Chapter 15
Sparse Relevance Kernel Machine-Based Performance Dependency Analysis of Analog and Mixed-Signal Circuits

Honghuang Lin, Asad Khan, and Peng Li

15.1 Introduction

As the complexity of *analog/mixed-signal* (AMS) circuits keeps increasing at a rapid pace, the tasks of design, verification, and test have become significant challenges. Nevertheless, it is essential to characterize the dependencies of circuit performances/specifications on various circuit and device parameters or test signatures for purposes such as design, verification, and test optimization. However, doing so is not trivial since the targeted dependencies are usually complex and nonlinear with deep-rooted correlations, making it arduous to reliably quantify the importance of numerous parameters.

For characterizing sophisticated circuit systems, machine learning techniques based on circuit simulations or measurements have been proven to be effective and produced promising outcomes. For example, *support vector machines* (SVMs) [34] are used as nonlinear classifiers in [19] to capture the mapping from input parameters to circuit performance. A regression extension to SVM is employed in [1] to rank circuit parameters based on their correlations with unexpected timing deviations. Additionally, Bayesian inference is often used to build statistical circuit models. For instance, a co-learning Bayesian model is proposed in [36] to efficiently model the performance of AMS circuits.

However, building machine learning-based circuit models faces two key challenges: (1) the availability of training data is limited, since circuit simulations or silicon measurements are usually expensive for sophisticated AMS systems; and (2) to describe an AMS system, it needs a huge number of analog signals and design/device parameters, leading to an extremely high dimensional problem. Since

H. Lin · A. Khan · P. Li (✉)
Texas Instruments, Dallas, TX, USA
e-mail: h-lin10@ti.com; a-khan1@ti.com; pli@tamu.edu

© Springer Nature Switzerland AG 2019

I. M. Elfadel et al. (eds.), *Machine Learning in VLSI Computer-Aided Design*,
https://doi.org/10.1007/978-3-030-04666-8_15

the sensitivities of circuit performances to various parameters may vary vastly, it is instrumental to reliably analyze circuit parameter criticality during the extraction of accurate circuit models. While traditional feature selection or importance ranking techniques may help to identify and select some important parameters out of a large parameter set, building models only with the selected parameters usually degrades the model performance and few of those techniques can guide the model to achieve higher accuracy. These difficulties present important roadblocks to analog/mixed-signal circuit characterization and performance dependency analysis with machine learning techniques.

In the machine learning domain, traditionally, feature selection [3, 5, 9] may be performed by combinatorial search [27, 35] to incrementally add or remove features from the selected subset, which is evaluated by the performance of its resulting predictor. Another kind of method [2, 24, 29] trims the feature space with regularization-based methods, like introducing new regularization terms into the cost function, to perform the feature selection. All these techniques are considered as *linear* feature selection methods since they handle or formulate the features in a linear manner, such as combinatorial search or L_1 norm regularization. To capture the nonlinear dependencies among features and their nonlinear "relevancy" to the targets, some other methods [6, 10, 17] switch the roles of features and samples in their learning models and apply the kernel method to the features instead of samples. These kinds of methods also belong to the category of regularization-based methods since their optimization models using Euclidean inner product as their kernel functions are equivalent to Lasso regression [29] based on the conclusion provided by Li et al. [17]. A drawback of such methods is that their results usually only improve the regularization of the learning model but not the accuracy, since they work independently as preceding filters of the training process.

Moreover, in complex scenarios, especially in complex circuit applications, where features' relevancy may vary in a large range, proper weighting of the features may improve the learning quality by balancing the impact of features on target. In this sense, feature selection is just a 0–1 binary weighting scheme, whose capability is limited when it comes to these scenarios. The task of assigning weights directly to the features and embed the weighted samples into a learning model is extremely computationally challenging, since most commonly used learning models are nonlinear, making weights difficult to manipulate and costly to optimize in the training process.

To address the above challenges, this work proposes a novel Bayesian learning framework for characterizing analog circuits with sparse statistical regression/classification models. The proposed framework is named *sparse relevance kernel machine* (SRKM) and can be considered as a significant extension to the SVM [34] and *relevance vector machine* (RVM) [30]. Instead of directly manipulating the features in a traditional way, the original kernel function is "atomized" into bilinear terms with weighting factors that obliquely reflect the relevancy of the features, leading to the newly defined feature kernel described in the next section. Then, the training model of the SRKM is developed following the RVM framework to achieve a sparse model for both regression and classification. The

SRKM simultaneously seeks relevant training samples (i.e., vectors) and parameters (i.e., features) to derive a sparse model in both the vector and parameter spaces. As a result, the SRKM not only produces accurate models learned from a moderate amount of simulation or measurement data, but also computes a probabilistically inferred weighting factor quantifying the criticality of each parameter as part of the overall learning framework, hence offering a powerful enabler for variability modeling, failure diagnosis, and test development. In addition, an iterative algorithm is developed for efficient training of the proposed SRKM.

The proposed SRKM is capable of solving both classification and regression problems. Compared to other popular kernel-based learning techniques, the SRKM produces more accurate models, requires less amount of training data, and extracts more reliable parametric ranking. The effectiveness of SRKM is demonstrated in examples including statistical variability modeling of a *low-dropout regulator* (LDO), *built-in self-test* (BIST) development of a charge-pump *phase-locked loop* (PLL), and applications of building statistical variability models for a commercial automotive interface design.

This chapter is organized as follows: In Sect. 15.2, we propose the feature kernel weighting scheme and learning model based on the kernel methods adopted in other SVM-related techniques. Then, in Sect. 15.3, the learning model of the SRKM is developed following the sparse Bayesian learning framework. An iterative efficient algorithm is developed to remedy the additional complexity that stems from the inclusion of more variables. Lastly, to illustrate the aforementioned advantages of the SRKM, details and results of three experiments are provided in Sect. 15.4.

15.2 Feature Kernel Weighting

Learning models of circuits are usually defined as: assuming that there are F circuit parameters of interest with which the circuit is described by a parameter (feature) vector $\mathbf{x}$, a sample of the circuit is defined by a pair $\{\mathbf{x}_i, t_i\}$ where t_i is the circuit performance under the configuration $\mathbf{x}_i$. By collecting a number of N samples, the objective of the learning task is to capture the mapping $\Psi : \mathbf{x} \rightarrow t$ with a function $y(\mathbf{x})$ whose output can be used as a prediction of the performance t. If the performance t is a quantified value, this falls into the regression category in machine learning. If t is a binary label, for example, *pass* or *fail* in the context of verification or test, then the learning is a binary classification task. Both regression and classification can be solved by kernel machines such as support vector machines.

15.2.1 Kernel Methods

Support vector machines (SVMs) [25, 33] have been widely used in the EDA domain as a powerful supervised learning toolbox solving classification and regres-

sion problems. According to a recent experiment conducted by Fernández-Delgado et al. [8], the comparison of 179 classifiers evaluated on 121 data sets shows that SVM is still among the top learning methods.

The excellence of SVM mainly relies on two portions of its model: the cost function and the famous "kernel trick." Similar to some other learning methods, the cost function of SVM is composed of a fitting loss term and a smoothing penalty term. By using different formulas for losses and penalties, several variants of SVM [12, 13, 26, 28] are derived from the original quadratic optimization problem. Such composition of cost functions provides exceptional robustness and regularization [39]. On the other hand, kernel trick or kernel method has shown great success in handling nonlinear problems.

By using the Lagrange multipliers, the exploration of the optimal separating hyperplane in the mapped higher dimensional space can be expressed as the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{minimize}} \quad & f_{\text{SVM}}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \hat{\mathbf{x}}_\mathbf{i}, \hat{\mathbf{x}}_\mathbf{j} \rangle - \sum_{i=1}^{N} \alpha_i, \\ \text{subject to} \quad & \sum_{i=1}^{N} y_i \alpha_i = 0, \\ & 0 \le \alpha_i \le C, \forall i. \end{aligned} \tag{15.1}$$

where y_i denotes the binary label of the training sample $\mathbf{x}_\mathbf{i}$, and $\hat{\mathbf{x}}_\mathbf{i}$ represents a projection of $\mathbf{x}_\mathbf{i}$ in a higher dimensional space.

In SVM, the idea of solving nonlinear problems is to map the original input vectors from the *input space*, which are not linearly solvable, into a higher dimensional space and explore a linear solution in that space. It is clearly shown in (15.1) that the optimization problem can be solved by merely knowing the inner product of any pair of the mapped input vectors $\langle \hat{\mathbf{x}}_\mathbf{i}, \hat{\mathbf{x}}_\mathbf{j} \rangle$, without explicitly defining the mapping $\mathbf{x} \rightarrow \hat{\mathbf{x}}$. Therefore, defining a kernel function K that satisfies or represents $K(\mathbf{x}_\mathbf{i}, \mathbf{x}_\mathbf{j}) = \langle \hat{\mathbf{x}}_\mathbf{i}, \hat{\mathbf{x}}_\mathbf{j} \rangle$ and substituting it into the optimization problem should achieve the same results as what is produced by making the mapping $\mathbf{x} \rightarrow \hat{\mathbf{x}}$. As long as the kernel function K satisfies Mercer's condition [33], there exists a feature space where the inner product is generated by K. In this light, a kernel function can be considered as an implicit definition of a certain mapping.

15.2.2 Weighting via Atomized Kernel

Although the implicit mapping defined by kernel methods is very powerful in solving nonlinear problems, similar to other learning techniques, it may easily be confused by irrelevant or redundant features, which are commonly seen in circuit

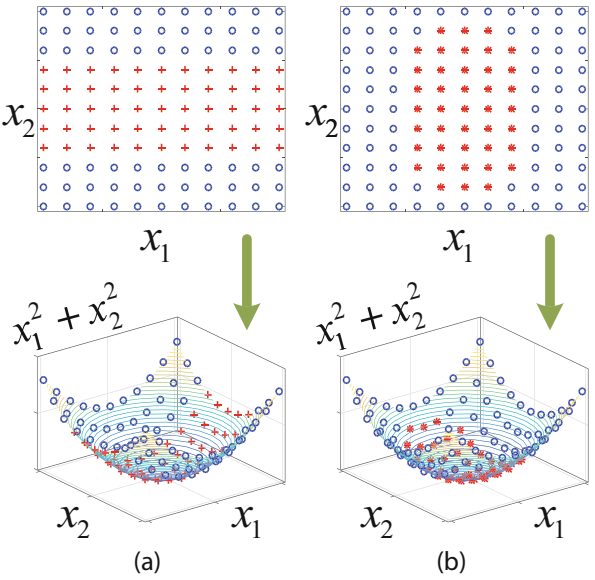


Fig. 15.1 Examples that cannot be linearly solved by the kernel function corresponding to the mapping $(x_1, x_2) \rightarrow (x_1, x_2, x_1^2 + x_2^2)$, where blue circles and red stars denote two different classes, respectively. (**a**) x_2 is the only relevant feature while x_1 is purely irrelevant. (**b**) x_1 is the dominant feature while x_2 provides subtle information to reflect the classes

applications. For example, in Fig. 15.1, if we only sample the center and the four corners, all the samples can be linearly separated by using the mapping $(x_1, x_2) \rightarrow (x_1, x_2, x_1^2 + x_2^2)$, which is equivalent to using kernel function $K(\mathbf{x}_a, \mathbf{x}_b) = \mathbf{x}_a^T \mathbf{x}_b + \mathbf{x}_a^T \mathbf{x}_a \mathbf{x}_b^T \mathbf{x}_b$. But if we keep sampling more evenly in the 2-D input space to include those small circles and asterisks in the training set, it's possible that, as Fig. 15.1 shows, one of the two features is actually irrelevant to the target, or features may have quite different relevancy. In both scenarios, by using the same kernel function or its equivalent mapping, the mapped input vectors can no longer be linearly solved in the 3-D feature space.

From another perspective, kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ is usually viewed as a representation of the covariance or similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. If $\mathbf{x}_i$ and $\mathbf{x}_j$ contain notable amount of noise or redundant features, the kernel function may not be able to correctly reflect the covariance or similarity. For example, Gaussian kernel, also known as *radial basis function* (RBF) and defined as $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, is a commonly used kernel function. Any noisy redundant feature included in $\mathbf{x}_i$ and $\mathbf{x}_j$ will directly affect the term $\|\mathbf{x}_i - \mathbf{x}_j\|^2$, leading to a kernel function value that cannot truly reflect the similarity of the two samples. Such vulnerability is critical and a flexible weighting scheme is needed especially when there are only limited number of samples.

To approach such problems, based on the implicit mapping mechanism, we propose to construct a new kernel by atomizing the existing kernel functions to achieve implicit feature weighting. Considering the examples shown in Fig. 15.1, to get rid of the interference from the redundant or much less relevant feature, the mapping can be further extended into a 6-dimensional space:

$$\hat{\mathbf{x}} = (x_1, \epsilon x_1, x_2, \epsilon x_2, x_1^2 + \epsilon x_2^2, \epsilon x_1^2 + x_2^2)^T,$$

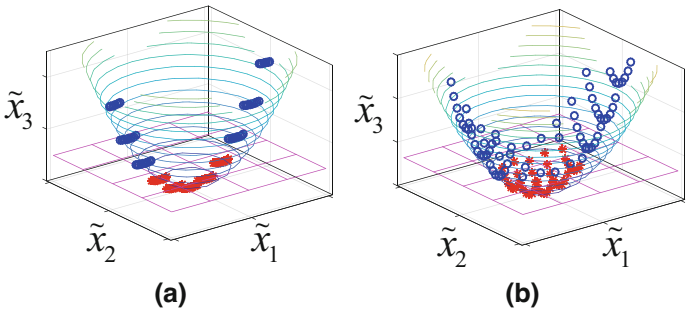


Fig. 15.2 3-D projections of the new 6-D mapping where the two examples in Fig. 15.1 are linearly solved by the purple planes: (**a**) $\tilde{x}_1 = x_2 + \epsilon x_1$, $\tilde{x}_2 = 0$, and $\tilde{x}_3 = x_2^2 + \epsilon^2 x_1^2$; (**b**) $\tilde{x}_1 = x_1 + \epsilon x_2$, $\tilde{x}_2 = \upsilon(x_2 + \epsilon x_1)$, and $\tilde{x}_3 = (x_1^2 + \epsilon^2 x_2^2) + \upsilon(x_2^2 + \epsilon^2 x_1^2)$, where υ is a small positive weight

where a small ϵ can weaken the interference of one feature to the other in the last two dimensions of $\hat{\mathbf{x}}$. As shown in Fig. 15.2a and b respectively corresponding to Fig. 15.1a and b, the 6-D space is linearly projected to 3-D spaces for the purpose of illustration, and the mapped samples are linearly solvable in the new 3-D spaces, meaning linear solutions exist in spaces which are linearly transformed from the 6-D space defined by $\hat{\mathbf{x}}$.

Since implicit mappings defined by kernel functions are more favorable, for the exact mapping $\mathbf{x} \rightarrow \tilde{\mathbf{x}}$ defined above, its corresponding kernel function is:

$$\begin{aligned}\tilde{K}(\mathbf{x_a}, \mathbf{x_b}) =& (\mathbf{x_a}^{(1)})^T \mathbf{x_b}^{(1)} + (\mathbf{x_a}^{(1)})^T \mathbf{x_a}^{(1)} (\mathbf{x_b}^{(1)})^T \mathbf{x_b}^{(1)} \\ &+ (\mathbf{x_a}^{(2)})^T \mathbf{x_b}^{(2)} + (\mathbf{x_a}^{(2)})^T \mathbf{x_a}^{(2)} (\mathbf{x_b}^{(2)})^T \mathbf{x_b}^{(2)},\end{aligned}$$

where $\mathbf{x}^{(1)} = diag(1, \epsilon) \cdot \mathbf{x}$ and $\mathbf{x}^{(2)} = diag(\epsilon, 1) \cdot \mathbf{x}$. By substituting the original kernel function K into the new kernel, it can be re-written as:

$$\begin{aligned}\tilde{K}(\mathbf{x_a}, \mathbf{x_b}) =& K(diag(1, \epsilon) \cdot \mathbf{x_a}, diag(1, \epsilon) \cdot \mathbf{x_b}) \\ &+ K(diag(\epsilon, 1) \cdot \mathbf{x_a}, diag(\epsilon, 1) \cdot \mathbf{x_b}),\end{aligned}$$

where the first term and second term are original kernels with vectors scaled by $diag(1, \epsilon)$ and $diag(\epsilon, 1)$, respectively. In other words, each terms scale down one feature with a small ϵ and map $\mathbf{x}$ to a portion of the dimensions in $\tilde{\mathbf{x}}$ which are insensitive to that feature.

More generally, we define a scaling diagonal matrix $\mathbf{S_i}(\epsilon)$ with:

$$\mathbf{S_i}(\epsilon) = diag(\mathbf{s_i}) \tag{15.2}$$

and

$$\mathbf{s_i}(j) = \begin{cases} 1, & j = i, \\ \epsilon, & j \neq i, \end{cases}$$

where $\epsilon \in [0, 1]$. For any existing kernel function $K(\mathbf{x_a}, \mathbf{x_b})$, we define a new *feature kernel* function as:

$$K_i(\mathbf{x_a}, \mathbf{x_b}; \epsilon) = K(\mathbf{S_i}(\epsilon) \cdot \mathbf{x_a}, \mathbf{S_i}(\epsilon) \cdot \mathbf{x_b}). \tag{15.3}$$

Such feature kernel mainly maintains the sensitivity to the i-th feature in both $\mathbf{x_a}$ and $\mathbf{x_b}$, with other features scaled by ϵ. If $\epsilon = 1$, such feature kernel is identical to the original kernel. If $\epsilon = 0$, the i-th feature is completely isolated since all the other features are zero out from the original kernel.

Assuming that kernel K maps samples from an F-dimensional input space to a d-dimensional space, which may be vulnerable to irrelevant or nonlinearly relevant features, we now atomize K into the sum of F weighted feature kernels by assigning one for each feature:

$$\tilde{K}(\mathbf{x_a}, \mathbf{x_b}; \epsilon) = \sum_{i=1}^{F} \upsilon_i K_i(\mathbf{x_a}, \mathbf{x_b}; \epsilon). \tag{15.4}$$

It will result in a mapping from the F-dimensional input space to another $(d \cdot F)$-dimensional space. In this much higher dimensional space, we are expecting that the inner product of any pair of vectors can be expressed or approximated by a linear combination of the feature kernels.

In addition, for the i-th feature kernel K_i, it mainly represents the information provided by the i-th feature since the influence of other features in the corresponding "axes" of the mapped space is scaled down if $\epsilon < 1$ or completely removed if $\epsilon = 0$. Therefore, we propose to perform feature weighting via the weighting parameters υ_i as demonstrated in (15.4). Larger $|\upsilon_i|$ means the i-th kernel is more important in the kernel model. For example, by using $\epsilon = 0.1$ as the scaling parameter, in Fig. 15.2a, the kernel function corresponding to the projection is $\tilde{K} = 0 \cdot K_1 + K_2$ while in Fig. 15.2b, it is $\tilde{K} = K_1 + \upsilon \cdot K_2$ where $\upsilon = 3/8$. These two atomized kernel functions clearly reflect the relevancy of the two features.

One of the advantages of this weighting scheme is that the weighting parameters are much easier to manipulate compared to the schemes that directly apply weighting parameters to the input vectors. Secondly, the parameter ϵ makes the model more flexible by smoothly morphing from regular kernels ($\epsilon = 1$) to feature selection ($\epsilon = 0$). Moreover, similar to the original kernel method, this weighting scheme actually avoids defining explicit feature weighting by instead weighting the linear combination of the feature kernels, which can be considered as an implicitly defined nonlinear weighting scheme.

For an existing kernel function K that satisfies Mercer's condition [33], the newly defined feature kernel K_i should also satisfy Mercer's condition. As a result, after $\mathbf{x}$ is normalized, for all square integrable $g(\mathbf{x})$ we have:

$$\int_{\chi\times\chi} \tilde{K}(\mathbf{x_a}, \mathbf{x_b}; \epsilon) g(\mathbf{x_a}) g(\mathbf{x_b}) d\mathbf{x_a} d\mathbf{x_a} = \sum_{i=1}^{F} \int_{\chi\times\chi} v_i K_i(\mathbf{x_a}, \mathbf{x_b}; \epsilon) g(\mathbf{x_a}) g(\mathbf{x_b}) d\mathbf{x_a} d\mathbf{x_a} \geq 0,$$

for all $\mathbf{x_a}, \mathbf{x_b} \in \chi$ as long as $v_i \geq 0, \forall i$.

15.2.3 Learning Model with Feature Kernels

In SVM, the training model is often solved in its dual form and, by leveraging *Karush–Kuhn–Tucker* (KKT) conditions, the prediction model is based upon the following decision function:

$$y(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{N} w_i K(\mathbf{x}, \mathbf{x_i}), \tag{15.5}$$

where $\{\mathbf{x_i}\}_{i=1}^{N}$ are the training examples and w_i are actually the Lagrange multipliers referred to as α_i in (15.1). The training process of learning methods using (15.5) as their decision function is to infer all the parameters w_i in (15.5) given the corresponding targets $\{t_i\}_{i=1}^{N}$ of the training examples $\{\mathbf{x_i}\}_{i=1}^{N}$.

Taking the error e_i between t_i and $y(\mathbf{x_i}; \mathbf{w})$ into consideration, the kernel-based machine can be written as:

$$\mathbf{t} = \Phi_w \cdot \mathbf{w} + \mathbf{e}, \tag{15.6}$$

where Φ_w is an $N \times N$ matrix defined by $\Phi_w(i, j) = K(\mathbf{x_i}, \mathbf{x_j})$, $\mathbf{t}$ is simply the target vector with $\mathbf{t}(i) = t_i$, and $\mathbf{e}$ is the error vector for the N training samples. Learning methods like SVM aim at minimizing $\mathbf{e}$ with fitting loss term and regularizing $\mathbf{w}$ with smoothing regularization term simultaneously.

As previously mentioned, most popular kernel functions such as Gaussian kernel (also known as *radial basis function*) and polynomial kernel are vulnerable to features with complicated relevancy. For training data with F features, we define a new learning model by embedding the sum of weighted feature kernels as a new kernel function into (15.6) to handle feature relevancy:

$$\mathbf{t} = \Phi_{wv}{}^{(\epsilon)} \cdot (\mathbf{w} \otimes \mathbf{v}) + \mathbf{e}, \tag{15.7}$$

where ${\Phi_{wv}}^{(\epsilon)}$ is an $N \times (NF)$ matrix defined by ${\Phi_{wv}}^{(\epsilon)}(i, (j-1)F + k) = K_k(\mathbf{x_i}, \mathbf{x_j}; \epsilon)$ with $i, j \in [1, N]$ and $k \in [1, F]$. Besides, $\mathbf{w} \otimes \mathbf{v}$ in (15.7) is the tensor product of vector $\mathbf{w}$ and $\mathbf{v}$ which yields an $(NF) \times 1$ row vector with the definition $(\mathbf{w} \otimes \mathbf{v})((j-1)F + k) = w_j v_k$ where $j \in [1, N]$ and $k \in [1, F]$.

A useful property of this new kernel machine is that the roles of $\mathbf{w}$ and $\mathbf{v}$ in the model (15.7) are relatively symmetric and transposable. Model (15.6) can be derived from model (15.7) by moving $\mathbf{v}$ from the tensor product to the design matrix defined by ${\Phi_w}^{(\epsilon)}(i, j) = \sum_{k=1}^{F} v_k K_k(\mathbf{x_i}, \mathbf{x_j}; \epsilon)$ with $i, j \in [1, N]$. Similarly, the following model can be derived from (15.7) by instead moving $\mathbf{w}$ to the design matrix:

$$\mathbf{t} = {\Phi_v}^{(\epsilon)} \cdot \mathbf{v} + \mathbf{e}, \tag{15.8}$$

where ${\Phi_v}^{(\epsilon)}$ is an $N \times F$ matrix defined by ${\Phi_v}^{(\epsilon)}(i, k) = \sum_{j=1}^{N} w_j K_k(\mathbf{x_i}, \mathbf{x_j}; \epsilon)$ with $i \in [1, N]$ and $k \in [1, F]$. This property indicates that the exploration process of $\mathbf{w}$ and $\mathbf{v}$ may be unified, which will be discussed in detail in the next section.

15.3 Sparse Relevance Kernel Machine

While the new kernel machine is capable of solving feature weighting during the training process, the searching space of the solution is inflated by the number of features. A sparse treatment is highly appealing since the sparsity may help to reduce the complexity. And in the scenario of feature selection, that is, in the cases of $\epsilon = 0$, it may help to filter out irrelevant or redundant features as much as possible if $\mathbf{v}$ tends to be sparse. Since Bayesian learning frameworks with sparse prior [30, 37, 38] are capable of producing highly sparse models, we develop a sparse relevance kernel machine under the Bayesian learning framework in this section.

15.3.1 Relevance Vector Machine

The *relevance vector machine* (RVM)[30] is a sparse Bayesian model providing a viable probabilistic framework, whose Bayesian network model is shown in Fig. 15.3.

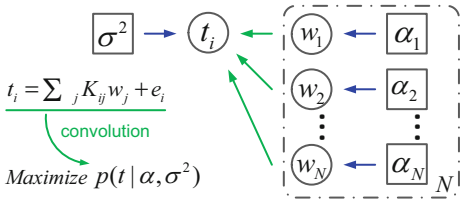


Fig. 15.3 The network model of the RVM where $K_{ij} = K(x_i, x_j)$. Circles denote random variables and squares denote deterministic model parameters

Given the input vectors $\{\mathbf{x_n}\}_{n=1}^{N}$ and their corresponding targets $\{t_n\}_{n=1}^{N}$, the RVM is used to probabilistically determine the model (15.6). The given target values are modeled as independent Bernoulli random variables with the following probability distribution:

$$P(\mathbf{t}|\mathbf{w}) = \prod_{i=1}^{N} \sigma[y(\mathbf{x_i}; \mathbf{w})]^{t_i} \cdot \sigma[1 - y(\mathbf{x_i}; \mathbf{w})]^{1-t_i}, \tag{15.9}$$

where σ is the sigmoid link function $\sigma(y) = (1 + e^{-y})^{-1}$.

Different from deterministic learning models (e.g., the SVM) that compute $\mathbf{w}$ directly, the RVM defines the prior distribution of $\mathbf{w}$ as independent zero-mean Gaussian random variables with variance $\boldsymbol{\alpha}$, and compute $\boldsymbol{\alpha}$ instead of $\mathbf{w}$ in the training process:

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{n=1}^{N} \mathcal{N}(0, \alpha_n^{-1}). \tag{15.10}$$

If $\alpha_i < \infty$, w_i is called a *relevance vector* since it has a variance greater than zero, allowing x_i making contributions to the decision function. Note that the RVM performs prediction with the posterior probability of the internal variables (i.e., the weights). Via convolution of Gaussian distributions, the covariance and mean of the posterior $p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha})$ can be shown to be respectively:

$$\Sigma = (\Phi^T \Gamma \Phi + \mathbf{A})^{-1} \tag{15.11}$$

$$\boldsymbol{\mu} = \Sigma \Phi^T \Gamma \mathbf{t}, \tag{15.12}$$

where $\mathbf{A} = diag(\boldsymbol{\alpha})$, $\Gamma = diag(\boldsymbol{\gamma})$, and $\boldsymbol{\gamma}(i) = \sigma[y(\mathbf{x_i}; \mathbf{w})] \cdot \sigma[1 - y(\mathbf{x_i}; \mathbf{w})]$.

The objective of the Bayesian network is to find the most probable model parameters with the given training samples, i.e., to maximize the posterior probability $p(\mathbf{w}, \boldsymbol{\alpha}|\mathbf{t})$. Based on the Bayes' rule, such objective is equivalent to:

$$\arg\max_{\boldsymbol{\alpha}} p(\mathbf{t}|\boldsymbol{\alpha}). \tag{15.13}$$

A training algorithm is proposed in [32] to compute the optimal $\boldsymbol{\alpha}$. After the training, for any $\hat{\mathbf{x}}$, its predicted probability of being labeled 1 is:

$$\sigma(\hat{y}) = \sigma(\boldsymbol{\mu}^T \boldsymbol{\phi}(\hat{\mathbf{x}})) \tag{15.14}$$

where $\boldsymbol{\phi}(\hat{\mathbf{x}})$ is a vector of size N whose i-th entry is defined by $\boldsymbol{\phi}(\hat{\mathbf{x}})(i) = K(\hat{\mathbf{x}}, \mathbf{x_i})$.

15.3.2 Bayesian Learning Model for SRKM

The RVM is a learning model based on the decision function (15.5) focusing on producing a sparse $\mathbf{w}$. Under the same framework, a feature selection technique called *relevance feature vector machine* (RFVM) [6] is proposed to exchange the roles of samples and features by defining the "feature vector" $\mathbf{f}_i = (\mathbf{x}_1(i), \mathbf{x}_2(i), \ldots, \mathbf{x}_F(i))^T$ and a new design matrix $\Phi'(i, j) = K(\mathbf{f}_i, \mathbf{f}_j)$. This is a regularization-based feature selection method, which focuses on building a filter method [5] by ranking the features.

To achieve high model accuracy and high quality feature weighting with the new kernel machine (15.7), we propose a *sparse relevance kernel machine* (SRKM) whose conceptual structure is shown in Fig. 15.4.

Without using the new kernel machine model (15.7), learning method with embedded feature weighting is often realized by directly assigning weights $\mathbf{v}$ to the parameters. However, as we discussed previously, since most kernel functions are nonlinear, it is extremely difficult to develop the models in the Bayesian learning contexts. For example, by assigning Gaussian prior to $\mathbf{v}$ in a similar way $p(\mathbf{v}|\boldsymbol{\beta}) = \prod_{n=1}^{F} \mathcal{N}(0, \beta_n^{-1})$, the Bayesian network described in Fig. 15.3 is extended to derive the new model on the left of Fig. 15.5. Assuming Gaussian kernel with linear direct feature weighting

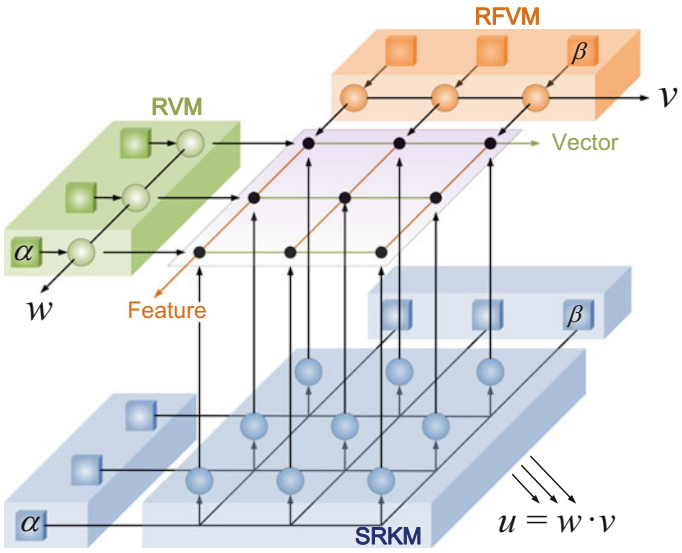


Fig. 15.4 The conceptual SRKM model. Small black dots denote training data, with each row representing a vector and each column representing a feature. Circles denote random variables and squares denote deterministic model parameters

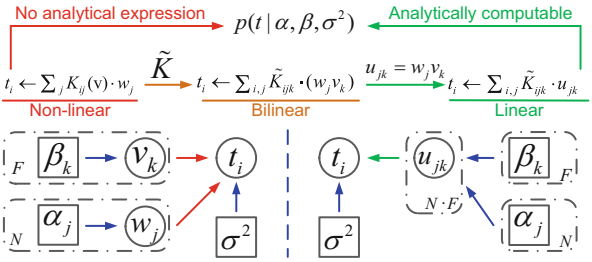


Fig. 15.5 Development of SRKM Bayesian network where $K_{ijk} = K_k(\mathbf{x}_i, \mathbf{x}_j; \epsilon)$. Circles denote random variables and squares denote deterministic model parameters

$$K(\mathbf{x_i}, \mathbf{x_j}) = e^{-\gamma \|diag(\mathbf{v}) \cdot (\mathbf{x_i} - \mathbf{x_j})\|^2},$$

is employed, the deterministic relationship from $\mathbf{w}$ and $\mathbf{v}$ to t is highly nonlinear. As a result, the following optimization objective of the Bayesian training process is not analytically computable and hence hinders the optimization-based training process:

$$p(\mathbf{t}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \sigma^2) = \iint p(\mathbf{t}|\mathbf{w}, \mathbf{v}, \sigma^2) \cdot p(\mathbf{w}|\boldsymbol{\alpha}) \cdot p(\mathbf{v}|\boldsymbol{\beta}) d\mathbf{w} d\mathbf{v}. \tag{15.15}$$

As the first step towards developing the SRKM, we embed the new kernel machine (15.7) into the Bayesian learning framework to handle the feature weighting problem. Here we treat ϵ in the feature kernel as a kernel parameter pre-defined before the training process, similar to the γ parameter in the Gaussian kernel, which can be selected by standard processes like cross validation. Leveraging the relevance kernel machine in the Bayesian network helps to simplify the highly nonlinear deterministic relationships into a bilinear form. As described in (15.7), $\mathbf{t}$ can be expressed as linear combinations of a series of $\mathbf{w}(j)\mathbf{v}(k)$ terms.

However, inference in Bayesian networks with nonlinear or even bilinear deterministic relationships requires great effort like piece-wise linearization [7] or dynamic discretization [23] to deal with the nonlinearity, which may greatly boost the complexity of the learning model. To address this problem, instead of defining $\mathbf{w}$ and $\mathbf{v}$ as separate Gaussian random variables, we replace the term $\mathbf{w}(j)\mathbf{v}(k)$ with a single random variable u_{jk} for all $j \in [1, N]$ and $k \in [1, F]$, which results in linear deterministic relationships from u_{jk} to $\mathbf{t}$. If we define a new vector $\mathbf{u}$ of size $(N \cdot F)$ whose entry $\mathbf{u}((j-1)N + k) = u_{jk} = \mathbf{w}(j) \cdot \mathbf{v}(j)$, the model becomes:

$$\mathbf{t} = \Phi_u{}^{(\epsilon)} \cdot \mathbf{u} + \mathbf{e}, \tag{15.16}$$

where the design matrix $\Phi_u{}^{(\epsilon)}$ is identical to $\Phi_{wv}{}^{(\epsilon)}$ in (15.7).

In the RVM, the zero-mean Gaussian prior distribution of $\mathbf{w}$ tends to help the model converge to a sparse $\mathbf{w}$ since the resulting marginal prior distribution over $\mathbf{w}$ is the product of *Student-t* distributions. Similarly, to achieve sparsity in $\mathbf{u}$, we also define their prior distributions as independent zero-mean Gaussian distributions. Considering the nature of $\mathbf{u}$, if $u_{j,k}$ is irrelevant, meaning the distribution of $u_{j,k}$

is infinitely peaked at zero, either the i-th sample ($\{u_{j,k}\}_{k=1}^{F}$) or the j-th parameter ($\{u_{j,k}\}_{j=1}^{N}$) should be irrelevant as well. To reflect this, we define a proper prior for $\mathbf{u}$ as:

$$p(\mathbf{u}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{j=1}^{N} \prod_{k=1}^{F} \mathcal{N}(0, \alpha_j^{-1} \beta_k^{-1}), \tag{15.17}$$

which leads to our proposed computable linear Bayesian network shown on the right of Fig. 15.5 and our conceptual model described in Fig. 15.4. An infinite $\alpha_j \beta_k = \infty$ means $u_{jk} = 0$ and the corresponding feature kernel is irrelevant to the final decision function. In addition, if $\alpha_j \to \infty$, all the $\{u_{j,k}\}_{k=1}^{F}$ are zero, meaning the j-th sample is discarded from the set of relevance vectors. Likewise, if $\beta_k < \infty$, the k-th parameter is relevant and there should be at least one non-zero $u_{i,j}$ for $i \in [1, N]$.

Under the same Bayesian inference framework, the posterior covariance and mean of $p(\mathbf{u}|\mathbf{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \sigma^2)$ in the proposed Bayesian network are found to be:

$$\Sigma_u = (\sigma^{-2} (\Phi_u{}^{(\epsilon)})^T \Phi_u{}^{(\epsilon)} + \mathbf{A_u})^{-1}, \tag{15.18}$$

$$\boldsymbol{\mu}_\mathbf{u} = \sigma^{-2} \Sigma_u (\Phi_u{}^{(\epsilon)})^T \mathbf{t}, \tag{15.19}$$

where $\mathbf{A_u} = diag(\alpha_1 \beta_1, \alpha_1 \beta_2, \ldots, \alpha_N \beta_F)$ and $\Phi_u{}^{(\epsilon)}$ is the new design matrix defined in (15.16). The formulas (15.18) and (15.19) are in the same form as the posterior covariance and mean of w in the RVM, and consequently solvable with the existing RVM algorithms.

The SRKM classification model behaves analogously to the regression model but using a Bernoulli likelihood instead of Gaussian for the target with the following sigmoid link function:

$$\varrho(y) = \frac{1}{1 + e^{-y}}. \tag{15.20}$$

As a result, the Bernoulli likelihood is:

$$P(\mathbf{t}|\mathbf{u}) = \prod_{i=1}^{N} \varrho[y(\mathbf{x_i}; \mathbf{u})]^{t_i} \cdot \varrho[1 - y(\mathbf{x_i}; \mathbf{u})]^{1-t_i}, \tag{15.21}$$

where the targets $t_i \in \{0, 1\}$ are for binary classifications. With likelihood in the form of (15.21), there is no closed-form expressions for $\mathbf{u}$ and hence the Laplace approximation procedure [21, 32] should be utilized and nested in each iteration of the training.

15.3.3 Efficient Training Algorithm

The marginal likelihood maximization [32] required in training the RVM model is solved in an iterative process similar to the well-known *expectation maximization* (EM) algorithm. Due to the required matrix operations, the worst case computational complexity in each iteration is $O(N^2F + N^2M)$ [14] if there are M relevance vectors in that iteration and F features in total. By performing one-time pre-computation of the full $N \times N$ design matrix Φ with the complexity of $O(N^2F)$ and pre-computing $(\Phi^T\Phi)$ (or normalizing Φ as the implementation of [31]), the complexity of each iteration can be reduced to $O(NM^2)$ by using $O(N^2)$ memory instead of $O(NM)$.

For the SRKM, by solving (15.16) with this algorithm, since the size of the vector $\mathbf{u}$ is $(N \cdot F)$, a large number of features F will blow the worst case computational complexity for each iteration from $O(NM^2)$ to $O(NFM^2E^2)$ if there are M relevance vectors and E relevance features in that iteration.

To address this computational challenge, our proposed efficient algorithm leverages the property that $\mathbf{w}$ and $\mathbf{v}$ are interchangeable in the bilinear Bayesian network (15.7), and that either vectors can be merged into the design matrix to reduce our model to (15.6) or (15.8). As Fig. 15.6 shows, fixing $\boldsymbol{\alpha}$ and moving the resulting expectation of $\mathbf{w}$ into the design matrix (i.e., converting $\Phi_u^{(\epsilon)}$ in (15.16) to $\Phi_v^{(\epsilon)}$ in (15.8)) will reduce every column in Fig. 15.6 to a single weight v_j with its prior β_j. Similarly, row-wise reduction by fixing $\boldsymbol{\beta}$ converts the proposed network to another RVM network with $\mathbf{w}$ and $\boldsymbol{\alpha}$.

The above discussion suggests an efficient two-level iterative training process. In each iteration of the top level, we reduce the model either row-wise or column-wise, and update $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$ subsequently. In the second level, the original algorithm [32] can be employed to solve either Model (15.6) or Model (15.8). The complexity in each iteration is now reduced to either $O(NM^2)$ or $O(FE^2)$.

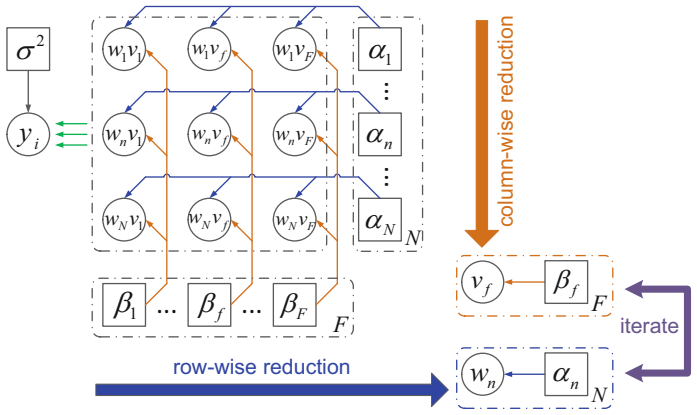


Fig. 15.6 Efficient SRKM model with network reduction. Circles denote random variables and squares denote deterministic model parameters

15.4 Experiments

To demonstrate the superiority of the proposed SRKM, we compare its performance with popular learning-based techniques including the SVM [34] and the RVM [30]. We also compare the SRKM with the RFVM [6] in terms of parameter (feature) ranking. For the purposes of parameter ranking and selection, we use feature kernel with $\epsilon = 0$ in all the experiments.

15.4.1 Variability Analysis of an LDO

Building an accurate regression model for a given analog performance and performing feature ranking among all sorts of process parameters are key to the understanding of the impacts of process variabilities on analog circuits. Since simulations or measurements are usually expensive, it is of great significance to build an accurate regression model and obtain reliable parameter weighting with a moderate amount of samples, which turns out to be a task well handled by the proposed method.

We investigate the process variations in a realistic *low-dropout regulator* (LDO) design (Fig. 15.7) proposed in [16]. We build SRKMs to analyze the impact of process variations on LDO specifications including its quiescent current, undershoot of the output voltage V_{out}, and load regulation. Channel length variations of all transistors in the LDO are modeled at the SPICE level using a commercial 90 nm CMOS technology design kit. We use various numbers of simulation samples to build a regression model relating the model process parameters with each targeted specification and test the accuracies of these models using a testing set of 1000 simulation samples. The results are shown in Fig. 15.8.

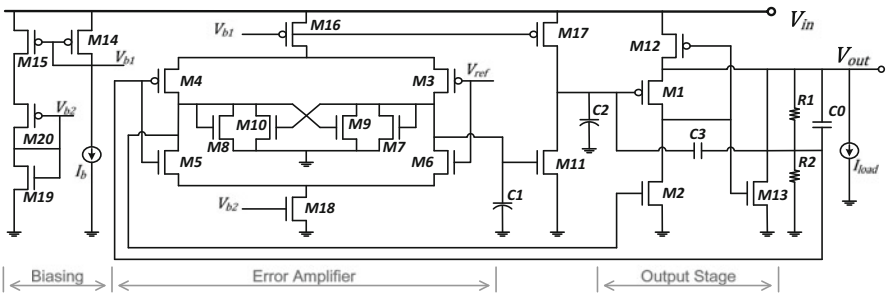


Fig. 15.7 Low-dropout regulator (LDO)

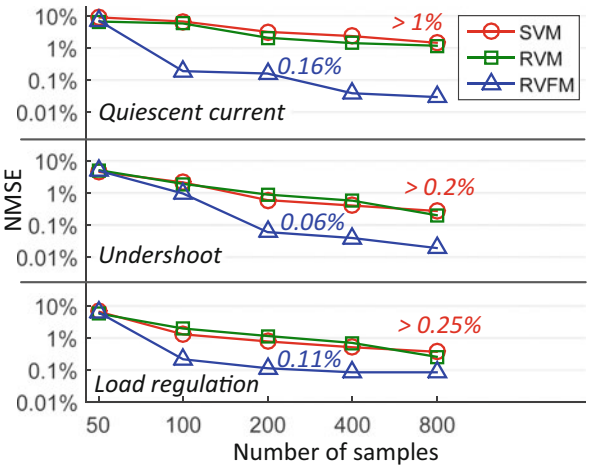


Fig. 15.8 Regression performance comparison

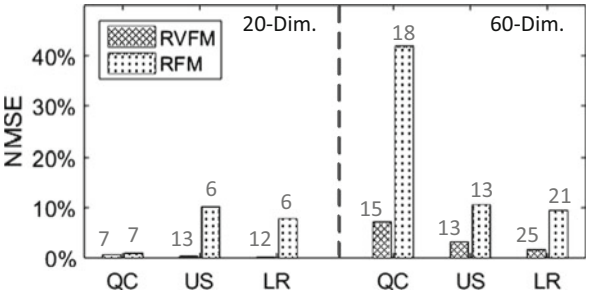


Fig. 15.9 Ranking quality comparison with the number of selected parameters (QC quiescent current, US undershoot, LR load regulation)

In this experiment, *normalized mean square error* (NMSE) is used as the metric to evaluate the performance of the predictors trained with different techniques. As Fig. 15.8 shows, the SRKM out-performs the popular SVM and RVM in all cases by achieving one order of magnitude lower NMSEs.

We compare the ranking produced by the SRKM and the RFVM on feature ranking in Fig. 15.10. To evaluate the quality of the ranking, for each design specification, we train two RVMs only in the process parameters selected by SRKM and RFVM, respectively. A parameter is selected by SRKM or RFVM if its expected weight is greater than 0.01. Such procedure is firstly applied to the regression model with 20 channel length variations, i.e., the three columns on the left of Fig. 15.9. Then, the same procedure is applied to an expanded full parameter set of 60 parameters involving variations of each transistor's channel length, oxide thickness, and threshold voltage (on the right of Fig. 15.9). The resulting NMSEs and the numbers of parameters selected indicate that the SRKM produces more reliable parameter weighting and reaches similar sparsity compared to the RFVM.

We use design knowledge to provide further insights and validation for the parameter rankings of the 20 channel length variations computed by the SRKM. For example, based on the analysis in [15] a majority portion of the multi-loop LDO's

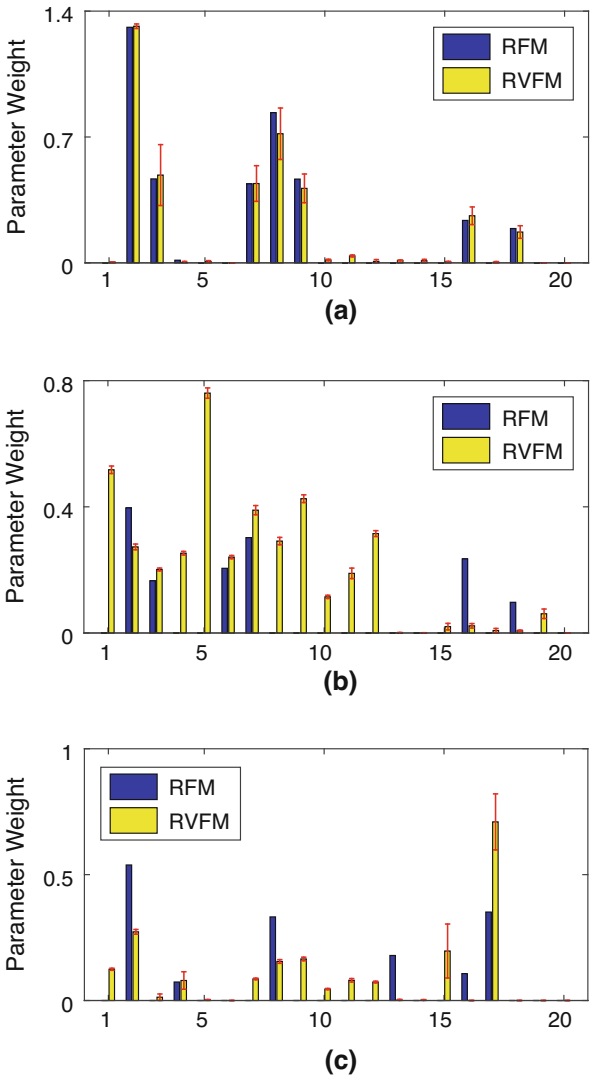


Fig. 15.10 Weights of transistor's channel length variation in the model of: (**a**) quiescent current; (**b**) undershoot; (**c**) load regulation. Red lines represent the 95% confidence intervals estimated by SRKM

quiescent current is consumed by the fastest two loops in the output stage and hence the variation on $M2$ has significant impact on the quiescent current. Moreover, variations on $M3$, $M7$, $M8$, and $M9$ may lead to mismatches and considerable changes at the two output nodes of the error amplifier, one of which is V_g of $M2$. This analysis matches the ranking shown in Fig. 15.10a.

The undershoot of the LDO is mainly determined by the load capacitor and the loop bandwidth, which is further determined by the error amplifier (involving $M3 \sim M10$), the fast loop in the output stage ($M12$), and the in-band zeros locations defined as:

$$\omega_{LCZ} \approx \sqrt{\frac{g_{m1} g_{m11} g_a}{g_{m2} C_{C2}(C_{C1} + C_{C3})}}, \tag{15.22}$$

where g_a is the output admittance of the error amplifier defined by the g_m of $M7 \sim M10$. The ranking of the SRKM in Fig. 15.10b is reliable since it captures all these relevant variations.

Load regulation of the LDO is mainly determined by the DC loop gain, which is the product of the gains of all stages in the loop. The gain of the EA stage is inversely proportional to the g_m of $M7 \sim M10$ and the second stage is comprised of $M17$ and $M11$. Again, the ranking of the SRKM as shown in Fig. 15.10c successfully identifies all these important variations.

15.4.2 PLL BIST Scheme Optimization

Built-in-self-test (BIST) is very effective in detecting operational failures of deployed analog/mixed-signal circuits. Based on the concept of alternative test, efficient BIST solutions can be formed by collecting low-cost test signatures and relating the signatures to targeted performance specifications using statistical prediction models. The effectiveness of BIST heavily depends on the quality of the selected signatures and the tradeoffs between accuracy, overhead, and test time. We apply the RFVM to the BIST of a charge-pump PLL targeting three key specifications: *lock-time* (LT), *frequency overshoot* (OVS), and *jitter* (JT) (Table 15.1).

Figure 15.11 shows the PLL along with three BIST schemes using various test signatures. Jitter, frequency overshoot, and lock-time are important specifications but cannot be easily measured directly on the chip. To capture failures in those specifications, the first candidate BIST scheme [40] collects the readouts of the counter in the divider as its test signature, while the second scheme [11] collects the accumulated *up* and *dn* phase detector outputs via integrators and *time-to-digital converters* (TDCs). The third scheme is an example of IDDQ testing, measuring the quiescent currents of the *charge-pump* (CP) and the *voltage control oscillator* (VCO) as test signatures similar to the approach of [22].

The first two schemes operate in a special test mode which instead of feeding back the divider output, it first feeds the one-buffer delayed reference clock to the phase detector for 8 reference cycles with a cycle time of 0.1 μs. Then, the reference

Table 15.1 BIST scheme synthesis

Spec.	Original best accuracy	Synthesized signatures			Test time (μs)			Test time reduction	Optimized accuracy
		Sch.1	Sch.2	Sch.3	Sch.1&2	Sch.3	Total		
JT	97.22%	1–3	1	VCO	0.3	0.6	0.9	43.75%	99.98%
OVS	98.00%	1–3	1–7	CP	0.4	0.6	1.0	37.50%	99.88%
LT	96.40%	1–4	1–2	VCO	0.4	0.6	1.0	37.50%	99.95%

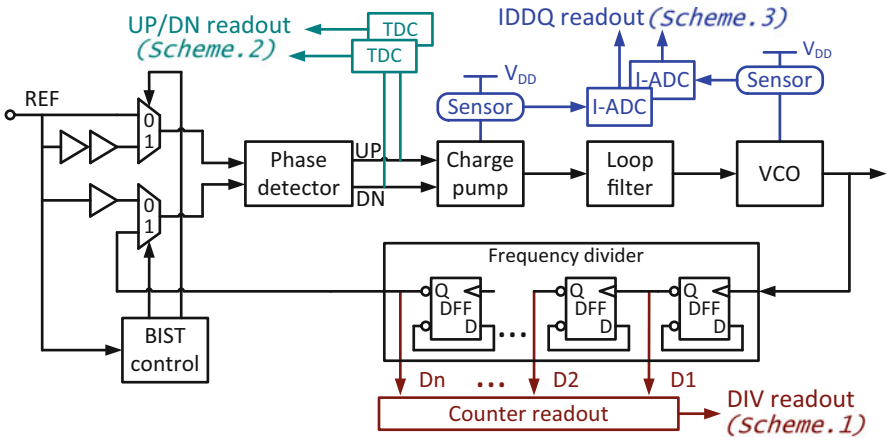
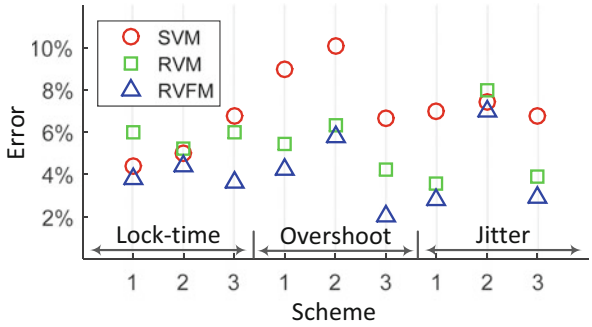


Fig. 15.11 A PLL and three BIST schemes

Fig. 15.12 Classifier performance comparison



clock input is replaced by the double delayed reference clock for another 8 cycles. Each cycle generates one signature for Scheme 1 and two for Scheme 2, making a total of 16 and 32 signatures for Scheme 1 and 2, respectively. Scheme 3 reads out two signatures, i.e., the CP and VCO quiescent currents, in the quiescent mode.

Recently, learning-based classifiers like the SVM have been trained to perform the failure detection in BIST [4, 40]. To make better usage of the collected test signatures, we apply the proposed SRKM in each scheme. We fit the target specification into a sigmoid function before we employ the SRKM as a classier for failure detection. Three classification techniques, the SVM, RVM, and SRKM, are trained with 200 simulation samples and tested with 4000 samples. The classifying errors are compared in Fig. 15.12 which shows the superior BIST classifier accuracy of the proposed SRKM.

In addition, the SRKM also produces reliable ranking among test signatures, which can be further leveraged to improve the efficiency of BIST schemes. For example, the SRKM ranks the 16 test signatures in Scheme 1 as shown in Fig. 15.13 when building the classifier for jitter failure detection. The tenth signature is the last one with a significant weight. After that, the remaining six signatures are of little importance and can be considered as redundant. Using the same procedure, we reduce the test time for each of the three specifications for Scheme 1 as reported in Table 15.2.

Assuming that realizing all three schemes on-chip does not lead to significant overhead, we seek to improve BIST accuracy by leveraging the signatures of all the schemes. While combing all the signatures can offer the best accuracy, it may not be completely efficient due to the existence of redundant signatures. For this, we train an SRKM on all the signatures across the three schemes to predict the jitter. Based on the signature ranking shown in Fig. 15.14, we collect the first three signatures in Scheme 1 and the first signature in Scheme 2. Although the third last signature in Scheme 2 also possesses a notable weight, collecting such signature is not cost-effective in terms of test time, and thus it is discarded. For Scheme 3, only the quiescent current of VCO is selected, which can be measured in 0.6 µs according to [22].

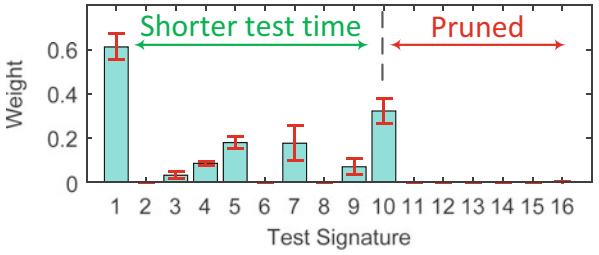


Fig. 15.13 Signature ranking for jitter prediction with Scheme 1

Table 15.2 Test time optimization of Scheme 1

Spec.	Original accuracy	Selected readouts	Resulting accuracy	Test time reduction
JT	97.22%	1–10	96.20%	37.5%
OVS	95.78%	1–12	94.89%	25.0%
LT	96.20%	1–6	97.00%	62.5%

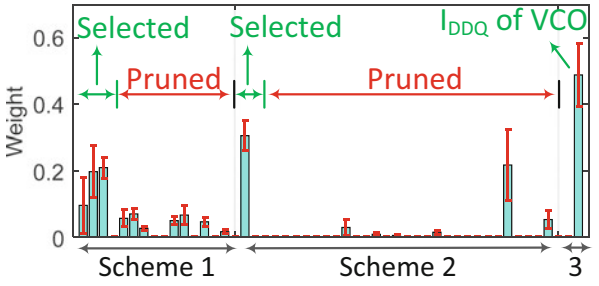


Fig. 15.14 Signature ranking for jitter prediction with all three schemes

Based on these five selected signatures, we synthesize an optimized combined BIST scheme for each specification and show the results in Table 15.1. As can be been, by using the proposed SRKM, the BIST accuracy can be boosted to over 99.88% with a test time reduction of about 40%.

15.4.3 Binary Classification for Functional Check

The above two examples illustrate the effectiveness of the regression version of SRKM. This experiment demonstrates the superiority of the proposed SRKM algorithm for classification by applying it to analyzing a commercial dual-lane data communication AMS system shown in Fig. 15.15. As a product designed for automotive applications, functional safety is a key requirement. One of the safety features, the *thermal shutdown* (TSD) function, is investigated in this experiment to analyze the impact of process variations on this feature.

Figure 15.15 shows the functional blocks that are electrically related to the TSD feature. The BG block is a bandgap reference that provides temperature independent voltage reference to the IREF block. The IREF block provides reference currents to multiple blocks, including the TX, RX, and TSD blocks. The Lane 1 and Lane 2 in the system share the same design, including the TSD implementation. The shutdown temperature threshold is designed to be 190 °C, meaning both Lane 1 and Lane 2 should turn off their driver to the bus when the temperature exceeds 190 °C.

We collected results from 1000 Monte Carlo simulations, and label them as 1 or 0 according to whether the lane can turn off the driver (TX block) at 190 °C or not. We use half of the simulations as the training data to build an SRKM classification model for each lane, and use the other half to test the performance of the trained classifier. Based on the *process design kit* (PDK) we use, there are 15,015 process parameters involved in each simulation. In other words, our goal is to build 15,015-dimensional classifiers with merely 500 samples.

As shown in Table. 15.3, the widely used SVM can only produce a classifier with an accuracy of about 60%, while the proposed SRKM can achieve an accuracy of around 85%. For the trained SVM model, all 500 samples become the support vectors, indicating that the SVM cannot find the regularity from the given training

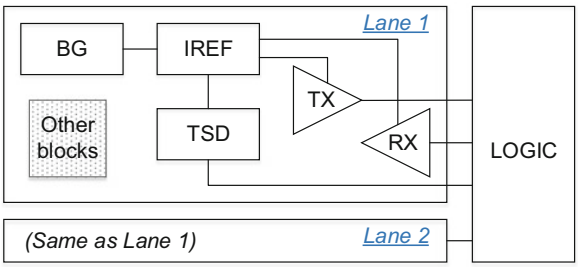


Fig. 15.15 Block diagram of a dual-lane data communication AMS system

data. On the contrary, the SRKM successfully achieves sparsity in the vector space (2 relevance vectors for Lane 1 and 23 for Lane 2 out of 500 samples) and the feature space (38 relevance features for Lane 1 and 45 relevance features for Lane 2 out of 15,015 process parameters).

In Fig. 15.16, we sum up the produced SRKM feature weights in each functional blocks and lanes to reflect the block-wise impact of process variations. For the SRKM trained for the Lane 1 TSD, relatively small feature weights have been assigned to the process parameters from Lane 2, which matches the structure of the system, where Lane 2 is indeed not directly connected to Lane 1 and hence should have minimal impact on the Lane 1 TSD. Similarly, in the SRKM trained for Lane 2 TSD, the impact of process parameters from Lane 1 is also minimal.

Based on the block-wise break-down of the sum of feature weights, it implies that the process parameters from the BG and IREF blocks, which determine the reference current of the TSD block, are more critical than the process parameters from the TSD block, which is the implementation of the TSD function. This also matches with the design intuition since the TSD block mainly involves components such as comparators whose performances are more resilient with respect to process variations, while currents are usually more vulnerable to process variations. Therefore, it can be anticipated that the feature weighting results are reliable.

Table 15.3 Learning model performance comparison (*SV/RV* support vectors/relevance vectors; *RF* relevance features)

	Model	Accuracy	# SV/RV	# RF
Lane 1	SVM	56.4%	500	–
	SRKM	85.8%	2	38
Lane 2	SVM	61.0%	500	–
	SRKM	84.6%	23	45

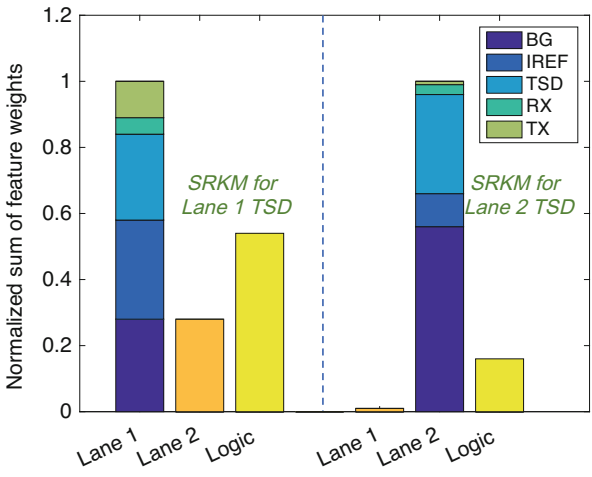


Fig. 15.16 Process variation impacts on TSD of various blocks

15.5 Conclusions

This paper proposes a novel sparse Bayesian learning framework named sparse relevance kernel machine to capture circuit characteristics and analyze circuit performance dependencies on assorted parameters or signatures via a statistical model. The advantages of the proposed framework are demonstrated in examples including statistical variability modeling of an LDO, a BIST scheme optimization of a charge-pump PLL, and building statistical variability models for a commercial automotive interface design.

The framework of the learning model was originally developed in our earlier works [18, 20]. Since then, we have extended the framework to handle both regression and classification problems, and have explored potentials of the new learning model in practical circuit applications. One major limitation of the current learning model is the computational complexity. Although we developed an iterative algorithm to remedy the complexity increased by the feature kernel weighting mechanism, based on the discussion in Sect. 15.3.3, the computational complexity of SRKM is one order higher than the widely used SVM in each iteration. And the overall convergence is also slower since SRKM has two levels of iterations. As a result, to solve applications with huge amount of data, the training algorithm needs to be further optimized for better efficiency.

Moreover, the feature kernel weighting mechanism proposed in Sect. 15.2 is very flexible. Since sparsity is very useful in our applications, we developed the learning model of SRKM in the Bayesian learning framework, but the feature kernel weighting mechanism can definitely fit into other kernel-based learning framework such as sequential minimal optimization (SMO), which poses potentials in other application scenarios where sparsity is not in need.

Acknowledgements This material is based upon work supported by the Semiconductor Research Corporation (SRC) through Texas Analog Center of Excellence at the University of Texas at Dallas (Task ID:2712.004).

References

1. P. Bastani, N. Callegari, L.C. Wang, M.S. Abadir, Statistical diagnosis of unmodeled systematic timing effects, in *Proceedings of the 45th Annual Design Automation Conference* (ACM, New York, 2008), pp. 355–360
2. C.M. Bishop, *Neural Networks for Pattern Recognition* (Oxford University Press, Oxford, 1995)
3. A.L. Blum, P. Langley, Selection of relevant features and examples in machine learning. Artif. Intell. **97**(1), 245–271 (1997)
4. A. Bounceur, B. Brahmi, K. Beznia, R. Euler, Accurate analog/RF BIST evaluation based on SVM classification of the process parameters, in *2014 9th International Design & Test Symposium (IDT)* (IEEE, Piscataway, 2014), pp. 55–60
5. G. Chandrashekar, F. Sahin, A survey on feature selection methods. Comput. Electr. Eng. **40**(1), 16–28 (2014)

6. H. Cheng, H. Chen, G. Jiang, K. Yoshihira, Nonlinear feature selection by relevance feature vector machine, in *Machine Learning and Data Mining in Pattern Recognition* (Springer, Berlin, 2007), pp. 144–159
7. B.R. Cobb, P.P. Shenoy, Nonlinear deterministic relationships in Bayesian networks, in *Symbolic and Quantitative Approaches to Reasoning with Uncertainty* (Springer, Berlin, 2005), pp. 27–38
8. M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. **15**(1), 3133–3181 (2014)
9. I. Guyon, S. Gunn, M. Nikravesh, L.A. Zadeh, *Feature Extraction: Foundations and Applications*, vol. 207 (Springer, Berlin, 2008)
10. S. Hochreiter, K. Obermayer, Nonlinear feature selection with the potential support vector machine, in *Feature Extraction* (Springer, Berlin, 2006), pp. 419–438
11. S.W. Hsiao, N. Tzou, A. Chatterjee, A programmable BIST design for PLL static phase offset estimation and clock duty cycle detection, in *2013 IEEE 31st VLSI Test Symposium (VTS)* (IEEE, Piscataway, 2013), pp. 1–6
12. X. Huang, L. Shi, J.A. Suykens, Ramp loss linear programming support vector machine. J. Mach. Learn. Res. **15**(1), 2185–2211 (2014)
13. S.S. Keerthi, O. Chapelle, D. DeCoste, Building support vector machines with reduced classifier complexity. J. Mach. Learn. Res. **7**, 1493–1515 (2006)
14. D.E. King, Dlib-ml: a machine learning toolkit. J. Mach. Learn. Res. **10**, 1755–1758 (2009)
15. S. Lai, Modeling, design and optimization of IC power delivery with on-chip regulation. Doctoral dissertation, Texas A&M University, 2014
16. S. Lai, P. Li, A fully on-chip area-efficient CMOS low-dropout regulator with fast load regulation. Analog Integr. Circuits Signal Process. **72**(2), 433–450 (2012)
17. F. Li, Y. Yang, E.P. Xing, From lasso regression to feature vector machine, in *Advances in Neural Information Processing Systems* (2005), pp. 779–786
18. H. Lin, Algorithms for verification of analog and mixed-signal integrated circuits. Doctoral dissertation, Texas A&M University, 2016
19. H. Lin, P. Li, Circuit performance classification with active learning guided sampling for support vector machines. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **34**(9), 1467–1480 (2015). https://doi.org/10.1109/TCAD.2015.2413840
20. H. Lin, P. Li, Relevance vector and feature machine for statistical analog circuit characterization and built-in self-test optimization, in *Proceedings of the 53rd Annual Design Automation Conference* (2016), pp. 11–16
21. D.J. MacKay, The evidence framework applied to classification networks. Neural Comput. **4**(5), 720–736 (1992)
22. S. Maltabas, O.K. Ekekon, K. Kulovic, A. Meixner, M. Margala, An IDDQ BIST approach to characterize phase-locked loop parameters, in *2013 IEEE 31st VLSI Test Symposium (VTS)* (IEEE, Piscataway, 2013), pp. 1–6
23. M. Neil, M. Tailor, D. Marquez, Inference in hybrid Bayesian networks using dynamic discretization. Stat. Comput. **17**(3), 219–233 (2007)
24. J. Neumann, C. Schnörr, G. Steidl, Combined SVM-based feature selection and classification. Mach. Learn. **61**(1–3), 129–150 (2005)
25. B. Schölkopf, A.J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* (MIT Press, Cambridge, 2002)
26. A. Smola, B. Scholkopf, G. Ratsch, Linear programs for automatic accuracy control in regression, in *Ninth International Conference on (Conf. Publ. No. 470) Artificial Neural Networks, 1999 (ICANN 99)*, vol. 2 (IET, Stevenage, 1999), pp. 575–580
27. P. Somol, P. Pudil, J. Novovičová, P. Paclık, Adaptive floating search methods in feature selection. Pattern Recogn. Lett. **20**(11), 1157–1163 (1999)
28. J.A. Suykens, J. Vandewalle, Least squares support vector machine classifiers. Neural Process. Lett. **9**(3), 293–300 (1999)
29. R. Tibshirani, Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Ser. B Methodol. **58**, 267–288 (1996)

30. M.E. Tipping, Sparse Bayesian learning and the relevance vector machine. J. Mach. Learn. Res. **1**, 211–244 (2001)
31. M.E. Tipping, An efficient matlab implementation of the sparse Bayesian modelling algorithm (version 2.0). Vector Anomaly, March 2009
32. M.E. Tipping, A.C. Faul et al., Fast marginal likelihood maximisation for sparse Bayesian models, in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics* (2003)
33. V.N. Vapnik, *Statistical Learning Theory* (Wiley, New York, 1998)
34. V. Vapnik, S. Golowich, A. Smola, Support vector method for function approximation, regression estimation, and signal processing, in *Advances in Neural Information Processing Systems* (1997), pp. 281–287
35. D. Ververidis, C. Kotropoulos, Fast and accurate sequential floating forward feature selection with the Bayes classifier applied to speech emotion recognition. Signal Process. **88**(12), 2956–2970 (2008)
36. F. Wang, M., Zaheer, X. Li, J.O. Plouchart, A. Valdes-Garcia, Co-learning Bayesian model fusion: efficient performance modeling of analog and mixed-signal circuits using side information, in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design* (IEEE Press, Piscataway, 2015), pp. 575–582
37. P.M. Williams, Bayesian regularization and pruning using a Laplace prior. Neural Comput. **7**(1), 117–143 (1995)
38. D.P. Wipf, B.D. Rao, An empirical Bayesian strategy for solving the simultaneous sparse approximation problem. IEEE Trans. Signal Process. **55**(7), 3704–3716 (2007)
39. H. Xu, C. Caramanis, S. Mannor, Robustness and regularization of support vector machines. J. Mach. Learn. Res. **10**, 1485–1510 (2009)
40. G. Yu, P. Li, A methodology for systematic built-in self-test of phase-locked loops targeting at parametric failures, in *IEEE International Test Conference, 2007 (ITC 2007)* (IEEE, Piscataway, 2007), pp. 1–10