t$. Second, the vector $\mathbf{w}$ is no longer constrained to a discrete set; instead, it can take any real value within the subinterval $\{l_{wm} \le w_m \le u_{wm}; m = 1, 2, \ldots, M\}$. For this reason, solving Equation (29) yields a lower bound of the cost function of the original mixed integer programming problem for the given subinterval defined in Equation (28).

Upper bound estimation: Similar to lower bound estimation, we can reuse the optimization formulation in Equation (29) to estimate the upper bound of the cost function with the parameter η set to

$$\eta = \inf_{l_t \le t \le u_t} t^2, \tag{31}$$

where inf(·) denotes the infimum (i.e., the greatest lower bound) of a set. In this case, the optimization in Equation (29) is again a convex second-order cone programming problem. After solving Equation (29), we further round the solution $\mathbf{w}$ to the discrete set defined in Equation (17). Next, we substitute the rounded $\mathbf{w}$ to the cost function in Equation (25). It, in turn, results in an upper bound of the cost function of Equation (25) given the subinterval defined in Equation (28).

Initial interval size estimation: Since the branch-and-bound method iteratively partitions the optimization variables $\mathbf{w}$ and t into subintervals, we need to determine the initial interval size for $\mathbf{w}$ and t, before starting the first iteration. Since $\mathbf{w}$ is represented in the format of $QK.F$ as shown in Figure 2, it must be within the following range:

$$-2^{K-1} \le w_m \le 2^{K-1} - 2^{-F} (m = 1, 2, \ldots, M). \tag{32}$$

On the other hand, since t is a linear function of $\mathbf{w}$ as defined in Equation (26), combining Equations (26) and (32) yields

$$-2^{K-1} \cdot \|\boldsymbol{\mu}_A - \boldsymbol{\mu}_B\|_1 \le t \le (2^{K-1} - 2^{-F}) \cdot \|\boldsymbol{\mu}_A - \boldsymbol{\mu}_B\|_1, \tag{33}$$

where $\|\cdot\|_1$ denotes the L_1-norm of a vector (i.e., the summation of the absolute values of all elements in the vector).

With the aforementioned heuristics, our proposed branch-and-bound method for solving Equation (25) can be summarized by Algorithm 1. Here, we iteratively shrink the search intervals until they are sufficiently small. During these iterations, the lower and upper bounds of the cost function are estimated to remove the irrelevant intervals and, hence, reduce the search space.

4. FIXED-POINT SVM

Following the idea of LDA-FP described in the previous section, we can derive the mathematical formulation for SVM-FP. As mentioned in Section 2.2, SVM aims to find the optimal classification boundary with maximum margin. Combining the SVM formulation in Equation (13), the rounding error in Equation (17) and the overflow constraints in Equations (22) and (24) yields the following constrained optimization

ALGORITHM 1: Branch-and-Bound Solver for LDA-FP

1. Start from two sets of training data $\{\mathbf{x}_A^{(n)}; n = 1, 2, \ldots, N_A\}$ and $\{\mathbf{x}_B^{(n)}; n = 1, 2, \ldots, N_B\}$ corresponding to two classes, and a given fixed-point format $Q_K.F$. Round the training data to their fixed-point representations.
2. Calculate $\boldsymbol{\mu}_A$ and $\boldsymbol{\mu}_B$ by Equations (3) and (4) and $\mathbf{S}_{W,\mathbf{x}}$ by Equation (2).
3. Initialize the search interval ξ by Equations (32) and (33). Estimate the lower bound l_f and upper bound u_f of the cost function for the given interval ξ. Set $\Xi = \{\xi\}$.
4. Select one interval from the set Ξ, and partition it into two subintervals. Remove the selected interval from Ξ.
5. For each of these two subintervals, estimate the lower bound l_f and upper bound u_f of the cost function. If l_f is no greater than the minimum upper bound over all intervals in the set Ξ, add the current subinterval to Ξ. Find all intervals in Ξ for which the lower bounds are greater than u_f, and remove them from Ξ.
6. If the sizes of all intervals in the set Ξ are sufficiently small, stop iteration. Otherwise, go to Step 4.

problem for SVM-FP:

$$
\begin{aligned}
\min_{\mathbf{w},b,\xi_A^{(n)},\xi_B^{(n)}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + \lambda \cdot \sum_{n=1}^{N_A} \xi_A^{(n)} + \lambda \cdot \sum_{n=1}^{N_B} \xi_B^{(n)} \\
\text{s.t.} \quad & \mathbf{w}^T \cdot \mathbf{x}_A^{(n)} + b \geq 1 - \xi_A^{(n)} && (n = 1, \ldots, N_A) \\
& \xi_A^{(n)} \geq 0 && (n = 1, \ldots, N_A) \\
& -\mathbf{w}^T \cdot \mathbf{x}_B^{(n)} - b \geq 1 - \xi_B^{(n)} && (n = 1, \ldots, N_B) \\
& \xi_B^{(n)} \geq 0 && (n = 1, \ldots, N_B) \\
& w_m \cdot \mu_{A,m} - \beta \cdot |w_m| \cdot \sigma_{A,m} \geq -2^{K-1} && (m = 1, 2, \ldots, M) \\
& w_m \cdot \mu_{B,m} - \beta \cdot |w_m| \cdot \sigma_{B,m} \geq -2^{K-1} && (m = 1, 2, \ldots, M) \\
& w_m \cdot \mu_{A,m} + \beta \cdot |w_m| \cdot \sigma_{A,m} \leq 2^{K-1} - 2^{-F} && (m = 1, 2, \ldots, M) \\
& w_m \cdot \mu_{B,m} + \beta \cdot |w_m| \cdot \sigma_{B,m} \leq 2^{K-1} - 2^{-F} && (m = 1, 2, \ldots, M) \\
& \mathbf{w}^T\boldsymbol{\mu}_A - \beta \cdot \sqrt{\mathbf{w}^T\boldsymbol{\Sigma}_A\mathbf{w}} \geq -2^{K-1} \\
& \mathbf{w}^T\boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T\boldsymbol{\Sigma}_B\mathbf{w}} \geq -2^{K-1} \\
& \mathbf{w}^T\boldsymbol{\mu}_A + \beta \cdot \sqrt{\mathbf{w}^T\boldsymbol{\Sigma}_A\mathbf{w}} \leq 2^{K-1} - 2^{-F} \\
& \mathbf{w}^T\boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T\boldsymbol{\Sigma}_B\mathbf{w}} \leq 2^{K-1} - 2^{-F} \\
& w_m \in \Omega && (m = 1, 2, \ldots, M)\,.
\end{aligned}
\tag{34}
$$

Equation (34) represents a mixed integer programming problem. Unlike the problem in Equation (25) where the cost function is nonconvex, the cost and constraint functions in Equation (34) are all convex except the last integer constraint. In this case, we can solve Equation (34) by using the standard branch-and-bound algorithm implemented by an existing toolbox such as MOSEK [MOSEK 2015].

5. FIXED-POINT LR

As shown in Section 2.3, the goal of LR is to find the parameters, namely, $\mathbf{w}$ and b, that have the smallest possible deviance between the training data and the predicted values. We rewrite the objective function in Equation (16) by using the Log-Sum-Exp (LSE) function $\Re^K \to \Re$:

$$
\mathrm{lse}\,(x_1, \ldots, x_K) = \log\left(\mathrm{e}^{x_1} + \cdots + \mathrm{e}^{x_K}\right). \tag{35}
$$

The function lse(·) is convex and can be interpreted as an analytic-differentiable approximation of the max function [Boyd and Vandenberghe 2004]. Using the bivariate LSE function, the objective function in Equation (16) can be reformulated as

$$\min_{\mathbf{w},b} \sum_{n=1}^{N_A} \left[\text{lse}\left(0, -\mathbf{w}^T \cdot \mathbf{x}_A^{(n)} - b\right)\right] + \sum_{n=1}^{N_B} \left[\text{lse}\left(0, -\mathbf{w}^T \cdot \mathbf{x}_B^{(n)} - b\right) - \left(-\mathbf{w}^T \cdot \mathbf{x}_B^{(n)} - b\right)\right] + \rho \cdot \mathbf{w}^T \mathbf{w}. \quad (36)$$

Since the function lse(·) is convex, it can be approximated by a Piecewise-Linear (PWL) function to any degree of accuracy if the PWL function contains a sufficiently large number of pieces. For a bivariate function $\text{lse}(y_1, y_2)$, a constructive algorithm has been proposed in the literature to find the optimal PWL approximation [Hsiung et al. 2008]. Given an integer $R \geq 2$, we solve the following optimization problem to find the optimal R-term PWL function to approximate the bivariate function $\text{lse}(y_1, y_2)$:

$$\begin{aligned} \min_{u_{r1}, u_{r2}, h_r} \quad & \sup_{y_1, y_2} \left\{ \text{lse}\,(y_1, y_2) - \max_{r \in \{1,\ldots,R\}} \{u_{r1} y_1 + u_{r2} y_2 + h_r\} \right\} \\ \text{s.t.} \quad & \text{lse}\,(y_1, y_2) > \max_{r \in \{1,\ldots,R\}} \{u_{r1} y_1 + u_{r2} y_2 + h_r\} \qquad \forall y_1, \forall y_2, \end{aligned} \quad (37)$$

where the optimization variables are $\{(u_{r1}, u_{r2}, h_r); r = 1, 2, \ldots, R\}$. The problem in (37) can be efficiently solved by an iterative algorithm described in Hsiung et al. [2008]. Once the parameters $\{(u_{r1}, u_{r2}, h_r); r = 1, 2, \ldots, R\}$ are determined, the optimal R-term PWL approximation is

$$\text{pwl}\,(y_1, y_2) = \max\{u_{11} y_1 + u_{12} y_2 + h_1, \ldots, u_{R1} y_1 + u_{R2} y_2 + h_R\}. \quad (38)$$

Now, we can substitute the LSE function in Equation (36) with the optimal R-term PWL approximation in Equation (38):

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \sum_{n=1}^{N_A} \left[\max\left\{-u_{12}\mathbf{w}^T \mathbf{x}_A^{(n)} - u_{12} b + h_1, \ldots, -u_{R2}\mathbf{w}^T \mathbf{x}_A^{(n)} - u_{R2} b + h_R\right\}\right] \\ & + \sum_{n=1}^{N_B} \left[\max\left\{-u_{12}\mathbf{w}^T \mathbf{x}_B^{(n)} - u_{12} b + h_1, \ldots, -u_{R2}\mathbf{w}^T \mathbf{x}_B^{(n)} - u_{R2} b + h_R\right\} - \left(-\mathbf{w}^T \cdot \mathbf{x}_B^{(n)} - b\right)\right] + \rho \cdot \mathbf{w}^T \mathbf{w}. \end{aligned} \quad (39)$$

Finally, combining the LR formulation in Equation (39), the rounding error in Equation (17) and the overflow constraints in Equations (22) and (24) yields the following constrained optimization problem for LR-FP:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \sum_{n=1}^{N_A} \left[\max\left\{-u_{12}\mathbf{w}^T \mathbf{x}_A^{(n)} - u_{12} b + h_1, \ldots, -u_{R2}\mathbf{w}^T \mathbf{x}_A^{(n)} - u_{R2} b + h_R\right\}\right] \\ & + \sum_{n=1}^{N_B} \left[\max\left\{-u_{12}\mathbf{w}^T \mathbf{x}_B^{(n)} - u_{12} b + h_1, \ldots, -u_{R2}\mathbf{w}^T \mathbf{x}_B^{(n)} - u_{R2} b + h_R\right\} - \left(-\mathbf{w}^T \cdot \mathbf{x}_B^{(n)} - b\right)\right] + \rho \cdot \mathbf{w}^T \mathbf{w} \\ & w_m \cdot \mu_{A,m} - \beta \cdot |w_m| \cdot \sigma_{A,m} \geq -2^{K-1} \qquad (m = 1, 2, \ldots, M) \\ & w_m \cdot \mu_{B,m} - \beta \cdot |w_m| \cdot \sigma_{B,m} \geq -2^{K-1} \qquad (m = 1, 2, \ldots, M) \\ & w_m \cdot \mu_{A,m} + \beta \cdot |w_m| \cdot \sigma_{A,m} \leq 2^{K-1} - 2^{-F} \quad (m = 1, 2, \ldots, M) \\ & w_m \cdot \mu_{B,m} + \beta \cdot |w_m| \cdot \sigma_{B,m} \leq 2^{K-1} - 2^{-F} \quad (m = 1, 2, \ldots, M) \\ \text{S.T.} \quad & \mathbf{w}^T \boldsymbol{\mu}_A - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_A \mathbf{w}} \geq -2^{K-1} \\ & \mathbf{w}^T \boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_B \mathbf{w}} \leq 2^{K-1} \\ & \mathbf{w}^T \boldsymbol{\mu}_A + \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_A \mathbf{w}} \leq 2^{K-1} - 2^{-F} \\ & \mathbf{w}^T \boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_B \mathbf{w}} \leq 2^{K-1} - 2^{-F} \\ & w_m \in \Omega \qquad (m = 1, 2, \ldots, M) \end{aligned} \quad (40)$$

Equation (40) represents a mixed integer programming problem where the cost and constraint functions are all convex except the last integer constraint. It can be solved using the standard branch-and-bound algorithm implemented by an existing toolbox such as MOSEK [MOSEK 2015]. In our implementation, CVX, a MATLAB-based convex solver [Grant and Boyd 2014] is used to first formulate the optimization problem in Equation (40) and then the formulation is passed to the MOSEK solver.

6. NUMERICAL EXAMPLES

In this section, two test cases are presented to demonstrate the efficacy of our proposed algorithms. When training fixed-point classifiers, both the input data and the weight values are appropriately scaled and rounded as fixed-point numbers. For testing and comparison purposes, the conventional algorithms, namely, LDA-R, SVM-R, and LR-R, are implemented where the weight vector $\mathbf{w}$ is first solved by Equations (11), (13), and (16), respectively, and then rounded to its fixed-point representation. On the other hand, the proposed LDA-FP, SVM-FP, and LR-FP methods solve the weight vector $\mathbf{w}$ from Equations (25), (34), and (40), respectively. In addition, we show the accuracy of LDA, SVM, and LR with double-precision floating-point arithmetic. All numerical values, including both the input data and the weight values, are represented as floating-point numbers, when training floating-point LDA, SVM, and LR. All numerical experiments run on a 2.9GHz Linux server with 8GB memory.

6.1. Movement Decoding

Brain Computer Interface (BCI) aims to establish a nonconventional communication path between human brain and external devices (e.g., prosthesis) [Nicolelis 2001; Wang et al. 2013]. It has been considered as a promising technology to restore hand and arm function for individuals with stroke or spinal cord injury. In this example, we consider a dataset where 32-channel electrical signals from the cerebral cortex are recorded by electrocorticography (ECoG) [Wang et al. 2013]. A 30-year-old right-handed male subject with tetraplegia caused by a complete C4 level spinal cord injury 10 years ago participates in the experiment for 28 days. There are 70 trials per movement direction in our dataset. A set of features are extracted from these signals by frequency-domain analysis and 42 features are selected based on Fisher criterion to decode the binary movement direction (i.e., left or right) [Bishop 2006; Won et al. 2014]. The dataset is small because there are only a limited number of human subjects who volunteer for ECoG research and collecting the ECoG data from one subject requires a complex neurosurgery, extensive post-surgery training and, hence, is time-consuming. For the aforementioned BCI application, it is extremely important to develop low-power portable/implantable circuits to implement the classification algorithm.

Table I shows the classification error and runtime for LDA-R and LDA-FP that are both implemented with fixed-point arithmetic. The classification error is estimated by using fivefold cross-validation. In Table I, the classification error of LDA-FP is not a strictly monotonic function of word length due to the randomness of our small dataset. In this example, LDA-FP outperforms LDA-R in classification accuracy for a given word length. To achieve the same classification error (i.e., 20.71%), the required word length is 8-bit for LDA-R and 6-bit for LDA-FP. On the other hand, the runtime of LDA-FP is substantially more than that of LDA-R. However, since the classifier training is offline and off-chip, the training cost is not a critical metric to our application of interest.

Table I further shows the classification error for LDA that is implemented with double-precision floating-point arithmetic. Compared to LDA-R and LDA-FP, LDA achieves similar classification error. This observation is consistent with our intuition that as the word length is sufficiently large for fixed-point arithmetic, the impact of

Table I. Classification Error and Runtime for Movement Decoding Using LDA-R, LDA-FP, and LDA

Word Length (*QK.F*)		LDA-R		LDA-FP		LDA
K	*F*	Error	Runtime (Sec)	Error	Runtime (Sec)	Error
1	2	50.00%	0.04	52.14%	40	19.29%
1	3	46.43%	0.06	37.17%	220	
1	4	40.71%	0.06	32.14%	1,914	
1	5	32.14%	0.05	20.71%	2,977	
1	6	21.43%	0.05	19.29%	153	
1	7	20.71%	0.07	20.00%	221	
1	8	21.43%	0.06	19.29%	183	
1	9	20.71%	0.06	18.57%	38	
1	10	20.71%	0.07	18.57%	11	
1	11	21.43%	0.08	19.29%	12	

Table II. Classification Error and Runtime for Movement Decoding Using SVM-R, SVM-FP, and SVM

Word Length (*QK.F*)		SVM-R		SVM-FP		SVM
K	*F*	Error	Runtime (Sec)	Error	Runtime (Sec)	Error
1	2	50.00%	0.60	50.00%	16	20.00%
1	3	50.00%	0.52	38.57%	1,504	
1	4	50.00%	0.58	23.57%	4,121	
1	5	26.43%	0.56	20.00%	3,450	
1	6	20.00%	0.48	20.00%	11,348	
1	7	22.14%	0.62	21.43%	3,163	
1	8	20.00%	0.58	19.29%	199	
1	9	20.00%	0.51	18.57%	87	
1	10	19.29%	0.59	19.29%	62	
1	11	19.29%	0.57	20.00%	62	

Table III. Classification Error and Runtime for Movement Decoding Using LR-R, LR-FP, and LR

Word Length (*QK.F*)		LR-R		LR-FP		LR
K	*F*	Error	Runtime (Sec)	Error	Runtime (Sec)	Error
1	2	50.00%	1.70	50.00%	109	19.29%
1	3	50.00%	1.70	35.71%	18,569	
1	4	50.00%	1.70	27.14%	35,481	
1	5	30.71%	1.74	20.72%	38,039	
1	6	21.43%	1.65	19.29%	103,346	
1	7	21.43%	1.60	20.71%	54,068	
1	8	19.29%	1.54	19.29%	8,041	
1	9	20.00%	1.58	20.00%	7,186	
1	10	19.29%	1.60	20.00%	5,616	
1	11	19.29%	1.61	19.29%	3,699	

rounding and overflow should be negligible and, therefore, both fixed-point and floating-point implementations should result in similar accuracy.

Tables II and III show the classification results for SVM-R, SVM-FP, SVM, LR-R, LR-FP, and LR. Generally speaking, LDA, SVM, and LR rely on different optimization formulations for classifier training. The overall classification error remains large (i.e., around 20%) in this example, because decoding brain states has been considered as an extremely complex and challenging task in the BCI community [Wang et al. 2013].

Table IV. Classification Error and Runtime for Drowsiness Detection Using LDA-R, LDA-FP, and LDA

Word Length ($QK.F$)		LDA-R		LDA-FP		LDA
K	F	Error	Runtime (sec)	Error	Runtime (sec)	Error
1	2	49.78%	0.06	34.83%	91	31.37%
1	3	49.36%	0.06	32.84%	207	
1	4	34.80%	0.06	31.37%	419	
1	5	32.21%	0.06	31.53%	18	
1	6	32.22%	0.06	30.98%	28	
1	7	30.48%	0.06	31.08%	19	

Considering the runtime, the computational cost of LDA-R, SVM-R, and LR-R is almost independent of the word length. Even when the word length varies, the same optimization problem is solved for LDA-R, SVM-R, and LR-R to determine the weight vector $\mathbf{w}$ and then round $\mathbf{w}$ to its fixed-point representation. On the other hand, both SVM-FP and LR-FP are more computationally expensive than LDA-FP when the word length is large due to the following reasons. First, the optimization formulation of SVM-FP in Equation (34) contains a large number of constraint functions, thereby resulting in high computational complexity. Second, the cost function of LR-FP in Equation (40) is composed of many $\max(\cdot)$ operators. Since these $\max(\cdot)$ functions are not smooth, they should be cast to a set of linear constraints by introducing additional slack variables [Boyd and Vandenberghe 2004]. It, in turn, leads to a large number of constraint functions and, consequently, expensive computational cost. Because LDA-FP, SVM-FP, and LR-FP are all formulated as MIP problems, the computational time required to solve them by the branch-and-bound method heavily depends on the problem size. In the worst case, the computational time grows exponentially with problem size and, hence, may become intractable.

6.2. Drowsiness Detection

The need for a reliable detection system of drowsiness is arising every day and it is considered as the leading objective in the development of new Advanced Driver Assistance systems [Correa et al. 2014]. There are several measures that can be used to assess the vigilance level, including behavioral measures, vehicle-based measures, and physiological measures [Sahayadhas et al. 2012]. In the literature, physiological measures are considered as the most reliable metrics for drowsiness detection, especially, across different individuals.

In this example, an EEG recording of 16 human subjects from the MIT-BIH Polysomnographic Database is used [Goldberger et al. 2000; Ichimaru and Moody 1999]. All participants are male and range between 32 and 56 years old. The database in total contains over 80 hours of polysomnographic recording for ECG, EEG, and respiration signals. The sleep stages are scored by experts. The epochs labeled as "Awake Stage" (AS) and "Stage I" (S1) have been used in this example where S1 corresponds to the drowsy stage. The AS epochs used in this article are only those preceding S1, and the S1 epochs used are only those subsequent to AS. All other epochs are excluded in our experiment. There are 5,435 trials where each trial contains a 10-second EEG segment. In total, 35 features are extracted from these EEG signals [Correa et al. 2014; Yeoa et al. 2009] and used to distinguish the two classes: AS and S1.

Tables IV, V, and VI show the classification error and runtime for different methods. Similar to the previous example, the classification error is estimated by using fivefold cross-validation. Several important observations can be made from these results. First, LDA-FP is more accurate than LDA-R with the same word length. To achieve the same

Table V. Classification Error and Runtime for Drowsiness Detection Using SVM-R, SVM-FP, and SVM

Word Length (*QK.F*)		SVM-R		SVM-FP		SVM
K	*F*	Error	Runtime (sec)	Error	Runtime (sec)	Error
1	2	50.00%	1.24	50.00%	13	31.51%
1	3	50.00%	1.21	36.30%	1,857	
1	4	34.41%	1.44	32.19%	1,485	
1	5	32.53%	1.33	32.30%	1,448	
1	6	32.01%	1.24	32.00%	1,056	
1	7	31.85%	1.19	31.76%	662	

Table VI. Classification Error and Runtime for Drowsiness Detection Using LR-R, LR-FP, and LR

Word Length (*QK.F*)		LR-R		LR-FP		LR
K	*F*	Error	Runtime (Sec)	Error	Runtime (Sec)	Error
1	2	50.00%	49.05	34.23%	9,971	31.77%
1	3	50.00%	45.90	32.64%	79,152	
1	4	34.69%	48.29	32.11%	185,551	
1	5	32.31%	50.29	32.72%	199,892	
1	6	31.67%	44.85	32.47%	75,917	
1	7	31.74%	44.73	32.47%	19,818	

classification accuracy, the required word length is 5-bit for LDA-R and 3-bit for LDA-FP.

Second, when the word length is sufficiently large, both the fixed-point and floating-point classifiers achieve similar classification error. In other words, further increasing the word length for our fixed-point classifiers would not improve the classification accuracy any more. The detection accuracy in this article is similar to that reported in the literature [Correa et al. 2014; Yeo et al. 2009].

Third, LDA, SVM, and LR offer similar classification accuracy in this example. However, the runtime of LDA-FP is substantially less than that of SVM-FP and LR-FP, when the word length is sufficiently large. Here, the optimization formulations of both SVM-FP and LR-FP involve a large number of constraint functions and, therefore, are more computationally expensive to solve than LDA-FP, similar to what we observed in the previous example.

7. CONCLUSIONS

In this article, we develop several novel machine-learning algorithms (e.g., LDA-FP, SVM-FP, and LR-FP) to train robust classifiers that are suitable for on-chip low-power implementation with fixed-point arithmetic. These training algorithms formulate the corresponding MIP problems that take into account the nonidealities (i.e., rounding and overflow) associated with fixed-point arithmetic. In addition, branch-and-bound methods with various heuristics are used to solve the MIP problems robustly (i.e., with guaranteed global optimum). Our numerical experiments demonstrate that the proposed methods are able to achieve up to 1.67× reduction in the word length compared to the conventional approaches without surrendering any classification accuracy. Our proposed techniques can be applied to a broad range of emerging applications such as the brain signal processing examples demonstrated in this article.

ACKNOWLEDGMENTS

The authors wish to thank Dr. Wei Wang from University of Pittsburgh for sharing his valuable experience and measurement data of ECoG-based BCI.

REFERENCES

Christopher Bishop. 2006. *Pattern Recognition and Machine Learning* (1st. ed.). Springer, New York, NY.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization* (1st. ed.). Cambridge University Press, Cambridge, UK. DOI:http://dx.doi.org/10.1017/cbo9780511804441

Lei Clifton, David A. Clifton, Marco A. F. Pimentel, Peter J. Watkinson, and Lionel Tarassenko. 2012. Gaussian process regression in vital-sign early warning systems. In *Proceedings of the 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 6161–6164. DOI:http://dx.doi.org/10.1109/embc.2012.6347400

George A. Constantinides, Peter Y. K. Cheung, and Wayne Luk. 2003. Wordlength optimization for linear digital signal processing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 22, 10 (Oct. 2003), 1432–1442. DOI:http://dx.doi.org/10.1109/tcad.2003.818119

Agustina Garcés Correa, Lorena Oroscoe, and Eric Laciar. 2014. Automatic detection of drowsiness in EEG records based on multimodal analysis. *Medical Engineering & Physics* 36, 2 (Feb. 2014), 244–249. DOI:http://dx.doi.org/10.1016/j.medengphy.2013.07.011

Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation* 101, 23 (June 2000), 215–220. DOI:http://dx.doi.org/10.1161/01.cir.101.23.e215

Gene H. Golub and Charles F. Van Loan. 2012. *Matrix Computations* (4th. ed.). Johns Hopkins University Press, Baltimore, MD.

Micheal Grant and Stephen Boyd. 2014. *CVX: MATLAB Software for Disciplined Convex Programming*, version 2.1. http://cvxr.com/cvx.

Kan-Lin Hsiung, Seung-Jean Kim, and Stephen Boyd. 2008. Tractable approximate robust geometric programming. *Optimization and Engineering* 9, 2 (June 2008), 95–118. DOI:http://dx.doi.org/10.1007/s11081-007-9025-z

Yuhei Ichimaru and George Moody. 1999. Development of the polysomnographic database on CD-ROM. *Psychiatry and Clinical Neurosciences* 53, 2 (April 1999), 175–177. DOI:http://dx.doi.org/10.1046/j.1440-1819.1999.00527.x

Hyejung Kim, Refet Firat Yazicioglu, Sunyoung Kim, Nick Van Helleputte, Antonio Artes, Mario Konijnenburg, Jos Huisken, Julien Penders, and Chris Van Hoof. 2011. A configurable and low-power mixed signal SoC for portable ECG monitoring applications. In *Proceedings of the 2011 Symposium on VLSI Circuits (VLSIC)*, 142–143.

Adam B. Kinsman and Nicola Nicolici. 2011. Automated range and precision bit-width allocation for iterative computations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30, 9 (Sept. 2011), 1265–1278. DOI:http://dx.doi.org/10.1109/tcad.2011.2152840

Morten L. Kringelbach, Ned Jenkinson, Sarah L. F. Owen, and Tipu Z. Aziz. 2007. Translational principles of deep brain stimulation. *Nature Reviews Neuroscience* 8, 8 (Aug. 2007), 623–635. DOI:http://dx.doi.org/10.1038/nrn2196

Arindam Mallik, Debjit Sinha, Prithviraj Banerjee, and Hai Zhou. 2007. Low-power optimization by smart bit-width allocation in a SystemC-based ASIC design environment. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26, 3 (March 2007), 447–455. DOI:http://dx.doi.org/10.1109/tcad.2006.888291

Tom Mitchell. 1997. *Machine Learning* (1st. ed.). McGraw-Hill, New York, NY.

MOSEK ApS. 2015. *The MOSEK Optimization Toolbox for MATLAB Manual*, version 7.1 (revision 28). http://docs.mosek.com/7.1/toolbox/index.html.

Miguel A. L. Nicolelis. 2001. Actions from thoughts. *Nature* 409 (2001), 403–407.

Wayne T. Padgett and David V. Anderson. 2009. *Fixed-Point Signal Processing*. Morgan & Claypool. DOI:http://dx.doi.org/10.2200/s00220ed1v01y200909 spr009

Taehwan Roh, Sunjoo Hong, Hyunwoo Cho, and Hoi-Jun Yoo. 2012. A 259.6μW HRV-EEG chaos processor with body channel communication interface for mental health monitoring. In *Proceedings of the 2012 IEEE International Solid-State Circuits Conference*, 294–296. DOI:http://dx.doi.org/10.1109/isscc.2012.6177020

Arun Sahayadhas, Kenneth Sundaraj, and Murugappan Murugappan. 2012. Detecting driver drowsiness based on sensors: A review. *Sensors* 12, 12 (2012), 16937–16953. DOI: http://dx.doi.org/10.3390/s121216937

Mohammed Shoaib, Niraj K. Jha, and Naveen Verma. 2012. A compressed-domain processor for seizure detection to simultaneously reduce computation and communication energy. In *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference*, 1–4. DOI: http://dx.doi.org/10.1109/cicc.2012.6330601

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley, New York, NY.

Wei Wang, Jennifer L. Collinger, Alan D. Degenhart, Elizabeth C. Tyler-Kabara, Andrew B. Schwartz, Daniel W. Moran, Douglas J. Weber, Brian Wodlinger, Ramana K. Vinjamuri, Robin C. Ashmore, John W. Kelly, and Michael L. Boninger. 2013. An electrocorticographic brain interface in an individual with tetraplegia. *PloS ONE* 8, 2 (Feb. 2013), e55344. DOI: http://dx.doi.org/10.1371/journal.pone.005534

Eric S. Winokur, Maggie K. Delano, and Charles G. Sodini. 2013. A wearable cardiac monitor for long-term data acquisition and analysis. *IEEE Transactions on Biomedical Engineering* 60, 1 (Feb. 2013), 189–192. DOI: http://dx.doi.org/10.1109/tbme.2012.2217958

Laurence A. Wolsey. 1998. *Integer Programming* (1st. ed.). Wiley, New York, NY.

Minho Won, Hassan Albalawi, Xin Li, and Donald E. Thomas. 2014. Low-power hardware implementation of movement decoding for brain computer interface with reduced-resolution discrete cosine transform. In *Proceedings of the 2014 Annual International Conference of the IEEE Engineering in Medicine and Biology Society,* 1626–1629. DOI: http://dx.doi.org/10.1109/EMBC.2014.6943916

Mervyn V. M. Yeoa, Xiaoping Lia, B. Kaiquan Shenb, and Einar P. V. Wilder-Smith. 2009. Can SVM be used for automatic EEG detection of drowsiness during car driving? *Safety Science* 47, 1 (Jan. 2009), 115–124. DOI: http://dx.doi.org/10.1016/j.ssci.2008.01.007

Jerald Yoo, Long Yan, Dina El-Damak, Muhammad Bin Altaf, Ali Shoeb, Hoi-Jun Yoo, and Anantha Chandrakasan. 2013. An 8-channel scalable EEG acquisition SoC with patient-specific seizure classification and recording processor. *IEEE Journal of Solid-State Circuit* 48, 1 (Feb. 2012), 214–228. DOI: http://dx.doi.org/10.1109/isscc.2012.6177019

Received August 2016; revised February 2017; accepted February 2017