# The Effects of Quantization On Support Vector Machines Using Polynomial Kernel

Ihab Nahlus

***Abstract*—In this paper, we apply a probabilistic method to predict the effect of quantization in a digital implementation of a Support Vector Machine (SVM). the quantization effects taken into consideration are both, input data and calculations done inside the processor. We derived a closed-form expression for these effects for an SVM using a $2^{nd}$ order polynomial Kernel and matched it with simulations.**

## I. Introduction

Although the subject of Support Vector Machines (SVMs) have started in the late seventies by Vapnik [1], the attention only sparkled after Vapnik's 1995 paper [2]. SVMs were used for classification mainly [3] [4] and later extended to regression [5]. In all of these cases, SVM was reported to have a better performance than all of the competing methods. There are recent hardware implementations of SVMs [6] [7] [8] that target a low-power design for the hope of development of embedded systems - general purpose processors are highly ineffective for embedding.

One might consider an analog implementation of a SVM, espcially that they are highly energy-efficient for low accuracies ($< 8$ bits) [9]. But the lack of a reliable analog storage device to store the required parameters makes the implementation currently impossible. A viable solution, however, would be the use of a mixed-signal implementation that combines the advantages of both digital and analog approaches. Low-Pass(LP) analog computing is a newly proposed method which is based on weighted charge transfer between analog inputs and output, set by the duty cycle of the digital select signal. A comparison of LP analog computing and digital computing has already been done and results are shown in Figure 1. We can see that LP analog is more energy efficient for accuracies less than 8 bits.

To choose which approach is more energy-efficient, we need to know how many bits the SVM would require. If this number is less than 8, one should go with the LP analog implementation and otherwise, just stick to the digital implementation. Hence, The objective is pretty clear ; predict how many bits are needed for the implementation of a SVM. In the past, quantization effects have been studied for a hardware implementation of a Multi-Layer Perceptron (MLP) and other neural networks. There are mainly two approaches to go about solving this problem. The first one takes a worst-case analysis by propagating the quantization effect from input to output and bounding its value [10] . The second approach assumes the quantization effect to be random with a specific Probability Distribution Function (PDF). This approach is very common in most Digital Signal Processing systems and the main assumption is that the PDF of the noise corresponds to a Uniform distribution [11] [12] . For SVMs, The first technique has already been done in [6] and the second technique has been done in [13]. However in [13], the author ignored quantization effects coming from calculations inside the processor and only considered the SVM with a Gaussian Kernel. In this paper, we will extend that analysis in [13] to SVM with polynomial Kernels and will analyze all quantization effects, namely coming from input and from fixed-point calculations.

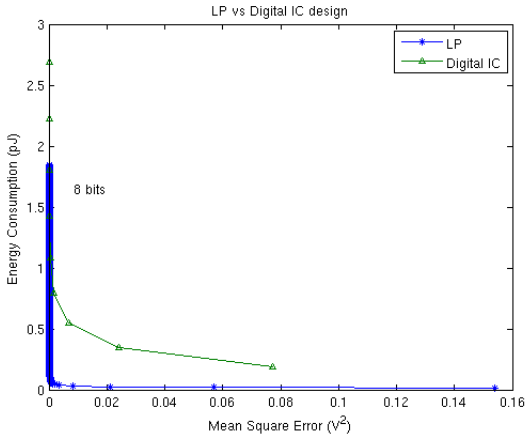


Figure 1. Comparison of LP analog computing with Digital computing

The following section reviews some Support Vector Machine basics and introduces some notation. In section III, quantization basics will be introduced. Section IV details the main result of this paper and we report the experimental results in Section V.

## II. Support Vector Machine

We have a setting where $\{\mathbf{X}\}_{i=1}^{n}$ are the inputs to the system s.t. $X_i \in \mathbb{R}^D$. Accordingly, we have $\{t_i\}_{i=1}^{n}$ which are the target classes s.t. $t_i \in \{-1, 1\}$. This is known as the binary classification problem.

The goal of the Support Vector Machine is to find the hyperplane separating the 2 classes. The boundary will have an equation $\langle \mathbf{W^T}, \Phi(\mathbf{X_i})\rangle + b = 0$ where $\mathbf{W}$ and $b$ are parameters we would like to determine and $\Phi(X)$ is some mapping , usually taking $\mathbf{X_i}$ to a higher dimensional space(more on it later).

Our decision function $y(\mathbf{X})$ is $sign(\langle \mathbf{W^T}, \Phi(\mathbf{X_i}\rangle + b)$. Let $\mathbf{X_1}$ and $\mathbf{X_2}$ be such that:

$$\begin{aligned} \langle \mathbf{W^T}, \Phi(\mathbf{X_1})\rangle + b &= -1 \\ \langle \mathbf{W^T}, \Phi(\mathbf{X_2})\rangle + b &= +1 \end{aligned}$$

where the data has been re-scaled such that no points lie between -1 and 1, and hence $\mathbf{X_1}$ and $\mathbf{X_2}$ represent the

boundaries of the 2 classes. We will see that these vectors are very important in SVMs, earning them a special name in the literature, Support Vectors.

One can show that the distance between the 2 boundaries is $\frac{2}{\|\mathbf{W}\|}$. Intuitively, we would like to maximize this quantity, or equivalently minimizing the inverse $\frac{\|\mathbf{W}\|}{2}$, which is in turn equivalent to minimizing $\frac{\|\mathbf{W}\|^2}{2}$. This leads to the following Quadratic Programming (QP) problem:

$$\begin{aligned} min_{w,b} \quad & \frac{1}{2}\|\mathbf{W}\|^2 \\ subject\,to \quad & t_i(\langle \mathbf{W^T}, \Phi(\mathbf{X_i})\rangle + b) \geq 1 \end{aligned}$$

This all assumes that the data was perfectly separable. A simple extension to this problem when the data are not perfectly separable is the soft-margin extension where we introduce slack variables $\epsilon_i$ for each $\mathbf{X_i}$. Our QP problem becomes:

$$\begin{aligned} min_{\mathbf{W},b,\epsilon} \quad & \frac{1}{2}\|\mathbf{W}\|^2 + C\sum_{i=1}^{n}\epsilon_i \\ subject\,to \quad & t_i(\langle \mathbf{W^T}, \Phi(\mathbf{X_i})\rangle + b) \geq 1 - \epsilon_i \\ & \epsilon_i \geq 0 \end{aligned} \tag{1}$$

where C serves as a limiting quantity to the penalty factors for misclassified data.

One can show that the solution to the problem in (1) is :

$$y(\mathbf{X}) = \sum_{i=1}^{n} \alpha_i t_i \langle \Phi(\mathbf{X}), \Phi(\mathbf{X_i})\rangle + b \tag{2}$$

where $\{\alpha\}_{i=1}^n$ are the lagrange multipliers of the Lagrangian solution to problem (1). We note that $\{\alpha\}_{i=1}^n$ are 0 for all vectors except the ones on the boundary, what we previously called the Support vectors. Hence, the complexity of this algorithm depends largely on the number of Support vectors, which we denote by L.

We notice that the solution in (2) only depends on the inner product of the $\Phi$ function and hence is attributed a special name in practice, Kernel function. By this definition, we can rewrite equation (2) as

$$y(\mathbf{X}) = \sum_{i=1}^{n} \alpha_i t_i K(\mathbf{X}, \mathbf{X_i}) + b \tag{3}$$

The Kernel function is usually chosen to operate on the lower dimension vectors $\mathbf{X}$ and $\mathbf{X_i}$ to produce a value equivalent to the dot-product of the higher-dimensional vectors. A well known family of Kernels is the Mercer Kernel family, where the kernel must be continuous,symmetric and positive-semi definite. Albeit having these requirements, we list some commons Kernels which have already been proved to be Mercer Kernels,

- $K(\mathbf{X}, \mathbf{X_i}) = (1 + \langle \mathbf{X}, \mathbf{X_i}\rangle)^d$ for some integer d, known as the **Polynomial** Kernel
- $K(\mathbf{X}, \mathbf{X_i}) = e^{(\gamma\|\mathbf{X}-\mathbf{X_i}\|^2)}$ for some $\gamma$, known as the **Gaussian** Kernel
- $K(\mathbf{X}, \mathbf{X_i}) = tanh(p_1\langle \mathbf{X}, \mathbf{X_i}\rangle - p_2)$ for some positive $p_1, p_2$, known as the **Perceptron** Kernel

We will restrict our attention in this paper to polynomial kernels, for ease of hardware implementations. Namely,

$$y(\mathbf{X}) = \sum_{i=1}^{n} \alpha_i t_i (1 + \langle \mathbf{X}, \mathbf{X_i}\rangle)^d + b \tag{4}$$

By noting that $(1 + \langle \mathbf{X}, \mathbf{X_i}\rangle)^d = \sum_{j=0}^{d} \binom{d}{j} \langle \mathbf{X}, \mathbf{X_i}\rangle^j$, we can rewrite (4) as:

$$y(\mathbf{X}) = \sum_{j=0}^{d} \binom{d}{j} y_j(\mathbf{X}) + b \tag{5}$$

where $y_j(\mathbf{X}) \triangleq \sum_{i=1}^{n} \alpha_i t_i \langle \mathbf{X}, \mathbf{X_i}\rangle^j = \sum_{i=1}^{n} \alpha_i t_i Z_i^j$. $Z_i$ was introduced just for notational convenience.

## III. Quantization Effects

### A. Basics

Let $q_{\epsilon_1,\epsilon_2}$ be the quantization error, going from $\epsilon_2$ bits to $\epsilon_1$ bits. We denote by $\Delta(\epsilon_1, \epsilon_2) \triangleq 2^{-\epsilon_1} - 2^{-\epsilon_2}$ to be the quantization step size.A few examples are shown below:

- Quantization from real value to N bits $\implies \epsilon_1 = N$ and $\epsilon_2 = \infty$
- Quantization after multiplication $\implies \epsilon_1 = N$ and $\epsilon_2 = 2N$
- Quantization after addition $\implies \epsilon_1 = N$ and $\epsilon_2 = N + 1$

The quantization is usually assumed to be Uniformly distributed in the interval $[-\frac{\Delta(\epsilon_1,\epsilon_2)}{2}, \frac{\Delta(\epsilon_1,\epsilon_2)}{2}]$. One can easily show that:

$$E[q_{\epsilon_1,\epsilon_2}] = 0 \tag{6}$$

$$\sigma^2_{\epsilon_1,\epsilon_2} \triangleq E[q^2_{\epsilon_1,\epsilon_2}] = \frac{\Delta^2(\epsilon_1,\epsilon_2)}{12} \tag{7}$$

We denote in what follows $\tilde{x}$ to be the quantized version of x

$$\tilde{x} \triangleq x + q_x$$

### B. Inner Product

Let $q_{IP,\mathbf{X}}$ be the calculation quantization error at the output of an Inner Product $\mathbf{X}$ of $D$ dimensions. We have a total of $D$ multiplications and $(D-1)$ additions. Hence, we can obtain $q_{IP,\mathbf{X}}$ as

$$q_{\mathbf{Z},IP} = \sum_{i=1}^{D} q^{(i)}_{N,2N} + \sum_{i=1}^{D-1} q^{(i)}_{N,N+1}$$

where $q^{(i)}_{N,2N}$ and $q^{(i)}_{N,N+1}$ are all assumed independent. We derive the mean and variance,

$$\begin{aligned} E[q_{\mathbf{Z},IP}] &= E[\sum_{i=1}^{D} q^{(i)}_{N,2N} + \sum_{i=1}^{D-1} q^{(i)}_{N,N+1}] \\ &= \sum_{i=1}^{D} E[q^{(i)}_{N,2N}] + \sum_{i=1}^{D-1} E[q^{(i)}_{N,N+1}] = 0 \end{aligned}$$

$$\begin{aligned}\sigma_{IP}^2 \triangleq &= E[q_{\mathbf{Z},IP}^2] = E[\sum_{i=1}^{D} q_{N,2N}^{2(i)} + \sum_{i=1}^{D-1} q_{N,N+1}^{2(i)} \\ &\quad +2\sum_{i=1}^{D} q_{N,2N}^{(i)} q_{N,N+1}^{(i)}] \\ &= \sum_{i=1}^{D} E[q_{N,2N}^{2(i)}] + \sum_{i=1}^{D-1} E[q_{N,N+1}^{2(i)}] \\ &= D\sigma_{N,2N}^2 + (D-1)\sigma_{N,N+1}^2\end{aligned}$$

## IV. Quantization in a $2^{nd}$ order polynomial SVM

We will restrict our derivations for an SVM using a $2^{nd}$ order polynomial Kernel. Equation (5) reduces to :

$$y(\mathbf{X}) = y_0(\mathbf{X}) + 2y_1(\mathbf{X}) + y_2(\mathbf{X}) + b$$

We note here that at the time of implementing the hardware system, the SVM has already been trained and we already know the indices for which $\{\alpha\}_{i=1}^{n}$ is non-zero. Hence, we can store the $\{\alpha\}_{i=1}^{L}$ which are non-zero with their according support vectors, $\{\mathbf{X}\}_{i=1}^{L}$, and targets, $\{t_i\}_{i=1}^{L}$. We will not consider the quantization effect on these parameters as it is a pre-determined process and not random by any means. The only quantization left to analyze is the input quantization and calculations inside the processor.

### A. Constant Term

$$y_0(\mathbf{X}) = \sum_{i=1}^{L} \alpha_i t_i \tag{8}$$

The quantity in equation (8) can be pre-determined and stored during the implementation of this system.

### B. Linear Term

$$y_1(\mathbf{X}) = \sum_{i=1}^{L} \alpha_i t_i Z_i \tag{9}$$

with $Z_i = \langle \mathbf{X}, \mathbf{X_i} \rangle$. We first calculate $\tilde{Z}_i$.

$$\begin{aligned}\tilde{Z}_i &= \langle \tilde{\mathbf{X}}, \mathbf{X_i} \rangle \\ &= \langle \mathbf{X}, \mathbf{X_i} \rangle + \langle q_{\mathbf{X}}, \mathbf{X_i} \rangle + q_{IP,Z_i}\end{aligned}$$

Now we replace back in (9),

$$\begin{aligned}\tilde{y}_1(\mathbf{X}) &= \sum_{i=1}^{L} \alpha_i t_i \tilde{Z}_i \\ &= \sum_{i=1}^{L} \alpha_i t_i Z_i + \sum_{i=1}^{L} \alpha_i t_i \langle q_{\mathbf{X}}, \mathbf{X_i} \rangle \\ &\quad + \sum_{i=1}^{L} \alpha_i t_i q_{IP,Z_i} + q_{IP,y_1(\mathbf{X})} \\ &\triangleq y_1(\mathbf{X}) + q_{y_1(\mathbf{X})}\end{aligned}$$

Now, we evaluate the mean and variance of $q_{y_1(\mathbf{X})}$,

$$\begin{aligned}E[q_{y_1(\mathbf{X})}] &= \sum_{i=1}^{L} \alpha_i t_i \langle E[q_{\mathbf{X}}], \mathbf{X_i} \rangle + \sum_{i=1}^{L} \alpha_i t_i E[q_{IP,Z_i}] \\ &\quad + E[q_{IP,y_1(\mathbf{X})}] \\ &= 0 \end{aligned} \tag{10}$$

$$\begin{aligned}E[q_{y_1(\mathbf{X})}^2] &= E\left[\left(\sum_{i=1}^{L} \alpha_i t_i \langle q_{\mathbf{X}}, \mathbf{X_i} \rangle\right)^2\right] + E[q_{IP,y_1(\mathbf{X})}^2] \\ &\quad + E\left[\left(\sum_{i=1}^{L} \alpha_i t_i q_{IP,Z_i}\right)^2\right] + \cancelto{0}{E[cross\ terms]} \\ &= \sigma_{N,\infty}^2 \sum_{k=1}^{D} \langle \alpha \mathbf{t}, \mathbf{X}^{(k)} \rangle^2 + \sigma_{IP}^2 + \sigma_{IP}^2 \|\alpha\|^2 \end{aligned} \tag{11}$$

### C. Quadratic Term

$$y_2(\mathbf{X}) = \sum_{i=1}^{L} \alpha_i t_i Z_i^2$$

Here, we compute $\tilde{Z}_i$ like before. To obtain $\tilde{Z}_i^2$, we multiply it by itself,

$$\begin{aligned}\tilde{Z}_i^2 &= \tilde{Z}_i \tilde{Z}_i \\ &= \langle \mathbf{X}, \mathbf{X_i} \rangle^2 + \langle q_{\mathbf{X}}, \mathbf{X_i} \rangle^2 + q_{IP,Z_i}^2 + 2q_{IP,Z_i} \langle \mathbf{X}, \mathbf{X_i} \rangle \\ &\quad + 2q_{IP,Z_i} \langle q_{\mathbf{X}}, \mathbf{X_i} \rangle + 2\langle \mathbf{X}, \mathbf{X_i} \rangle \langle q_{\mathbf{X}}, \mathbf{X_i} \rangle\end{aligned}$$

By a similar derivation to that in the previous sub-section, we can write $q_{y_2(\mathbf{X})}$ as,

$$\begin{aligned}q_{y_2(\mathbf{X})} &= \sum_{i=1}^{L} \alpha_i t_i \Big( \langle q_{\mathbf{X}}, \mathbf{X_i} \rangle^2 + q_{IP,Z_i}^2 + 2q_{IP,Z_i} \langle \mathbf{X}, \mathbf{X_i} \rangle \\ &\quad + 2q_{IP,Z_i} \langle q_{\mathbf{X}}, \mathbf{X_i} \rangle + 2\langle \mathbf{X}, \mathbf{X_i} \rangle \langle q_{\mathbf{X}}, \mathbf{X_i} \rangle \Big) \\ &\quad + q_{IP,y_2(\mathbf{X})}\end{aligned}$$

Now, we calculate the mean and variance of $q_{y_2(\mathbf{X})}$,

$$E[q_{y_2(\mathbf{X})}] = \sigma_{N,\infty}^2 \sum_{k=1}^{D} \langle \alpha \mathbf{X^{(k)}}, \mathbf{t} \mathbf{X}^{(k)} \rangle + \sigma_{IP}^2 \langle \alpha, \mathbf{t} \rangle \tag{12}$$

(Only the Squared terms contribute to the mean, others are 0)

$$\begin{aligned}E[q_{y_2(\mathbf{X})}^2] &= \sigma_{N,\infty}^4 \left( \sum_{k=1}^{D} \langle \alpha \mathbf{X^{(k)}}, \mathbf{t} \mathbf{X}^{(k)} \rangle \right)^2 + \sigma_{IP}^4 \langle \alpha, \mathbf{t} \rangle^2 \\ &\quad + 4\sigma_{N,\infty}^2 \sum_{k=1}^{D} \langle \alpha \mathbf{Z}, \mathbf{t} \mathbf{X}^{(k)} \rangle + \sigma_{IP}^2 \left(1 + 4\langle \alpha \mathbf{Z}, \alpha \mathbf{Z} \rangle\right) \\ &\quad + \sigma_{N,\infty}^2 \sigma_{IP}^2 \sum_{k=1}^{D} 4 \langle \alpha \mathbf{t} \mathbf{X^{(k)}}, \alpha \mathbf{t} \mathbf{X^{(k)}} \rangle \\ &\quad + \sigma_{N,\infty}^2 \sigma_{IP}^2 \sum_{k=1}^{D} \langle \alpha, \mathbf{t} \rangle \langle \alpha \mathbf{X^{(k)}}, \mathbf{t} \mathbf{X^{(k)}} \rangle \end{aligned} \tag{13}$$

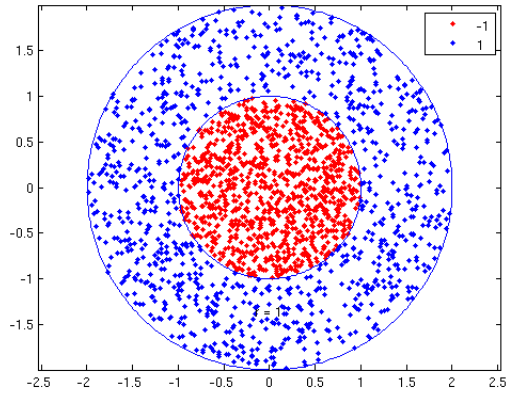


Figure 2. Experimental Setup

We assumed in the above that

$$E[q^4_{\epsilon_1,\epsilon_2}] = \left(E[q^2_{\epsilon_1,\epsilon_2}]\right)^2 \frac{16\times 5}{12^2} \approx \left(E[q^2_{\epsilon_1,\epsilon_2}]\right)^2$$

### *D. $2^{nd}$ order polynomial system*

By ignoring the quantization effects of the last stage of calculations , we can write $q_{y(\mathbf{X})}$ as

$$q_{y(\mathbf{X})} = q_{y_1(\mathbf{X})} + q_{y_2(\mathbf{X})}$$

Now, we calculate the mean and variance

$$E[q_{y(\mathbf{X})}] = E[q_{y_1(\mathbf{X})}] + E[q_{y_2(\mathbf{X})}] \tag{14}$$

$E[q_{y_1(\mathbf{X})}]$ and $E[q_{y_2(\mathbf{X})}]$ are found in equations (10) and (12).

$$E[q^2_{y(\mathbf{X})}] = E[q^2_{y_1(\mathbf{X})}] + E[q^2_{y_2(\mathbf{X})}] + 2E[q_{y_1(\mathbf{X})}q_{y_2(\mathbf{X})}] \tag{15}$$

$E[q^2_{y_1(\mathbf{X})}]$ and $E[q^2_{y_2(\mathbf{X})}]$ are found in equations (11) and (13). We still need to calculate $E[q_{y_1(\mathbf{X})}q_{y_2(\mathbf{X})}]$

$$\begin{aligned} E[q_{y_1(\mathbf{X})}q_{y_2(\mathbf{X})}] &= 2\sigma^2_{N,\infty}\sum_{k=1}^{D}\langle\alpha\mathbf{Z},\mathbf{tX^{(k)}}\rangle\langle\alpha,\mathbf{tX^{(k)}}\rangle \\ &\quad +2\sigma^2_{IP}\langle\alpha\mathbf{tZ},\alpha\mathbf{t}\rangle \end{aligned}$$

## V. Experimental Results

Let $x_i = \left(x_i^{(1)} \triangleq r_i cos(\theta_i), x_i^{(2)} \triangleq r_i sin(\theta_i)\right) \in \mathbb{R}^2$ such that

$$r_i \in \begin{cases}[0,1], & \text{if } t_i = -1. \\ [1,2], & \text{if } t_i = 1.\end{cases} \quad , \quad \theta_i \in [0, 2\pi]$$

The experimental setup is shown in Figure 2. We show the training of our data vectors with a $1^{st}$ order polynomial Kernel in Figure 3. One clearly see that this classifier will not work properly. In Figure 4 and 5, the training of our data vectors with a $2^{nd}$ order polynomial Kernel and Gaussian Kernel ,respectively ,are shown. We can see that both classifiers have a good separating hyperplane but the number of Support vectors for the polynomial Kernel is half as much ($6\%$ compared to $12\%$). A total of 2000 vectors were used for training the SVM in each case. This justifies our choice for a $2^{nd}$ order polynomial Kernel.

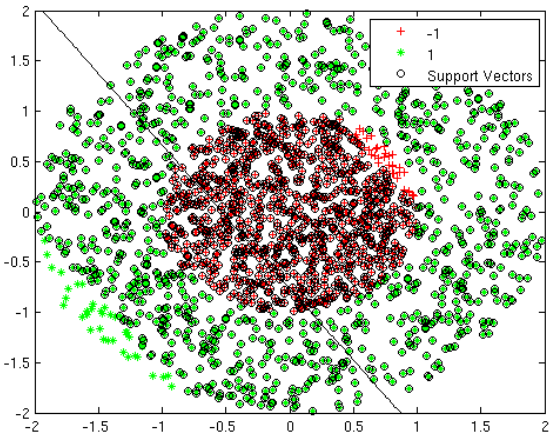


Figure 3. SVM training with $1^{st}$ order polynomial

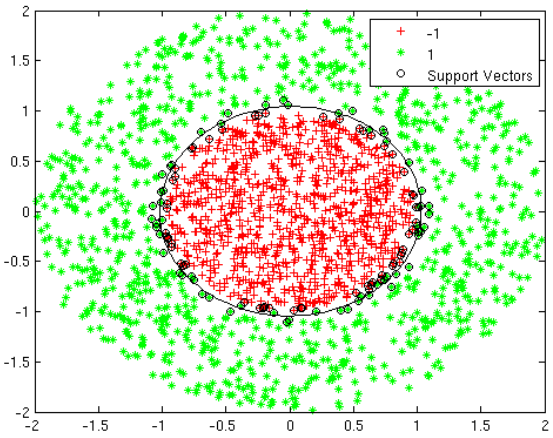


Figure 4. SVM training with $2^{nd}$ order polynomial

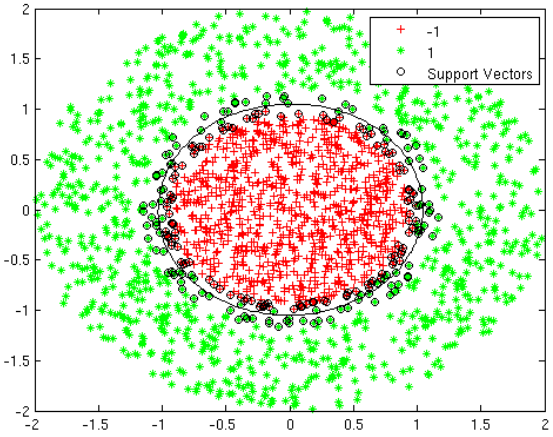


Figure 5. SVM training with Gaussian Kernel

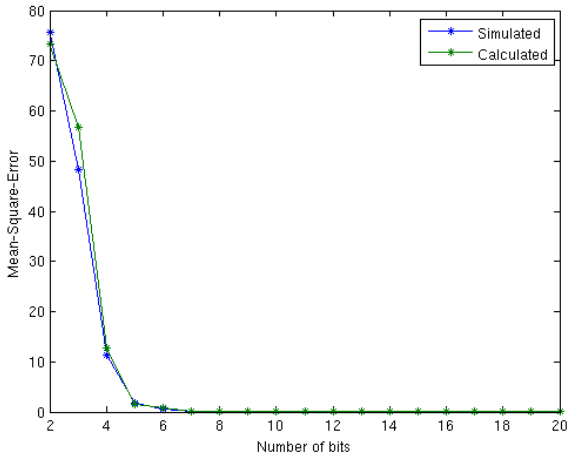


Figure 6. Quantization Simulation vs Theoretical

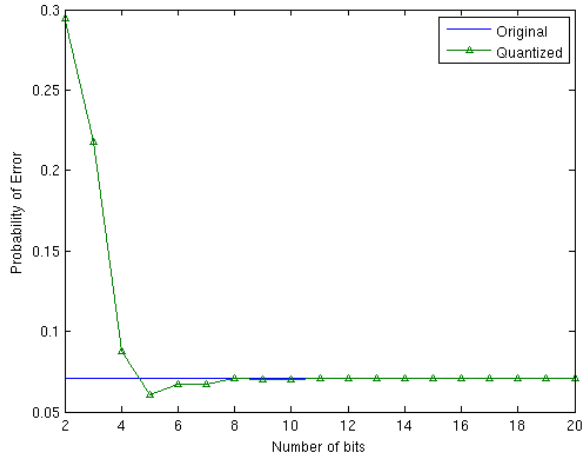


Figure 7. Probability of Error

In Figure 6, we show the quantized output of the network by simulations and the predicted output by our analysis above. The simulation has been done for 1000 test vectors and what we show in the figure is the average Mean-Square-Error. We can see that the analysis and simulations do match. Now, what can one do with this analysis?

We usually have a requirement on the Probability of error , $P_e$. We can use the analysis above to predict $P_e$ of the network for all different bit widths. This is shown in Figure 7. The Original curve represents the performance of the network without any quantization done and the quantized version is obtained by the above analysis. If the requirement on $P_e$ require us to use less than 8 bits, we are better off with the LP Analog approach (mixed-signal technique) and otherwise, just stick to the digital implementation.

## VI. Conclusion and Future work

In this paper, we derived the quantization effects in a SVM implementing $2^{nd}$ order polynomial kernel, and then verified our results with simulations. Although in practice a $2^{nd}$ order polynomial kernel can give us the required accuracy (also shown in the experimental results), We plan on extending the analysis presented in this paper for polynomial kernels for the sake of completeness. Finally, the analysis of the link between probability of error and quantization variance is still missing. A more advanced analysis would be required for that, in addition to some further assumptions.

## References

[1] V. Vapnik. Estimation of Dependences Based on Empirical Data [in Russian]. Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).

[2] C. Cortes and V. Vapnik. Support vector networks. Machine Learning, 20:273–297, 1995.

[3] V. Blanz, B. Sch¨olkopf, H. B¨ulthoff, C. Burges, V. Vapnik, and T. Vetter. Comparison of view–based object recognition algorithms using realistic 3d models. In C. von der Malsburg, W. von Seelen, J. C. Vorbr¨uggen, and B. Sendhoff, editors, Artificial Neural Networks — ICANN'96, pages 251 – 256, Berlin, 1996. Springer Lecture Notes in Computer Science, Vol. 1112.

[4] T. Joachims. Text categorization with support vector machines. Technical report, LS VIII Number 23, University of Dortmund, 1997. ftp://ftp-ai.informatik.uni-dortmund.de/pub/Reports/report23.ps.Z.

[5] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. Advances in Neural Information Processing Systems, 9:155–161, 1997.

[6] D. Anguita, A. Boni, S. Ridella, "A digital architecture for support vector machines: theory, algorithm, and FPGA implementation" IEEE Trans. on Neural Networks, vol. 14, pp. 993–1009, Sept. 2003.

[7] R. Genov, G. Cauwenberghs, "Kerneltron: support vector machine in silicon" IEEE Trans. on Neural Networks, vol. 14, pp. 1426–1434, Sept. 2003.

[8] Yoo, Jerald, Long Yan, Dina El-Damak, Muhammad Awais Bin Altaf, Ali H. Shoeb, and Anantha P. Chandrakasan. "An 8-Channel Scalable EEG Acquisition SoC With Patient-Specific Seizure Classification and Recording Processor." (2013): 1-15.

[9] Hosticka, Bedrich J. "Performance comparison of analog and digital circuits." Proceedings of the IEEE 73, no. 1 (1985): 25-29.

[10] D. Anguita, S. Ridella, S. Rovetta, "Worst case analysis of weight inaccuracy effects in multilayer perceptrons" IEEE Trans. on Neural Networks, vol. 10, pp. 415–418, Mar. 1999.

[11] O.-J. Kwon, S.-Y. Bang, "Comments on: The effects of quantization on multilayer neural networks" IEEE Trans. on Neural Networks, vol. 9, pp. 718–719, Jul. 1998.

[12] G. Dündar, K. Rose, "The effects of quantization on multilayer neural networks" IEEE Trans. on Neural Networks, vol. 6, pp. 1446–1451, Nov. 1995.

[13] Anguita, Davide, and Giovanni 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, pp. 681-684. IEEE, 2005.