Computer-Aided Design of Machine Learning Algorithm: Training Fixed-Point Classifier for On-Chip Low-Power Implementation

Hassan Albalawi, Yuanning Li and Xin Li
Electrical & Computer Engineering Department, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213 USA
{halbalaw, ynli, xinli}@cmu.edu

ABSTRACT

In this paper, we propose a novel linear discriminant analysis algorithm, referred to as LDA-FP, to train on-chip classifiers that can be implemented with low-power fixed-point arithmetic with extremely small word length. LDA-FP incorporates the non-idealities (i.e., rounding and overflow) associated with fixed-point arithmetic into the training process so that the resulting classifiers are robust to these non-idealities. Mathematically, LDA-FP is formulated as a mixed integer programming problem that can be efficiently solved by a novel branch-and-bound method proposed in this paper. Our numerical experiments demonstrate that LDA-FP substantially outperforms the conventional approach for the emerging biomedical application of brain computer interface.

1. INTRODUCTION

Machine learning is an important technique that has been continuously growing during the past several decades [1]-[2]. Large-scale machine learning 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 [3]-[7], 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 [8] and deep brain stimulation [9]. 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., cloud server) for remote processing.
- **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 10 μW), 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 [10]-[13], 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 paper, we consider a case study of *linear discriminant analysis* (LDA) [2] for binary classification. 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 LDA algorithm developed for double-procession floating-point operation must be "re-designed" in order to be made "robust" to rounding error. Similar "numerical" issues have been extensively observed 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 [14]. The fundamental question here is how we can improve the robustness of LDA and make it suitable for on-chip low-power implementation.

Towards this goal, we propose a new LDA algorithm for fixed-point computation, which is referred to as LDA-FP in this paper. LDA-FP is formulated as a mixed integer programming problem with consideration of the non-idealities (i.e., rounding and overflow) posed by fixed-point arithmetic. Furthermore, a branch-and-bound method with several efficient heuristics is developed to find the globally optimal classification boundary of LDA-FP. With our re-designed training algorithm, LDA-FP is made maximally robust to the non-idealities associated with fixed-point arithmetic and, hence, can be efficiently implemented with extremely small word length for on-chip low-power operation.

Our proposed LDA-FP algorithm is validated for two different test cases, including (i) a synthetic data set, and (ii) a practical application of movement decoding for brain computer interface [15]-[16]. As will be demonstrated by our experimental results in Section 5, LDA-FP is able to reduce the word length by up to 3× (i.e., equivalent to 9× power reduction) compared to the conventional LDA algorithm, without surrendering any classification accuracy.

The remainder of this paper is organized as follows. In Section 2, we briefly review the background of LDA. In Section 3, we derive our proposed LDA-FP formulation and then develop an efficient branch-and-bound solver for LDA-FP in Section 4. The efficacy of LDA-FP is demonstrated by several numerical examples in Section 5. Finally, we conclude in Section 6.

DAC '14, June 01 - 05 2014, San Francisco, CA, USA

http://dx.doi.org/10.1145/2593069.2593110

2. BACKGROUND

LDA is a machine learning algorithm that has been extensively applied to a large number of binary classification problems [2]. 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 $\mathbf{w}$ [2], as shown in Figure 1.

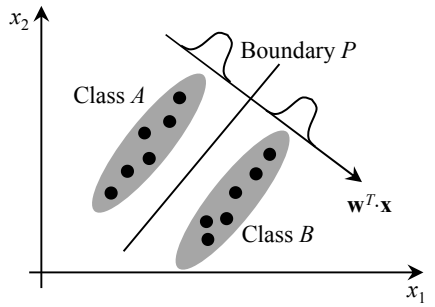

Figure 1. LDA aims to maximally separate two classes by projecting the feature vector $\mathbf{x}$ along the optimal direction $\mathbf{w}$.

Towards 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 $\mathbf{x}$ [2]:

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

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

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 [2].

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 [2]:

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

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

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 $S_{B,y}$ [2]:

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

There are two important clarifications that should be made regarding the optimization formulation in (10). First, it is easy to verify that the optimal solution $\mathbf{w}$ of (10) is not unique, since the cost function in (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 (10), multiplying $\mathbf{w}$ by any non-zero scalar λ (i.e., $\lambda\cdot\mathbf{w}$) does not change the cost function and, hence, is also an optimal solution of (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 $\|\bullet\|_2$ denotes the L_2-norm of a vector) so that the optimal solution becomes unique.

Second, even though the optimization formulation in (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) [2]. Assuming that the within-class scatter matrix $\mathbf{S}_{W,\mathbf{x}}$ is full-rank, the optimal $\mathbf{w}$ of (10) can be proven to be [2]:

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

The weight vector $\mathbf{w}$ in (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.

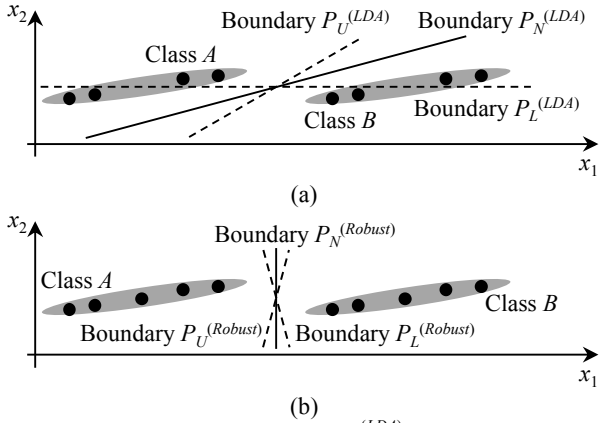

Figure 2. (a) The optimal boundary $P_N^{(LDA)}$ determined by LDA is extremely sensitive to rounding error. (b) A robust boundary $P_N^{(Robust)}$ can be found and it is insensitive to rounding error.

The conventional LDA algorithm is able to find the optimal decision boundary in (12), 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 (12) is no longer optimal.

To understand the reason, we consider a simple 2-D example in Figure 2, where 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 2(a). In

addition to this nominal boundary $P_N^{(LDA)}$, Figure 2(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 2(b) shows a robust boundary $P_N^{(Robust)}$ that is less sensitive to rounding error than $P_N^{(LDA)}$. Note that 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 LDA may result in large classification error, since it does not explicitly consider the rounding error during classifier training. It, in turn, motivates us to re-design the conventional LDA algorithm in order to generate a robust classifier that is least sensitive to rounding error.

3. FIXED-POINT CLASSIFIER

The non-idealities (i.e., rounding and overflow) posed by fixed-point arithmetic can be broadly classified into two different categories: (i) the non-idealities associated with the feature vector **x**, and (ii) the non-idealities associated with the weight vector **w**. For the feature vector **x**, all features in **x** can be carefully scaled to avoid overflow. In addition, the feature vector **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 **x**. In other words, the conventional LDA algorithm is able to address the non-idealities for the feature vector **x** appropriately.

On the other hand, mitigating the non-idealities of the weight vector **w** is not trivial. If we simply follow the conventional LDA algorithm, **w** is determined by (11) and then normalized/rounded to its fixed-point representation. In this case, large classification error may occur, as is demonstrated by the simple example in Figure 2. Hence, our focus of this section is to derive a new LDA-FP algorithm to solve the optimal **w** that is suitable for fixed-point implementation.

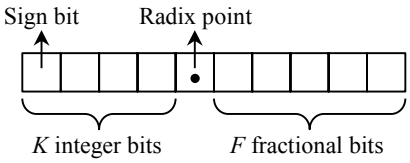

Figure 3. An example is shown for the fixed-point format of *QK.F* based on two's complement.

Without loss of generality, we assume that a fixed-point number is represented in the format of *QK.F* based on two's complement [13]. It has *K* integer bits (including the sign bit) and *F* fractional bits, as shown in Figure 3. The total word length is $K + F$. In this paper, we further assume that all fixed-point operations in the classifier are implemented 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 **w** can be assigned with different word lengths. However, since we mainly focus on classifier training in this paper, the problem of word length optimization should be considered as a separate topic for our future research.

Given the fixed-point representation *QK.F* shown in Figure 3, we must consider a number of important constraints, when deriving our proposed optimization formulation for LDA-FP.

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

$$w_m \in \Omega \quad (m = 1, 2, \cdots, M), \tag{13}$$

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

$$\mathbf{x} \sim \begin{cases} Gauss(\boldsymbol{\mu}_A, \boldsymbol{\Sigma}_A) & (Class\ A) \\ Gauss(\boldsymbol{\mu}_B, \boldsymbol{\Sigma}_B) & (Class\ B) \end{cases}. \tag{14}$$

Note that the probability distribution of **x** depends on the class that **x** belongs to. Such a Gaussian model has been widely applied in the literature by the machine learning community [2].

Based on (14), each multiplication $w_m \cdot x_m$, where $m \in \{1, 2, \ldots, M\}$, yields a Gaussian distribution:

$$w_m \cdot x_m \sim \begin{cases} Gauss\left(w_m \cdot \mu_{A,m}, w_m^2 \cdot \sigma_{A,m}^2\right) & (Class\ A) \\ Gauss\left(w_m \cdot \mu_{B,m}, w_m^2 \cdot \sigma_{B,m}^2\right) & (Class\ B) \end{cases}, \tag{15}$$

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 (15), we can statistically define a confidence interval for $w_m \cdot x_m$:

$$\beta = \Phi^{-1}(0.5 + 0.5 \cdot \rho) \tag{16}$$

$$\begin{matrix} \left[w_m\mu_{A,m} - \beta \cdot |w_m| \cdot \sigma_{A,m},\ w_m\mu_{A,m} + \beta \cdot |w_m| \cdot \sigma_{A,m}\right] (Class\ A) \\ \left[w_m\mu_{B,m} - \beta \cdot |w_m| \cdot \sigma_{B,m},\ w_m\mu_{B,m} + \beta \cdot |w_m| \cdot \sigma_{B,m}\right] (Class\ B) \end{matrix}, \tag{17}$$

where ρ denotes the confidence level and $\Phi^{-1}(\bullet)$ 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 (17) "almost" covers the range of $w_m \cdot x_m$. Hence, the confidence interval in (17) must be within the range of the fixed-point representation *QK.F* to avoid overflow:

$$\begin{aligned} w_m \cdot \mu_{A,m} - \beta \cdot |w_m| \cdot \sigma_{A,m} &\geq -2^{K-1} \\ w_m \cdot \mu_{B,m} - \beta \cdot |w_m| \cdot \sigma_{B,m} &\geq -2^{K-1} \\ w_m \cdot \mu_{A,m} + \beta \cdot |w_m| \cdot \sigma_{A,m} &\leq 2^{K-1} - 2^{-F} \\ w_m \cdot \mu_{B,m} + \beta \cdot |w_m| \cdot \sigma_{B,m} &\leq 2^{K-1} - 2^{-F} \end{aligned}. \tag{18}$$

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:

$$\mathbf{w}^T \cdot \mathbf{x} \sim \begin{cases} Gauss\left(\mathbf{w}^T \cdot \boldsymbol{\mu}_A, \mathbf{w}^T \cdot \boldsymbol{\Sigma}_A \cdot \mathbf{w}\right) & (Class\ A) \\ Gauss\left(\mathbf{w}^T \cdot \boldsymbol{\mu}_B, \mathbf{w}^T \cdot \boldsymbol{\Sigma}_B \cdot \mathbf{w}\right) & (Class\ B) \end{cases}. \tag{19}$$

Similar to (15)-(18), we can statistically define the confidence interval for y and then set up the following constraints to prevent y from overflowing:

$$\begin{aligned} \mathbf{w}^T\boldsymbol{\mu}_A - \beta \cdot \sqrt{\mathbf{w}^T\boldsymbol{\Sigma}_A\mathbf{w}} &\geq -2^{K-1} \\ \mathbf{w}^T\boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T\boldsymbol{\Sigma}_B\mathbf{w}} &\geq -2^{K-1} \\ \mathbf{w}^T\boldsymbol{\mu}_A + \beta \cdot \sqrt{\mathbf{w}^T\boldsymbol{\Sigma}_A\mathbf{w}} &\leq 2^{K-1} - 2^{-F} \\ \mathbf{w}^T\boldsymbol{\mu}_B + \beta \cdot \sqrt{\mathbf{w}^T\boldsymbol{\Sigma}_B\mathbf{w}} &\leq 2^{K-1} - 2^{-F} \end{aligned}. \tag{20}$$

On the other hand, it is important to note that we do not need

to explicitly set up any overflow constraint for the intermediate result 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.

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

Finally, combining (10), (13), (18) and (20) yields the following constrained optimization problem for our proposed LDA-FP algorithm:

$$\begin{aligned}
\min_{\mathbf{w}} \quad & \frac{\mathbf{w}^T \cdot \mathbf{S}_{W,\mathbf{x}} \cdot \mathbf{w}}{\left[\left(\boldsymbol{\mu}_A - \boldsymbol{\mu}_B\right)^T \cdot \mathbf{w}\right]^2} \\
\text{S.T.} \quad & w_m \cdot \mu_{A,m} - \beta \cdot |w_m| \cdot \sigma_{A,m} \geq -2^{K-1} \qquad (m = 1, 2, \cdots, M) \\
& w_m \cdot \mu_{B,m} - \beta \cdot |w_m| \cdot \sigma_{B,m} \geq -2^{K-1} \qquad (m = 1, 2, \cdots, M) \\
& w_m \cdot \mu_{A,m} + \beta \cdot |w_m| \cdot \sigma_{A,m} \leq 2^{K-1} - 2^{-F} \qquad (m = 1, 2, \cdots, M) \\
& w_m \cdot \mu_{B,m} + \beta \cdot |w_m| \cdot \sigma_{B,m} \leq 2^{K-1} - 2^{-F} \qquad (m = 1, 2, \cdots, M) \\
& \mathbf{w}^T \boldsymbol{\mu}_A - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_A \mathbf{w}} \geq -2^{K-1} \\
& \mathbf{w}^T \boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_B \mathbf{w}} \geq -2^{K-1} \\
& \mathbf{w}^T \boldsymbol{\mu}_A + \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_A \mathbf{w}} \leq 2^{K-1} - 2^{-F} \\
& \mathbf{w}^T \boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_B \mathbf{w}} \leq 2^{K-1} - 2^{-F} \\
& w_m \in \Omega \qquad (m = 1, 2, \cdots, M)
\end{aligned} \tag{21}$$

Note that the optimization formulation in (21) represents a mixed integer programming problem [17]-[18] that is difficult to solve due to the following two reasons. First, the solution **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 (21) and find the global optimum. The details of our proposed solver will be discussed in the next section.

4. BRANCH-AND-BOUND SOLVER

Branch-and-bound method has been widely used to solve mixed integer programming problems [17]. The key idea is to iteratively partition the optimization variables into sub-intervals. For each sub-interval, the lower and upper bounds of the cost function are estimated in order to efficiently remove the irrelevant sub-intervals that do not contain the optimal solution from the search space. However, when the branch-and-bound method is applied to solve (21), it is not trivial to efficiently estimate the lower and upper bounds for a given sub-interval of **w**, because the cost function in (21) is the ratio of two quadratic functions and, hence, not convex [18]. In what follows, we will propose several novel ideas to address this technical challenge so that the optimization problem in (21) can be solved by the branch-and-bound method both efficiently (i.e., with low computational cost) and robustly (i.e., with guaranteed global optimum).

- **Lower bound estimation**: We introduce an additional variable:

$$t = \left(\boldsymbol{\mu}_A - \boldsymbol{\mu}_B\right)^T \cdot \mathbf{w}, \tag{22}$$

and re-write the cost function in (21) as:

$$\min_{\mathbf{w},t} \quad \frac{\mathbf{w}^T \cdot \mathbf{S}_{W,\mathbf{x}} \cdot \mathbf{w}}{t^2}. \tag{23}$$

Note that both **w** and t are now considered as optimization variables and they should be partitioned into sub-intervals when searching for the optimal solution by the branch-and-bound method.

With a given sub-interval:

$$\begin{aligned} l_{wm} \leq w_m \leq u_{wm} \quad (m = 1, 2, \cdots, M) \\ l_t \leq t \leq u_t \end{aligned}, \tag{24}$$

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:

$$\begin{aligned}
\min_{\mathbf{w},t} \quad & \frac{1}{\eta} \cdot \mathbf{w}^T \cdot \mathbf{S}_{W,\mathbf{x}} \cdot \mathbf{w} \\
\text{S.T.} \quad & w_m \cdot \mu_{A,m} - \beta \cdot |w_m| \cdot \sigma_{A,m} \geq -2^{K-1} \qquad (m = 1, 2, \cdots, M) \\
& w_m \cdot \mu_{B,m} - \beta \cdot |w_m| \cdot \sigma_{B,m} \geq -2^{K-1} \qquad (m = 1, 2, \cdots, M) \\
& w_m \cdot \mu_{A,m} + \beta \cdot |w_m| \cdot \sigma_{A,m} \leq 2^{K-1} - 2^{-F} \qquad (m = 1, 2, \cdots, M) \\
& w_m \cdot \mu_{B,m} + \beta \cdot |w_m| \cdot \sigma_{B,m} \leq 2^{K-1} - 2^{-F} \qquad (m = 1, 2, \cdots, M) \\
& \mathbf{w}^T \boldsymbol{\mu}_A - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_A \mathbf{w}} \geq -2^{K-1} \\
& \mathbf{w}^T \boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_B \mathbf{w}} \geq -2^{K-1} \\
& \mathbf{w}^T \boldsymbol{\mu}_A + \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_A \mathbf{w}} \leq 2^{K-1} - 2^{-F} \\
& \mathbf{w}^T \boldsymbol{\mu}_B - \beta \cdot \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_B \mathbf{w}} \leq 2^{K-1} - 2^{-F} \\
& t = \left(\boldsymbol{\mu}_A - \boldsymbol{\mu}_B\right)^T \cdot \mathbf{w} \\
& l_{wm} \leq w_m \leq u_{wm} \qquad (m = 1, 2, \cdots, M) \\
& l_t \leq t \leq u_t
\end{aligned} \tag{25}$$

where η is a constant defined by:

$$\eta = \sup_{l_t \leq t \leq u_t} t^2. \tag{26}$$

In (26), 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 $l_t \leq t \leq u_t$ and determine the value of η.

The optimization in (25) is a convex second-order cone programming problem [18] and, hence, can be solved efficiently. Compared to (21), the formulation in (25) is "relaxed" in two different ways. First, the denominator of the cost function is relaxed to the maximum possible value η over the interval $l_t \leq t \leq u_t$. Second, the vector **w** is no longer constrained to a discrete set; instead, it can take any real value within the sub-interval $\{l_{wm} \leq w_m \leq u_{wm};\ m = 1, 2, \ldots, M\}$. For this reason, solving (25) yields a lower bound of the cost function of the original mixed integer programming problem for the given sub-interval defined in (24).

- **Upper bound estimation**: Similar to lower bound estimation, we can re-use the optimization formulation in (25) to estimate the upper bound of the cost function with the parameter η set to:

$$\eta = \inf_{l_t \leq t \leq u_t} t^2, \tag{27}$$

where inf(•) denotes the infimum (i.e., the greatest lower bound) of a set. In this case, the optimization in (25) is again a convex second-order cone programming problem. After solving (25), we further round the solution **w** to the discrete set defined in (13). Next, we substitute the rounded **w** to the cost function in (21). It, in turn, results in an upper bound of the cost function of (21) given the sub-interval defined in (24).

- **Initial interval size estimation**: Since the branch-and-bound method iteratively partitions the optimization variables **w** and t into sub-intervals, we need to determine the initial interval size for **w** and t, before starting the first iteration. Since **w** is represented in the format of $QK.F$ as shown in Figure 3, it must be within the following range:

$$-2^{K-1} \le w_m \le 2^{K-1} - 2^{-F} \quad (m = 1, 2, \cdots, M). \tag{28}$$

On the other hand, since t is a linear function of **w** as defined in (22), combining (22) and (28) yields:

$$-2^{K-1} \cdot \left\| \boldsymbol{\mu}_A - \boldsymbol{\mu}_B \right\|_1 \le t \le \left(2^{K-1} - 2^{-F}\right) \cdot \left\| \boldsymbol{\mu}_A - \boldsymbol{\mu}_B \right\|_1, \tag{29}$$

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

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 $QK.F$. Round the training data to their fixed-point representations.
2. Calculate $\boldsymbol{\mu}_A$ and $\boldsymbol{\mu}_B$ by (3)-(4) and $\mathbf{S}_{W,\mathbf{x}}$ by (2).
3. Initialize the search interval ξ by (28)-(29). 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 sub-intervals. Remove the selected interval from Ξ.
5. For each of these two sub-intervals, 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 sub-interval 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.

With the aforementioned heuristics, our proposed branch-and-bound method for solving (21) 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. It is important to mention that our implementation of Algorithm 1 includes a number of additional heuristics to speed up the search process. Due to the page limit, these additional heuristics are not discussed in this paper.

5. NUMERICAL EXAMPLES

In this section, two test cases are presented to demonstrate the efficacy of our proposed LDA-FP algorithm. For testing and comparison purposes, two different LDA algorithms are implemented: (i) the conventional LDA where the weight vector **w** is solved by (11) and then rounded to its fixed-point representation, and (ii) the proposed LDA-FP where the weight vector **w** is solved from (21) by Algorithm 1. All numerical experiments are run on a 2.9GHz Linux server with 8GB memory.

5.1 Synthetic Data

In this sub-section, a synthetic example is constructed to provide deep insights for comparing LDA and LDA-FP. In our example, we define three features x_1, x_2 and x_3:

$$x_1 = \begin{cases} -0.5 + 0.58 \cdot \varepsilon_1 + 0.58 \cdot \varepsilon_2 + 0.58 \cdot \varepsilon_3 & (Class\ A) \\ 0.5 + 0.58 \cdot \varepsilon_1 + 0.58 \cdot \varepsilon_2 + 0.58 \cdot \varepsilon_3 & (Class\ B) \end{cases} \tag{30}$$

$$x_2 = 0.001 \cdot \varepsilon_2 + \varepsilon_3 \quad (Class\ A\ and\ B) \tag{31}$$

$$x_3 = \varepsilon_3 \quad (Class\ A\ and\ B), \tag{32}$$

where ε_1, ε_2 and ε_3 are three independent random variables with standard Gaussian distributions. Studying (30)-(32), we notice that the discriminant information required for classification is carried by the first feature x_1 only. However, even though the other two features x_2 and x_3 do not carry discriminant information, they can possibly be used to cancel the noise terms ε_2 and ε_3. In other words, by combining x_1, x_2 and x_3 together, the weighted sum $w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3$ may be independent of ε_2 and ε_3 if the weight values w_1, w_2 and w_3 are appropriately chosen.

Table 1. Classification error and runtime for synthetic data set based on fixed-point arithmetic

Word Length (Bit)	LDA	LDA-FP	
	Error	Error	Runtime (Sec)
4	50.00%	27.04%	0.81
6	50.00%	26.83%	5.87
8	50.00%	25.98%	20.42
10	50.00%	22.62%	29.16
12	24.46%	19.60%	29.11
14	19.48%	19.33%	0.06
16	19.33%	19.33%	0.06

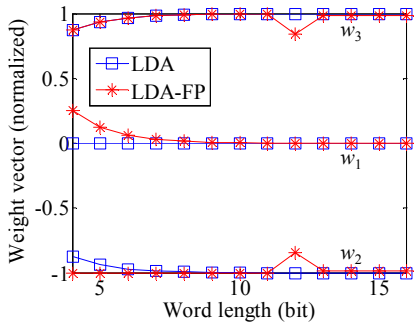

Figure 4. The three weight values w_1, w_2 and w_3 are plotted as functions of the word length.

Table 1 shows the classification error and runtime as functions of the word length of fixed-point representation. The conventional LDA algorithm only needs to solve a linear equation to determine the weight vector **w**, as shown in (11). Hence, its runtime is almost negligible and is not reported here. As shown in Table 1, LDA cannot generate a meaningful classification result that is above the chance level (i.e., 50% for binary classification) until the word length reaches 12. On the other hand, LDA-FP provides good classification accuracy, even when the word length is as small as 4. In this example, LDA-FP successfully reduces the required word length by up to 3×. Since the power consumption of on-chip fixed-point arithmetic is almost a quadratic function of the word length [13], LDA-FP reduces the power consumption by up to 9× in this example.

To further understand the reason why LDA-FP outperforms LDA in this example, we plot the three weight values w_1, w_2 and w_3 in Figure 4. Remember that only the feature x_1 carries the

discriminant information, while the other two features x_2 and x_3 are useful for noise cancellation only. In order to perfectly remove the noise terms ε_2 and ε_3, w_2 and w_3 must be extremely large, while w_1 is relatively small. If the word length is sufficiently large, LDA is able to accomplish the aforementioned noise cancellation. However, as the word length decreases, the small value of w_1 is simply rounded to zero by LDA, resulting in large classification error. On the other hand, the proposed LDA-FP algorithm is able to increase w_1 to a non-zero value as shown in Figure 4, when the word length is extremely small. Therefore, LDA-FP is able to achieve good classification accuracy with small word length, even though the noise terms ε_2 and ε_3 cannot be perfectly removed in this case.

5.2 Brain Computer Interface

Brain computer interface (BCI) aims to establish a non-conventional communication path between human brain and external devices (e.g., prosthesis) [15]-[16]. It has been considered as a promising technology to restore hand and arm function for the individuals with stroke or spinal cord injury. In this example, we consider a data set where the electrical signals from the cerebral cortex are recorded by electrocorticography (ECoG) and 42 features are extracted from these signals to decode the binary movement direction (i.e., left or right) [16]. There are 70 trials per movement direction in our data set. For such a BCI application, it is extremely important to develop low-power portable/implantable circuits to implement the classification algorithm.

Table 2. Classification error and runtime for brain computer interface (BCI) based on fixed-point arithmetic

Word Length (Bit)	LDA	LDA-FP	
	Error	Error	Runtime (Sec)
3	50.00%	52.14%	39.9
4	46.43%	37.17%	219.7
5	40.71%	32.14%	1913.5
6	32.14%	20.71%	2977.0
7	21.43%	19.29%	152.8
8	20.71%	20.00%	221.1

Table 2 shows the classification error and runtime for LDA and LDA-FP that are both implemented with fixed-point arithmetic. The classification error is estimated by using 5-fold cross-validation. In Table 2, the classification error of LDA-FP is not a strictly monotonic function of word length due to the randomness of our small data set. In this example, LDA-FP again outperforms LDA in classification accuracy for a given word length. To achieve the same classification error (e.g., 20.71%), the required word length is 8-bit for LDA and 6-bit for LDA-FP. As the word length is reduced from 8-bit to 6-bit, the power consumption can be reduced by 1.8×. It, in turn, demonstrates the superior performance of LDA-FP over LDA.

6. CONCLUSIONS

In this paper, we develop a novel LDA-FP algorithm to train robust classifiers that are suitable for on-chip low-power implementation with fixed-point arithmetic. LDA-FP is formulated as a mixed integer programming problem that takes into account the non-idealities (i.e., rounding and overflow) associated with fixed-point arithmetic. In addition, a branch-and-bound method with various heuristics is proposed to solve the LDA-FP problem both efficiently (i.e., with low computational cost) and robustly (i.e., with guaranteed global optimum). Our numerical experiments demonstrate that LDA-FP is able to achieve up to 9× reduction in power consumption compared to the conventional LDA algorithm without surrendering any classification accuracy. LDA-FP can be applied to a broad range of emerging applications such as the brain computer interface example demonstrated in this paper.

7. ACKNOWLEDGEMENTS

This work is supported in part by the National Science Foundation under contract CCF–1314876 and by the CMU-SYSU Collaborative Innovation Research Center at Carnegie Mellon University. The authors wish to thank Dr. Wei Wang from University of Pittsburgh for sharing his valuable experience and measurement data of ECoG-based BCI.

8. REFERENCES

[1] T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.
[2] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.
[3] H. Kim, R. Yazicioglu, S. Kim, N. Helleputte, A. Artes, M. Konijnenburg, J. Husken, J. Penders and C. Hoof, "A configurable and low-power mixed signal SoC for portable ECG monitoring applications," *IEEE VLSI Symposium*, pp. 142-143, 2011.
[4] E. Winokur, M. Delano and C. Sodini, "A wearable cardiac monitor for long-term data acquisition and analysis," *IEEE Trans. on BME*, vol. 60, no. 1, pp. 189-192, Jan. 2013.
[5] T. Roh, S. Hong, H. Cho and H. Yoo, "A 259.6μW HRV-EEG chaos processor with body channel communication interface for mental health monitoring," *IEEE ISSCC*, pp. 294-295, 2012.
[6] M. Shoaid, N. Jha and N. Verma, "A compressed-domain processor for seizure detection to simultaneously reduce computation and communication energy," *IEEE CICC*, 2012.
[7] J. Yoo, L. Yan, D. El-Damak, M. Altaf, A. Shoeb and A. Chandrakasan, "An 8-channel scalable EEG acquisition SoC with patient-specific seizure classification and recording processor," *IEEE JSSC*, vol. 48, no. 1, pp. 214-228, Jan. 2013.
[8] L. Clifton, D. Clifton, M. Pimentel and P. Watkinson, "Gaussian process regression in vital-sign early warning systems," *IEEE EMBC*, pp. 6161-6164, 2012.
[9] M. Kringelbach, N. Jenkinson, S. Owen and T. Aziz, "Translational principles of deep brain stimulation," *Nature*, vol. 8, pp. 523-635, Aug. 2007.
[10] G. Constantinides, P. Cheung and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Trans. on CAD*, vol. 22, no. 10, pp. 1432-1442, Oct. 2003.
[11] A. Mallik, D. Sinha, P. Banerjee and H. Zhou, "Low-power optimization by smart bit-width allocation in a SystemC-based ASIC design environment," *IEEE Trans. on CAD*, vol. 26, no. 3, pp. 447-455, Mar. 2007.
[12] A. Kinsman and N. Nicolici, "Automated range and precision bit-width allocation for iterative computations," *IEEE Trans. on CAD*, vol. 30, no. 9, pp. 1265-1278, Sep. 2011.
[13] W. Padgett and D. Anderson, *Fixed-point Signal Processing*, Morgan and Claypool Publishers, 2009.
[14] G. Golub and C. Loan, *Matrix Computations*, Johns Hopkins University Press, 1996.
[15] M. Nicolelis, "Actions from thoughts," *Nature*, vol. 409, pp. 403-407, Jan. 2001.
[16] W, Wang, J. Collinger, Al. Degenhart, E. Tyler-Kabara, A. Schwartz, D. Moran, D. Weber, B. Wodlinger, R. Binjamuri, R. Ashmore, J. Kelly and M. Boninger, "An electrocorticographic brain interface in an individual with tetraplegia," *Plos One*, vol. 8, no. 2, Feb. 2013.
[17] L. Wolsey, *Integer Programming*, John Wiley and Sons, 1998.
[18] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.