An Outlier Detection Method and its Application to Multicore-Chip Power Estimation

Yaguang Li, Pingqiang Zhou

School of Information Science and Technology, ShanghaiTech University, Shanghai, China
{liyg, zhoupq}@shanghaitech.edu.cn

Abstract—Machine learning has been recently applied to solve challenging research problems in the EDA area. The performance of the machine learning algorithms are vulnerable to effects such as the outliers and the insignificant features in the input training data set. In this paper, we propose a model averaging method, together with an outlier detection method, to make the machine learning process more robust to such contaminating effects. Results on artificial and chip power estimation data sets show that our method behave much better than the conventional ordinary least square method which is widely used in the machine learning community.

1. Introduction

Machine learning [1], [2] has been recently applied to tackle challenging research problems in the EDA area, such as high level synthesis [3], physical design [4], circuit test [5], [6] and design for manufacturing [7]–[9]. Prior works [10], [11] compare the sensitivity of different machine learning methods to the outliers (also known as noise), and their results show that when the noise level increase, the accuracy of the models generated by the machine learning methods deteriorate fast. Therefore, to ensure the machine learning methods behave well, all these work require *clean* training sets to fit the models which are typically known as prior knowledge in the EDA community.

However, in the real world it is inevitable that the collected data sets contain a certain level of noise, which arises because of human error, instrument error, fraudulent behavior and faults in systems, etc. [12] Prior work on purifying the data can be roughly categorized into parametric methods and non-parametric methods. Different from non-parametric methods, parametric methods need a priori data knowledge [12], i.e., a specific distribution model. For instance, Minimum Volume Ellipsoid estimation (MVE) [13] fits the smallest permissible ellipsoid volume around the majority of the data.

In this work, we propose a parametric method named Outlier Detection Method (ODM) for building robust models in presence of outliers. Different from [14] which removes one outlier after another, in ODM we use a user-defined parameter to isolate the outliers. To further minimize the uncertainty brought by the insignificant features in the data, upon top of ODM, we propose AODM, the averaged ODM method, which tries to build a weighted model over all the possible feature sets. Finally, we apply our AODM method to several artificial data sets and also one of the important research problems in EDA – the power estimation of a muliticore chip – and show its effectiveness by comparing its performance with the commonly used Ordinary Least Square (OLS) method in machine learning.

The rest of the paper is organized as follows. We introduce preliminaries about power estimation and OLS in Section 2. We then discuss our AODM method in Section 3. After that we verify the effectiveness of our proposed methodology by results presented in Section 4. Finally, we make conclusions in Section 5.

2. Preliminaries

Fast power analysis is critical for performance and reliability optimization in a multicore chip. Rather than running slow micro-architectural-level and RTL-level simulations, in the past decade, people have explored the usage of performance counters to estimate the chip power [15]–[20]. Their results show that a linear model like

$$p = \sum_{i=1}^{K} x_i \cdot c_i + b \tag{1}$$

is sufficient for system-level power estimation. In (1), p is the power consumption of a processor chip, K is the number of available performance counters, c_i is the value for the i-th counter, x_i is the coefficient for i-th counter and b is a constant.

Given the fact that it is easy to obtain enough data sets of (counter, power) by running a certain period of simulations, the coefficients in model (1) can be trained by regression methods such as Ordinary Least Square (OLS) in machine learning, as [15], [18]

$$\min_{x,b} \|P - Cx - B\|_2 \tag{2}$$

978-1-5090-6625-4/17/$31.00 ©2017 IEEE

$$\boldsymbol{C} = \begin{bmatrix} c_1^{(1)} & c_2^{(1)} & \dots & c_M^{(1)} \\ c_1^{(2)} & c_2^{(2)} & \dots & c_M^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ c_1^{(N)} & c_2^{(N)} & \dots & c_M^{(N)} \end{bmatrix} \quad (3)$$

$$\boldsymbol{x} = [x_1 \quad x_2 \quad \dots \quad x_M]^T \quad (4)$$

$$\boldsymbol{B} = [b \quad b \quad \dots \quad b]^T \quad (5)$$

Matrix C denotes N samples with M features ($i.e.$, performance counters) that may contribute to the power for the functional block or system of interest. P denotes the corresponding power consumption of each sample, thus an $N*1$ vector. x denotes the coefficient vector and B is a constant vector of b in (1).

3. Proposed Method

Suppose a training set $\{X, y\}$ is given for a machine learning process, where

- matrix $X \in R^{N \times M}$ is the training data, whose each row represents a training sample and every column represents a training feature, and
- $y \in R^N$ is the corresponding vector of training labels.

The performance of a machine learning approach used for model fitting is affected by two factors:

- *Outliers in given training set*: There may exist mismatch between an observed label and the real label, which has contaminating effect on the model accuracy.
- *Uncertainty in features*: the training data X may contain both significant and insignificant features. In high dimension space, the dimension of insignificant features might disturb the learning process, thus diverge the learned model from real model, which decreases the performance of a trained model.

We present our solutions to minimizing the side effect of the above two factors respectively in Section 3.1 and Section 3.2.

3.1 Outlier Detection Method (ODM)

The modeling process of applying a machine learning approach can be theoretically represented as

$$y = f(X) + \epsilon \quad (6)$$

where $f(X)$ represents a certain relationship (more specifically, model) between X and y, and ϵ is the model error which reflects the uncertainty of the training process when building a model from X and y, and is often modeled as a Gaussian variable, i.e. $x \sim N(0, \sigma^2)$.

As stated before, due to the existence of the outliers in the input data set $\{X, y\}$, the trained model $f(X)$ may not be accurate. So the question is, given the noise in the data set, how to learn an accurate model using a machine learning algorithm? A conventional way to solve this problem is to minimize a least square problem as

$$\tilde{f}(X) = \arg\min_f \|y - f(X)\|_2 \quad (7)$$

where $\tilde{f}(X)$ is the optimal model trained on the data set $\{X, y\}$. However, such approach fails to exclude the outliers in building the model and therefore suffers from accuracy issue. In our work, we propose to address this issue by introducing an outlier detection matrix Q to the problem (7) as

$$\min_f \|Q(y - f(X))\|_2 \quad (8)$$

where Q is a diagonal matrix whose element $Q(i,i)$ indicates whether the i-th training sample $\{X(i,:), y(i,1)\}$ is an outlier point.

The matrix Q is defined as follows

$$Q = \begin{bmatrix} \varphi(e_1) & 0 & \cdots & 0 \\ 0 & \varphi(e_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \varphi(e_m) \end{bmatrix} \quad (9)$$

where vector

$$e = y - \tilde{f}(X) \quad (10)$$

represents the residual vector between training label vector y and the value predicted by model $\tilde{f}(X)$ (See Equation (7)), and $\varphi(x)$ is an outlier sample detection function defined as

$$\varphi(x) = \begin{cases} 0 & , |x| > \alpha\sigma \\ 1 & , else \end{cases} \quad (11)$$

In our work, to get matrix Q, we first solve the traditional problem (7) to get an inaccurate model, then use Equation (10) to find the residual vector e. Finally, we can obtain matrix Q using Equation (11).

In Equation (11), α is a user-defined parameter to filter out the outliers. A small α will identify more samples as outliers, while a big α will be more strict to identify outliers. By introducing an outlier matrix Q in Equation (8), we isolate outliers samples in the machine learning processing, thus result in improved model performance. For the special case of $Q = I$, our method is degenerated to the conventional method shown in (7).

3.2 Average Outlier Detection Method (AODM)

In Section 3.1, we discuss our solution to the issue of outliers in the input training data set. In this section, we present our solution to the uncertainty arises from the feature selection process. Traditionally, a good training set is often given by a specialist. However, if this is not the case and the training set has unimportant or even self-contradicting features, then a robust method is required to

build a good model on a bad training set. In our work, we solve this problem by a general model averaging method that is adapted from Bayesian Average Modeling [21].

Suppose Σ_k is another diagonal feature selection matrix whose i-th element is 1 only if the i-th feature in X is selected. Based on the ODM model presented in Equation (8), we can define the new outlier detection model with feature selection as

$$E_k = \frac{1}{\text{Trace}(\Sigma_k)} \min_f \|Q(y - f(X\Sigma_k))\|_2 \qquad (12)$$

where E_k is the error in the best model found by the least square method, normalized by the number of selected features for the modeling. Then the model averaged over all the possible feature selection results can be written as

$$M_{average} = \frac{\sum_{k=1}^{2^M} \frac{1}{E_k} M_k}{\sum_{j=1}^{2^M} \frac{1}{E_j}} \qquad (13)$$

where M_k represents the model coefficients trained from the data set $\{X\Sigma_k$,y$\}$. Note that $X\Sigma_k$ is always a $N \times M$ matrix, hence the number of coefficients for models of different selected feature set sizes is the same and thus addable. Besides, averaging over all the models provides better average predictive ability than using any single model [22], this is because, by building an averaged weighted model, model M_k with big E_k has small contribution to the averaged model $M_{average}$, hence uncertainty in X can be eliminated.

4. Experimental Results

In this section, we will present results to show the effectiveness of our proposed methods and show how AODM can be used to build a better power prediction model in a multicore chip.

4.1 Effectiveness of our ODM and AODM Methods

In this section we compare our methods with traditional machine learning method OLS (see Section 2) on artificial data sets.

It should be noted that, α in Equation (11) is determined by the given training set $\{X, y\}$ and largely affects the the model accuracy. If outliers are far from inliers, then a big α may work. In contrast, if outliers are not far from inliers, a big α makes the model fail to find outliers, thus its performance will be the same as OLS. In this section, we found that $\alpha = 0.6$ worked well for our experiments.

Fig. 1 shows the results. We artificially added some noise to a real model and get the observed values. Then we trained a model with $\{X, y\}$ by ODM and OLS respectively. As can be seen in Fig. 1, our method gives a better-explained model than OLS, which is more closer to the the real model. We also compare the L2 norm of the predicted result to the real value in Table 1, results

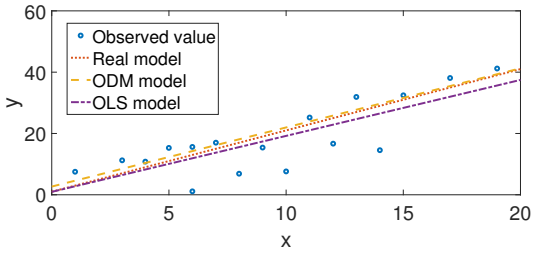

Figure 1. Comparison between our ODM method and OLS.

show that our ODM is more robust to the presence of outliers than OLS.

Table 1
Comparison between ODM and OLS.

	ODM	OLS
$\|\bullet\|_2$	4.40	7.80

Further, we extend the training set to 2D dimension. We use the same training data $x1$ in Fig. 1 and add an uncorrelated randomly sampled data $x2$. Fig. 2 shows the results. As can be seen, the learned model by our AODM method (AODM model plane) is between real model plane and OLS model plane, which indicates that our AODM method is more robust than OLS. This is also confirmed by the data results presented in Table 2.

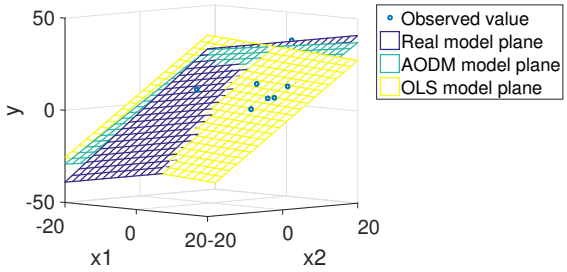

Figure 2. Observed value, real plane and trained planes.

Table 2
Comparison between AODM and OLS.

	AODM	OLS
$\|\bullet\|_2$	7.30	15.59

4.2 Application to Power Prediction

In this section, we further apply our AODM method to power modeling.

We use a full system multicore simulator GEM5 [23] to simulate the standard SPEC CPU2006 [24] benchmarks on a multicore chip to generate the performance counters. We then use McPAT [25] to generate the power profiles of the processor chip. Since the power profiles are simulated and no real power information is available, in order to compare the result of our method and OLS, a criterion is applied to estimate the result as

$$E_{method} = \frac{1}{\text{Trace}(Q)} \|Q(y - f_{method}(X))\|_2 \qquad (14)$$

In Equation (14), E_{method} and f_{method} represent the optimal error and used method (AODM or OLS in our experiment) respectively. Note that this criterion does not include the outliers. To compare the performance between AOMD and OLS, we use the same testing samples in our experiment, hence Q_{LSQ} is the same with Q_{AODM} at a certain α. Fig.3 shows the results

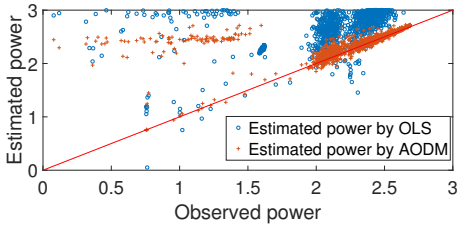

Figure 3. Observed power and estimated power.

when $\alpha = 0.4$. As we can seen, due to the outliers and uncertainty feature in feature matrix C (see Equation (3)), model solved by OLS is skewed. In contrast, our method is more robust and it can build well-explained model for majority of the observed power samples.

Further, we present how α affects the model accuracy as represented by E_{method} in Fig.4. When α is

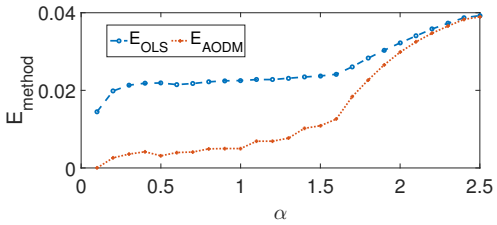

Figure 4. Relationship between E_{method} and α.

small, $E_{ourmethod}$ is much smaller then E_{OLS}. When α increases, $E_{ourmethod}$ gets closer to E_{OLS}. This is because the outliers are not distinguished due to large α. But when α is large, $E_{ourmethod}$ is still lower than E_{OLS} because of the elimination of insignificant feature by our model averaging strategy.

5. Conclusion

Performance of a model found by a machine learning approach is pre-limited by the quality of the input training set. In this paper, we propose two methods to address the side effects caused by the uncertainty of training set: an outlier detection method to minimize the disturbance of learning process caused by the noise in the training data, and a model averaging method to control the disturbance caused by the unnecessary features in the training data. Experiment results that our methods are more robust than traditional method, resulting in better-explained models.

References

[1] T. M. Mitchell, "Machine learning." *New York, NY, USA: McGraw-Hill,*, 1997.

[2] C. M. Bishop, "Pattern recognition and machine learning (information science and statistics)." *Secaucus, NJ, USA: Springer*, 2006.

[3] H. Y. Liu *et al.*, "On learning-based methods for design-space exploration with high-level synthesis," in *Design Automation Conference*, 2013, pp. 1–7.

[4] B. Yu *et al.*, "Machine learning and pattern matching in physical design," in *Design Automation Conference*, 2015, pp. 286–293.

[5] L. C. Wang, "Data mining in functional test content optimization," in *Design Automation Conference*, 2015, pp. 308–315.

[6] F. Ye *et al.*, "Self-learning and adaptive board-level functional fault diagnosis," in *Design Automation Conference*, 2015, pp. 294–301.

[7] S. Y. Lin *et al.*, "A novel fuzzy matching model for lithography hotspot detection," in *Design Automation Conference*, 2013, pp. 1–6.

[8] Y. T. Yu *et al.*, "Machine-learning-based hotspot detection using topological classification and critical feature extraction," in *Design Automation Conference*, 2013, pp. 1–6.

[9] Z. Xiao *et al.*, "Directed self-assembly (dsa) template pattern verification," in *Design Automation Conference*, 2014, pp. 1–6.

[10] A. Atla *et al.*, "Sensitivity of different machine learning algorithms to noise," *Journal of Computing Sciences in Colleges*, vol. 26, no. 5, pp. 96–103, 2011.

[11] E. Kalapanidas *et al.*, "Machine learning algorithms: A study on noise sensitivity," in *Proc. 1st Balcan Conference in Informatics*, 2003, pp. 356–365.

[12] V. Hodge *et al.*, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[13] G. F. Piepel, *Robust Regression and Outlier Detection*. Wiley-Interscience, 2003.

[14] P. S. Schenker, "Outlier detection and motion segmentation," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 2059, pp. 432–443, 1993.

[15] G. Contreras *et al.*, "Power prediction for intel xscale®; processors using performance monitoring unit events," in *Proceedings of the 2005 International Symposium on Low Power Electronics and Design*, 2005, pp. 221–226.

[16] W. L. Bircher *et al.*, "Runtime identification of microprocessor energy saving opportunities," in *Proceedings of the 2005 International Symposium on Low Power Electronics and Design*, 2005, pp. 275–280.

[17] K. Rajamani *et al.*, "Online power and performance estimation for dynamic power management," IBM Research Division, Tech. Rep., July 2006.

[18] B. C. Lee *et al.*, "Accurate and efficient regression modeling for microarchitectural performance and power prediction," *Acm Sigarch Computer Architecture News*, vol. 34, no. 5, pp. 185–194, 2006.

[19] R. Rodrigues *et al.*, "A study on the use of performance counters to estimate power in microprocessors," *IEEE Transactions on Circuits Systems II Express Briefs*, vol. 60, no. 12, pp. 882–886, 2013.

[20] B. Ramon *et al.*, "Counter-based power modeling methods," *Computer Journal*, vol. 56, no. 2, pp. 198–213, Feb. 2013.

[21] A. H. Jennifer *et al.*, "Bayesian model averaging: a tutorial," *Statistical Science*, vol. 14, no. 4, pp. 382–417, 1999.

[22] D. Madigan *et al.*, "Model selection and accounting for model uncertainty in graphical models using occam's window," *Journal of the American Statistical Association*, vol. 89, no. 428, pp. 1535–1546, 1994.

[23] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[24] http://www.spec.org/cpu2006/, "[online]."

[25] L. Sheng *et al.*, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," *IEEE Micro*, pp. 469–480, 2009.