

Available online at www.sciencedirect.com

Procedia Computer Science 4 (2011) 508–517

Procedia
Computer Science

International Conference on Computational Science, ICCS 2011

Effects of Reduced Precision on Floating-Point SVM Classification Accuracy

Bernd Lesser, Manfred Mücke, Wilfried N. Gansterer*

University of Vienna, Research Lab Computational Technologies and Applications

Abstract

There is growing interest in performing ever more complex classification tasks on mobile and embedded devices in real-time, which results in the need for efficient implementations of the respective algorithms. Support vector machines (SVMs) represent a powerful class of nonlinear classifiers, and reducing the working precision represents a promising approach to achieving efficient implementations of the SVM classification phase. However, the relationship between SVM classification accuracy and the arithmetic precision used is not yet sufficiently understood. We investigate this relationship in floating-point arithmetic and illustrate that often a large reduction in the working precision of the classification process is possible *without* loss in classification accuracy. Moreover, we investigate the adaptation of bounds on allowable SVM parameter perturbations in order to estimate the lowest possible working precision in floating-point arithmetic. Among the three representative data sets considered in this paper, none requires a precision higher than 15 bit, which is a considerable reduction from the 53 bit used in double precision floating-point arithmetic. Furthermore, we demonstrate analytic bounds on the working precision for SVMs with Gaussian kernel providing good predictions of possible reductions in the working precision without sacrificing classification accuracy.

Keywords: SVM, machine learning, reduced precision floating-point arithmetic, perturbation analysis, quantization effects

1. Introduction

Support Vector Machines (SVMs) [1, 2] are a well established machine learning technique and have enjoyed growing acceptance as a versatile classifier in a diverse range of applications. As embedded and portable computing platforms become more and more powerful, there is an increasing interest in performing complex classification tasks on embedded devices, often in real time. In this context, it is very important to minimize hardware resource usage and power consumption while keeping classification accuracy comparable PC-based implementations typically using double precision floating-point arithmetic. One way to decrease hardware resources and power consumption is to reduce the working precision of arithmetic operations on reconfigurable hardware (FPGAs), which can lead to super-linear performance improvements [3]. Detailed analysis of the effects of reduced working precision on the results is important for being able to make optimal choices for SVM classification implementations.

*Corresponding author

Email addresses: bernd.lesser@univie.ac.at (Bernd Lesser), manfred.muecke@univie.ac.at (Manfred Mücke), wilfried.gansterer@univie.ac.at (Wilfried N. Gansterer)

In this paper, our focus is on the classification phase of SVMs (we do not consider the SVM training phase in this paper). Experiments throughout literature have shown that SVM classification is in general relatively robust against quantization effects (see Section 1.1). Several hardware implementations of SVM classification have used number formats other than double precision floating-point to minimize execution time and/or hardware resource usage. There is, however, still little understanding of the relation between the arithmetic precision used and the SVM classification accuracy achieved. Most authors rely on experiments exploring only a small set of precision levels and/or do not accurately model rounding error in digital implementations.

Our work aims at unifying and furthering the knowledge on how parameter quantisation and rounding error affect SVM classification accuracy and how this knowledge can be used to derive reliable reduced precision floating-point implementations of SVM classification. Specifically, we perform more comprehensive experiments than previously reported by Anguita et al. [4] and investigate the effect of reduced precision floating-point SVM classification over a wide range of precisions.

We compare our results against existing experimental results by Anguita et al. and further investigate the applicability of their theoretical classification error model to our refined settings.

For the set of benchmarks and the Gaussian kernel parameters we consider, the results allow for a reliable derivation of a reduced floating-point working precision which in turn leads to a potential minimization of hardware resource usage while conserving the classification accuracy of a double precision floating-point implementation.

1.1. Related Work

Many approaches in the literature consider reducing the bitwidth of SVM classification parameters, which has lead to additional performance speedup due to less hardware resource usage on a number of different target platforms for different number formats. In most cases, reduced bit width classification accuracy has been investigated in fixed precision arithmetic and compared with the classification accuracy of a corresponding reference implementation (often in IEEE double precision) on various benchmark data sets. It has been illustrated experimentally that in many SVM classification problems a significant reduction of the working precision is possible without a loss of classification accuracy.

Investigating the maximum number of bits that can be saved on a given benchmark data set often requires extensive classification accuracy testing using SVM implementations supporting the adaption of the working precision on a fine-grained level. Implementing such a system (e.g. on FPGAs), can represent tremendous effort. Therefore, most approaches proposing custom SVM hardware implementations test the effects of reducing the working precision on their final implementation on a trial- and error basis and for a very small set of possible working precisions only. Thus most work in literature lack any sort of methodological approach on how many bits could be saved while still keeping a certain classification accuracy.

In [5], the authors propose an integer based SVM compression algorithm as a pre-processing step for input data for both training and classification. Results are being tested on standard hardware (PC). The proposed algorithm shows high bit width savings for the considered benchmark data sets, but there is no methodological investigation of the question how much the working precision can be reduced. SVM bit width reduction experiments for fixed-point arithmetic have been discussed in [6, 7, 8, 9] for FPGA based platforms, while [10] targets SVM classification on PIC-microprocessors.

Anguita et al. published a series of papers on the development and efficient implementation of SVM classifiers for embedded hardware using fixed-point arithmetic [11, 12, 13, 14, 15]. Beside proposing optimizations for the SVM training phase, they repeatedly performed bit width reduction experiments similar to the previously mentioned approaches, and compared floating- and fixed-point SVM results experimentally.

To the best of our knowledge, [4] and [16] are the only two publications which methodologically investigate how to *predict* the required bit width in order to avoid a loss of classification accuracy. In [16], statistical methods are used for investigating the effect of quantization (i. e., the conversion/rounding to fixed-point format). The authors estimate the effect of SVM parameter quantization on the classification accuracy by computing the expectation and the variance of the corresponding output noise (i. e., the classification error), which can be represented as a function that depends (among others) on the number of bits used for representing the SVM parameters. This function, which was originally derived to investigate quantization effects, is then used to derive estimates of the number of bits needed to achieve the required classification performance. In [4], a worst case analysis of SVM parameter perturbation/quantization is

performed. The quantization effects are propagated from the input to the output of the SVM classification phase, and bounds for allowable fixed-point perturbations (not increasing the classification error) are derived using interval arithmetic. Although numerical experiments summarized in [4] indicate that the derived bounds may be too pessimistic for many real world problems, such guaranteed worst case bounds are better suited for the development of self-adaptive algorithms. The bounds derived in [4] are intended to predict the required bit width in fixed-point arithmetic. In this paper, we investigate the adaptation of these bounds to floating-point arithmetic and we also consider more general types of perturbation of the data, in particular, a permanent reduction of the working precision for all operations. (In Section 3.1 we review the results of [4] in more detail.)

There is only very little work investigating the effects of reduced working precision on SVM classification in *floating-point* arithmetic. In [17, 18] no significant loss in classification accuracy compared to the IEEE double precision floating-point classification has been reported when experimentally reducing the floating-point mantissa length to $\approx 30\%$ and 50% respectively, but also here this is done by experiments only.

Similar results have been reported for SVM classification performed in a logarithmic number system [19, 20].

1.2. Contributions

In this work, we systematically investigate the effects of using reduced precision on a floating-point SVM classification process for a large range of precisions ($[53, 52, \dots, 4]$, where $p = 53$ corresponds to IEEE double precision), and for three well-known publicly available benchmark data sets. We further compare our results with results obtained by Anguita et al. [4], but we also consider more general types of perturbations beyond one-time random element wise additive perturbations (reduction of the working precision for the entire SVM classification phase). Finally, we investigate the adaptation and applicability of the worst case bound derived by Anguita et al. [4] to floating-point SVM classification and to a reduction of the working precision.

To the best of our knowledge, this is the first work investigating the impact of floating-point rounding error on SVM classification accuracy. We show over a range of benchmarks, that the bound suggested by Anguita et al. [4] can indeed be used to derive floating-point number formats to implement SVM classification, although it is not considering rounding error.

The rest of this paper is organized as follows: Section 2 reviews the basics of SVM classifiers. Section 3 discusses effects of SVM parameter quantisation and rounding error on achieved classification accuracy, including a summary of the only existing theoretical approach for predicting the effects of parameter quantization from [4]. Section 4 describes our experimental setup in detail and discusses our experimental results. Section 5 finally concludes this paper.

2. Background and Problem Setting

In the following we shortly review the basics of SVMs, detail our application scenario and derive the respective requirements to design and analysis of SVM classification. We denote matrices using capital letters, vectors using lower case letters and scalars using greek letters.

2.1. Support Vector Machines

A Support Vector Machine [1, 2] constructs a decision boundary as a linear discriminant function (also called a hyperplane for dimensions larger than three) in vector space to separate given data-points into two classes.

Starting from a set of labeled data points in a given feature space, SVM training constructs (learns) a decision boundary between the two classes. The solution to this can be considered optimal in the sense that SVM training finds the boundary which globally maximizes the distance between the boundary and the two classes. The data-driven design of the SVM and the involved quadratic optimization problem result in high computational requirements for the SVM training phase.

After the training phase, the decision boundary is represented by a subset of the entire training set, called the *support vectors*. The support vectors together with a weight vector w and a bias term b form the *SVM classification parameters*. Use of kernel-functions (i.e. a transformation function to map the input data from its original vector space into a higher dimensional target space) allows for extending SVMs to non-linear classification problems in an efficient and elegant way by omitting the need to explicitly computing the transformation.

For a wide range of applications retraining is not required. In this case the training phase can be carried out off-line and only the (static) classification parameters of the SVM fully define the classifier. In this paper, we exclusively focus on the SVM classification phase and consider a standard SVM classification function $\hat{\Phi}(z)$ of the form

$$\hat{\Phi}(z) = \text{sign} \left(\sum_{i=1}^N w_i K(X_i, z) + \beta \right), \quad (1)$$

where $N \in \mathbb{N}$ denotes the number of support vectors, w is the weight vector ($w_i = \alpha_i y_i \in \mathbb{R}$ with α_i is the i^{th} Lagrange multiplier resulting from the SVM training phase and $y_i \in \{-1, +1\}$ denotes the class membership of the i^{th} support vector), $K(X_i, z)$ is the kernel function, X is the support vector matrix with X_i the i^{th} support vector, z is the data instance to be classified and $\beta \in \mathbb{R}$ is the scalar bias. $\hat{\Phi}(z)$ gives a data instance's class membership.

In this paper, we focus on the *Gaussian* (or *radial bias function (RBF)*) kernel function K :

$$K(X_i, z) = e^{\left(\frac{-\|X_i - z\|^2}{2\sigma^2} \right)}$$

where the factor σ^2 represents the degree of the kernel. This results in the following SVM classification function considered in this work:

$$\hat{\Phi}(z) = \text{sign} \left(\sum_{i=1}^N w_i e^{\left(\frac{-\|X_i - z\|^2}{2\sigma^2} \right)} + \beta \right). \quad (2)$$

2.2. Implementation in Floating-Point Arithmetic

The implementation of Equation (2) on a computer needs to approximate operations in \mathbb{R} using some limited precision number format. Standard PCs provide IEEE double precision floating-point arithmetic [21] which provides a precision of 53 bits. Real-time or power-constrained applications often resort to computing platforms like embedded CPUs, digital signal processors or FPGAs allowing for a more custom-tailored implementation achieving higher performance and/or lower power consumption. Using smaller number formats usually results in smaller memory footprints, higher performance [3] and lower power consumption on hardware platform which support and exploit them. Especially on such platforms one of the key tasks of every efficient implementation is to identify the smallest acceptable number format.

Our work does not target a specific hardware platform but aims at identifying the lowest working precision acceptable to achieve a classification accuracy comparable to a reference implementation in double precision floating-point arithmetic. The specific number format chosen for an implementation should then be picked considering both the minimum acceptable precision and the cost function of the specific hardware architecture for implementing arithmetic functions.

In this paper, we consider evaluating Equation (2) in floating-point arithmetic. The reference implementation is assumed to use IEEE double precision floating-point arithmetic, resulting in a baseline working precision of 53 bit. We consider every discrete value of the precision p in the range $[4, 5, \dots, 53]$ to be a viable choice for the working precision.

3. SVM Classification Error Analysis

There are two effects causing the classification error of an implementation of SVM classification in digital hardware to deviate from a reference implementation's classification error: Parameter quantization and rounding error. Parameter quantization effectively alters the classifier (which remains unchanged for all classifications) while rounding adds a data-dependent noise component. The effect of parameter quantization on the hyperplane can be quantified exactly after learning while the effect of rounding is data dependent and can therefore only be quantified exactly when the data point to be classified is known.

In the following, we discuss the impact of parameter quantization and rounding on the SVM classification error.

3.1. Parameter Quantisation

To the best of our knowledge, the work by Anguita et al. [4, 16] is the only one providing a theoretical analysis of SVM parameter quantization to date. In [4], the authors use interval arithmetic to evaluate the effect of SVM parameter quantization on the classification error. The SVM classification parameters w , k (vector of kernel function results k , where $k_i = K(X_i, z)$) and β (cf. Equation (1)) are replaced by intervals: $\alpha \rightarrow [\alpha - \Delta_\alpha, \alpha + \Delta_\alpha]$, $k \rightarrow [k - \Delta_k, k + \Delta_k]$, $\beta \rightarrow [\beta - \Delta_\beta, \beta + \Delta_\beta]$, respectively, yielding:

$$[\hat{\Phi}(z) - \Delta_\Phi, \hat{\Phi}(z) + \Delta_\Phi] = \sum_{i=1}^m y_i [\alpha_i - \Delta_\alpha, \alpha_i + \Delta_\alpha] [k_i - \Delta_k, k_i + \Delta_k] + [\beta - \Delta_\beta, \beta + \Delta_\beta] \quad (3)$$

Assuming a Gaussian kernel, and by applying standard methods of interval arithmetic, Equation (3) can be simplified to:

$$[-\Delta_\Phi, +\Delta_\Phi] \subseteq (\mu^+ - \mu^-) \Delta_\alpha \Delta_k + (\mu(\Gamma \Delta_k + \Delta_\alpha) + \Delta_\beta) [-1, +1] \quad (4)$$

where μ^+ and μ^- denote the number of support vectors of the positive- and of the negative class, respectively. Γ denotes a SVM learning parameter, which forms an upper bound for the α_i values ($\alpha_i \leq \Gamma$) and is a free parameter in the SVM training phase. Relation (4) describes the size of the interval, in which the deviation of the reference classification function $\hat{\Phi}(z)$ must lie depending on these factors. For $\mu = \mu^+ + \mu^-$, assuming $\mu^+ \approx \mu^-$ and setting $\Delta = \Delta_\alpha = \Delta_k = \Delta_\beta$ (i.e. all SVM parameters are disturbed by perturbations of comparable magnitude) this relation can be simplified to the following worst-case bound:

$$\Delta \leq \frac{1}{\mu(\Gamma + 1) + 1} \quad \mu \in \mathbb{N}; \Delta, \Gamma \in \mathbb{R} \quad (5)$$

Equation (5) states that the maximum size of the SVM parameter quantization Δ , must be smaller or equal than a value depending on the total number of support vectors μ and the SVM learning parameter Γ to guarantee no additional deviation from SVM reference classification error due to parameter quantization.

The authors of [4] verify the derived bound in Equation (5) experimentally on the *SONAR* data set. They train the SVM once on the trainings data set and then add 10^6 times an element wise, uniformly distributed random perturbation $[-\Delta.. +\Delta]$ to the SVM parameters (w , k and β) for testing the deviation from the reference classification error on the test data set (see Section 4.1).

This experiment is repeated for each chosen value of Δ and average-, minimum- and maximum deviations from the reference classification errors are reported (see [4], Figure 7). While not explicitly stated, all experiments are most likely performed on standard PC hardware using IEEE double precision arithmetic.

Their results show that the derived bound holds and is conservative, i.e. SVM classification shows to be much more robust (for the chosen dataset) against parameter quantization than the bound suggests (by about a factor of 2^3).

From Perturbation Analysis to Number Format. The work by Anguita et al. [4] may give the impression that the acceptable parameter quantization Δ 's logarithm of base two ($\log_2(\Delta)$) gives the number of bits required to implement the SVM classification. This neglects rounding effects in the SVM classification process and (depending on the number format chosen) over- and underflow effects. To compensate for over- and underflow, range analysis of SVM classification would be required and a suitable number of bits would be needed to represent the integer values (in case of fixed-point) or exponent (in case of floating-point). The perturbation analysis gives no information on the additional classification error caused by a finite precision's arithmetic rounding error. Our experiments (described in section 4) illustrate the effects of additional rounding error.

3.2. Evaluating Classification Accuracy

There are various approaches for evaluating classification accuracy and classification error.

Leave-one-out cross-validation temporarily removes one data point from the data set's n data points. The SVM is trained on the remaining $n - 1$ data points and the removed data point is classified using the achieved parameters. This procedure is repeated for all n data points in the data set. The classification accuracy P is evaluated as the number of

correctly classified data points i_{pass} relative to the total number of data points n ($P = i_{pass}/n$). The classification error E corresponds to the number of incorrectly classified data points i_{fail} over the total number of data points ($E = i_{fail}/n$).

Anguita et al. [4] have tested their SVM classification accuracy by a-priori splitting the data set into two disjoint sets, one used for training the SVM and one used for testing the resulting SVM classifier's accuracy. The classification accuracy is then evaluated as the average number of correctly classified data points over all trials. The classification error is defined as the average percentage of incorrectly classified data points over all trials. In contrast to the approach pursued by Anguita et al., the classification accuracy achieved with the leave-one-out method depends less on a specific splitting of the data. We therefore consistently use leave-one-out cross-validation in the following.

To better demonstrate the effect of limited precision with respect to a reference implementation, we normalise the classification error by the reference implementation's classification error. This yields the *deviation from SVM classification error*. The classification error in some experiments can actually decrease compared to the reference implementation, resulting in a negative deviation from SVM classification error.

3.3. Reducing the Working Precision

Performing SVM classification in finite precision arithmetic results in parameter quantization as well as rounding error accumulating over the sequence of arithmetic operations performed. Equation (3) incorporates parameter quantization only. Closed forms describing the effect of rounding error on the classification accuracy do not exist yet. In this work we explore experimentally the potential effect of rounding on classification accuracy. While rounding error of dot-products, which form the core of SVM classification, has been widely investigated, we are not aware of any work applying rounding error analysis to SVM classification.

The two dominant number formats chosen when implementing signal processing in digital logic are fixed-point and floating-point. The two error models differ fundamentally: While use of fixed-point results a fixed absolute error, use of floating-point results in a fixed relative error. While hardware implementations of fixed-point arithmetic are less complex than their respective floating-point counterparts, algorithm design and verification is more complex for the former as frequent rescaling is necessary but not made explicit in the stored numbers. As digital logic integration continues at an exponential rate, the cost difference importance between fixed- and floating-point arithmetic decreases. We therefore chose floating-point as our target number format.

Floating-point arithmetic represents numbers using a sign bit s , exponent e and mantissa f ($f \in [1..2)$ for normalized representation). The number of bits used to represent the mantissa f is called the precision p ($p=53$ for IEEE double precision). The real value of a binary floating-point number can be computed as $-1^s \cdot 2^e \cdot f$. The rounding error e at any floating-point calculation is in the range of $[-2^{e-p}..2^{e-p}]$ when using round to nearest. See [21, 22] for full details on floating-point numbers.

We simulate the effect of reduced precision floating-point arithmetic by use of the MPFR library allowing for arbitrary settings of precisions (i.e. p is a free parameter).

4. Experiments

In the following, we investigate experimentally effects of parameter quantisation and rounding error on classification accuracy by performing the following experiments:

- Additive random elementwise perturbation
Different SVM parameters are perturbed once drawing a noise component from $[-\Delta \dots + \Delta]$, but all subsequent arithmetic operations are performed using IEEE double precision arithmetic, thus arithmetic operations imply no additional rounding error. To sample the huge space of possible perturbation combinations, we repeat this procedure 10^6 times. Recording the resulting classification accuracy over different maximum perturbations Δ gives a rather complete image of influence parameter perturbations can have on classification accuracy. This is the type of perturbation which has been considered in [4]. We perform this experiment using both 2-fold-2 (E1, for comparison with [4]) and leave-one-out (E2 to achieve reproducible results) cross validation.
- Reduction of floating-point precision
SVM classification is performed using a floating-point number format with the exponent identical to double precision (11 bit) while the precision p is smaller than the one used for double precision (E3). Compared

to SVM classification using IEEE double precision floating-point arithmetic, this leads to additional rounding errors in each floating-point operation during the SVM classification process.

4.1. Setup

We performed our experiments using the following standard benchmark data sets: *SONAR*, *IRIS* (classes 2 and 3, which are not linearly separable) and *MUSK*. All data sets are publicly available [23]. *SONAR* represents a difficult classification problem challenging SVMs due to its high dimensionality and tight margin and consists of 208 data-points with 60 features each. *SONAR* was also chosen to allow for comparison of results by Anguita et al. [4]. The *SONAR* benchmark was complemented by *IRIS* and *MUSK*, having considerably less and more features, respectively. While the *IRIS* benchmark consists of only 100 data-points with four features each, the *MUSK* benchmark provides 476 data-points described by 168 features each. While *SONAR* is already normalized, we also normalized *IRIS* and *MUSK* by dividing each feature value by the maximum value contained in the data set.

All experiments described were performed and evaluated using Matlab. To perform SVM training and classification in double precision floating-point arithmetic, we used the freely available *SVM and Kernel Methods Matlab Toolbox* (SVM-KM) [24]. For computing classification results in reduced precision floating-point, we implemented the Gaussian SVM classification function (2) using the *GNU MPFR Library* [25] and made it accessible from Matlab via mex.

For SVM training, we use the same settings as Anguita et al. [4] ($\sigma^2 = 1$, $\Gamma = 10$). Computing the reference classification accuracy in double precision, we achieved about 86% (*SONAR*), 95% (*IRIS*) and 94% (*MUSK*).

Anguita et al. [4] present the classification accuracy over $\log_2(\Delta)$ while the natural parameter for experiments using MPFR is the precision p . As Δ represents the maximum quantization error when converting to a given number format, the equation $\Delta = 2^{e-p}$ holds. As the data sets used are all normalized, the largest exponent used for SVM parameters having a non-zero mantissa f is -1 (i.e. $e \leq 0$ which gives $\Delta = 2^{-p-1}$). Based on this we can directly compare parameter quantisation- and round-off experiments, and can further map Anguita et al.'s worst-case bound to this experimental data.

For each experiment, the reference classification error is computed using double precision arithmetic. The classification error gives the relative deviation of an experiment relative to the reference classification accuracy. A classification error of zero therefore corresponds to a result identical to computation in double precision.

E1 - Data Perturbation (2-fold cross validation) The data set is split in two disjoint sets at random and a reference classification error is computed using double precision. Using the resulting number of support-vectors the error prediction bound (5) is computed. We then add a random noise component drawn from an equal distribution $[+\Delta, -\Delta]$ to each parameter of the SVM and record the respective SVM classification error using 2-fold cross validation. This is repeated 10^6 times to sample sufficiently many combinations of noise components. This procedure is repeated for all integer values in the range $\log_2(\Delta) = -12.. -1$.

E2 - Data Perturbation (leave-one-out cross validation) Using leave-one-out cross validation the SVM is trained n times. Classification of n data points is performed in double-precision yielding the reference classification error. Since leave-one-out cross validation can result in a different number of support vectors for each data point tested, we average μ over the full leave-one-out cycle of the reference classification for computing the error prediction bound 5. Then, a random noise component drawn from an equal distribution $[+\Delta, -\Delta]$ is added to each parameter of the SVM in each classification performed and the respective classification error is recorded. This is repeated 10^6 times to sample sufficiently many combinations of noise components. This procedure is repeated for all integer values in the range $\log_2(\Delta) = -12.. -1$.

E3 - Reduced Floating-Point Precision (leave-one-out cross validation) The reference classification error is computed as in E2. Then the classification error is computed using the Gaussian classifier implemented in MPFR with $p = 53$ (double precision) using leave-one-out cross validation to verify proper working of our setup. Subsequently, we calculate the classification error for each precision p in the range $p = 4 \dots 54$ bits.

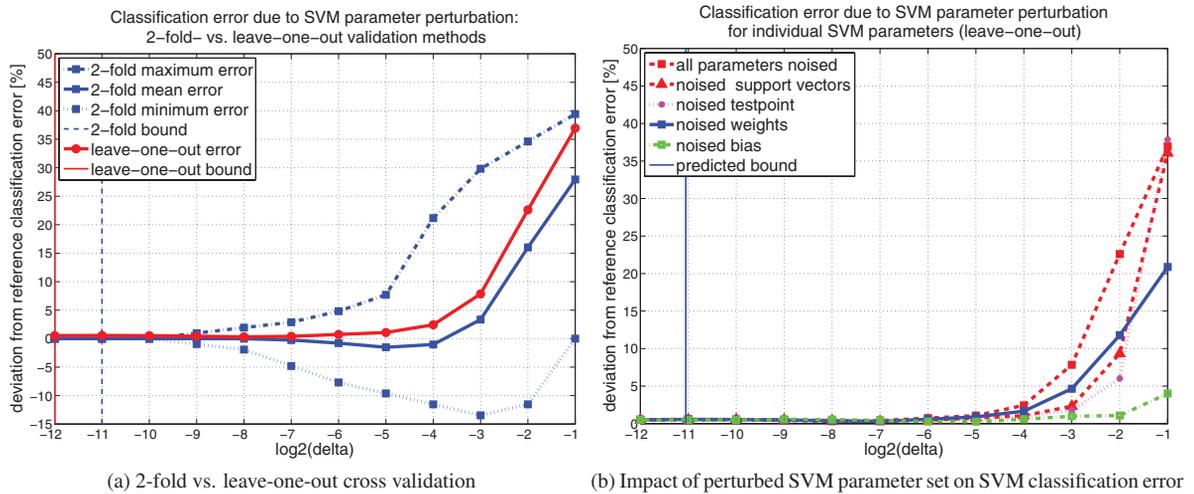


Figure 1: Deviation from SVM reference classification error due to parameter perturbation for SONAR

4.2. SVM Classification Parameter Perturbation

Figure 1 shows the deviation from the reference classification error over maximum noise added to the SVM parameters for the SONAR data set. Figure 1a compares leave-one-out- and 2-fold validation. For the 2-fold validation we also plot the mean-, minimum- and maximum deviation from reference classification errors over 10^6 trials for each $\log_2(\Delta)$. The deviated classification error shows similar characteristics for the the leave-one-out- and the averaged 2-fold method, which starts deviating at about $\Delta = 2^{-6}$. The bounds for both methods are -12 and -11, respectively. They differ by about 1 bit as the leave-one-out method results in a larger number of support vectors than 2-fold. Figure 1b shows the deviation of the reference classification error over $\log_2(\Delta)$ when perturbing selected parameters of the SVM only.

4.3. Reduced Working Precision SVM Classification

Figure 2 shows the deviation of the SVM reference classification error over working precision p for the SONAR, the IRIS and the MUSK data set when performing classification in reduced precision using MPFR.

For all data sets, a deviation from the reference implementation only becomes evident once the precision is lower than 15 bits. For a wide range of precisions ($[53, 52, \dots, 15]$), the result is identical to the reference implementation. This is in agreement with literature showing that very moderate-sized number formats can accurately implement SVM classification. The bounds are much tighter for reduced precision experiments, but are still acceptable.

Figure 3 shows the deviation from the SVM reference classification error over working precision p for the SONAR, the IRIS and the MUSK data set when performing classification in reduced precision as well as the deviation when perturbing SVM parameters with a value corresponding to the maximum quantisation error at the respective precision ($\Delta = 2^{-p-1}$). This plot allows to quantify the additional deviation in classification error caused by rounding error compared to parameter quantisation only.

4.4. Discussion

Perturbation analysis indicate that the derived bound overestimates the effect of parameter quantisation by five to seven bits. Performing SVM classification in reduced precision floating-point arithmetic shows a much smaller distance between calculated bound and experimental data. For the IRIS data set, an error deviation up to about 3% on the left side of the predicted bound can be observed. Comparing the effects of parameter quantisation with and without rounding error, we can identify the additional effect of rounding. Overall, although the bound as derived in [4] does not take rounding error into consideration, it can – when ignoring error derivations of insignificant magnitude – give an acceptable estimate of the required working precisions for SVM classification.

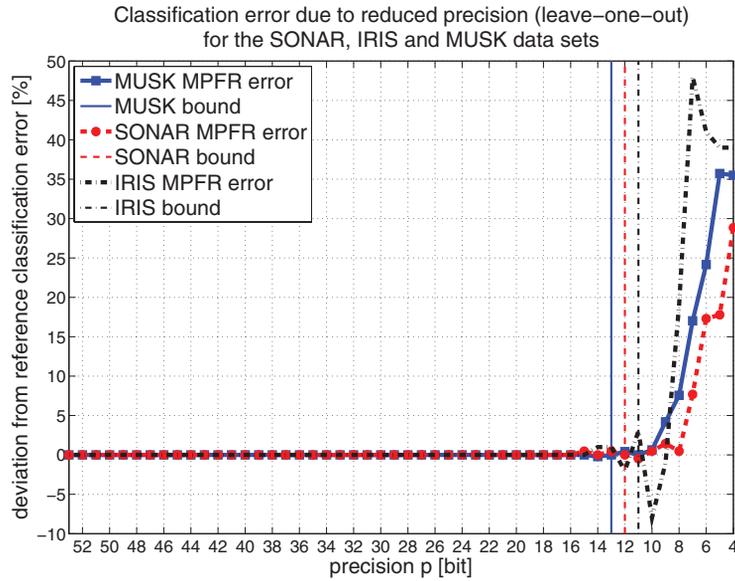


Figure 2: Deviation from the SVM reference classification error due to reduced working precision

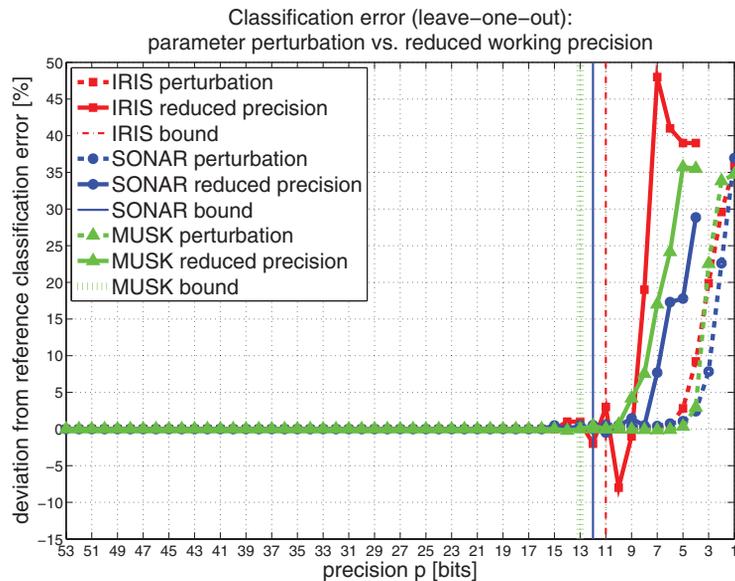


Figure 3: Deviation from the SVM reference classification error: perturbation vs. reduced working precision

5. Conclusions

We investigated the effects of parameter quantisation and of reduced working precision on the accuracy of floating-point SVM classification for a wide range of working precisions ([53, 52, ..., 4]). Experimental results for three well-known publicly available benchmark data sets showed that quite large reductions in the working precision of the SVM classification are possible before a loss in classification accuracy can be observed.

Moreover, our results indicate that there is big potential for adapting the perturbation bounds derived by Anguita et al. [4] to a reduction of the working precision in floating-point arithmetic. Already a simple adaptation which we propose in this paper is quite successful in indicating the limiting precision level below which a loss of classification

accuracy can be expected. Once a proven reliable bound for the maximum allowable perturbation for SVM classification in floating-point arithmetic is developed based on our first results, fully adaptive SVM classification algorithms can be designed which automatically select the lowest possible working precision level and thus achieve the highest possible performance on architectures where reducing the working precision leads to performance improvements (such as FPGAs).

Thus, our future work will focus on the theoretical derivation of bounds for allowable perturbations in floating-point arithmetic, on the generalization of our results to other frequently used SVM kernel functions (like the polynomial kernel), and on the investigation of the resulting runtime performance improvements.

Acknowledgements. This work was supported by project 819469 (MixSVM) of the Austrian Research Promotion Agency (FFG) and by project S10608 (NFN SISE) of the Austrian Science Fund FWF.

References

- [1] V. N. Vapnik, *The nature of statistical learning theory*, Springer-Verlag New York, 1995.
- [2] C. Cortes, V. Vapnik, Support-vector networks, *Machine Learning* 20 (1995) 273–297.
- [3] M. Muecke, B. Lesser, W. N. Gansterer, Peak Performance Model for a Custom Precision Floating-Point Dot Product on FPGAs, *Third Workshop on UnConventional High Performance Computing 2010 (UCHPC 2010)* (to appear in *Lecture Notes in Computer Science*, Springer-Verlag).
- [4] D. Anguita, A. Boni, S. Ridella, A digital architecture for support vector machines: theory, algorithm, and FPGA implementation, *Neural Networks, IEEE Transactions on* 14 (5) (2003) 993–1009.
- [5] T. Luo, L. Hall, D. Goldgof, A. Remsen, Bit reduction support vector machine, in: *Data Mining, Fifth IEEE International Conference on*, 2005, p. 4 pp.
- [6] J. W. Wee, C. H. Lee, Concurrent support vector machine processor for disease diagnosis, in: *ICONIP, 2004*, pp. 1129–1134.
- [7] I. Biasi, A. Boni, A. Zorat, A reconfigurable parallel architecture for svm classification, in: *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, Vol. 5, 2005, pp. 2867–2872 vol. 5.
- [8] S. Cadambi, I. Durdanovic, V. Jakkula, M. Sankaradass, E. Cosatto, S. Chakradhar, H. Graf, A massively parallel FPGA-based coprocessor for support vector machines, in: *Field Programmable Custom Computing Machines, 2009. FCCM '09. 17th IEEE Symposium on*, 2009, pp. 115–122.
- [9] R. Pedersen, M. Schoeberl, An embedded support vector machine, in: *Proceedings of the Fourth Workshop on Intelligent Solutions in Embedded Systems (WISES 2006)*, 2006, pp. 79–89.
- [10] A. Boni, L. Gasparini, R. Pianegiani, D. Petri, Low-power and low-cost implementation of svms for smart sensors, in: *Instrumentation and Measurement Technology Conference, 2005. IMTC 2005. Proceedings of the IEEE*, Vol. 1, 2005, pp. 603–607.
- [11] D. Anguita, A. Boni, S. Ridella, Learning algorithm for nonlinear support vector machines suited for digital vlsi, *Electronics Letters* 35 (16) (1999) 1349–1350.
- [12] D. Anguita, A. Boni, A. Zorat, Mapping lssvm on digital hardware, in: *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, Vol. 1, 2004, pp. –571.
- [13] D. Anguita, A. Ghio, S. Pischitta, A learning machine for resource-limited adaptive hardware, in: *Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference on*, 2007, pp. 571–576.
- [14] D. Anguita, A. Ghio, S. Pischitta, S. Ridella, A hardware-friendly support vector machine for embedded automotive applications, in: *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, 2007, pp. 1360–1364.
- [15] D. Anguita, A. Ghio, S. Pischitta, S. Ridella, A support vector machine with integer parameters, *Neurocomputing* 72 (1-3) (2008) 480–489, *machine Learning for Signal Processing (MLSP 2006) / Life System Modelling, Simulation, and Bio-inspired Computing (LSMS 2007)*.
- [16] D. Anguita, G. Bozza, The effect of quantization on support vector machines with gaussian kernel, in: *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, Vol. 2, 2005, pp. 681–684 vol. 2.
- [17] J. Manikandan, B. Venkataramani, V. Avanthi, FPGA implementation of support vector machine based isolated digit recognition system, in: *VLSI Design, 2009 22nd International Conference on*, 2009, pp. 347–352.
- [18] B. Catanzaro, N. Sundaram, K. Keutzer, Fast support vector machine training and classification on graphics processors, in: *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, Helsinki, Finland, 2008, pp. 104–111.
- [19] F. Khan, M. Arnold, W. Pottenger, Finite precision analysis of support vector machine classification in logarithmic number systems, in: *Digital System Design, 2004. DSD 2004. Euromicro Symposium on*, 2004, pp. 254–261.
- [20] K. Irick, M. DeBole, V. Narayanan, A. Gayasen, A hardware efficient support vector machine architecture for FPGA, in: *Field-Programmable Custom Computing Machines, 2008. FCCM '08. 16th International Symposium on*, 2008, pp. 304–305.
- [21] IEEE, IEEE standard for floating-point arithmetic, Tech. rep. (2008).
- [22] J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé, S. Torres, *Handbook of Floating-Point Arithmetic*, Birkhäuser Boston, 2010.
- [23] UCI Machine Learning Repository, <http://archive.ics.uci.edu/ml/>.
- [24] SVM and Kernel Methods Matlab Toolbox, <http://asi.insa-rouen.fr/enseignants/arakotom/toolbox/index.html>.
- [25] The GNU MPFR Library, <http://www.mpfr.org/>.