Constructive Derivation of Analog Specification Test Criteria

Haralampos-G. D. Stratigopoulos
Electrical Engineering Dept.
Yale University
New Haven, CT 06520-8285
Email: haralampos-g.stratigopoulos@yale.edu

Yiorgos Makris
Electrical Engineering Dept.
Yale University
New Haven, CT 06520-8285
Email: yiorgos.makris@yale.edu

***Abstract*— We discuss the design of a neural system that learns to separate nominal from faulty instances of an analog circuit in a low dimensional measurement space. The key novelty of the proposed system is that it successively establishes a separation hypersurface of order that adapts to the intrinsic complexity of the problem. Thus, it performs excellent classification even in the presence of complex distributions. The test criterion for classifying a circuit is simply the location of its measurement pattern with respect to the separation hypersurface. Despite its simplicity, this criterion is, by construction, strongly correlated to the performance parameters of the circuit and does not rely on fault models.**

I. Introduction

Implicit functional test methods [1], [2], [3], [4], [5] advocate a learning-based approach to the problem of analog circuit testing. Such methods examine whether a circuit meets its specifications without explicitly measuring its performance parameters and without assuming a prescribed fault model that, inevitably, induces a bias [6]. Rather, they rely on a learning process that involves a small set of fully characterized circuit instances, called *training set*. This learning process utilizes a low dimensional discriminative set of measurements obtained on each training instance and culminates in test criteria that correlate these measurements to the performance parameters of the circuit. Once these test criteria are established, the same measurements suffice to examine a new circuit instance with regards to compliance to its specifications. Thus, test application time and cost are significantly reduced, as compared to the traditional functional test method, which is now necessary only to characterize the circuits in the training set. Ultimately, the objective of implicit functional test methods is to construct criteria that classify correctly previously unseen instances, i.e. that exhibit good *generalization* performance.

Towards this direction, the methods proposed in [1], [2], [3] separate the populations of nominal and faulty instances of the training set in a measurement space, $\tilde{x} \in \Re^d$, by a set of M hyperplanes, b_i, $i = 1, .., M$, where M is the number of single-ended specifications of the circuit. The union of these hyperplanes divides the measurement space into regions A^n and A^f, $\bigcup_{i=1}^{M} b_i : \tilde{x} \mapsto \left(A^n, A^f\right)$. A decision on a new circuit instance is made based on the region that its measurement pattern falls in. Circuits that fall into A^n (A^f) are classified as nominal (faulty). However, as we show in the following section, the separation boundary of the two regions is, in general, non-linear[1], and a crude approximation with hyperplanes may result in a high percentage of misclassified circuits. Furthermore, we show that individually optimizing these hyperplanes amplifies the classification error.

In [5], the authors developed a two-layer neural network that potentially can draw separation boundaries of any order. However, the network topology (i.e. the number of hidden units) that generalizes well on new circuit instances is not known *a priori* and can only be identified by a trial and error procedure that requires significant computational effort.

As opposed to [1], [2], [3], in this work, we develop a neural classifier that separates nominal from faulty instances of a circuit in a measurement space, by allocating a single *hypersurface* that can be of arbitrary order, $b : \tilde{x} \mapsto \left(A^n, A^f\right)$. As opposed to [5], the proposed neural classifier is constructed successively, in a way that enables it to adaptively acquire the necessary network connectivity that produces a separation boundary of appropriate order. This particular architecture provably converges to a boundary that yields perfect separation of the circuits in the training set. Once this is achieved, the network is pruned down to the level where it achieves the best generalization on a validation set.

The remainder of this paper is organized as follows. In section II, we provide further motivation for this work and we pinpoint its novel contributions. In sections III and IV, we present the neural network topology and its learning algorithm. In section V, we implement a measurement selection algorithm that finds discriminative sets of measurements within an initial set. Our method is validated on a state variable filter in section VI.

II. Contributions of the proposed method

Fig. 1(a) to (c) illustrate a few distribution examples in a two-dimensional measurement space $x_1 - x_2$. These scatter plots were obtained by performing a realistic Monte Carlo analysis, letting various parameters of the circuit follow a normal distribution centered at their nominal values. Each data

[1]We note that the non-linearity of the decision boundaries was first mentioned in [4], where regression is applied to approximate the non-linear function $\tilde{f} : \tilde{x} \mapsto \tilde{p}$ that maps the measurement pattern, $\tilde{x}$, to the performance parameter vector, $\tilde{p}$.

Proceedings of the 23rd IEEE VLSI Test Symposium (VTS'05)
1093-0167/05 $ 20.00 **IEEE**

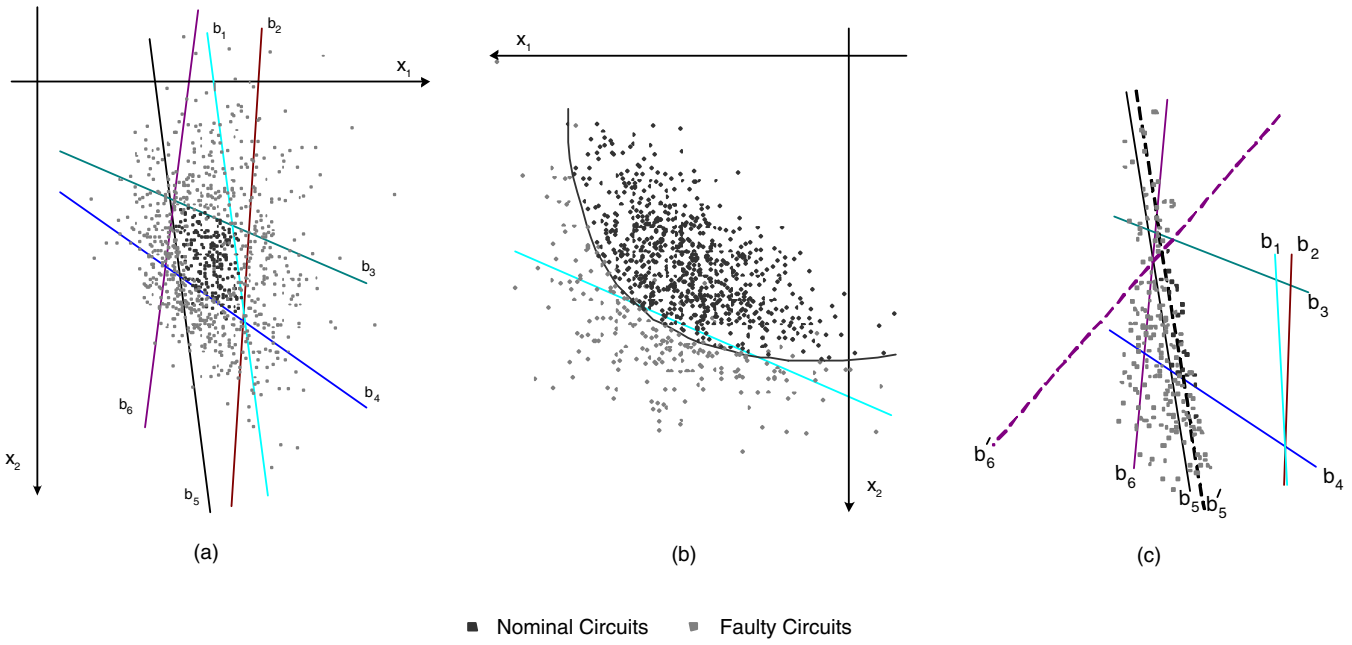

Fig. 1. Distributions of nominal and faulty instances for the state variable filter with $M = 6$ that appears in section VI. (a) Piecewise linear approximation of the acceptance region. (b) Individual boundaries are, in general, non-linear. (c) Allocating boundaries individually induces an error.

point represents the measurement pattern of a Monte Carlo pass.

In Fig. 1(a), instances are labelled as nominal if they satisfy all specifications and faulty otherwise. It can be seen that the acceptance region is bounded by an ellipsoid, such that a single linear boundary would completely fail to separate the two distributions. To overcome this problem, methods in [1], [2], [3] decompose the classification task into M parts, where M is the number of single-ended specifications. In particular, for each single-ended specification, μ, a hyperplane, b_μ, is allocated in the measurement space, such that the maximum possible separation between nominal and faulty instances in the training set (with respect to specification μ) is obtained. In essence, each hyperplane b_μ creates two regions $A^n(\mu)$ and $A^f(\mu)$: instances that fall into $A^n(\mu)$ ($A^f(\mu)$) are classified as nominal (faulty) with respect to specification μ. Therefore, *M two-class separation problems are solved in parallel*. The overall acceptance region is approximated by the union $\bigcup_{\mu=1}^{M} A^n(\mu)$. Since the M individual boundaries are linear, the resulting approximation is piecewise linear, as shown in Fig. 1(a). The misclassification error of this approach can be broken down into two factors.

The *first* factor corresponds to patterns that are misclassified across the acceptance region boundaries due to the insufficiency of the linear approximation. For instance, consider the two populations in Fig. 1(b), whose patterns are labelled with respect to one single-ended specification. The actual decision boundary is non-linear and a simple linear approximation leads to high misclassification.

The *second* factor stems from optimizing the location of each of the M boundaries individually. In particular, each boundary is allocated such that it minimizes misclassification throughout the entire area of patterns. Thus, it also tries to minimize misclassification in areas that are distant from the actual acceptance region. In these areas, correct classification is interpreted as mapping the patterns into the right faulty class. This, however, is unnecessary and, in addition, it degrades the classification ability of the system around the acceptance region boundary, where the unique nominal class is separated from the $2^M - 1$ faulty classes. To view this, consider Fig. 1(c), where Fig. 1(a) is redrawn showing the boundary of the acceptance region and the patterns distributed across boundaries b_5 and b_6 only. It can be seen that boundaries $b_5^{'}$ and $b_6^{'}$ would yield a better classification with regards to all specifications. Yet the system chose b_5 and b_6 in place of $b_5^{'}$ and $b_6^{'}$ because, with respect to individual specifications, b_5 and b_6 provide better classification throughout the measurement space. In short, instead of solving M distinct two-class separation problems in parallel, as in [1], [2], [3], it would be preferable to only tune the boundary around the acceptance region. However, this requires a classifier capable of drawing highly non-linear boundaries (an ellipsoid in the case of Fig. 1(a)).

In this work, we will be describing the architecture of a neural network that employs the instances in the training set to construct a *single* decision boundary that can be of *arbitrary order*.

III. Neural Network Topology

The neural system is trained using N fully characterized circuit instances. Each instance is associated with a boolean variable, t, indicating whether it satisfies all specifications or violates at least one, and a d-dimensional measurement vector,

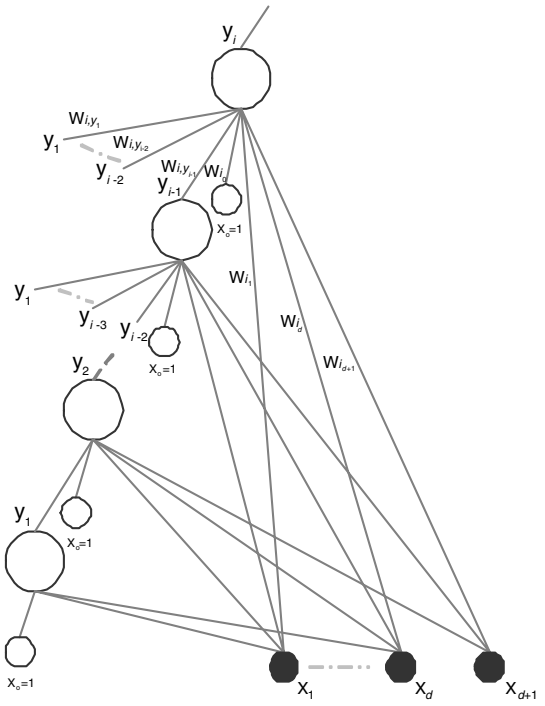

Fig. 2. Topology of the neural network.

$\tilde{x} \in \Re^d$. The training set $\left(\tilde{x}^1, t^1\right), \left(\tilde{x}^2, t^2\right), ..., \left(\tilde{x}^N, t^N\right)$ is used to optimize the adaptive parameters of the neural network. The cost function for this optimization is the classification rate achieved on the circuit instances in the above set.

Since the shape of the acceptance region boundary is not known *a priori*, assuming a fixed network topology limits the range of feasible boundaries. Instead, the proposed neural classifier learns the boundary constructively, starting with the input layer and dynamically adding layers until it matches the intrinsic complexity of the problem at hand. A comprehensive general discussion on constructive algorithms for neural networks can be found in [7].

In particular, the proposed neural classifier is constructed using the *pyramid* algorithm [8] that successively places layers of single neurons above existing ones. The first neuron, y_1, receives inputs from the d measurements. Each successive neuron, y_k, receives inputs from the d measurements and from each neuron below itself. In order for the algorithm to handle the real-valued measurements, each neuron above the first layer also receives an extra attribute that is the projection of the d-dimensional measurement vector onto a parabolic surface:

$$x_{d+1} = \sum_{i=1}^{d} x_i^2 \tag{1}$$

Each newly added neuron takes over the role of the output neuron and the network growth continues until a satisfactory solution for the learning problem is found. The complete architecture of the network is shown in Fig. 2.

The neuron model used herein is an ℓ-input threshold logic unit, also known as *perceptron* [9], that computes the threshold function of the weighted sum of its inputs $\tilde{v}_i \in \Re^\ell$: $y_i(\tilde{x}) = -1$ for $\tilde{w}_i^T \tilde{v}_i \leq 0$ and $y_i(\tilde{x}) = +1$ for $\tilde{w}_i^T \tilde{v}_i > 0$. $\tilde{w}_i^T = [w_{i_0}, w_{i_1}, ..., w_{i_\ell}]$ is the adaptive weight vector and the weight w_{i_0} is referred to as the bias. Here, $\tilde{v}_1 = \tilde{x}$, $\tilde{v}_i = (\tilde{x}, x_{d+1}, y_{i-1}, ..., y_1)$ for $i > 1$ and $\tilde{w}_1^T = [w_{1_0}, w_{1_1}, ..., w_{1_d}]$, $\tilde{w}_i^T = \left[w_{i_0}, ..., w_{i_{d+1}}, w_{i,y_{i-1}}, ..., w_{i,y_1}\right]$ for $i > 1$. If the $+1$ and -1 refer to the pass and fail decisions respectively, then, we want to select weights such that $\tilde{w}_i^T \tilde{v}_i^n > 0$ for all $\tilde{x}^n \in \boldsymbol{C}_n$ and $\tilde{w}_i^T \tilde{v}_i^n < 0$ for all $\tilde{x}^n \in \boldsymbol{C}_f$, where $\boldsymbol{C}_n$ and $\boldsymbol{C}_f$ denote the nominal and faulty classes. Letting t be $t^n = +1$ for $\tilde{x}^n \in \boldsymbol{C}_n$ and $t^n = -1$ for $\tilde{x}^n \in \boldsymbol{C}_f$, the inequality:

$$\left(\tilde{w}_i^T \tilde{v}_i^n\right) t^n > 0 \tag{2}$$

should hold for all measurement patterns, in order for the i-th output neuron to separate the training distributions perfectly. The perceptron has a simple geometrical representation: it divides its input space by a hyperplane, such that the activation function is $y_i = +1$ on one side of the hyperplane, and $y_i = -1$ on the other side. Therefore, at the i-th layer, the populations of nominal and faulty circuits are divided by a boundary, f_i, composed of the set of solutions to equation $\tilde{w}_i^T \tilde{v}_i = 0$. Because of the extra attribute in (1) and the input from the preceding neurons, this boundary is non-linear in the original space of measurements.

This constructive algorithm produces a sequence of boundaries $\{f_n\}$ that converges, in the limit, to the boundary of the acceptance region, f, as closely as desired, i.e. $lim_{n\to\infty}||f - f_n|| = 0$ [8].

Convergence proof For each pattern, $\tilde{x}^p$, define $\varepsilon_p = min_{q\neq p} \sum_{i=1}^{d} \left(x_i^p - x_i^q\right)^2$ and $k = max_{p,q} \sum_{i=1}^{d} \left(x_i^p - x_i^q\right)^2 > \varepsilon_p$. Suppose that pattern $\tilde{x}^p$ is misclassified at the $(i-1)$-th layer, i.e. $y_{i-1}(\tilde{x}^p) = -t^p$. Then, if we select the following weights:

$$\begin{aligned} w_{i_0} &= t^p \left(k + \varepsilon_p - \sum_{j=1}^{d} \left(x_j^p\right)^2\right) \\ w_{i_j} &= 2t^p x_j^p, \;\; j = 1, .., d \\ w_{i_{d+1}} &= -t^p \\ w_{i,y_{i-1}} &= k \\ w_{i,y_j} &= 0, \;\; j = i-2, i-3, ..., 1 \end{aligned}$$

the net input of the i-th neuron is:

$$\begin{aligned} I_{y_i}^p &= w_{i_0} + \sum_{j=1}^{d+1} w_{i_j} x_j^p + \sum_{j=1}^{i-1} w_{i,y_j} y_j(\tilde{x}^p) \\ &= t^p \left(k + \varepsilon_p - \sum_{j=1}^{d} \left(x_j^p\right)^2\right) + \sum_{j=1}^{d} 2t^p \left(x_j^p\right)^2 \\ &\quad -t^p \sum_{j=1}^{d} \left(x_j^p\right)^2 + k y_{i-1}(\tilde{x}^p) \\ &= t^p \varepsilon_p \end{aligned}$$

Since $I_{y_i}^p t^p > 0$, the pattern $\tilde{x}^p$ is correctly classified by the new layer i. Consider now the pattern $\tilde{x}^q \neq \tilde{x}^p$ that is correctly classified at layer $(i-1)$, i.e. $y_{i-1}(\tilde{x}^q) = t^q$. Then,

$$\begin{aligned} I_{y_i}^q &= w_{i_0} + \sum_{j=1}^{d+1} w_{i_j} x_j^q + \sum_{j=1}^{i-1} w_{i,y_j} y_j(\tilde{x}^q) \\ &= t^p \left(k + \varepsilon_p - \sum_{j=1}^{d} \left(x_j^p\right)^2 \right) + \sum_{j=1}^{d} 2 t^p x_j^p x_j^q \\ &\quad -t^p \sum_{j=1}^{d} \left(x_j^q\right)^2 + k y_{i-1}(\tilde{x}^q) \\ &= t^p \left(k + \varepsilon_p - \varepsilon^{'} \right) + k t^q \end{aligned}$$

where $\varepsilon^{'} = \sum_{i=1}^{d} \left(x_i^p - x_i^q\right)^2 > \varepsilon_p$ and, thus, $0 < k^{'} = k + \varepsilon_p - \varepsilon^{'} < k$. Therefore, $I_{y_i}^q t^q = \left(t^p k^{'} + k t^q \right) t^q > 0$, which means that the pattern $\tilde{x}^p$ continues to be classified correctly after the addition of layer i.

The above proof shows the existence of weights that will reduce the misclassification error whenever a new layer is added to the network. In the following section we discuss a training algorithm that finds such weights. Since the number of training patterns is finite, eventual convergence to zero errors is guaranteed.

IV. Learning

In order to minimize circuit misclassification at layer i, equation (2) suggests that we select a weight vector $\tilde{w}_i$ that minimizes the following error function, which is known as *perceptron criterion*:

$$E^{perc}(\tilde{w}_i) = - \sum_{\tilde{x}^n : y_i(\tilde{x}^n) \neq t^n} \left(\tilde{w}_i^T \tilde{v}_i^n \right) t^n \qquad (3)$$

Here, the summation is over all patterns in the training set that are misclassified by the current weight vector $\tilde{w}_i$. The error function is the sum of a number of positive terms and is equal to zero if all patterns are correctly classified. The search in the space of weights is performed by applying the *thermal perceptron learning rule* [10]:

$$w_{i_j}^{(\tau+1)} = w_{i_j}^{(\tau)} + \frac{\alpha}{2} \tilde{v}_j^n \left(t^n - y_i(\tilde{x}^n) \right) e^{\frac{-|y_i(\tilde{x}^n)|}{T}} \qquad (4)$$

where $\alpha > 0$. This corresponds to a simple learning procedure: we cycle through all patterns in the training set and test each pattern in turn using the current set of weight values. If the pattern $\tilde{x}^n$ is correctly classified then we proceed to the next, otherwise, we add $\alpha \tilde{v}_j^n e^{-|y_i(\tilde{x}^n)|/T}$ to the current weight vector if $\tilde{x}^n \in C_n$, or we subtract $\alpha \tilde{v}_j^n e^{-|y_i(\tilde{x}^n)|/T}$ if $\tilde{x}^n \in C_f$. It can be seen that this procedure successively reduces the error in (3) [9].

The exponential tail in (4) controls the correction of weights based on the location of the misclassified pattern $\tilde{x}^n$ with respect to the decision boundary. $|y_i(\tilde{x}^n)|$ is a measure of this distance. In turn, the temperature T controls how strongly the changes are attenuated for large values of $|y_i(\tilde{x}^n)|$. As an intuition, one can imagine a zone surrounding the decision boundary. The boundary moves only if an erroneously classified pattern falls within this zone. The temperature is annealed from an initial value T_o to 0, causing a gradual reduction of the extent of the sensitive zone. In the limit of $T \to 0$, the zone disappears altogether and the perceptron is stable, i.e. its training has been completed. Best results are obtained when α is reduced from 1 to 0 at the same time as T is. The reader is referred to [10] for the rationale supporting this approach.

The thermal learning rule outperforms other known perceptron based learning algorithms [11], provided that the temperature is chosen appropriately. In particular, T should be of the same order of magnitude as the range of values of $y_i(\tilde{x}^n)$. We followed the suggestion in [12]. T decreases from T_o (initially 1) to 0 during 500 cycles through the training set. Since $y_i(\tilde{x}^n)$ might vary considerably for different instances, we calculate the average value of $|y_i(\tilde{x}^n)|$ over the set of instances, $\langle |y_i(\tilde{x}^n)| \rangle_n$, over each cycle. At the end of each cycle, T_o is set to $T_o = \left(2T_o + 2\langle |y_i(\tilde{x}^n)| \rangle_n\right)/3$. The temperature T is then set to γT_o, where γ (initially 1) decreases linearly with each cycle to reach 0 after 500 cycles. α is set to $0.1T/T_o$.

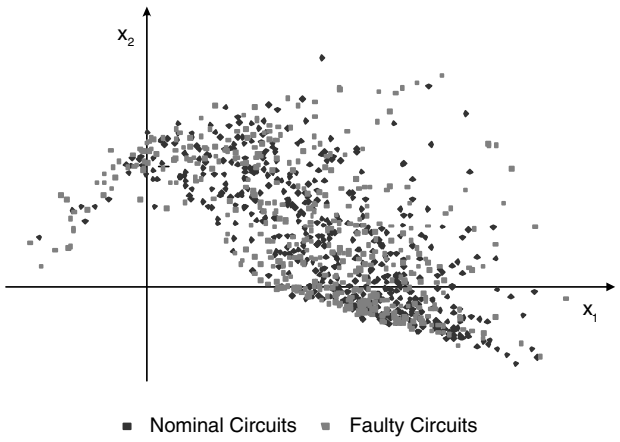

Fig. 3. Adding measurement x_2 does not provide additional discrimination ability over x_1. Both measurements are samples of the response of the state variable filter when it is configured to oscillate [13].

V. Selection of measurements

Increasing the dimensionality of the measurement space does not necessarily improve classification. For instance, the added measurement may be strongly correlated to the current set of measurements. This is illustrated in Fig. 3 where, in essence, the two populations fall upon each other. Moreover, increasing the dimensionality of the measurement space may lead to the point where the distributions are very sparse, in which case the boundary representation will be poor. This phenomenon, which has been termed as *curse of dimensionality* [9], will cause an adverse effect on the generalization performance. Therefore, the separation problem should be solved in a measurement space that is both highly-discriminative and low-dimensional.

The problem of selecting the most effective $d^{'}$ measure-

TABLE I

THE STATE VARIABLE FILTER HAS SIX SINGLE-ENDED SPECIFICATIONS.

Performance parameter	Specification limits
Dc gain (in V)	$0.9 - 1.1$
Maximum gain (in V)	$1.0 - 1.3$
f_{3db} (in KHz)	$1.83 - 2.23$

ments from a given set of d measurements, $d^{'} < d$, is called *feature selection*. We implemented a method called *floating search* [14]. A comparative study [15] shows that this method yields near-optimal results, yet is much faster than other feature selection approaches based on branch-and-bound algorithms.

The algorithm uses two basic procedures, namely the *sequential forward selection* (SFS) and the *sequential backward selection* (SBS). Given a subset of measurements, SFS selects from the remaining measurements and includes the one that is the most significant with respect to this subset. Similarly, SBS selects from the current subset of measurements and excludes the one that is the least significant with respect to this subset. Each measurement subset, X_i, is evaluated on the training set by the Fisher criterion [9], which examines class separation by projecting the two populations on the one-dimensional space that maximizes the distance of their means and minimizes the within-class scatter:

$$J\left(X_i\right) = \max_{\tilde{u}} \frac{\tilde{u}^T S_B \tilde{u}}{\tilde{u}^T S_W \tilde{u}} \tag{5}$$

where S_B and S_W are the between-class and within-class covariance matrices respectively. The criterion is calculated with respects to each single-ended specification separately and is then averaged over all specifications. This is necessary in order to account for distributions with equal means, such as the one in Fig. 1(a). A subset, X_i, is deemed better than another subset, X_j, if and only if $J\left(X_i\right) > J\left(X_j\right)$. The algorithm is a bottom-up search procedure which starts with an empty feature set and includes new features by means of applying the basic SFS procedure. The feature inclusion phase is followed by a series of successive conditional exclusions of the worst feature in the newly updated set through the SBS procedure, provided that further improvement can be made to previous sets of lower cardinality. Upon termination, the algorithm reports the best identified subsets of $d^{'}$ measurements for every $d^{'} \in \{1, ..., d-1\}$.

VI. SIMULATION RESULTS

In this section, the proposed classification method is evaluated on a state variable filter [16]. The list of specifications is given in Table I. The course of the experiment is as follows:

1) Our experiment starts with an initial set of thirty measurements. We obtained this set by applying a realistic statistical simulation of the circuit and identifying test stimuli responses (and their respective sampling times) which are the most sensitive to process variations [17]. Various types of measurements are examined in this analysis: *ac*, *dc*, impulse response, frequency and magnitude of oscillation [13]. The generalization performance of our classifier is bounded by the discrimination ability of this initial set of measurements.
2) We then obtain the circuit instances that constitute the training and the validation set for the classifier. In a production environment, the training set will comprise a small number of circuits that will be exhaustively tested through traditional specification test and on which the selected measurements will be carried out. For the purpose of this experiment, we generate these circuit instances through a Monte Carlo simulation with 3000 runs. One third of these instances is used as the training set and the rest as the validation set. Parameters are modelled by a normal distribution centered at their nominal value with a deviation of 10%.
3) The vector $(\tilde{x}^n, t^n)$ is computed for each circuit instance n and is subsequently normalized. Normalization is required for the comparison of different subsets of measurements through the criterion in (5) and, moreover, it speeds up in practice the training phase of the classifier.
4) The normalized training patterns serve as input to the feature selection algorithm, which searches for efficient measurement subsets. Upon completion, the algorithm reports the best subset of $d^{'}$ measurements that it has identified for every $d^{'} \in \{1, ..., 29\}$.
5) The validation set is split in two equal sets, namely the *hold-out set* and the *test set*. For each best subset of $d^{'}$ measurements, and for every $d^{'} \in \{1, ..., 30\}$, we train the neural network and, at each layer, we record the classification rate on the training and the hold-out sets. The growth of the network stops when all training patterns are classified correctly or when it reaches the 100-th layer. At the end of training, we prune the network down to the layer that has the highest generalization on the hold-out set. The generalization ability of the classifier is assessed at this layer by measuring its accuracy on the test set. It is important to note that the test set is unbiased since it is not used at all during training.
6) For comparison purposes, we also implemented the two-layer neural network in [2], [3] that partitions the measurement space with individual hyperplanes, one for each single-ended specification, and then obtains their union to approximate the acceptance region. The neural network was trained using the thermal perceptron rule for each best subset of $d^{'}$ measurements and for every $d^{'} \in \{1, ..., 30\}$. At the end of training, the network reports the classification rate on the training and the validation sets obtained in step 2 (half of the validation set is used here). We note that, in practice, these rates are similar to the ones achieved by the classifier in [1] since the latter also considers the union of individual hyperplanes. In the following, [1], [2], [3] are referred to as *linear* methods.

Table II shows the classification rate achieved on the training and validation sets for the proposed classifier and for the linear methods. The experiment illustrates the following:

1) Upon completion of construction, the proposed classifier achieves almost perfect classification on the training set. This is expected since the final boundary can be of arbitrary order. The respective classification rates achieved by the linear methods are lower due to the limited flexibility of the hyperplanes. The rate in brackets in the fourth column refers to the accuracy of training at the layer where the best generalization on the hold-out set is observed. As a result of pruning, this rate is no longer close to 100%.
2) The proposed network achieves the highest generalization rate for the best subset of three measurements (94.9% of new instances are classified correctly), which is considerably higher than the maximal generalization rate for linear methods, achieved for the best subset of ten measurements (89.3% of new instances are classified correctly). The underlying reasons that the linear methods misclassify a larger number of circuits were discussed in section II.
3) The classification rate on the training set increases monotonically with the number of measurements, i.e. $J(X^+) \geq J(X)$, where X and X^+ denote sets of measurements, with $|X| \leq |X^+|$. However, for the selected best subsets of measurements, monotonicity is not satisfied on the validation set. In particular, regarding the proposed classifier, the classification rate on the validation set presents a peak, which verifies the existence of the curse of dimensionality. Regarding the linear methods, the classification rate on the validation set oscillates for subsets of cardinality larger than four. We attribute this to the fact that the validation peak occurs at subsets of different cardinalities for each of the M two-class problems. Thus, the resulting validation curve is not convex. This manifestation of the curse of dimensionality suggests that, for this type of test methods, an efficient combination of measurements is desired, rather than a large set.

VII. Conclusion

We designed a neural system that learns test criteria for analog circuits through a small set of fully characterized circuit instances. These test criteria translate into a single hypersurface in a low dimensional measurement space. We demonstrated that a single hypersurface offers great advantages over the union of distinct hyperplanes used in previously reported methods. A circuit is tested by examining the location of its measurement pattern with regards to the hypersurface. Since the latter is learned through fully characterized circuit instances, this simple test is strongly correlated to the circuit specifications and does not depend on underlying fault-model assumptions.

TABLE II

Classification rates for the state variable filter.

	Linear Methods		Proposed Constructive Method		
# of meas.	train. set	valid. set	train. set (pruned net.)	test set	net. size (in layers)
1	41.2	41.5	86.0 (82.4)	79.8	54
2	74.6	72.4	98.5 (93.4)	91.8	26
3	84.5	83.6	99.5 (97.3)	94.9	10
4	90.2	88.2	99.5 (92.1)	92.1	15
5	90.7	87.2	99.9 (94.8)	90.9	12
6	91.0	88.8	99.8 (90.0)	87.9	10
7	91.2	87.6	99.9 (91.2)	85.0	11
8	91.2	88.6	99.7 (88.9)	84.7	5
9	91.3	88.8	99.8 (86.8)	84.6	2
10	91.6	89.3	99.8 (86.8)	84.6	5

References

[1] W. M. Lindermeir, H. E. Graeb, and K. J. Antreich, "Analog testing by characteristic observation inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 9, pp. 1353–1368, 1999.

[2] C. Y. Pan and K. T. Cheng, "Test generation for linear time-invariant analog circuits," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 46, no. 5, pp. 554–564, 1999.

[3] P. N. Variyam and A. Chatterjee, "Specification-driven test generation for analog circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 10, pp. 1189–1201, 2000.

[4] P. N. Variyam, S. Cherubal, and A. Chatterjee, "Prediction of analog performance parameters using fast transient testing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 3, pp. 349–361, 2002.

[5] V. Stopjakova, P. Malosek, D. Micusik, M. Matej, and M. Margala, "Classification of defective analog integrated circuits using artificial neural networks," *Journal of Electronic Testing*, vol. 20, pp. 25–37, 2004.

[6] S. Sunter and N. Nagi, "Test Metrics for Analog Parametric Faults," in *IEEE VLSI Test Symposium*, 1999, pp. 226–234.

[7] V. Honavar and L. Uhr, "Generative learning structures and processes for generalized connectionist networks," *Information Sciences*, vol. 70, pp. 75–108, 1993.

[8] R. Parekh, J. Yang, and V. Honavar, "Constructive neural-network learning algorithms for pattern classification," *IEEE Transactions on Neural Networks*, vol. 11, no. 2, pp. 436–451, 2000.

[9] C. M. Bishop, *Neural Networks for Pattern Pecognition*, Oxford University Press, 1995.

[10] M. Frean, "A "thermal" perceptron learning rule," *Neural Computation*, vol. 4, pp. 946–957, 1992.

[11] S. I. Gallant, "Perceptron-based learning algorithms," *IEEE Transactions on Neural Networks*, vol. 1, no. 2, pp. 179–191, 1990.

[12] N. Burgess, "A constructive algorithm that converges for real-valued input patterns," *International Journal of Neural Systems*, vol. 5, no. 1, pp. 59–66, 1994.

[13] K. Arabi and B. Kaminska, "Oscillation-test methodology for low-cost testing of active analog filters," *IEEE Transactions on Instrumentation and Measurement*, vol. 48, no. 4, pp. 798–806, 1999.

[14] P. Pudil, J. Novovicova, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, pp. 1119–1125, 1994.

[15] A. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance," *IEEE Transactions on Pattern Analysis an Machine Intelligence*, vol. 19, no. 2, pp. 153–158, 1997.

[16] A. Raghunathan, H. J. Shin, J. A. Abraham, and A. Chatterjee, "Prediction of analog performance parameters using oscillation based test," in *IEEE VLSI Test Symposium*, 2004, pp. 377–382.

[17] Z. Jaworski, M. Niewczas, and W. Kuzmicz, "Extension of inductive fault analysis to parametric faults in analog circuits with application to test generation," in *IEEE VLSI Test Symposium*, 1997, pp. 172–176.