Support Vector Machines for Analog Circuit Performance Representation

F. De Bernardinis[†§] fdb@eecs.berkeley.edu
M. I. Jordan[†] jordan@cs.berkeley.edu
A.SangiovanniVincentelli[†] alberto@eecs.berkeley.edu

[†]Department of Electrical Engineering and Computer Science
University of California, Berkeley

[§]Dipartimento di Ingegneria dell'Informazione
Università di Pisa, Italy

ABSTRACT

The use of Support Vector Machines (SVMs) to represent the performance space of analog circuits is explored. In abstract terms, an analog circuit maps a set of input design parameters to a set of performance figures. This function is usually evaluated through simulations and its range defines the feasible performance space of the circuit. In this paper, we directly model performance spaces as mathematical relations. We study approximation approaches based on two-class and one-class SVMs, the latter providing a better tradeoff between accuracy and complexity avoiding "curse of dimensionality" issues with 2-class SVMs. We propose two improvements of the basic one-class SVM performances: conformal mapping and active learning. Finally we develop an efficient algorithm to compute projections, so that top-down methodologies can be easily supported.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids-Verification

General Terms

Algorithms

1. INTRODUCTION

Analog design has been traditionally a difficult discipline of IC design. While in digital designs functionality depends on discrete sequences of discrete (binary) signals, continuous sequences (waveforms) of continuous values encode the information we need to manipulate and use in the analog case. For this reason, any second-order physical effect may have a significant impact on function and performance of an analog circuit. The effect of the choice of design parameters such as transistor size and layout on performance is usually computed by simulation. Since simulations require completely specified circuits to compute performances, the complexity of setting up simulations for system level exploration is daunting even if we do not consider its computational cost and the size of the search space. For this reason, behavioral models are currently used at the first stages of designs to partition heuristically system constraints on individual blocks. However, this is carried out in a pure functional way, without any notion of the underlying feasible performance space and tradeoffs. Building an approximate representation of the performance space is of great interest for providing behavioral models with architectural constraints that can fully enable top-down design methodologies [1]. In the 70s and early 80s, a number of approaches [2] have been proposed to approximate the relation between circuit performance and parameters in explicit form. In particular, the design problem tackled was to identify the value of the design parameters that would yield a point in the performance space meeting a set of requirements expressed as inequalities. Assuming that the region of points satisfying the requirements is convex, the boundaries of the region were approximated by a set of hyperplanes (simplicial approximation [3]) or by quadratic functions. If the exploration of the feasible region was part of an optimization procedure, the simplicial or quadratic approximation was refined locally to obtain a smaller error and yield an optimization procedure that would eventually converge to an optimal point.

In this paper, we take a slightly different view of the problem by directly modeling performances themselves in place of parameter-performance relations. We improve upon the existing techniques by exploiting recent results on learning and approximation techniques. In particular, we want to directly represent the feasibility of achieving particular performances for a given topology of a circuit. This can be viewed as the problem of estimating the support of a set, and treated statistically as a quantile estimation problem.

2. BACKGROUND

System level analog design is a process largely dominated by heuristics. Given a set of specifications/requirements that describe the system to be realized, the selection of an optimal implementation architecture comes mainly out of experience. Usually, what is achieved is just a feasible point at the system level, while optimality is sought locally at the circuit level. The reason for this is the difficulty in the analog world of knowing if something is realizable without actually attempting to design the circuit. The number of effects to consider and their complex interrelations make this problem approachable only through the experience coming from past designs. System level optimization is extremely

DAC 2003, June 2–6, 2003, Anaheim, California, USA.

hard because of the difficulties that hierchical designs face in abstracting single block behaviors and establishing cost and feasibility without further knowledge of the implementation architecture and topology.

To cope with these problems, the Analog Platform concept has been proposed in [4] extending the one developed for the digital world. The term Analog Platform (AP) indicates an architectural resource to map analog functionalities during early system level exploration phases and constrain exploration to the feasible space of the considered implementation (or set of implementations). Therefore, an AP consists of a set of behavioral models and of matching performance models. *Behavioral models* are parameterized mathematical models of the analog functionality that introduce at the functional level a number of non-idealities attributable to the implementation and not intrinsic in the functionality itself, such as distortion, bandwidth limitations, and noise. Non-ideal effects, as well as the principal performance figures of the functionality (e.g. gain, power consumption and area) are embedded in the behavioral models in the form of parameters. *Performance models* constrain these parameters to satisfy a mathematical relation so that only feasible instances of the behavioral model (models with a complete set of parameter values) can be selected with respect of the considered architecture. In this paper we provide the details of the performance representation mechanism presented in [4], showing how they can be derived from sample data.

Other methods for building analog models from simulation data are mostly based on regression mechanisms. A given performance figure is fit on a set of regressors that are function of physical parameters of the circuit. An interesting model based on boosting methods has been recently proposed [5]. However, regression approaches are not suited to top-down methodologies where abstraction levels introduce opaqueness in the system hierarchy that hides the implementation details needed for regression. Furthermore, each performance figure is fitted independently from the others, so that errors in capturing relations among different feasible performances may cumulate. An approach to model the feasible space has been proposed in [6]. In this paper we present a novel approach to directly model performances that allows capturing high dimensional performance spaces with efficient representations and provides the necessary means to support the multiple levels of abstraction required in top-down flows. Furthermore, it does not require that designers generate explicit models for their circuits, nor it asks them to cast their problems into specific mathematical frameworks.

3. PROBLEM FORMULATION

Top-down flows incrementally process designs through a sequence of abstraction levels until final implementation is achieved. At each level of abstraction top-down methodologies only deal with parameters present at the current level to transform a given set of requirements into next-level constraints. For example, if we are exploring the use of an amplifier at a given point in a design, our main interest is in selecting the optimal amplifier instance with respect to a given set of performance figures, such as gain, power, noise, etc. Looking at the next (more detailed) level of abstraction, gain, power and noise can be placed in effect/cause relation with a given amplifier topology, bias point and device sizing, which should not be considered at the current level of abstraction. This implies that classic regression schemes for

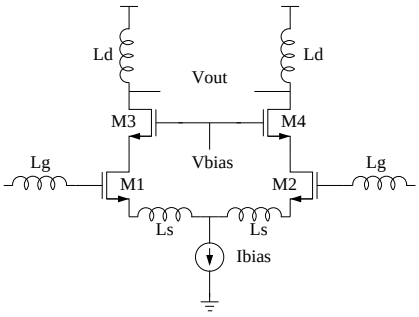

Figure 1: Schematic of the LNA modeled in this paper. Throughout the modeling process, the input impedance is matched to 50Ω and the differential pair is assumed with 2% maximum mismatch.

representing analog circuit performances are not the most suitable means of performance representation in top-down flows. Therefore, we propose modeling the performances themselves as mathematical relations on performance figures (effects), discarding the information on the relative parameters (causes) as not pertinent at the current level of abstraction. To approach the problem quantitatively, we introduce the following definitions:

1. *Input space* $\mathcal{I}$ - Given a circuit $\mathcal{C}$ and m parameters controlling its instances, $\mathcal{I}_\mathcal{C} \subseteq \mathbb{R}^m$ is the set of m-tuples (parameter values) over which we want to characterize $\mathcal{C}$.

2. *Output space* $\mathcal{O}$ - Given a circuit $\mathcal{C}$ and n performance figures completely characterizing its behavioral model, $\mathcal{O}_\mathcal{C} \subseteq \mathbb{R}^n$ is the set of m-tuples (performance values) that are achievable by $\mathcal{C}$.

3. *Evaluation function* ϕ - Given a circuit $\mathcal{C}$, $\mathcal{I}_\mathcal{C}$ and $\mathcal{O}_\mathcal{C}$, $\phi_\mathcal{C} : \mathcal{I} \rightarrow \mathcal{O}$ allows translating a parameter m-tuple set into a performance n-tuple set.

4. *Performance relation* $\mathcal{P}$ - Given a circuit $\mathcal{C}$, $\mathcal{I}_\mathcal{C}$, $\mathcal{O}_\mathcal{C}$ and $\phi_\mathcal{C}$, we define the performance relation of $\mathcal{C}$ given $\mathcal{I}_\mathcal{C}$ and $\phi_\mathcal{C}$ to be $\mathcal{P}_\mathcal{C}$ on $\mathbb{R}^n$ that hold only for points $o \in \mathcal{O}_\mathcal{C}$. With a little abuse in notation, we will denote both the performance relation characteristic function $\chi_\mathcal{P}(x) : \mathbb{R}^n \rightarrow \{0, 1\}$ and the relation itself with $\mathcal{P}_\mathcal{C}(x)$.

Unfortunately, very little is known a priori about $\phi_\mathcal{C}$; circuits are nonlinear systems so complex that it is difficult to derive any strong property concerning $\mathcal{O}_\mathcal{C}$ or $\mathcal{P}_\mathcal{C}$. However, it is usually the case that the control set $\mathcal{I}_\mathcal{C}$ of interest is a region of $\mathbb{R}^m$ where $\phi_\mathcal{C}$ can be assumed to be continuous. If we also assume $\mathcal{I}$ to be a connected set (or a union of connected sets), then $\mathcal{O}$ will be a (union of) connected set as well, which is the only property we will exploit for building the approximation of $\mathcal{P}_\mathcal{C}$.

Even when the continuity assumption does not strictly hold in $\mathcal{I}$, the set of performances that we are interested in modeling usually relates to circuits with well determined operating characteristics (performance invariant $\mu(x) = 1$). For example, when we look at an oscillator we are not concerned in modeling points where it does not oscillate and that would introduce an abrupt discontinuity in its performance figures. As long as the circuit remains in that operating mode, we can assume its performance to vary with

continuity when we span $\mathcal{I}$. Therefore, we are actually interested in an effective $\mathcal{I}^* = \phi^{-1}(\{x \in \mathcal{O} \text{ s.t. } \mu(x) = 1\})$, which makes the continuity assumption true.

In this paper we use the Low Noise Amplifier (LNA) for wireless applications shown in Figure 1 as a case study. For simplicity, $\mathcal{I}_{LNA}$ has been taken as a hypercube in $\mathbb{R}^7$, and a large number of simulations randomly spanning this hypercube were run to collect data for the experiments. $\mathcal{I}_{LNA}$ includes the sizes of M1-M2 and M3-M4 (2% maximum mismatch), the bias current, the output load and the current mirror size. $\mathcal{O}_{LNA}$ also lies in $\mathbb{R}^7$ and includes gain, noise, power, second and third order harmonic distortion, bandwidth and ripple in frequency response. Therefore, $\phi_{LNA} : \mathbb{R}^7 \to \mathbb{R}^7$ and $\mathcal{P}_{LNA} : \mathbb{R}^7 \to \{0, 1\}$.

4. SUPPORT VECTOR MACHINES

Support Vector Machines were first proposed in 1992 [7] to solve machine learning problems. Machine learning consists in classifying points in a large input space as satisfying a potentially complex unknown relation given a set of experiments that answer the question for a *training* set in the input space. The training set populates the "performance" function space with points that are either satisfying or not the relation. The task is to approximate the performance function based on the knowledge of the training set, *so that a point in the input space not part of the training set could be correctly classified.* A general principle to select an approximant that is consistent with the training set and has good properties in classifying inputs, is the so called Occam's razor that can be used to set up an appropriate optimization problem.

A most important characteristic of an approximation scheme is the choice of the basis functions, i.e., of those building blocks that can be chosen to optimize the likelihood of being correct on the inputs not in the training set. We propose Support Vector Machines (SVMs) as a way of approximating the performance relation $\mathcal{P}$. These approximating functions are of the form

$$f(x) = sign(\sum_i \alpha_i e^{-\gamma|x-x_i|^2} - \rho) \tag{1}$$

where x_i are input samples, α_is are weighting multipliers, ρ is a constant, γ is a parameter controlling the fit of the approximation and the sum is over a proper set of samples (support vector set).

SVMs belong to the class of hyper-plane classifiers that separate data points according to which side of a hyperplane they fall. Non-linear mappings $\psi(\cdot)$ of the input samples into a high dimensional feature space are exploited so to increase data separability through hyperplanes. More specifically, SVMs exploit mapping to Hilbert spaces through *kernel* functions, so that a kernel $k(x, \cdot)$ is associated at each point x. In this paper, because of the weak properties that we have on $\mathcal{P}$, the Gaussian (Radial Basis Function - RBF) kernel is chosen,

$$k(\boldsymbol{x}, \boldsymbol{x'}) = e^{-\gamma \cdot \|\boldsymbol{x}-\boldsymbol{x'}\|^2} \tag{2}$$

where γ is a parameter of the kernel and controls the "width" of the kernel function around x. In paragraph 5.3 we show a method to modify the kernel function to improve the approximation accuracy. The optimal hyperplane is represented by a small fraction of the original training data and can be

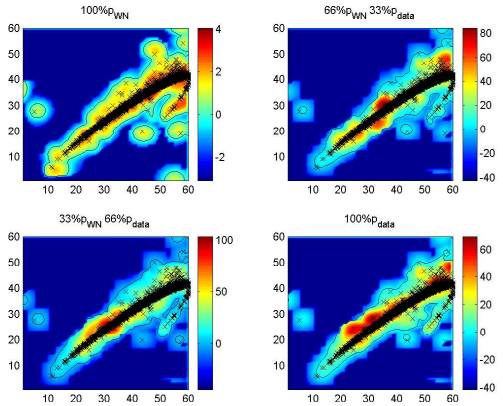

Figure 2: Plots of HD_3 vs HD_2 used as a reference pattern for the visual inspection for false positives. The projections have been obtained from 4-dimensional SVMs trained using mixtures of $p_{compl}(\cdot)$ and $p_{WN}(\cdot)$ to generate -1 samples.

computed very efficiently through the following optimization problem:

$$\min_{\boldsymbol{w},\boldsymbol{\xi},\rho,b} \{\frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_i \xi_i - \nu\rho\} \tag{3}$$
$$\text{s.t. } y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq \rho - \xi_i, \qquad \rho, \xi_i \geq 0$$

where $\boldsymbol{x_i} = \psi(x_i)$ are the training samples, y_i are the training labels, m is the number of samples, $\boldsymbol{w} = \sum_{i=1}^{nSV} y_i \alpha_i \boldsymbol{x_i}$ defines the optimal hyperplane in terms of a linear combination of samples and nSV is the number of support vectors. The parameter ν provides an intuitively appealing control on some of properties of the SVM; most importantly ν provides an upper bound on the fraction of false negatives (outliers).

In Section 5.1 we rely on a standard two-class SVM classifiers to build an approximation for $\mathcal{P}$. A more direct approach, however, is to make use of the so-called "one-class SVM" [8]. Since the characteristic functions of the performance relations being considered are deterministic, the support regions show sharp boundaries. The SVM approach also holds open the promise of obtaining representations of the performance space at varying levels of abstraction, providing needed flexibility for the design process. In Section 6, we investigate an approach for achieving such abstraction that makes use of the computational properties of the SVM approach.

5. SVM APPROXIMATIONS OF $\mathcal{P}$

The accuracy of an approximation $\hat{\mathcal{P}}$ is ideally assessed by two quantities: the rate of *false negatives* and the rate of *false positives*. In our problem, false positives represent a more serious problem than false negatives. In fact, top-down methodologies rely on predicted performances being achievable when partitioning constraints over all the blocks in the system; therefore, a false positive would make the resulting design not feasible. On the other hand, false negatives may prevent us to achieve a better point but the design would still be feasible. Although false positives are of great interest, they are also problematic to measure—since we use simulation to generate samples, there is no constructive way of generating points in $\mathbb{R}^n \setminus \mathcal{O}$.

In the attempt of gaining some initial insight into the rate of occurrence of false positives, we generated plots of some

Number of -1 random samples	#SV	%Train	%Test	FP with 40k noise samples
20,000	2,164	98.76	93.55	748
30,000	2,626	97.08	92.07	522
40,000	3,056	93.70	90.11	335

Table 1: Performance of two-class ($\gamma = 20, \nu = 0.05$, 5,000 +1 samples). Large negative sets decrease the number of false positives, but for sets larger than 30,000 samples the false negative rates become significantly worse.

characteristic 2-D and 3-D projections of $\mathcal{P}_{LNA}(\cdot)$. Examples of such plots are reported in Figure 2, where we show the projection on a plane representing second-order harmonic distortion (HD_2) and third-order (HD_3) for four different estimators to be described in the following section. Estimators that place the estimated boundary around regions that contain few data points are oversmoothed, and provide visual evidence of the need to choose smaller values of regularization parameters.

We have also explored a more quantitative approach, based on the following intuition. Since $\mathcal{O}$ generally lies in a low-dimensional manifold in the ambient Euclidean space, if data are sampled throughout $\mathbb{R}^n$, the number of true positives (n_{TP}) is expected to be small relative to the number of false positives (n_{FP}). Also, for reasonable values of the SVM parameters (γ — controls the width of the kernel around the i^{th} support vector—, and ν — controls false positives), such that the performance space is approximated as a connected set, it is reasonable to expect that n_{TP} should be nearly independent of the SVM parameters. This suggests that we may be able to partition the number of positives, $n_P(\gamma, \nu)$, from samples in $\mathbb{R}^n$ as follows:

$$n_P(\gamma, \nu) = n_{TP} + n_{FP}(\gamma, \nu). \quad (4)$$

Thus, based on the rate of variation of $n_P(\gamma, \nu)$ with γ and ν, we can obtain a more quantitative basis for setting optimal values for γ and ν and thereby controlling the problematic false positive rate.

5.1 Two-class SVM

One approach to finding an approximation of $\mathcal{O}$ is to attempt to use the powerful tools of two-class SVMs to form a discriminant boundary that separates $\mathcal{O}$ from its complement. This requires generating artificial "negative-class" data from the complement, and given that we are unable to generate data from the complement set algorithmically (the simulator only generates "positive-class" data), we must develop heuristics for generating such data. We have studied three different heuristics for specifying suitable density functions $p(\boldsymbol{x})$, ($\boldsymbol{x} \in [-1, 1]^7$) that can be used to generate "negative-class" data:

- $p_{WN}(\boldsymbol{x}) = constant$;
- $p_{data}(\boldsymbol{x}) = \prod_i \hat{p}_i(x_i)$, where $\hat{p}_i(x_i)$ is the empirical marginal density function of points in $\mathcal{O}$ on the i^{th} dimension;
- $p_{compl}(\boldsymbol{x}) = \prod_i \hat{q}_i(x_i)$, where $\hat{q}_i(x_i) = \frac{\hat{p}_{max} - \hat{p}_i(x_i)}{\hat{p}_{max}\Delta x_i - 1}$, $\Delta x_i =$ range of variable x_i, is a *complementary* density function of $\hat{p}_i(x_i)$ (in the sense that $\text{argmax}\{\hat{p}_i(x_i)\} \equiv \text{argmin}\{\hat{q}_i(x_i)\}$).

The first approach places negative points independently of the actual $\hat{\mathcal{O}}$ so that the number of points near the boundary of $\mathcal{P}$ tends to be too small unless a very large number of samples is used. The second approach approximates the region where $\hat{\mathcal{O}}$ is located to reduce the number of samples needed but tends to blur the region. The third approach places most points out of $\mathcal{O}$, but tends to make the two classes "too separable" and overestimates $\mathcal{P}$. All the approaches have been experimentally evaluated and mixtures have been considered as well. The color gradients in Figure 2 show the effects of different complement generation heuristics on the projection accuracy. The color scale shows that increasing the percentage of $p_{data}(\cdot)$ noise decreases the effect of the positive pattern. Even though the region contour seems acceptable, the margins of the prediction (color shades in the picture) are not as sharp as in the case obtained with $p_{const}(\cdot)$. Analogous plots obtained with $p_{compl}(\cdot)$ (not included) show regions that are too smooth.

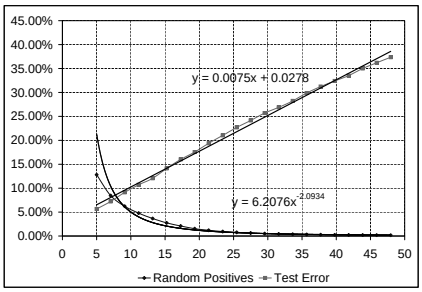

Figure 3: Test error rates (false negatives) and random positive rates achieved by a one-class SVM trained with 5,000 samples and $\nu = 0.05$ as a function of γ. False negative rates are computed presenting the SVM with 10,000 fresh samples, while random positives with 80,000 random samples.

Unfortunately, there is an intrinsic problem with 2-class SVM when applied to high dimensional problems related to curse-of-dimensionality issues. Considering a reduced $\mathcal{O} \in \mathbb{R}^4$ for the LNA, a ratio of 5,000/30,000 positive-to-negative points is needed to provide a reasonable tradeoff between FP and FN rates (cf. Table 1). Given that $\mathcal{O}$ is a manifold of low-dimension, we expect this imbalance to be exponentially worse for higher dimensionality of $\mathcal{O}$. Moreover, the imbalance in number of data skews the classification boundary in ways that are difficult to predict. In general, none of the heuristics for generating "negative-class" data provide intuitive, effective general parameters for controlling performance tradeoffs, and are subject to possibly large, ill-defined biases.

5.2 One-class SVM

One-class SVMs were originally introduced as a means of estimating quantiles of a probability density function [8]. One-class SVMs require samples from only one class of data (positive class) (assuming negative class elsewhere) and compute the optimal hyperplane maximizing the separation of data from the origin. As a consequence, even if minimum support estimation is attempted through the training process, there is no explicit penalty in having a large support regions in $\mathcal{O}$. In fact, all the bounds available for one-class SVM predictors concern the probability of false negatives, i.e. that a sample drawn from $p(x)$ might be misclassified by the machine.

Training of one-class SVMs is achieved by solving a quadratic optimization problem whose structure is analogous to

that of two-class SVMs:

$$\min_{\boldsymbol{w},\boldsymbol{\xi},\rho} \{\frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{\nu m}\sum_i \xi_i - \rho\} \quad (5)$$
$$\text{s.t.} \quad \boldsymbol{w}\cdot\boldsymbol{x}_i \geq \rho - \xi_i, \qquad \xi_i \geq 0$$

As with two-class SVMs, there are efficient algorithms to solve this quadratic programming problem that exploit structural properties of SVMs, e.g. [9] [10]. We based our implementation on libsvm [11], and we trained SVMs with up to 50,000 samples in very reasonable time (10's of minutes).

Differently from the two class formulation (3), there is no explicit penalties for false positives in (5) — all y_is are $+1$. As a consequence, larger values of γ in the RBF kernel are required to achieve tight approximations for the performance region. In our case, for moderate values of γ ($< 5 \approx 8$), the number of positives (eqn. 4) represents a major problem, as can be seen from Figure 3. Increasing γ reduces the number of positives. However, the performance in terms of generalization becomes worse than in the two-class case, as reported in Table 2 (false negatives are about 8% higher than comparable two-class SVMs). Finally, SVMs tend to degenerate into Parzer window estimators as larger and larger values for γ are used.

5.3 Improving accuracy

The accuracy of the estimator can be improved by enhancing the resolution in the support region boundaries. Since the underlying problem is deterministic, the transition across the boundary should be sharp. One way to achieve improved resolution is via *conformal transformation*. This approach was described in the context of the two-class SVM by [12] and [13], but the basic idea is also applicable to the one-class SVM. In particular, an initial estimate of the boundary is provided by prior training of an SVM. Since support vectors are located on the boundary of the hyperplane in feature space, they provide information about the region that is to undergo expansion. This expansion is achieved via a conformal transformation:

$$\tilde{k}(\boldsymbol{x},\boldsymbol{x}') = c(\boldsymbol{x})\cdot k(\boldsymbol{x},\boldsymbol{x}')\cdot c(\boldsymbol{x}'). \quad (6)$$

Possible conformal transformations include:

$$c_1(\boldsymbol{x}) = \sum_i \bar{\alpha}_i \cdot e^{-\gamma\|\boldsymbol{x}-\bar{\boldsymbol{x}}_i\|^2}, c_2(\boldsymbol{x}) = \sum_i e^{-\zeta_i\|\boldsymbol{x}-\bar{\boldsymbol{x}}_i\|^2} \quad (7)$$

where $\zeta_i = \frac{1}{M}\sum_j \|\bar{\boldsymbol{x}}_{\eta|j} - \bar{\boldsymbol{x}}_j\|^2$, and where $\boldsymbol{x}_{\eta|j}$ is the j^{th} nearest neighbor of $\bar{\boldsymbol{x}}_i$. $\bar{\boldsymbol{\alpha}}_i$ and $\bar{\boldsymbol{x}}_i$ refer to an SVM previously trained on the same dataset. Our results with both conformal transformations show significant performance improvements (see Table 2), with $c_1(\boldsymbol{x})$ performing slightly better than $c_2(\boldsymbol{x})$. However, the results achievable with active learning schemes (sec.5.4) and the complexity of projections over transformed kernels (sec.6) make their use less attractive.

5.4 Active learning

Because one-class SVMs estimate the degree of novelty of a given vector, it is possible to exploit misclassified samples as seeds for further sample generation, thus forcing the sampling scheme to place more points where fewer are present [2]. Ideally, all the training samples should be correctly classified, but the need to regularize forces a fraction of the training samples to be misclassified (controlled via the parameter ν). We can exploit such false negatives as follows.

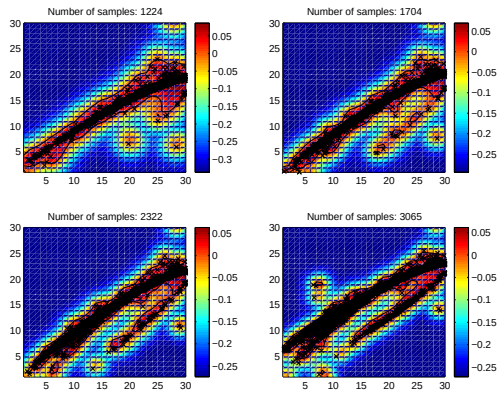

Figure 4: Results achieved interleaving training with generation of new data samples. Starting from 1,000 points randomly sampled, 4 iterations were completed with $\nu = \{0.1, 0.08, 0.06, 0.04\}$ and $\gamma = \{10, 15, 20, 25\}$. The final SVM emphasizes a complex pattern in the performance space that was not present in SVMs obtained from a larger number of random samples.

For each such false negative o_s, we can reinforce the information content of the data point (i.e. its novelty) by generating new samples $\{o_n\}$, in the neighborhood of o_s. One way to achieve this is by applying small random perturbations to the input vector i_s. Even if the simulation function $\phi_{\mathcal{C}}$ is not bijective, we can store the mapping $\{i_t\} \rightarrow \{o_t\}$ to get i_s. The set $\{i_n\}$ is then simulated to generate a new sample $\mathcal{O}'$. A new SVM is then trained on $\mathcal{O}\cup\mathcal{O}'$ and the process is iterated. As a consequence, the accuracy of the approximation $\hat{\mathcal{P}}$ of $\mathcal{P}$ is increased.

We tested the approach by running five simulation/train iterations, starting from a data set consisting of 1,000 samples and selecting 5% more samples each time. We started with relatively small values of γ and relatively large values for ν because these combinations of parameters produce very smooth support regions that allow to better isolate "novel" points. In successive iterations, γ was increased while ν decreased using a sort of annealing schedule leading to the final SVM reported in Figure 4. For each selected sample a variable number of new simulation points was generated depending on the value of the decision function (the more negative, the more simulations). The resulting $\hat{\mathcal{P}}_{LNA,5}$ (trained on 3,065 samples, Figure 4) yields excellent performance, displaying features that were not present in SVMs trained with uniformly sampled datasets containing 5,000 samples.

6. PROJECTIONS ON $\mathcal{P}$

Given the nature of top-down flows, the same circuit may be considered at different levels of abstraction in the design process. As a consequence, given an analog circuit $\mathcal{C}$ and its performance relation $\mathcal{P}_{\mathcal{C}}(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^n$, it is often the case that at a high level of abstraction we are interested in a lower-dimensional relation $\mathcal{P}'_{\mathcal{C}}(\boldsymbol{x}')$ with $\boldsymbol{x}' \in \mathbb{R}^{n'}$, $n' < n$. $\boldsymbol{x}'$ is a projection of $\boldsymbol{x}$ that refers to a more abstract view of the circuit. For example, $\boldsymbol{x} = [$*gain, bandwidth, noise, distortion*$]$, $\boldsymbol{x}' = [$*gain, bandwidth*$]$. The formal definition of $\mathcal{P}'(\boldsymbol{x}')$ is:

$$\mathcal{P}'(\boldsymbol{x}') = \begin{cases} 1 & \text{if } \exists \boldsymbol{x}'' \text{ s.t. } \mathcal{P}([\boldsymbol{x}'\boldsymbol{x}'']) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Support Vector Machine	%Train	%Test	$\mathcal{O}$	ν	γ	#SV
Two-Class $p_n(\cdot)$	97.08	92.07	$\mathbb{R}^4$	0.05	20	2,626
Two-Class 0.66 $p_n(\cdot)$ 0.33 $p_c(\cdot)$	95.66	93.06	$\mathbb{R}^4$	0.05	20	2,346
Two-Class 0.66 $p_n(\cdot)$ 0.33 $p_d(\cdot)$	71.12	60.58	$\mathbb{R}^4$	0.05	20	4,873
Two-Class 0.6 $p_n(\cdot)$ 0.2 $p_d(\cdot)$ 0.2 $p_c(\cdot)$.	97.84	91.81	$\mathbb{R}^4$	0.05	20	2,859
One-Class	95.72	85.11	$\mathbb{R}^4$	0.05	20	764
One-Class	95.20	79.67	$\mathbb{R}^7$	0.04	20	929
One-Class prior $c_1(\cdot)$	96.15	84.57	$\mathbb{R}^7$	0.04	20	791
One-Class prior $c_2(\cdot)$	95.97	84.39	$\mathbb{R}^7$	0.04	20	819

Table 2: Summary of SVM performance. The false negative rate on the training set is very good due to the ν-SVM formulation used, with the exception of the third row, which relies on $p_{data}(\cdot)$ to generate random samples (random samples blur the data pattern). All two-class SVMs were trained with 30,000 random negative samples.

When working with SVMs, the problem becomes that of finding $\boldsymbol{x}''$ that makes the decision function positive given $\boldsymbol{x}'$. An equivalent condition is that the maximum of $f(\boldsymbol{x}) = \sum_{Sup.Vec.} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) - \rho$ is positive, which generates a non-linear global optimization problem. However, the decision function is a sum of Gaussian kernels that, in our case, have large γ parameters. Therefore, each point where such a Gaussian is centered (i.e., each support vector) provides a small neighborhood of a possible local optimum. Exploring these points running a simple steepest ascent method from the support vectors, we can keep track of local optima and find the global optimum more rapidly. We have implemented this approach to compute projections. Some heuristics have also been adopted to limit the number of support vectors to be checked for optimality. If we rewrite the decision function as:

$$f([\boldsymbol{x}'\boldsymbol{x}'']) = \sum_i \underbrace{\alpha_i e^{-\gamma\|\boldsymbol{x}'-\boldsymbol{x}'_i\|^2}}_{\tilde{\alpha}_i} e^{-\gamma\|\boldsymbol{x}''-\boldsymbol{x}''_i\|^2}, \qquad (8)$$

then if $\|\boldsymbol{x}' - \boldsymbol{x}'_i\|$ is "large" ($\boldsymbol{x}'$ is fixed) $\tilde{\alpha}_i$ will be very small and so the support vector $\boldsymbol{x}_i$ will not be a good candidate to get to the maximum. Furthermore, if the SVM had a global optimum in the neighbor of $\boldsymbol{x}''_i$ because of $\boldsymbol{x}_i$, then the SVM would classify $\boldsymbol{x}$ based on a support vector whose components $\boldsymbol{x}'_i$ are "far" from $\boldsymbol{x}'$. This would be an inconsistent behavior for an SVM based on RBF modeling $\mathcal{O} = \phi(\mathcal{I})$ under the assumptions made in section 3. The condition $\|\boldsymbol{x}' - \boldsymbol{x}'_i\|_\infty > 0.1$ provides a good heuristic for discarding support vectors. Given a candidate support vector $\boldsymbol{x}_i$ and the smoothness of the decision function, we found that a simple steepest ascent local optimizer suffices for finding the optimum in fewer than 15 iterations. The complexity of the implemented algorithm is (n is the dimensionality of $\mathcal{P}$ and l the number of support vectors) $O(l^2 n)$ for the normal case and $O(l^3 n)$ for the conformal mapping case.

7. CONCLUSIONS

We presented a novel approach for modeling the performance space of an analog circuit based on SVMs. The resulting model provides a clear separation of abstraction levels, directly modeling performance relations in place of regressions on implementation parameters. An efficient projection algorithm allows considering the same circuit at different levels of abstraction using the same underlying model. Therefore, the approach has preferred utilization in top-down design flows and analog platforms. SVMs are trained on simulation data, and false positives are controlled based on a randomized testing procedure. By augmenting the basic one-class SVM to exploit conformal mappings and an active learning methodology, we have obtained a satisfactory solution both in terms of accuracy and of reduction of number of simulations required to model the circuit. Overall we feel that one-class SVMs are quite promising as an approach to finding representations of the performance space and as a component in an IC design system where top-down constraint mapping is molded with bottom-up circuit block characterizations.

8. REFERENCES

[1] A. Sangiovanni-Vincentelli, "Defining platform-based design," *EE-Design*, March 2002.

[2] R. Brayton and A. Hachtel, G.D.and Sangiovanni-Vincentelli, "A survey of optimization techniques for integrated-circuit design," *Proceedings of the IEEE*, vol. 69, pp. 1334–62, October 1981.

[3] S. Director, "The simplicial approximation approach to design centering," *IEEE Transactions on Circuits and Systems*, vol. 24, pp. 363–72, July 1977.

[4] L. Carloni, F. De Bernardinis, A. Sangiovanni Vincentelli, and M. Sgroi, "The art and science of integrated systems design," in *to be presented at ESSCIRC'02*, September 2002.

[5] H. Liu, A. Singhee, R. Rutenbar, and L. R. Carley, "Remembrance of circuits past: Macromodeling by data mining in large analog design spaces," in *Proceedings of DAC*, 2002.

[6] R. Harjani and T. Tibshirani, "Feasibility and performance region modeling of analog and digital circuits," in *Analog Integrated Circtuis and Signal Processing*, Kluwer Academic Publishers, 1996.

[7] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *5th Annual ACM Workshop on COLT* (D. Haussler, ed.), (Pittsburgh, PA), pp. 144–152, ACM Press, 1992.

[8] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," Tech. Rep. MSR-TR-99-87, Microsoft Research, 1999.

[9] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Tech. Rep. MSR-TR-98-14, Microsoft Research, 1998.

[10] T. Joachims, "Making large-scale svm learning practical," in *Advances in Kernel Methods - Support Vector Learning*, MIT Press, 1998.

[11] C. Chang and C. Lin, "Training ν-support vector classifiers: Theory and algoritms," *Neural Computation*, vol. 13, no. 9, pp. 2119–2147, 2001.

[12] S. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, no. 12, pp. 783–789, 1999.

[13] S. Amari and S. Wu, "Conformal transformation of kernel functions: A data-dependent way to improve support vector machine classifiers," *Neural Processing Letters*, no. 15, pp. 59–67, 2002.