City University of New York (CUNY)

CUNY Academic Works

Publications and Research

New York City College of Technology

2016

Performance modeling of CMOS inverters using support vector machines (SVM) and adaptive sampling

Ashwin Satyanarayana

CUNY New York City College of Technology

[How does access to this work benefit you? Let us know!](#)

More information about this work at: https://academicworks.cuny.edu/ny_pubs/78

Discover additional works at: <https://academicworks.cuny.edu>

This work is made publicly available by the City University of New York (CUNY).

Contact: AcademicWorks@cuny.edu

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Performance modeling of CMOS inverters using support vector machines (SVM) and adaptive sampling

Ashwin Satyanarayana*

New York City College of Technology (CUNY), N-913, Computer Systems Technology, 300 Jay Street, Brooklyn, NY 11201, United States

ARTICLE INFO

Article history:

Received 16 October 2015

Revised 16 January 2016

Accepted 3 March 2016

Available online xxx

Keywords:

CMOS design

Big data

Chernoff bound

Support vector machines

Adaptive sampling

ABSTRACT

Integrated circuit designs are verified through the use of circuit simulators before being reproduced in real silicon. In order for any circuit simulation tool to accurately predict the performance of a CMOS design, it should generate models to predict the transistor's electrical characteristics. The circuit simulation tools have access to massive amounts of data that are not only dynamic but generated at high speed in real time, hence making fast simulation a bottleneck in integrated circuit design. Using all the available data is prohibitive due to memory and time constraints. Accurate and fast sampling has been shown to enhance processing of large datasets without knowing all of the data. However, it is difficult to know in advance what size of the sample to choose in order to guarantee good performance. Thus, determining the smallest sufficient dataset size that obtains the same accurate model as the entire available dataset remains an important research question. This paper focuses on adaptively determining how many instances to present to the simulation tool for creating accurate models. We use Support Vector Machines (SVMs) with Chernoff inequality to come up with an efficient adaptive sampling technique, for scaling down the data. We then empirically show that the adaptive approach is faster and produces accurate models for circuit simulators as compared to other techniques such as progressive sampling and Artificial Neural Networks.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction and motivation

Circuit model simulations have led to huge improvements in integrated circuit performance, as critical transistor dimensions scale below the 100 nm (nanoscale) regime. The output of these simulations are performance prediction models of integrated circuits that predict the performance as functions of the design parameters. This task of circuit simulation is computationally (both time and memory) expensive. As a result, simulation tools require sampling techniques [1,2] for creating models that are accurate and efficient. In this paper, we use adaptive sampling with Support Vector Machines (SVM) for generating the models of nanoscale CMOS inverter circuits for generating not only accurate, but also computationally efficient models.

Sampling can be a powerful technique for fast simulations as it avoids unnecessary processing of the whole data. However, one of the challenges of sampling is the way it is evaluated, as sampling involves a risk of reaching improper conclusions [3]. One way of evaluating a sampling strategy is called the Probably Close Enough (PCE) criterion (which is modeled after the Probably Approximately

Correct (PAC) criteria [1]). The PCE idea is to think about taking a sample that is probably good enough, meaning that there is only a small chance that the data mining algorithm could do better by using the entire database instead. We would like the smallest sample size $|D_i|$ such that:

$$Pr[|acc(D) - acc(D_i)| \geq \epsilon] \leq \delta \quad (1)$$

where $acc(D_i)$ refers to the accuracy of our mining algorithm after seeing a sample of size D_i (where D_i subset of D), $acc(D)$ refers to the accuracy after seeing all records in the database, ϵ is a parameter to be specified describing what close enough means, and δ is a parameter describing what probably means. PCE is similar to the PAC bound in computational learning theory [15]. One of the goals of this paper is to determine the sample size $|D_i|$ that satisfies Eq. (1), given the approximation parameter ϵ and confidence parameter δ .

The remainder of the paper is organized as follows. Section 2 discusses the contributions of this paper. Section 3 discusses prior research relevant to this work. Section 4 formulates the problem. In Section 5 we introduce adaptive sampling using Chernoff bound for support vector machines. We then perform our experiments on SPICE data in the next section. We finally summarize our results and conclude.

* Tel.: +0015183307907.

E-mail address: asatyanarayana@citytech.cuny.edu

2. Contributions of this paper

The novel contributions of this paper are as follows:

1. Adaptive sampling with SVM is proposed, which can be used to develop faster and accurate simulations instead of using all of the available data. Bootstrap samples are used with Chernoff inequality to make the samples independent of one another, which is required for streaming (or online) learning. This technique is demonstrated on five medium and two massive datasets.
2. Two prior progressive sampling methodologies and one prior adaptive sampling methodology using Artificial Neural Network (ANN) are investigated and compared with our technique with regard to computational time and number of instances required for convergence.
3. We demonstrate the accuracy of the constructed SVM model (on a 45 nm CMOS inverter) by showing the correlations between the SVM predicted results and the SPICE simulations.

3. Related prior research

3.1. Sampling

During the last decade, managing large data streams has been one of the most time-consuming and non-trivial activities which usually requires expert knowledge. One of the solutions is to sample by analyzing the distribution of data in a database [25], another solution exploits adaptivity to focus on highly informative columns which enables to do away with the usual incoherence assumptions on the row-space while achieving competitive sample complexity bounds [22,26]. Sub-modular functions have been used in connection to data subset selection by connecting sub-modularity to likelihood functions of classifiers [14,27].

3.2. Using machine learning techniques for circuit simulation tools

ANN has been widely used for simulating circuits for tackling the complexity of circuit optimization [19]. It has been used for selecting the channel length and width [4], modeling signal and noise behavior [7], circuit sizing for technology independent circuits [8], for simulation of nanoscale circuits [9] and modeling on-chip spiral inductors [10]. Although ANN has shown to have many benefits, it suffers from being very slow. For fast, yet accurate design optimization, we propose adaptive sampling for improving speed and using support vector machines for improving accuracy.

3.3. Learning curve phenomenon

A learning curve as shown in Fig. 1 shows the relationship between sample size and model accuracy. The horizontal axis represents n , the number of instances in a given dataset, which can vary between 0 and N , the total number of available instances. The vertical axis represents the accuracy of the model produced by a data mining algorithm when given a sample of size n . Learning curves typically have a steeply sloping portion early in the curve, a more gently sloping middle portion, and a plateau late in the curve [6]. The plateau occurs when adding additional data instances does not improve accuracy. When a learning curve reaches its final plateau, we say it is converged. We denote the training set size at which convergence occurs as n_{min} .

Progressive sampling [5,6] starts with a small random sample of data and sequentially adds new data points (using arithmetic or geometric series) until a test of convergence is passed. John and Langley [5] introduced arithmetic sampling which uses a fixed schedule $S_a = \{|D_1|, |D_1| + \beta, |D_1| + 2\beta, \dots, |D_1| + k \cdot \beta\}$, where $|D_1|$

Fig. 1. Hypothetical learning curve.

is the starting sample size, and β is the fixed difference between successive terms. However, as argued by Provost et al. [5] the main drawback of arithmetic sampling is that if n_{min} is a large multiple of $|D_1|$ then the approach will require many runs of the underlying classification mining algorithm. Provost, Jensen and Oates [6] considered using geometric sampling which uses the schedule $S_g = a^k \cdot |D_1| = \{|D_1|, a \cdot |D_1|, a^2 \cdot |D_1|, a^3 \cdot |D_1|, \dots, a^k \cdot |D_1|\}$, where a is the common ratio. They show that geometric schedules are robust for medium to large datasets. One of the limitations of this approach is overshooting. For example, in the KDD CUP dataset, where $n_{min} = 56,600$, the geometric schedule is as follows: $\langle 100, 200, 400, 800, 1600, 3200, 6400, 12800, 25600, 102400 \rangle$. Notice here that the last sample has overshoot n_{min} by 45,800 instances. However, the both sampling schedules are determined a priori, and not dependent on the dataset at hand or the data mining algorithm used.

3.4. Limitations of progressive sampling

There are certain drawbacks of progressive sampling:

- (a) *Sampling schedules determined a priori:* The sample sizes in progressive sampling can be determined beforehand, and are not dependent on the dataset at hand.
- (b) *Overshooting:* As described in the previous section, geometric sampling has the tendency to overshoot the optimum sample size n_{min} as the sample size increases exponentially with each iteration.
- (c) *Sample measure:* There is no measure of uncertainty such as bias or variance used for picking any sample. A good sampling algorithm is expected to have low bias and low sampling variance. These characteristics have not been explored in progressive sampling.
- (d) *Convergence tests:* The tests of convergence used in progressive sampling are not full proof, and the algorithm could converge at a local optimum (if the learning curve is not smooth) instead of the actual global optimum (plateau region).

In order to overcome the limitations of progressive sampling we use adaptive sampling which attempts to determine the optimal sample size n_{min} and SVM. SVM is important because they are robust to very large number of variables and small samples [13], can learn both simple and highly complex classification models and employ sophisticated mathematical principles to avoid overfitting.

4. Problem formulation

Let $\bar{X}$ be the input vector containing the following circuit design parameters:

- (a) the channel width of the NMOS transistor (W_n),
- (b) the channel width of the PMOS transistor and (W_p),
- (c) the output load capacitor (C_L).

Let $\bar{Y}$ be the output vector containing the performance parameters of the design:

- (a) output rise time (τ_R),
- (b) fall time (τ_F),
- (c) inverter switching point (V_{SP}) and
- (d) average power consumption (P_{av}).

Thus the inputs and outputs of the performance model are as follows:

$$\bar{X} = [W_n, W_p, C_L] \quad (2)$$

$$\bar{Y} = [\tau_R, \tau_F, V_{SP}, P_{av}] \quad (3)$$

The performance model is thus written as

$$\bar{Y} = f(\bar{X}) \quad (4)$$

This relationship between the circuit design parameter and the performance parameter is generally strongly nonlinear and multi-dimensional. Traditionally this is evaluated through SPICE simulation. The corresponding support vector machine model is written as

$$\bar{Y} = f_{SVM}(\bar{X}) \quad (5)$$

where f_{SVM} is a support vector machine, $\bar{Y}$ is a q dimensional output vector of the SVM $\bar{X}$ is the SVM input vector. This work, therefore, attempts to construct f_{SVM} such that it is a faithful approximation of the original function f .

5. Adaptive sampling using Chernoff inequality for support vector machines

5.1. Support vector machines vs artificial neural networks

Traditional neural networks have issues with generalization and can produce models that tend to overfit the data. SVMs are based on the Structural Risk Minimization (SRM) principle (i.e. minimizing an upper bound on the empirical risk), which has been shown to be better than the Empirical Risk Minimization (ERM) principle (i.e. minimizing the error on the training data) employed by ANN [16]. This makes the SVM generalize the data better and has outperformed ANN in many applications [17,18].

For evaluating the performance of the SVM classifiers, we see how closely the predicted values are to the actual values by using the following metric: Accuracy (Acc) is defined as the percentage of correctly classified instances in the dataset:

$$Acc = \frac{|T_p| + |T_n|}{N} \quad (6)$$

where $|T_p|$ denotes the number of true positives, $|T_n|$ denotes the number of true negatives and N denotes the size of the test dataset.

5.2. Utility confidence interval and Chernoff inequality

Definition 1 (Utility confidence interval). Let u be the utility function. Let $u(D)$ denote the true quality when using all of the data, and let $\hat{u}(D_i)$ denote its estimated quality based on a sample $D_i \subseteq$

D of size m . Then θ is a utility confidence bound for u iff for any δ , $0 < \delta \leq 1$,

$$Pr[|u(D) - \hat{u}(D_i)| \leq \theta] \geq 1 - \delta \quad (7)$$

Eq. (7) says that θ provides a two-sided confidence interval on $\hat{u}(D_i)$ with confidence δ . In other words, the probability of drawing a sample D_i , such that the difference between the true and estimated utility of any hypothesis disagree by θ or more (in either direction) lies below δ .

The utility function we consider is the average over all instances, of some instance function $f(x_i)$, where $x_i \in D$. The utility is then defined as

$$\hat{u}(D_i) = \frac{1}{|D_i|} \sum_{i=1}^{|D_i|} f(x_i) \quad (8)$$

For the rest of this paper, the utility function that we will be using is the classification accuracy $acc(x_i)$ (i.e. $f(x_i) = acc(x_i)$).

In order to overcome some of the limitations of progressive sampling, we introduce a dynamic adaptive sampling schedule, which selects instances to be included in the sample that depends on data characteristics obtained from the current sample. In our preliminary work in this area [11], we showed that the primary purpose of adaptive (that varies with the problem) sampling is to take advantage of data such as classification accuracy in order to obtain more precise estimates of the next sample. In this paper, we extend our previous approach by using adaptive sampling on SVM by bootstrap sampling for circuit simulation.

Definition 2 (Chernoff inequality [12]). Consider the independent Bernoulli trials $X_1, X_2, \dots, X_m$ with $Pr[X_i = 1] = p$ (probability of success) and $Pr[X_i = 0] = 1 - p$ (probability of failure). Let X be the sum of the outcomes of these m trials: $X = X_1 + X_2 + \dots + X_m$. If we denote p' as $\frac{X}{m}$, then the general form of Chernoff Bounds (where the expected value of X , $E[X] = mp$) is

$$Pr[|p - p'| \geq \epsilon] \leq e^{-\frac{m\epsilon^2}{2}} \quad (9)$$

Chernoff inequality is used to bound the chance that an arbitrary random variable X takes a value that is far away from its expected value $E[X]$. We can combine the two concepts, namely, confidence bounds and the Chernoff inequality because of the following observations:

1. 0–1 loss function for classification accuracy: We treat classification accuracy of each instance as an independent Bernoulli trial which gets 1 if the predicted values equals actual value and 0 otherwise.
2. The utility function is the average over all the instances of the 0–1 loss function, as shown in Eq. (8).

Combining Eqs. (7) and (9) we get

$$Pr[|u(D) - \hat{u}(D_i)| \geq \epsilon] \leq e^{-\frac{m\epsilon^2}{2}} \leq \delta \quad (10)$$

Since we do not know $u(D)$, the utility for the entire dataset, we consider only pairwise samples at stages i and $i - 1$ respectively. We use the terminology $\hat{u}(D_i)$ to represent the utility at stage i and $\hat{u}(D_{i-1})$ the utility at stage $i - 1$ to reflect our approach. The probability of failure that the distance between these utilities differ greater by distance ϵ is

$$Pr[|\hat{u}(D_i) - \hat{u}(D_{i-1})| \geq \epsilon] \quad (11)$$

Using Chernoff inequality (Eq. (9)), we can obtain the number of instances (drawn independently) needed at each iteration to be

$$m \geq \frac{2}{\hat{u}(D_i)} \left[\frac{1}{\epsilon^2} \log \frac{1}{\delta} \right] \quad (12)$$

Algorithm DASA (D, ϵ, δ)**Input:** D (Dataset), ϵ (approximation parameter), δ (probability of failure)**Output:** A (a detected noisy subset of E)1: $\hat{u}(D_0) \leftarrow 0$ 2: Randomly select $\frac{1}{10} * |D_1|$.3: Apply Support Vector Machine on this sample and determine $\hat{u}(D_1)$ using:

$$\hat{u}(D_1) = \frac{1}{|D_1|} \sum_{i=1}^{D_1} acc(x_i)$$

4: **for** each iteration $i(\geq 1)$ **do**

5: Check for convergence using the criteria:

$$\left| \frac{1}{|D_i|} \sum_{i=1}^{|D_i|} acc(x_i) - \frac{1}{|D_{i-1}|} \sum_{i=1}^{|D_{i-1}|} acc(x_i) \right| \leq \epsilon$$

6: If the above convergence test succeeds then we have reached convergence (return the mean computation time and the number of instances. EXIT)

7: Else Compute the next sample size m using:

$$m = \frac{2}{\hat{u}(D_i)} \left[\frac{1}{\epsilon^2} \log \frac{1}{\delta} \right]$$

8: Randomly draw m instances by bootstrapped sampling to form the new sample D_{i+1} 9: Apply SVM on this new sample and determine $\hat{u}(D_{i+1})$ 10: **end for****Fig. 2.** Adaptive sampling algorithm using Chernoff inequality and support vector machine.

An important ingredient for sampling algorithms is how to determine when to stop at a particular sample size. If the expected utility in moving from stage i to $i+1$ is less than a small number, we stop. That is, we would stop when

$$|\hat{u}(D_{i+1}) - \hat{u}(D_i)| \leq \epsilon \quad (13)$$

Using the definition of utility from Eq. (8), and setting $f(x_i) = acc(x_i)$, the classification accuracy (a 0–1 loss function), we obtain (we refer to ϵ as the stopping threshold and Eq. (14) as the stopping criterion)

$$\left| \frac{1}{|D_{i+1}|} \sum_{i=1}^{|D_{i+1}|} acc(x_i) - \frac{1}{|D_i|} \sum_{i=1}^{|D_i|} acc(x_i) \right| \leq \epsilon \quad (14)$$

The dynamic adaptive sampling algorithm is shown in Fig. 2.

5.3. Bootstrap sampling

In order to use the Chernoff bounds to solve for the number of instances at each iteration, we need to draw samples that are independent of one another, so that they can work with an incremental learner like SVM. To achieve this, we use bootstrapping which is basically random sampling with replacement. A great advantage of bootstrap is its simplicity. It is a straightforward way to derive estimates of standard errors and confidence intervals for complex estimators of complex parameters of the distribution, such as percentile points, proportions, odds ratio, and correlation coefficients [15]. Bootstrap is also an appropriate way to control and check the stability of the results. Although, for most problems it is impossible to know the true confidence interval, bootstrap is asymptotically

more accurate than the standard intervals obtained using sample variance and assumptions of normality.

Theorem 1. The $DASA(D, \epsilon, \delta)$ algorithm produces a series of utilities $\hat{u}(D_0), \hat{u}(D_1), \hat{u}(D_2), \dots, \hat{u}(D_m)$, such that,

1. $\hat{u}(D_m) \geq \hat{u}(D_i), 0 \leq i \leq m-1$
2. the sample mean $\frac{1}{n} * \sum_{i=1}^m [\hat{u}(D_{i+1}) - \hat{u}(D_i)]$ converges to the population mean as $n \rightarrow \infty$, where n is the number of samples.

Proof. Our approach moves from D_i to D_{i+1} iff the expected utility $u(D_{i+1})$ is better than $u(D_i)$ by at least ϵ , and hence (1) follows. We define $q_i = \hat{u}(D_{i+1}) - \hat{u}(D_i)$. Let S be the sample mean over n samples given by

$$S = \frac{1}{n} * \sum_{i=1}^m q_i$$

This average tends to the true population mean as $n \rightarrow \infty$ at the rate of convergence given by Chernoff bounds: The probability that “ q_i is more than $\mu + \gamma$ ” goes to 0 exponentially fast as n increases; and for a fixed n , exponentially as γ increases and hence (2) follows. Formally we have

$$Pr[q_i > \mu + \gamma] \leq e^{-2n(\frac{\gamma}{\Lambda})^2}$$

where Λ is the range of possible values for $\hat{u}(D_{i+1}) - \hat{u}(D_i)$. $\square$

6. Empirical results

We present our results to demonstrate the use of dynamic adaptive sampling with SVM by first testing our approach on

Table 1

Comparison of the total number of instances required for the different methods to reach convergence.

Dataset	Full: $S_N = \{N\}$	Geo: $S_g = a^k \cdot D_i $	Chernoff (ANN)	Chernoff (SVM)	Oracle $S_0 = \{n_{min}\}$
LED	100,000	6,300	5,100	3,193	2,000
Waveform	100,000	25,500	16,108	14,294	12,000
Census	32,000	25,500	10,014	7,324	8,000
Amazon access	30,000	26,200	12,453	10,453	7,700
Bank marketing	45,211	42,891	30,295	29,027	27,934
KDD cup	235,000	204,700	67,800	61,320	56,600
NASA HTTP	461,612	409,500	158,345	149,035	130,645
US census	2,458,285	2,001,492	1,003,343	908,921	830,423
Youtube comedy slam	1,138,562	945,872	532,187	501,392	482,871

Table 2

Comparison of the mean computational time (in CPU seconds) required for the different methods to obtain the same accuracy (averaged over 20 runs of the experiment).

Dataset	Full: $S_N = \{N\}$	Geo: $S_g = a^k \cdot D_i $	Chernoff (ANN)	(SVM)	Oracle $S_0 = \{n_{min}\}$
LED	46.51	15.67	25.87	15.34	5.72
Waveform	558.91	89.76	156.73	97.38	32.85
Census	48.76	10.77	27.84	20.14	13.87
Amazon access	32.98	12.45	21.44	15.31	8.56
Bank marketing	40.23	17.98	29.44	20.35	15.33
KDD cup	17870.59	5616.89	3116.84	2518.31	1826.16
NASA HTTP	38160.78	13482.49	8713.00	6126.74	4719.85
US census	179729.34	148923.35	94728.31	72519.44	69293.11
Youtube comedy slam	82491.51	26341.23	14120.57	13108.22	10923.13

some datasets from UCI repository [23] and real world web trace datasets [24]. We used WEKA platform for our experiments [28]. In WEKA 3.6.4, the implementation of SVM is called SMO; the implementation of C4.5 Decision Tree is J48. In our experiments, we ran these implemented methods using our prepared DASA code (written in Java) on the WEKA platform. We then show in the following section that this technique works with modeling CMOS inverters.

6.1. SVM with Chernoff bounds on medium and massive datasets

We present our technique of using SVM on five medium sized datasets from the UCI repository [23]: LED, WAVEFORM, AMAZON, BANK and CENSUS (adult) and two large datasets YOUTUBE and US CENSUS. We also use two massive real world Web traces: KDD CUP 2000 and NASA-HTTP [24]. We use LS-SVM learner for all the datasets. We compare our method with other methods (Full, Geo and Oracle), and the results are shown in Tables 1 and 2. We compare the different approaches with our method using the following two performance criteria.

- The mean computational time:** The runtimes were averaged over 20 runs of each of the five datasets. The experiments were run on a 3.00 GHz Pentium Dual Core CPU with 1.96 GM of RAM running under the x86 Windows Operating System. Linux time command was used to produce the results in Table 2.
- The total number of instances needed to converge:** If the sampling schedule is $S = \langle |D_1|, |D_2|, |D_3|, \dots, |D_k| \rangle$, then the total number of instances would be $|D_1| + |D_2| + |D_3| + \dots + |D_k|$.

We compare our convergence method with other methods, which are as follows:

- (Full):** $S_N = \{N\}$, a single sample with all the instances. This is the most commonly used method. This method suffers from both speed and memory drawbacks.
- (Geo):** Geometric Sampling [6], in which the sample size is created geometrically, $S_g = \{|D_1|, a \cdot |D_1|, a^2 \cdot |D_1|, \dots, a^k \cdot |D_1|\}$. We use $|D_1| = 100$ and $a = 2$ as used by Provost et al. [6].

Table 3

Range of circuit design parameters.

Parameters	Min	Max
$W_n(nm)$	90	1000
$W_p(nm)$	90	1000
$C_i(pf)$	1	5

- (Chernoff using ANN):** Adaptive sampling using Chernoff bounds [11], where we use $\epsilon = 0.001$ and $\delta = 0.05$ (95% probability).
- (Chernoff using SVM and Bootstrap Sampling):** Adaptive sampling using Chernoff bounds and SVM, where we use $\epsilon = 0.001$ and $\delta = 0.05$ (95% probability).
- (Oracle):** $S_0 = \{n_{min}\}$, the optimal sample size determined by the omniscient oracle; we determined n_{min} empirically by analyzing the full learning curve beforehand.

6.2. SVM with Chernoff inequality on CMOS inverter:

Data Generation: For our data generation, we used CMOS inverters that are constructed corresponding to the circuit design parameters listed in Table 3. The channel length of both the transistors is fixed at 45 nm which is the minimum of the process technology. The other process technology parameters are taken from Berkeley Predictive Technology model file [20]. Based on Halton sequence generator [21], uniformly distributed samples are generated within the specified range. T-SPIICE simulation is used to generate the data corresponding to those sample points. Transient analysis and DC transfer sweep analysis are performed in order to extract the performance parameters.

Scaling: SVMs assume that the data it works with is in a standard range, usually either 0 to 1, or -1 to 1 (roughly). Therefore, the normalization of feature vectors prior to feeding them to the SVM is very important. (This is often called whitening, although there are different types of whitening.) Hence, for each dimension, the values are scaled to lie roughly within this range. It is observed from Table 3 that the input parameters vary over a wide range. Similarly, the output performance parameters vary over a wide range. Therefore, data scaling is required for efficient construction

Fig. 3. Adaptive sampling algorithm using Chernoff inequality and support vector machine.

Table 4

Comparison of the total number of instances required for the different methods to reach convergence.

Dataset	Full: $S_N = \{N\}$	Arith: $S_a = D_1 + k \cdot \beta$	Geo: $S_g = a^k \cdot D_1 $	Chernoff(ANN)	Chernoff (SVM)	Oracle $S_0 = \{n_{min}\}$
T-SPICE	1030	1000	800	752	672	659

Table 5

Comparison of the cpu time required for the different methods to reach convergence.

Dataset	Full: $S_N = \{N\}$	Arith: $S_a = D_1 + k \cdot \beta$	Geo: $S_g = a^k \cdot D_1 $	Chernoff (ANN)	Chernoff (SVM)	Oracle $S_0 = \{n_{min}\}$
T-SPICE	693.2	178.39	152.9	142.8	108.4	64.2

of the SVM model. In this work, we have used linear scaling of the data between 0 and 1, described by the following formula:

$$\tilde{x} = \tilde{x}_{min} + \frac{x - x_{min}}{x_{max} - x_{min}} (\tilde{x}_{max} - \tilde{x}_{min}) \quad (15)$$

And the corresponding de-scaling formula is given by

$$x = x_{min} + \frac{\tilde{x} - \tilde{x}_{min}}{\tilde{x}_{max} - \tilde{x}_{min}} (\tilde{x}_{max} - \tilde{x}_{min}) \quad (16)$$

where x , x_{min} , x_{max} represent the original data and $\tilde{x}$, $\tilde{x}_{min}$, $\tilde{x}_{max}$ represent the scaled data.

Data Organization: k -fold cross-validation: We use k -fold cross-validation, in which the original sample is randomly partitioned into k equal sized sub-samples. Of the k sub-samples, a single sub-sample is retained as the validation data for testing the model, and the remaining $k - 1$ sub-samples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k sub-samples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once. In this paper, we use 10-fold cross-validation.

Results: The learning curve results of the three sampling techniques are shown in Figs. 3(a)–3(d) for each of the chosen per-

formance parameters. Arithmetic samples reaches convergence (or the plateau region of the learning curve) using the most number of iterations as shown in Table 4. Geometric sampling overshoots our optimum n_{min} by many instances. In our approach at each iteration, we draw a new sample (bootstrapped) of size $|D_{i+1}|$ based on our DASA algorithm. This sample will be based on the dataset on hand, and not independent as with progressive sampling approaches. Table 5 shows the times used by the different approaches to reach convergence.

Fig. 4 shows the scatter plots between the SVM predicted values and the SPICE simulation. We notice that there is nearly perfect correlation with unity correlation coefficient. These demonstrate the accuracy of the constructed SVM model.

7. Conclusion

Circuit simulation tools have access to massive amounts of data that are not only dynamic but generated at high speed in real time, hence making fast simulation a bottleneck in integrated circuit design. Using all the available data is prohibitive due to memory and time constraints. In this paper, we use an adaptive sampling technique and demonstrate that it enhances processing large datasets.

One of the defining problems of data mining is determining the smallest training set size for massive datasets. We attempt to address this problem for massive datasets using our incremental

Fig. 4. Adaptive sampling algorithm using Chernoff inequality and support vector machine.

dynamic adaptive sampling approach and demonstrate it on CMOS inverters for SVM learners, that addresses two key questions: 1) Has convergence occurred and 2) If not, how many more are required at each iteration of the algorithm. In this paper we address these two questions together using Chernoff bounds and bootstrapping. We have shown that convergence detection can be done effectively and moderately efficiently using the adaptive technique. Our empirical results show that although geometric sampling performs well for medium sized datasets in terms of computational size required, the adaptive approach outperforms other approaches in the case of massive datasets.

In our future work we will work on the challenges that the introduction of simulation-based analog synthesis tools creates for analog modeling. These tools routinely visit 10^3 to 10^5 fully simulated circuit solution candidates. We will extend our current work to adaptively sample from large-scale data mining to build models that capture significant regions of this visited performance space, parameterized by variables manipulated by synthesis, trained by the data points visited during synthesis. Our future work will also focus on challenge from several application domains which exhibit the property of inherent application resilience, offering entirely new avenues for performance and power optimization by relaxing the conventional requirement of exact (numerical or Boolean) equivalence between the specification and hardware implementation.

References

- [1] N. Vladimir, Vapnik: Statistical Learning Theory, John Wiley and Sons, New York, NY, 1998.
- [2] L.G. Valiant, A theory of the learnable, Commun. ACM 27 (11) (1984) 1134–1142.
- [3] P. Tryfos, Sampling Methods for Applied Research: Text and Cases, John Wiley and Sons, Inc, New York, NY, 1996, p. 440.
- [4] M. Avci, T. Yildirim, Neural network based MOS transistor geometry decision for TSMC 0.18 process terminology, in: Proc of ICCS, 2006, pp. 615–622.
- [5] G.H. John, P. Langley, Static versus dynamic sampling for data mining, in: KDD, 96, 1996, August, pp. 367–370.
- [6] F. Provost, D. Jensen, T. Oates, Efficient progressive sampling, Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 1999, August, pp. 23–32.
- [7] F. Güneş, F. Gürgeç, H. Torpi, Signal-noise neural network model for active microwave devices, IEE Proc. - Circuits Devices Syst. 143 (1) (1996) 1–8.IET.
- [8] N. Kahraman, T. Yildirim, Technology independent circuit sizing for standard cell based design using neural network, Digit. Signal Process. 19 (1996) 1–8.
- [9] F. Djefalla, M. Chahdib, A. Benhayat, M.L. Hafiane, An approach based on neural computation to simulate the nanoscale CMOS circuit, Solid State Electron. 51 (2007) 48–56.
- [10] S.K. Mandal, S. Sural, A. Patra, ANN- and PSO-based synthesis of on-chip spiral inductors for RF ICs, IEEE Trans. CADICS 27 (2008) 188–192.
- [11] A. Satyanarayana, I. Davidson, A dynamic adaptive sampling algorithm (DASA) for real world applications: finger print recognition and face recognition, in: Proc. of 15th International Symposium on Methodologies for Intelligent Systems (ISMIS 2005), 2005, pp. 631–640.
- [12] H. Chernoff, A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations, Ann. Math. Stat. 23 (1952) 493–507.
- [13] S. Klement, A.M. Mamlouk, T. Martinez, Reliability of cross-validation for SVMs in high-dimensional, low sample size scenarios, Artificial Neural Networks-ICANN 2008, Springer, Berlin Heidelberg, 2008, pp. 41–50.
- [14] K. Wei, Y. Liu, K. Kirchhoff, C. Bartels, J. Bilmes, Submodular subset selection for large-scale speech training data, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2014, IEEE, 2014, pp. 3311–3315.
- [15] I. Davidson, An ensemble approach for stable learners, Proc. The National Conference on A.I. (AAAI), San Jose, 2004.
- [16] C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery, Volume 2, Kluwer Academic Publishers, Boston, 1998.
- [17] B. Pradhan, A comparative study on the predictive ability of the decision tree, support vector machine and neuro-fuzzy models in landslide susceptibility mapping using GIS, Comput. Geosci. 51 (2013) 350–365.
- [18] R.M. Balabin, E.I. Lomakina, Support vector machine regression (LS-SVM) an alternative to artificial neural networks (ANNs) for the analysis of quantum chemistry data? Phys. Chem. Chem. Phys. 13 (24) (2011) 11710–11718.
- [19] D. Dhabak, S. Pandit, Adaptive sampling algorithm for ANN-based performance modeling of nano-scale CMOS inverter, World Acad. Sci. Eng. Technol. 80 (2011).
- [20] W. Zhao, Y. Cao, New generations of predictive technology model for sub-45nm early design exploration, IEEE Trans. Electron Devices 53 (2006) 2816–2823.
- [21] L. Kuipers, H. Niederreiter, Uniform distribution of sequences, Dover Publications, 2006.
- [22] A. Satyanarayana, Intelligent sampling for big data using bootstrap sampling and chebyshev inequality, 2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE), IEEE, 2014, pp. 1–6.
- [23] M. Lichman, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2013. <http://archive.ics.uci.edu/ml>.

- [24] NASA-HTTP Logs. HTTP Requests to the NASA Kennedy Space Center WWW Server in Florida (1995), <http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html>.
- [25] J. Murphy, M. Kristiansen, Towards realistic sampling: generating dependencies in a relational database, Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, ACM, 2013, p. 12.
- [26] A. Krishnamurthy, A. Singh, Low-rank matrix and tensor completion via adaptive sampling, in: Advances in Neural Information Processing Systems, 2013, pp. 836–844.
- [27] K. Wei, R. Iyer, J. Bilmes, Submodularity in data subset selection and active learning, ICML, 2015, 2015.
- [28] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. Witten, The WEKA data mining software: An update, SIGKDD Explorations, 11, 2009.

Ashwin Satyanarayana is currently an Assistant Professor with the Department of Computer Systems Technology, New York City College of Technology (CUNY). Prior to this, Dr. Satyanarayana was a Research Scientist at Microsoft in Seattle from 2006 to 2012, where he worked on several Big Data problems including Query Reformulation on Microsoft's search engine Bing. He holds a PhD in Computer Science (Data Mining) from SUNY, with particular emphasis on Data Mining, Machine Learning and Applied Probability with applications in Real World Learning Problems. He is an author or co-author of over 20 peer reviewed journal and conference publications and co-authored a textbook—"Essential Aspects of Physical Design and Implementation of Relational Databases." He has four patents in the area of Search Engine research. He is also a recipient of the Math Olympiad Award, and is currently serving as Secretary/Treasurer of the ASEE (American Society of Engineering Education) Mid-Atlantic Conference.