Hardware-Based Support Vector Machine Classification in Logarithmic Number Systems

Faisal M. Khan, Mark G. Arnold and William M. Pottenger
Computer Science and Engineering
Lehigh University
Bethlehem, Pennsylvania
faisalmkhan@gmail.com,{marnold,billp}@cse.lehigh.edu

***Abstract*—Support Vector Machines are emerging as a powerful machine-learning tool. Logarithmic Number Systems (LNS) utilize the property of logarithmic compression for numerical operations. We present an implementation of a digital Support Vector Machine (SVM) classifier using LNS in which considerable hardware savings are achieved with no significant loss in classification accuracy.**

I. INTRODUCTION

Cognitive systems capable of gathering information, detecting significant events, making decisions and/or coordinating operations are of value in a wide variety of application domains, from biomedical devices to automated military units. The core functionality of such machine learning involves mathematical kernels employing commonly used operators [6], typically implemented as software-based solutions executing on general-purpose machines. Unfortunately, such solutions require significant resources for execution and may consequently be unsuitable for portable applications. Efficient hardware implementations of machine-learning techniques yield a variety of advantages over software solutions: increased processing speed, reliability and battery life as well as reduced cost and complexity.

However, aside from a plethora of work in neural-network implementations [4], there are few hardware-based machine-learning technologies. This paper describes preliminary research towards the development of robust, hardware-based kernel solutions beyond neural networks for application-specific deployment. Specifically, the research employs Support Vector Machines (SVMs), a representative kernel-based machine-learning technique especially suited to high-dimensional data [6].

We use logarithmic arithmetic for its energy-efficient properties [3,9]. Successful deployment of logarithmic functionality in neural networks has been shown to increase reliability and reduce power usage [2]. We anticipate further progress in kernel-based SVMs since the majority of machine-learning kernels employ multiplication and/or exponentiation operators, the performance of which logarithmic computation significantly improves.

In the following sections, we review SVM and LNS backgrounds along with related work in hardware-based machine-learning. We then present our design, its implementation and its verification. We follow with a conclusion and a discussion of future work.

II. SUPPORT VECTOR MACHINES

The Support Vector Machine (SVM) algorithm is based on statistical learning theory [8]. It has a simple and intuitive algorithm. It performs excellently for complex real-world problems that may be difficult to analyze theoretically.

SVMs are an extension of linear models that are capable of nonlinear classification. Linear models are incapable of representing a concept with nonlinear boundaries between classes. SVMs employ linear models to represent nonlinear class boundaries by transforming the input, or *instance space*, into a new space using a nonlinear mapping.

This transformation is facilitated through the use of kernels. The SVM algorithm can be treated linearly within the instance space, whereas the choice of various kernels may map the core operations transparently to a higher dimensional space. Consequently, complex pattern recognition and classification approaches can abstractly be represented linearly.

Following this transformation, a Maximum Margin Hyperplane (MMH) that separates the instances by class is learned, thereby forming a decision boundary. The MMH comes no closer to a given instance than it must; in the ideal case it optimally separates classes. *Support vectors* are the training instances closest to the MMH. A set of support vectors thus defines the decision boundary for a given set of instances. This simplifies the representation of the decision boundary since other training instances can be disregarded.

SVM training is a complex quadratic optimization problem for obtaining the support vectors $\vec{X}$ (with class values Y), their coefficients α, and a threshold value *b*.

Support Vector classification (in a simple two-class problem) simply looks at the sign of a decision function. A test instance $\vec{T}$ is classified by the following decision function [6], [8]:

$$f(\vec{T}) = sign(\sum_i \alpha_i Y_i K(\vec{T}, \vec{X}_i) + b) \quad (1)$$

The choice of the kernel function $K(\vec{X}_i, \vec{X}_j)$ and the resultant feature space determines the functional form of the support vectors; thus, different kernels behave differently. Some common kernels are [6], [8]:

Linear: $K(\vec{X}, \vec{Y}) = (\vec{X} \bullet \vec{Y})$ (2)

Polynomial: $K(\vec{X}, \vec{Y}) = (\vec{X} \bullet \vec{Y})^d$ (3)

Radial Basis Function (RBF):

$$K(\vec{X}, \vec{Y}) = \exp(-\| \vec{X} - \vec{Y} \|^2 / (2\sigma^2)) \quad (4)$$

Sigmoid: $K(\vec{X}, \vec{Y}) = \tanh(\mathrm{K}(\vec{X} \bullet \vec{Y}) + \Theta)$ (5)

III. HARDWARE-BASED MACHINE LEARNING

There exists a significant lack of hardware-based machine-learning systems. With the exception of neural networks [4], the advantages of portable, dedicated machine-learning ASICs are yet to be explored.

The Kerneltron [5], developed at John Hopkins is a recent SVM classification module. The internally analog, externally digital computational structure employs a massively parallel kernel computation structure. It implements linear and RBF kernels. Due to the internal analog computation, the system is able to achieve a system precision resolution of no more than 8 bits.

Anguita et al. [1] present a recent endeavor in the field. They propose the design of a fully digital architecture for SVM training and classification employing the linear and RBF kernels. The result is a highly optimal SVM ideal for hardware synthesis. The minimal word size they are able to use is 20 bits.

IV. LOGARITHMIC NUMBER SYSTEMS

In contrast to [1,5], we use logarithmic arithmetic due to its high degree of suitability for machine-learning-kernel operations. Based on the once ubiquitous engineer's slide rule [3], Logarithmic Number Systems (LNS) are an alternative to fixed- and floating-point arithmetic. LNS utilize the property of logarithmic compression for numerical operations. Within the logarithmic domain, multiplication and division can be treated simply as addition or subtraction. Hardware computation of these operations is significantly faster with reduced complexity. Employing LNS involves an overhead of conversion to and from the logarithmic domain that is insignificant relative to the reduction in kernel computational complexity [3].

Unlike Floating-Point (FP) systems, the relative error of LNS is constant and LNS can often achieve an equivalent signal-to-noise ratio with fewer bits of precision compared to conventional FP or fixed-point architectures [3,9]. Similar to FP architectures, LNS implementations can represent numbers with relative precision; numbers closer to zero such as those used in SVMs, are represented with better precision in LNS than in fixed-point systems.

LNS provide other benefits conducive to a low-power, reliable application. The logarithmic conversion is inherently a compression algorithm as well. LNS are particularly cost effective when an application performs acceptably with reduced precision. Given successful analog implementations of SVMs [5], we suspected digital low-precision LNS SVMs would be feasible. Such reduced precision permits a diminished word size. In turn, this offers lower power consumption. Furthermore, in CMOS technology, power is consumed when individual bits switch. Conventional multiplication involves extensive computation and bit switching. In LNS, since multiplication is a simple addition, the number of bits and the frequency of their switching are significantly reduced [9].

A disadvantage of LNS is that more hardware is required for addition and subtraction than for multiplication and division. Addition and subtraction in LNS are handled through lookup tables: $s(z) = \log_2(1+2^z)$ and $d(z) = \log_2|1-2^z|$. For systems that tolerate low precision [2,3,9,10], this lookup often requires minimal hardware. Upper case are for reals used in the SVM algorithm and lower case are for their corresponding LNS representations. Thus, let $x = \log_2| X |$ and $y = \log_2| Y |$. LNS uses $X+Y = Y(1+X/Y)$, equivalent to $\log_2(|X| + |Y|) = y + s(x-y)$, and $\log_2|X - Y| = y + d(x-y)$. The function $s(z)$ is used for sums, and $d(z)$ is used for differences, depending on the signs of X and Y.

Neural-network implementations using LNS already exist [2] that exploit properties of $s(z)$ and $d(z)$ to approximate a sigmoid related to the RBF- and sigmoid-SVM kernels. The mathematical nature of kernel-based operations, given the emphasis on multiplication and exponentiation operations, make LNS an attractive technology for SVMs.

V. HARDWARE DESIGN AND SIMULATION

A. *Finite Precision Analysis*

In our previous work [7], we explored the necessary precision requirements for an implementation of SVM classification in LNS. Additional statistical analysis presented here employing Analysis of Variance leads to the conclusion that in general an LNS architecture of three precision bits (leading to a word size of nine bits) has a

statistically similar performance to a traditional floating-point architecture. Thus, for the hardware implementation, it was decided to design an architecture with four bits of LNS precision. Four bits of precision would require an LNS word size of ten bits, which is an attractively small requirement when compared with traditional architectures and existing analog and digital SVMs.

B. Hardware Design

We employed the linear SVM kernel. Experiments in [7] indicated that the linear kernel was often either the best or second-best choice for implementation. The linear kernel is a desirable choice for implementation due to a shortcut possible during classification. During SVM training with a linear kernel, it is possible to store a weight vector representing the weight of each attribute. As a result, linear-kernel classification only requires a dot product of the unknown instance against the weight vector; there is no need to store all the support vectors and their alpha values. Consequently, hardware realization of a linear kernel SVM is simpler than other kernels, making it an attractive choice.

The classifier can be utilized in a variety of situations. It is conceivable that unknown instances may be presented in different manners. For example, within the context of classifying whether a patient has diabetes, a medical lab could have many tests available that are systematically input, or the classifier could be part of a device that is applied to an individual person, testing one instance at a time. A classifier architecture should be capable of being deployed in both environments.

Our design is capable of being utilized in such multiple environments. It can be part of a system systematically receiving instances to classify, perhaps from a ROM. The system would provide the classifier a clock signal, and would input a new instance every set number of cycles. On the other hand, the classifier could be part of a device that tests an instance intermittently. In this case, the classifier would receive a new instance along with a clock signal. Upon completion of the classification, the clock signal would cease to be applied, thereby "deactivating" the classifier until a new instance is ready.

Fig. 1 illustrates our architecture for a linear kernel SVM classifier in LNS. In general when an unknown instance is presented to the classifier, the Register is initialized to *b*, from (1), and the Counter is initialized to zero. The instance to be classified is stored in the unknown-instance buffer, and the support-vector information holds the weights. Thus the Multiply-and-Accumulate (MAC) unit performs as many operations as there are attributes of the instance vectors. Finally, the SVM classification result is the sign of the value stored in the Register.

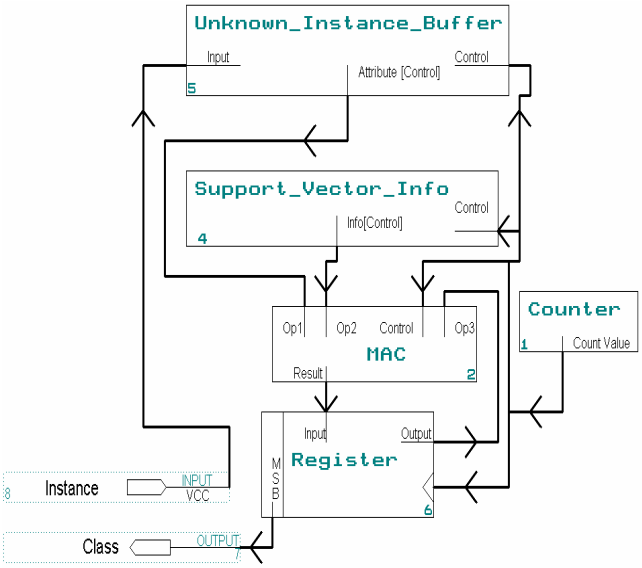

Figure 1. Linear Kernel SVM Classification LNS Architecture

C. Design Simulations and Verification

The design was simulated to ascertain the performance. Our work in [7] utilized certain machine-learning datasets to ascertain the classification accuracy of LNS implementations with varying precisions versus traditional floating-point results. We thus compared the results of our four-bit precision hardware design against the results [7] of floating-point software and four-bit precision LNS simulations. This was done for four machine-learning datasets. Table I summarizes the results.

As Table I indicates, the LNS hardware realization seems to yield results comparable to experimental LNS software results [7]. The differences have been traced to the fact that an exact representation of zero is not available in LNS. Thus results extremely close to or exactly zero may be misclassified. A study of SVM operations, however, indicated that this is a rare circumstance.

A statistical analysis of the results was performed using a two-sample *t*-test. The *t*-test indicated a *p*-value of 0.9155 when comparing the LNS hardware and FP software results, and a *p*-value of 0.9528 when comparing LNS hardware and software results. The *p*-value is also known as the observed significance level. A simplified explanation of the *p*-value that compares two methods is that the closer a *p*-value is to 1.0, the more similar the two results are. Thus, it can be stated with confidence that the differences between the accuracies of the LNS hardware and LNS software implementations are statistically insignificant.

TABLE I. CLASSIFICATION ACCURACY LINEAR KERNEL WITH 4-BIT PRECISION

Dataset	LNS Hardware	LNS Software Simulation	Floating Point Software
Diabetes	75.0	78.5	79.2
Votes	93.8	94.3	94.4
Heart Risk	79.2	81.2	81.2
SONAR	83.6	78.2	74.0

The hardware synthesis of the various LNS SVM classifiers indicates a reduced and efficient hardware complexity. Table II indicates the slices out of the 768 used on a simple Xilinx FPGA, the Spartan3 XC3s50pq208-5 device. In order to evaluate the complexity of the LNS-based operations, Table II also lists the slices used for comparison systems using conventional arithmetic.

TABLE II. SLICES USED ON XILINX FPGA DURING SYNTHESIS

Dataset	10-bit LNS	10-bit FX	20-bit FX
Diabetes	218	181	358
Votes	244	245	490
Heart Risk	246	232	455
SONAR	629	638	1244

D. *Comparision with Fixed-Point Implementation*

In order to evaluate the efficiency of an LNS-based SVM classifier, it is necessary to compare it against a traditional standard. To that extent, 10- and 20-bit Fixed Point (FX) implementations were synthesized, and the resulting number of slices are shown in Table II. The 10-bit FX version classifies less accurately than is acceptable and is shown here for comparison because the LNS design also uses a 10-bit word size. For example, on the diabetes problem, the 10-bit FX has an accuracy of only 34.9. According to [1], a 20-bit FX version offers adequate accuracy, and therefore is a more realistic benchmark to compare the LNS against. It is noteworthy that on most datasets, the fully functional LNS version takes roughly the same number of slices as the inadequate 10-bit FX version. When compared against the more realistic 20-bit FX version, the LNS classifiers are about one-half the size of the FX classifiers. (The 20-bit FX SONAR dataset is too big to fit in the Spartan3 XC3s50pq208-5 device.) Such area savings should translate into equivalent reduction in power consumption.

VI. CONCLUSIONS AND FUTURE WORK

A. *Conclusions*

We have presented a novel digital SVM employing logarithmic arithmetic. The LNS SVM compares favorably with the only other work done in digital SVM hardware [1], and a sample fixed-point implementation. The hardware realization of the LNS SVM Classification module is efficient in terms of hardware complexity and word size, and statistically has the same performance as software simulations. Logarithmic Number Systems represent an extremely attractive technology for realizing digital hardware implementations of SVMs and possibly other machine learning approaches.

B. *Future Work*

This paper has described the first steps towards developing robust, kernel-based hardware machine-learning platforms employing logarithmic arithmetic. These platforms will serve as foundations for low-power machine-learning research, and for porting software solutions to hardware configurations.

Our future goals include exploring precision requirements for hardware LNS-based SVM training. With the singular exception of the (non-LNS) recent work in [1], to the best of our knowledge no research into hardware-based training has been accomplished. Our other goals include employing an increased range of possible kernels, and expanding LNS hardware architectures to other machine-learning algorithms.

ACKNOWLEDGMENTS

The authors would like to acknowledge Jie Ruan and Philip Garcia for their contributions. Co-author William M. Pottenger gratefully acknowledges His Lord and Savior, Yeshua the Messiah (Jesus the Christ).

REFERENCES

[1] Davide Anguita, et al., "A Digital Architecture for Support Vector Machines: Theory, Algorithm, and FPGA Implementation," *IEEE Trans. Neural Networks*, vol. 14, no. 5, pp. 993-1009, Sept. 2003.

[2] Mark Arnold, et al. "On the Cost Effectiveness of Logarithmic Arithmetic for Back Propagation Training on SIMD Processors," *Intl. Conf. Neural Networks*, Houston, TX, pp. 933-936, June 9-12, 1997.

[3] Mark Arnold. "Slide Rules for the 21st Century: Logarithmic Arithmetic as a High-speed, Low-cost, Low-power Alternative to Fixed Point Arithmetic," *Online Seminar Elec. Engr.*, 2001.
`http://www.techonline.com/community/20140`

[4] Gert Cauwenberghs, Ed., *Learning on Silicon: Adaptive VLSI Neural Systems*. Boston: Kluwer Academic Publishers, 1999.

[5] Roman Genov and Gert Cauwenberghs. "Kerneltron: Support Vector Machine in Silicon," *IEEE Trans. Neural Networks*, vol. 14, no. 5, pp. 1426-1434, 2003.

[6] Ralf Herbich. *Learning Kernel Classifiers: Theory and Algorithms.* Cambridge: The MIT Press, 2002.

[7] Faisal M. Khan, Mark G. Arnold and William M. Pottenger. "Finite Precision Analysis of Support Vector Machine Classification in Logarithmic Number Systems," *IEEE Euromicro Symp.Digital System Design (DSD)*, pp. 254-261, Sept. 2004.

[8] Vladimir Vapnik, *The Nature of Statistical Learning Theory*, Springer Verlag, 1995.

[9] V. Paliouras and T. Stouraitis, "Low-power Properties of the Logarithmic Number System," *Proc. 15th Symp. ARITH*, Vail, CO, pp. 229-236, June 2001.

[10] V. Paliouras, "Optimization of LNS Operations for Embedded Signal Processing Applications," *Proc. ISCAS*, Scottsdale, AZ, pp. 744-747, 2002.