Machine Learning and Pattern Matching in Physical Design

Bei Yu[1], David Z. Pan[1], Tetsuaki Matsunawa[2], and Xuan Zeng[3]

[1]ECE Department, University of Texas at Austin, Austin, TX, USA
[2]Center for Semiconductor Research and Development, Toshiba Corp., Kawasaki, Japan
[3]State Key Laboratory of ASIC & Systems, Microelectronics Department, Fudan University, China
{bei,dpan}@cerc.utexas.edu, tetsuaki.matsunawa@toshiba.co.jp, xzeng@fudan.edu.cn

Abstract

Machine learning (ML) and pattern matching (PM) are powerful computer science techniques which can derive knowledge from big data, and provide prediction and matching. Since nanometer VLSI design and manufacturing have extremely high complexity and gigantic data, there has been a surge recently in applying and adapting machine learning and pattern matching techniques in VLSI physical design (including physical verification), e.g., lithography hotspot detection and data/pattern-driven physical design, as ML and PM can raise the level of abstraction from detailed physics-based simulations and provide reasonably good quality-of-result. In this paper, we will discuss key techniques and recent results of machine learning and pattern matching, with their applications in physical design.

I. Introduction

As the feature size of semiconductor process technology nodes further scales down, the industry is greatly challenged in terms of manufacturing and design [1]. On one hand, since the mainstream lithography technology is still limited by the 193nm wavelength lithography, what-you-see at the design is not necessarily what-you-get at the fab. Even with complicated design rules, various resolution enhancement techniques (RETs), and multiple patterning lithography (MPL) techniques, there may still be complex layout-dependent lithography hotspots which may cause opens, shorts, and yield loss. Therefore, how to detect and remove lithography hotspots during physical design is critical to ensure high yield. On the other hand, since modern VLSI circuits have billions of transistors/interconnects, it is simply impossible to run detailed lithography and other physics-based simulations at the full-chip scale during the physical design stage to deal with potential lithography hotspots and other design constraints.

Machine learning and pattern matching techniques provide reasonably good abstraction and quality-of-result, which make them suitable to perform lithography hotspot detection, as shown by many recent studies. Meanwhile, the machine learning and pattern matching principles have also been used for lithography aware routing and other areas of physical design, such as datapath placement and clock optimization with success. In this paper, we will present some commonly used machine learning and patterning matching techniques, and discuss their applications in physical design and verification.

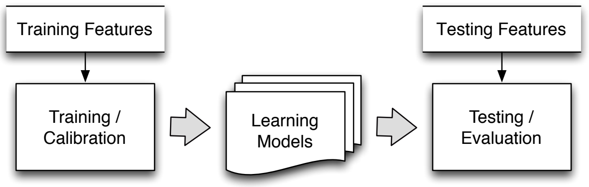


Fig. 1. Classical supervised learning flow consists of training and testing stages.

The rest of the paper is organized as follows. In Section II we introduce some basic algorithms of machine learning and pattern matching. In Section III we discuss the physical design applications. In Section IV we will further discuss some advanced issues. Section V concludes this paper.

II. Basic Algorithms

A. Machine Learning Techniques

Machine learning is a computer science discipline that deals with the construction and study of algorithms that can learn from data [2]. The widely applied machine learning techniques in physical design is called *supervised learning*, where the training data contains explicit examples of what the correct output should be for given inputs. A classical supervised learning flow is illustrated in Fig. 1 consisting of training/calibration and testing/evaluation stages. In the training stage, given the input training features a set of learning models are built. In the testing stage, the constructed learning models are to make predictions or decisions, rather than following only explicitly programmed instructions. In this subsection we introduce some typical and popular learning models in supervised learning.

A.1 Artificial Neural Network (ANN)

The artificial neural network (ANN) is inspired by human brain to estimate or approximate functions that can depend on a large number of inputs and the details of the functions are usually unknown. ANN creates a leveled network of neurons, and each neuron i can be trained to model a function $f_i : X \rightarrow Y$ through backpropagation algorithm [3]. An ANN with multiple levels can thus model highly non-linear functions. The input features are presented to ANN via the input layer, which

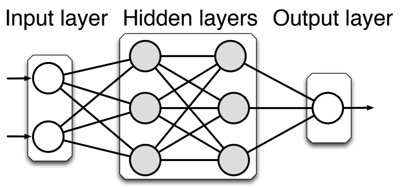


Fig. 2. An example of four layer ANN.

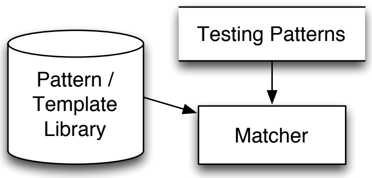


Fig. 3. Classical pattern matching flow.

communicates to one or more hidden layers where the actual processing is done. Then the hidden layers are connected to an output layer, which scales its input to the desired response. An example of a four layer feedforward ANN is shown in Fig. 2.

Compared with other machine learning techniques, ANN usually has good noise-robustness. However, it takes more time in the training and calibration to reach optimal or close-to-optimal solutions.

A.2 Support Vector Machine (SVM)

The support vector machine (SVM) is one of the most popular classification and learning techniques. In SVM, data vectors are mapped into a higher-dimensional space using a kernel function, and an optimal linear discrimination function in the space or an optimal hyperplane that fits the training data is built [4]. The objective is to maximize the margin between the separating hyper-plane and the nearest data vectors from both classes. If the data is not separable, a C-type SVM can be applied to the classification problem as follows [5].

$$\begin{array}{rll} \min\limits_{\alpha}: & \frac{1}{2}\alpha^T Q\alpha - e^T\alpha & \\ \text{s.t.} & z^T\alpha = 0 & \\ & Q_{ij} = z_i z_j K(x_i, x_j) & i,j = 1,\ldots,n \\ & 0 \le \alpha_i \le C, & i = 1,\ldots,n \end{array} \tag{1}$$

where $x_i \in \mathbf{R}^d$ are training vectors, $z \in \{1, -1\}$ is indicator vector. e is the vector of all ones, Q is an $n \times n$ positive semidefinite matrix, and the parameter C controls the trade-off between allowing training errors and forcing rigid separating margins. For each element $Q_{ij} \in Q$, $Q_{ij} = z_i z_j K(x_i, x_j)$. The kernel function K maps the data into the different space so a hyperplane can be used to do the separation. After solving the optimization problem in (1), all $\alpha_i \in \alpha$ can be calculated. Then given a new testing vector t, the decision function is sgn$(\sum_i z_i\alpha_i K(x_i, t))$, where sgn function is defined as follows:

$$\text{sgn}(x) = \begin{cases} -1, & \text{if } x \le 0 \\ 1, & \text{if } x > 0 \end{cases}$$

In theory, SVM guarantees the global optimum but is sensitive to data noise.

A.3 Boosting

The Boosting is another method of ensemble learning for classification problem in supervised learning. The basic concept is that a set of weak classifiers are used as a means for creating a single strong classifier. A weak classifier is indicted to be a classifier which has slightly better performance than random guessing. A strong classifier is generated by conjunction with these weak classifiers. Although there are many Boosting algorithms, a learning process is common to most of them. In the learning process of weak classifiers, the weight of training data is updated iteratively so as to increase their weight for misclassified data and to decrease their weight for correctly classified data. A final strong classifier consists of weak classifiers which are learned with training data of different weights.

Although Boosting has a lot of flexibility in the design of learning algorithm, the weak classifier must be a classifier which can be trained with weighted data. The widely used algorithms for Boosting and the weak classifier are AdaBoost [6] and Decision Tree classifier [7]. In AdaBoost, the weak classifier y_m is learned with the data weight $D_m(x_i)$, where x is the training vectors $x_i \in \mathbf{R}^d$, $i = 1, ..., n$ and m is the number of weak classifiers $m = 1, ..., M$. Then the weighting coefficient is evaluated as $\beta_m = \epsilon_m/(1 - \epsilon_m)$, where ϵ is the misclassification ratio for weighted training vectors x_i. The data weight of the next weak classifier y_{m+1} is reweighted as $D_{m+1}(x_i) = \beta_m D_m(x_i)$ when the data x_i is misclassified by the previous classifier y_m and $D_{m+1}(x_i) = D_m(x_i)$ otherwise. Finally, $D_{m+1}(x_i)$ is normalized, and the final strong classifier $Y(x)$ is formed as the result of y_m weighted by $\alpha_m = \ln(1/\beta_m)$.

$$Y(x) = \text{sgn}\left[\sum_{m=1}^{M} \alpha_m y_m(x)\right] \tag{2}$$

Generally, Boosting is sensitive to the data including lots of noise and outliers. However, it can be superior with respect to the over-fitting issue compared to other classification algorithms.

B. Pattern Matching Techniques

The machine learning techniques are to assign an unknown pattern to one of the possible classes, while the pattern matching techniques in this subsection are of a slightly different nature. A typical pattern matching flow is illustrated in Fig. 3, where a set of patterns or templates are available in library. Different from that in machine learning, usually there is no training stage to build up the library. Given a new testing pattern, we need to decide which pattern in library matches the new one best. Depending on whether the testing pattern is exactly matching to one pattern in the library, pattern matching can be divided into exact pattern matching and fuzzy pattern matching.

B.1 Exact Pattern Matching

In exact pattern matching, a testing pattern is matched if and only if there is an exact same pattern in the library. If a set of features can be extracted from one pattern, the exact pattern matching is the well known string matching problem [8]. Although the exact pattern matching is of high performance at detecting pre-determined patterns in the library, it lacks the capability to identify never seen patterns.

B.2 Fuzzy Pattern Matching

In some applications, usually it is difficult to identify a pattern exactly matching a given testing pattern. Instead of reporting "unmatched", in fuzzy pattern matching problem, the pattern in library matching the testing pattern best would be identified. Fuzzy pattern matching has been widely applied in speech recognition, face detection, and data processing [9, 10]. The critical step in this problem is to define a cost to measure the "similarity" between two patterns.

III. Applications in Physical Design

In this section we will show some key applications of machine learning and pattern matching in physical design (including physical verification).

A. Lithography Hotspot Detection

A.1 Machine Learning Approach

In physical design and verification stages, the hotspot detection problem is to locate hotspots on a given layout with fast turn-around-time. Conventional lithography simulation [11, 12] obtains pattern images using complicated lithography models. Although it is accurate, full-chip lithography simulation is computational expensive, and thus cannot provide quick feedback to guide the early physical design stages. Hotspot detection plays an essential role in bridging the wide gap between modeling and process-aware physical design tool.

There have been a lot of machine learning based hotspot detection works. Machine learning techniques construct a regression model based on a set of training data. This method can naturally identify previous unknown hotspots. However, it may generate false alarms, which are not real hotspots. How to improve the detecting accuracy is the main challenge when adopting machine learning techniques.

Many recent approaches utilize SVM and ANN techniques to construct the hotspot detection kernel. In [13], a 2-D distance transform and histogram extraction is performed on pixel-based layout images, which are used to construct the SVM-based hotspot detection. [14] presents a neural network judgment based detection flow, where 2-D hotspot patterns are directly used to train an ANN model. [15] proposes a multi-level and hybrid method adopting both SVM and ANN to further improve the performance. In [16, 17], SVM is employed through extraction and classification of layout density-based metrics. In [18] the machine learning based methodologies are extended to the directed self-assembly (DSA) hotspot detection. New DSA model with point correspondence and segment distance features are proposed for robust learning.

In [19–21], principle component analysis (PCA) is applied for feature extraction and data reduction. Combining PCA with SVM may help to improve the detection accuracy significantly. Very recently, [22] proposes a Boosting based classification model. Through a simplified layout feature, the utilization of the weakly nonlinear learning algorithm is able to detect hotspots accurately with low false alarm.

A.2 Pattern Matching Approach

Pattern matching based methods are also widely applied in hotspot detection. A layout graph is proposed in [23] to reflect pattern-related CD variation. The resulted graph can be used to find hotspots including closed features, L-shaped features and complex patterns. The concept of range pattern [24] is proposed to incorporate process-dependent specifications, and is enhanced in [25] to represent new types of hotspots. A range pattern is a two-dimensional layout of rectangles with additional specifications encoded by strings. Each range pattern is associated with a scoring mechanism to reflect the problematic regions according to yield impact. The hotspot patterns are stored in a pre-defined library and the detection process performs string matching to find hotspots. This approach is accurate, but the construction of range patterns relies on a grid-based layout matrix, and may be time-consuming when the number of grids is large. Yu et al. [26] propose a DRC-based hotspot detection by extracting critical topological features and modeling them as design rules. Therefore, hotspot detection can be viewed as a rule checking process through a DRC engine.

In [27], a pattern matching based hotspot classification scheme is proposed. The hotspots are classified into clusters by data mining methods. The representative hotspot in each cluster is then identified and stored in a hotspot library for future hotspot detection. The hotspot classification approach in [27] relies on a distance metric of different pattern samples, which is defined as a weighted integral over the area where a pair of hotspot patterns differs (XOR of patterns). It is sensitive to the small variations or shifts of the shapes. In [28], an Improved Tangent Space (ITS) based metric is proposed for hotspot classification. It is an extension of the well-developed tangent space methods [29, 30] in computer vision community. The ITS metric defines a distance metric of a pair of polygons, which is the L_2 norm of the difference of the corresponding turning functions of the polygons [29] [30]. The turning function of polygon measures the angle of the counterclockwise tangent as a function of the normalized arc length, measured from some reference point of the polygon. The ITS based metric is easy to compute and is tolerant with small variations or shifts of the shapes. With the ITS based metric, the hotspot classification can achieve higher accuracy.

A.3 Hybrid Machine Learning and Pattern Matching

Some works combined both the pattern matching and machine learning methods. Recently, a fuzzy matching with some learning technique is proposed in [31] which can dynamically tune appropriate fuzzy regions around known hotspots in multi-dimensional space. Fig. 4 shows an example with known layout patterns of hotspots and non-hotspots in a 2-dimensional space. A machine learning method would divide the space into

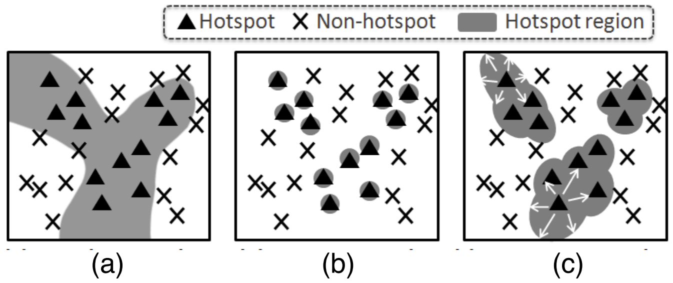


Fig. 4. A 2D-space example of hotspot region decision. (a) Machine learning; (b) Pattern matching; (c) Fuzzy matching model. [31]

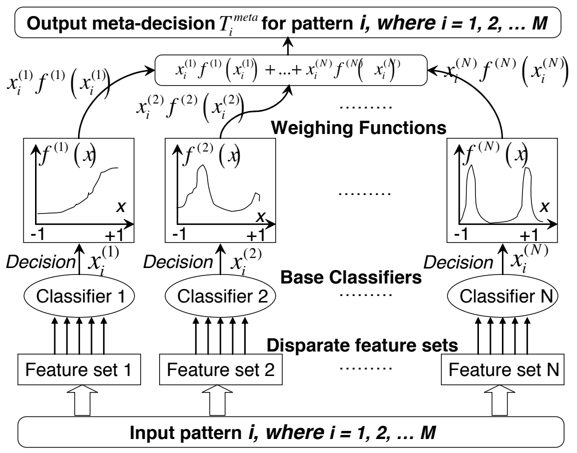

Output meta-decision T_i^{meta} for pattern i, where i = 1, 2, ... M
$x_i^{(1)} f^{(1)}(x_i^{(1)}) + ... + x_i^{(N)} f^{(N)}(x_i^{(N)})$
$x_i^{(1)} f^{(1)}(x_i^{(1)})$
$x_i^{(2)} f^{(2)}(x_i^{(2)})$
$x_i^{(N)} f^{(N)}(x_i^{(N)})$
Weighing Functions
Base Classifiers
Classifier 1
Classifier 2
Classifier N
Decision
Disparate feature sets
Feature set 1
Feature set 2
Feature set N
Input pattern i, where i = 1, 2, ... M

Fig. 5. Meta-classifier construction via a combination of disparate base classifiers [32].

two regions of hotspots and non-hotspots as shown in Fig. 4(a), while a conventional pattern matching approach would construct an individual pattern to match each known hotspot as shown in (b). The fuzzy matching model in Fig. 4(c) includes groups of hotspots, where the fuzzy region of each group will iteratively grows to provide better detection accuracy. In [32], data samples are fed to a pattern matcher first, then machine learning classifiers are used to examine the non-hotspots left by the pattern matcher. Motivated by the fact that different hotspot classifiers have different objectives and strengths, [32] further proposes a unified meta-classifier that enables several classifiers to work together. Fig. 5 illustrates the construction flow of the meta-classification, which is composed of multiple *base classifiers* and *weighting functions*. For each layout pattern, certain hotspot features are extracted and then fed into each base classifier, which calculates the prediction decision and generates a weight based on the weighting functions. The final meta-decision is based on the weighed sum of base classifiers.

B. Lithography Friendly Routing

Lithography hotspot mitigation can be performed at the post-routing stage, e.g., [33]. In [34] [35], design rule checker is integrated with the routing engine at the post-routing stage to identify and correct hotspots. However, fixing hotspots at the post-routing stage has limited flexibility as only limited rip-up

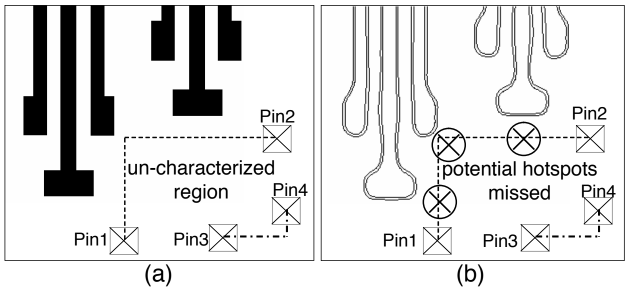


Fig. 6. The hotspot detection challenge in the detailed routing stage [36].

and reroute may be performed. With efficient hotspot predictions, it will be interesting to integrate lithography hotspot detection together with routing.

One challenge of lithography-aware routing is that hotspots are difficult to be detected before a real routing path is obtained. Fig. 6(a) gives a layout region with metal blockages and unrouted pins Pin1-Pin4. Since some nets are not yet routed, there is an un-characterized region where no hotspots would be identified by general hotspot detection methods. Consequently, potential hotspots may be caused by route Pin1-Pin2 as in Fig. 6(b). [36] proposes a lithography-friendly detailed routing based on a pre-built hotspot prediction model and a routing path prediction model. The hotspot detection model is trained to evaluate the pattern printability based on a set of post-RET data. To overcome the issue of un-characterized regions, the routing path prediction model is established using the following steps: (1) explore the possible routing solutions given the available routing resources; (2) perform accurate lithography simulation for the possible layout results; (3) identify preferable routes according to results of hotspots and routing congestion. Because the data that need to be processed for building the routing path prediction model is huge, an ANN classifier is constructed to guide the routing engine.

C. Datapath Placement

The typical objective function of placement is to minimize total half-perimeter wire length (HPWL), which is a good indicator of placement quality for random logic designs. However, in datapath logic designs, generally cells are characterized by a high degree of bit-wise parallelism that conventional placement has shown to be sub-optimal [37]. In designs where there are many embedded datapaths, extracting the datapaths and placing them with random logics appropriately has the potential to significant improve the overall StWL [38–40].

Fig. 7 shows a toy example where modern placers are not able to handle datapaths effectively [38]. Fig. 7(a) displays the datapath circuit, where the input/output pins are fixed and there are three bit-stacks corresponding to cells: ($\{2, 3, 4, 5\}$, $\{6, 7, 8, 9\}$, and $\{10, 11, 12, 13\}$. Fig. 7(b) displays the PADE placement solution, where each bit-stack is tightly packed and aligned producing an StWL solution of 524. Fig. 7(c) displays the placement solution from Fast-Place3 [41] where the bit-stack is not carefully aligned producing StWL of 612.

A new placement flow with automatic datapath extraction is proposed in [38], which evaluates and ranks all the first-order, important data paths, and optimizes them along with general-

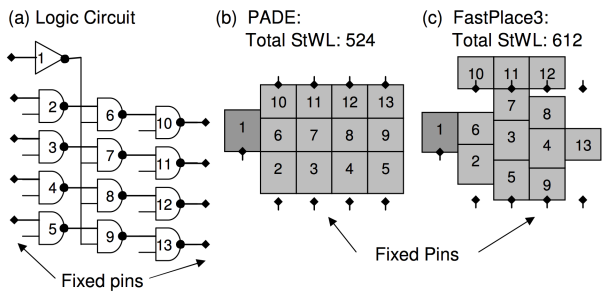


Fig. 7. Datapath driven placement example showing a 14% steiner wire length improvement compared to conventional placement [38].

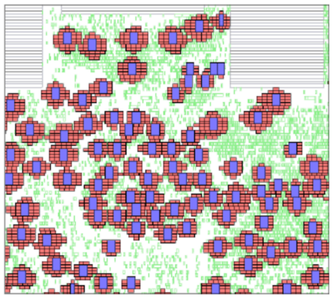


Fig. 8. Multi-GHz design showing clustered latches, where red cells are latches and purple cells are local clock buffers (LCB) [45].

purpose wirelength driven placement. In the training stage, to classify and evaluate the datapath patterns in the initial netlist, SVM and ANN techniques are combined to build compact and run-time efficient models. In SVM model, an error tolerant technique is combined with a special working set selection. ANN works through configuring complex networks of neurons to achieve a high dimensional decision diagram-like data structure given training samples and decision hints. The optimization objective for both SVM and ANN is to maximize the evaluation accuracies of datapath and non-datapath patterns. In the testing stage, the data learning models will be applied directly to test whether a new unknown design patterns is datapath pattern. A pattern is evaluated to be datapath if and only if both SVM and ANN evaluation scores are above certain thresholds.

D. Clock Optimization

Clock network design for high performance microprocessors is one of the most challenging problems in VLSI design. To bound the skew and power consumption of the bottom level stage, latches are often clustered and placed next to a common local clock buffer (LCB) in a structured fashion [42–44]. As shown in Fig. 8, small groups of latches are tightly clustered around LCBs to significantly reduce the total local clock tree length. [45] proposes a scalable machine learning solution to the latch optimization problem. First, for each latch cluster, a genetic algorithm is proposed to search for the latch placement solution. Second, a large set of initial latch placement templates are generated, and the template number can be significantly reduced by using a set theoretic to remove the redundancy. Last, a machine learning technique called decision tree induction is developed using similarity metric for quickly selecting the correct template during design automation.

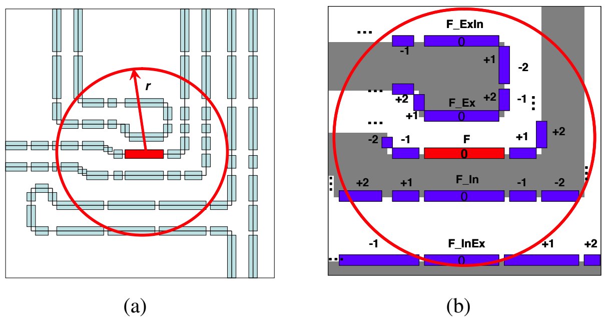


Fig. 9. Fragmentation based layout feature, where the layout is cut into fragments [15].

IV. Advanced Issues

A. Machine Learning or Pattern Matching?

The machine learning, especially supervised learning, consists of training and testing stages. In the training stage, given the input pattern, a set of learning models are built. Then the constructed learning models are applied in testing stage to make predictions or decisions. Machine learning based methods can be applied to complexed nonlinear classification, or the applications where the classification functions are unknown. However, to maintain stable performance, machine learning requires additional parameter tuning for different applications.

The pattern matching based method defines and stores precharacterized patterns into library. Given a testing pattern, the library pattern matched is identified. The fuzzy pattern matching can be applied to find a very similar library pattern to the input pattern. Pattern matching based method is very fast and accurate to detect known templates. But even fuzzy pattern matching is applied, it may lack the capability to identify never seen patterns.

Generally speaking, machine learning can obtain higher performance, especially when predicting unseen data. But machine learning may take longer time in training to achieve high performance. If the library patterns are sufficient and the runtime is a concern, pattern matching based method is a good option. Pattern matching tools have already been integrated in industry [46–49]. On the other hand, pattern matching is more sensitive to process changes, as new set of patterns will have to be built after process tuning. Machine learning can be more robust under process changes, and it can handle unseen patterns more naturally. Hybrid approaches of machine learning and pattern matching will be desirable.

B. Feature Extraction

In both machine learning and pattern matching techniques, we need to analyze patterns. Each pattern is extracted into a specific feature, which is an individual measurable heuristic property describing the pattern. Feature extraction is a task to represent the patterns using simple but comprehensive feature information, so that all the selected features can be distinguished. Therefore, feature extraction is one of the most critical steps in both machine learning and pattern matching. In this subsection we introduce and analyze some classical feature extraction techniques in physical design and physical verification.

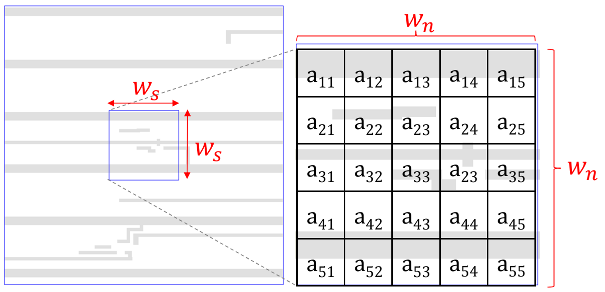


Fig. 10. Density-based layout feature. Feature Vector is represented as: $X = \{a_{11}, a_{12}, ..., a_{54}, a_{55}\}$ [22].

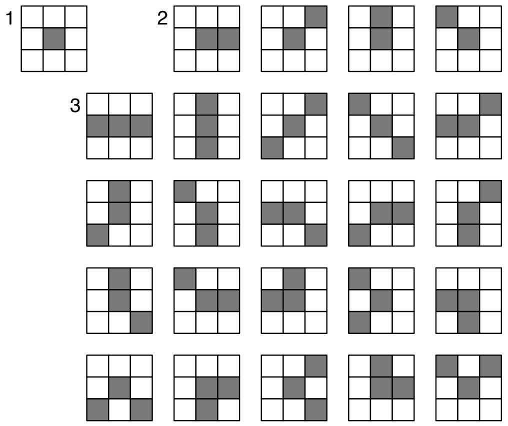


Fig. 11. HLAC based layout feature.

A fragmentation based layout extraction scheme is presented in [15], where the layout is cut into fragments. For each fragment F, an effective radius r is defined to cover the neighboring fragments which need to be considered in the context characterization of F as shown in Fig. 9. A complete representation of F includes the geometric characteristic of fragments inside r, including pattern shapes, the distance between patterns, corner information (convex or concave), and so on.

Density based layout features are presented in [16, 22], where a layout pattern is represented as a vector of layout density values of its surrounding area. Given a layout clip with predefined grids, the method calculates the layout density covered in each grid. An ordered list of density values then forms the final vector that represents the corresponding layout pattern. Fig. 10 illustrates an example of density based layout feature. The goal of this layout feature is not to identify the geometrical features that may degrade the printability of a pattern. Instead, it aims at providing a compact representation of layout patterns to enable measurement of pattern similarities.

A *higher-order local autocorrelation* (HLAC) based layout feature was proposed in [21]. As illustrated in Fig. 11, in HLAC based layout feature a configuration of 3×3 pixels forms 25 local mask patterns that take the shift-invariant properties into consideration. In each mask, *black* represents the pixels engaged in multiplication and *white* represents "do not care". The masks are used to scan across the whole image, calculating x value at each location and adding up all. The total values for all masks are concatenated to from a vector as HLAC features.

Fig. 12 compares the performance of different feature extractions. We can see from Fig. 12 (a) that fragmentation based layout feature are separated into several clusters, and the hotspot features are mixed with non-hotspot features. Therefore, if not carefully designed, this feature may be too complicated for single classification model to separate hotspots and non-hotspots with high accuracy. From Fig. 12 (b) and (c) we can see that for this case, density based feature and HLAC based feature can define decision boundary easily.

In hotspot detection application, [22] proposes a method to quantitatively evaluate the extracted features. The average distance of hotspot features to non-hotspot features, Z, is defined as follows.

$$Z = \frac{1}{N}\sum_{i=1}^{N} d_i \tag{3}$$

where N is the total number of real hotspots in the training set. d is the Mahalanobis distance [50] normalized by non-hotspot features, as in (4).

$$d_i = \frac{\sqrt{(x_i-\mu)^T V^{-1}(x_i-\mu)} - d_{NHS_{min}}}{d_{NHS_{max}} - d_{NHS_{min}}} \tag{4}$$

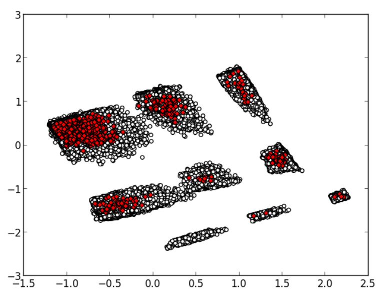
where μ is the center of mass of non-hotspot features, V is the variance covariance matrix of non-hotspot features, $d_{NHS_{max}}$ is the maximum Mahalanobis distance of non-hotspot features and $d_{NHS_{min}}$ is the minimum Mahalanobis distance of non-hotspot features.

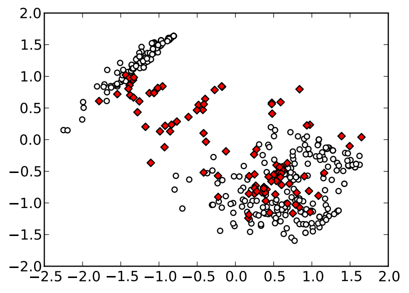
Generally speaking, $Z < 1$ indicates that it is difficult to separate hotspots and non-hotspots linearly as the most hotspot features are within the non-hotspot feature space. In contrast, $Z > 1$ shows linear separation friendly features due to its distance of hotspot features from non-hotspot features. In other words, to define an appropriate layout feature, Z is preferably larger than 1 but not too large [22].

C. Overcome Overfitting in Machine Learning

Overfitting is the phenomenon where fitting the training patterns well no longer indicates that the learning models can work well for the testing patterns [51]. Generally speaking, overfitting happens when the learning models or the extracted features are more complex than necessary to represent the problem. In other words, if the learning model uses additional degrees of freedom to fit noise in the training patterns, we may observe a bad fitting in testing patterns. There are several reasons why overfitting happens, i.e., noise in the data, lack of representative samples, too complex feature, and too complex learning models [52].

Fig. 13 shows an example of overfitting in a hotspot detection test case. The density based layout feature and AdaBoost learning models are utilized in a hotspot detection framework. To investigate the effect of overfitting, different levels of feature complexity can be applied. The larger Wn value, the more complex feature is extracted. We can see that when the feature complexity is very low, both the training accuracy and testing accuracy are bad, which is known as *model underfitting*. However, once the feature complexity becomes too large, although the training accuracy is still high, the testing accuracy begins to decrease.


(a)


(b)

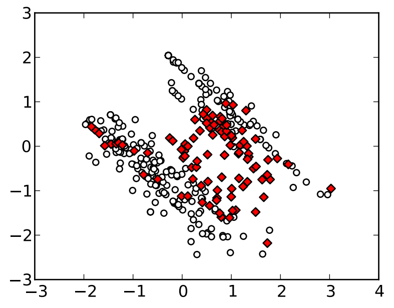

(c)

Fig. 12. Comparison of different feature extraction methods, where red points are hotspot features and the white points are non-hotspot features: (a) Fragmentation based feature; (b) Density based feature; (c) HLAC based feature.

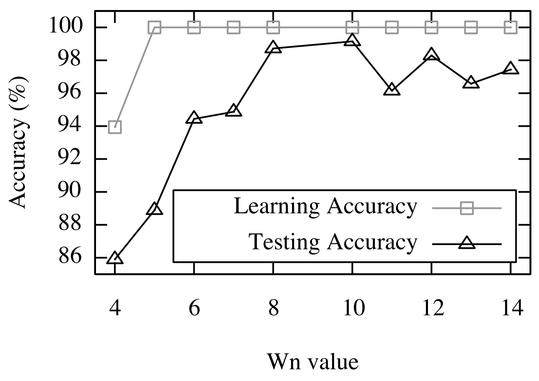


Fig. 13. An example of overfitting that once the feature complexity (Wn) becomes too large, the testing accuracy begins to decrease.

Several techniques can be applied to overcome the overfitting issue. The first one is called *regularization*, which introduces additional conditions in constraints or objective functions in the learning models [53]. For example, the objective function in (1) can be rephrased as follows.

$$\min_{\alpha} : \frac{1}{2}\alpha^T Q\alpha - e^T\alpha + \lambda\alpha^T\alpha \quad (5)$$

where λ is a parameter to control the amount of regularization. The penalty term $\alpha^T\alpha$ enforces a trade-off between making higher training accuracy and simpler learning model. Another technique handling overfitting is called *cross validation* [54], where the training data is partitioned into training set and validation set. The validation set can be used to estimate the performance for testing data. In addition, the validation set can be also applied to select appropriate learning models.

V. Conclusion

In this paper we have surveyed some commonly used machine learning and pattern matching techniques. We have also introduced their applications in physical design and verification, e.g., lithography hotspot detection, datapath placement, and clock optimization. In addition, we have discussed some advanced issues, including feature extraction, and overfitting problem. Since modern VLSI circuits have billions of transistors/interconnects, machine learning and pattern matching techniques have gained more and more attention to provide reasonably good abstraction and quality-of-result. We hope this paper will stimulate more studies on application specific machine learning and pattern matching techniques in physical design and VLSI CAD.

Acknowledgment

This work is supported in part by National Science Foundation (NSF), Semiconductor Research Corporation (SRC), National Natural Science Foundation of China (NSFC) research project 61125401, 61376040 and 61228401, Shanghai Science and Technology Committee project 13XD1401100, and Toshiba.

References

[1] D. Z. Pan, B. Yu, and J.-R. Gao, “Design for manufacturing with emerging nanolithography,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 32, no. 10, pp. 1453–1472, 2013.

[2] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. MIT press, 2012.

[3] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural Network Design*. Pws Boston, 1996, vol. 1.

[4] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in *Conference on Learning Theory*. ACM, 1992, pp. 144–152.

[5] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[6] Y. Freund and R. E. Schapire, “A desicion-theoretic generalization of on-line learning and an application to boosting,” in *Conference on Learning Theory*, 1995, pp. 23–37.

[7] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. CRC press, 1984.

[8] D. E. Knuth, J. H. Morris, Jr, and V. R. Pratt, “Fast pattern matching in strings,” *SIAM Journal on Computing*, vol. 6, no. 2, pp. 323–350, 1977.

[9] H. Wu, Q. Chen, and M. Yachida, “Face detection from color images using a fuzzy pattern matching method,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, pp. 557–563, 1999.

[10] D. Singh, J. Pandey, and D. Chauhan, “Topology identification, bad data processing, and state estimation using fuzzy pattern matching,” *IEEE Transactions on Power Systems*, vol. 20, no. 3, pp. 1570–1579, 2005.

[11] J. Kim and M. Fan, “Hotspot detection on Post-OPC layout using full chip simulation based verification tool: A case study with aerial image simulation,” in *Proceedings of SPIE*, vol. 5256, 2003.

[12] E. Roseboom, M. Rossman, F.-C. Chang, and P. Hurat, "Automated full-chip hotspot detection and removal flow for interconnect layers of cell-based designs," in *Proceedings of SPIE*, vol. 6521, 2007.

[13] D. G. Drmanac, F. Liu, and L.-C. Wang, "Predicting variability in nanoscale lithography processes," in *IEEE/ACM Design Automation Conference (DAC)*, 2009, pp. 545–550.

[14] D. Ding, X. Wu, J. Ghosh, and D. Z. Pan, "Machine learning based lithographic hotspot detection with critical-feature extraction and classification," in *IEEE International Conference on IC Design and Technology (ICICDT)*, 2009, pp. 219–222.

[15] D. Ding, J. A. Torres, and D. Z. Pan, "High performance lithography hotspot detection with successively refined pattern identifications and machine learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, pp. 1621–1634, 2011.

[16] J.-Y. Wuu, F. G. Pikus, A. Torres, and M. Marek-Sadowska, "Rapid layout pattern classification," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2011, pp. 781–786.

[17] Y.-T. Yu, G.-H. Lin, I. H.-R. Jiang, and C. Chiang, "Machine-learning-based hotspot detection using topological classification and critical feature extraction," in *IEEE/ACM Design Automation Conference (DAC)*, 2013, pp. 671–676.

[18] Z. Xiao, Y. Du, H. Tian, M. D. Wong, H. Yi, H.-S. P. Wong, and H. Zhang, "Directed self-assembly (DSA) template pattern verification," in *IEEE/ACM Design Automation Conference (DAC)*, 2014, pp. 1–6.

[19] J.-R. Gao, B. Yu, and D. Z. Pan, "Accurate lithography hotspot detection based on PCA-SVM classifier with hierarchical data clustering," in *Proceedings of SPIE*, vol. 9053, 2014.

[20] B. Yu, J.-R. Gao, D. Ding, X. Zeng, and D. Z. Pan, "Accurate lithography hotspot detection based on principal component analysis-support vector machine classifier with hierarchical data clustering," *Journal of Micro/Nanolithography, MEMS, and MOEMS (JM3)*, vol. 14, no. 1, p. 011003, 2015.

[21] H. Nosato, H. Sakanashi, E. Takahashi, M. Murakawa, T. Matsunawa, S. Maeda, S. Tanaka, and S. Mimotogi, "Hotspot prevention and detection method using an image-recognition technique based on higher-order local autocorrelation," *Journal of Micro/Nanolithography, MEMS, and MOEMS (JM3)*, vol. 13, no. 1, p. 011007, 2014.

[22] T. Matsunawa, J.-R. Gao, B. Yu, and D. Z. Pan, "A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction," in *Proceedings of SPIE*, 2015.

[23] A. B. Kahng, C.-H. Park, and X. Xu, "Fast dual graph based hotspot detection," in *Proceedings of SPIE*, vol. 6349, 2006.

[24] H.Yao, S. Sinha, C. Chiang, X. Hong, and Y. Cai, "Efficient process-hotspot detection using range pattern matching," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2006, pp. 625–632.

[25] J. Xu, S. Sinha, and C. C. Chiang, "Accurate detection for process-hotspots with vias and incomplete specification," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2007, pp. 839–846.

[26] Y.-T. Yu, Y.-C. Chan, S. Sinha, I. H.-R. Jiang, and C. Chiang, "Accurate process-hotspot detection using critical design rule extraction," in *IEEE/ACM Design Automation Conference (DAC)*, 2012, pp. 1167–1172.

[27] N. Ma, "Automatic IC hotspot classification and detection using pattern-based clustering," Ph.D. dissertation, Engineering and Mechanical Engineering, University of California, Berkeley, 2008.

[28] J. Guo, F. Yang, S. Sinha, C. Chiang, and X. Zeng, "Improved tangent space based distance metric for accurate lithographic hotspot classification," in *IEEE/ACM Design Automation Conference (DAC)*, 2012, pp. 1173–1178.

[29] E. M.Arkin, L. Chew, D. P.Huttenlocher, K. Kedem, and J. S.B.Mitchell, "An efficiently computable metric for comparing polygonal shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 3, pp. 209–216, 1991.

[30] L. J. Latecki and R. Lakamper, "Shape similarity measure based on correspondence of visual parts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 10, pp. 1–6, 2000.

[31] S.-Y. Lin, J.-Y. Chen, J.-C. Li, W.-Y. Wen, and S.-C. Chang, "A novel fuzzy matching model for lithography hotspot detection," in *IEEE/ACM Design Automation Conference (DAC)*, 2013, pp. 681–686.

[32] D. Ding, B. Yu, J. Ghosh, and D. Z. Pan, "EPIC: Efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2012, pp. 263–270.

[33] J. Mitra, P. Yu, and D. Z. Pan, "RADAR: RET-aware detailed routing using fast lithography simulations," in *IEEE/ACM Design Automation Conference (DAC)*, 2005, pp. 369–372.

[34] J. Yang, N. Rodriguez, O. Omedes, F. Gennari, Y.-C. Lai, and V. Mankad, "DRCPlus in a router: automatic elimination of lithography hotspots using 2D pattern detection and correction," in *Proceedings of SPIE*, vol. 7641, 2010, p. 76410Q.

[35] J.-R. Gao, H. Jawandha, P. Atkar, A. Walimbe, B. Baidya, O. Rizzo, and D. Z. Pan, "Self-aligned double patterning compliant routing with in-design physical verification flow," in *Proceedings of SPIE*, vol. 8684, 2013, p. 868408.

[36] D. Ding, J.-R. Gao, K. Yuan, and D. Z. Pan, "AENEID: a generic lithography-friendly detailed router based on post-RET data learning and hotspot detection." in *IEEE/ACM Design Automation Conference (DAC)*, 2011, pp. 795–800.

[37] S. I. Ward, D. A. Papa, Z. Li, C. N. Sze, C. J. Alpert, and E. Swartzlander, "Quantifying academic placer performance on custom designs," in *ACM International Symposium on Physical Design (ISPD)*, 2011, pp. 91–98.

[38] S. Ward, D. Ding, and D. Z. Pan, "PADE: a high-performance placer with automatic datapath extraction and evaluation through high dimensional data learning," in *IEEE/ACM Design Automation Conference (DAC)*, 2012, pp. 756–761.

[39] S. Chou, M.-K. Hsu, and Y.-W. Chang, "Structure-aware placement for datapath-intensive circuit designs," in *IEEE/ACM Design Automation Conference (DAC)*, 2012, pp. 762–767.

[40] H. Xiang, M. Cho, H. Ren, M. Ziegler, and R. Puri, "Network flow based datapath bit slicing," in *ACM International Symposium on Physical Design (ISPD)*, 2013, pp. 139–146.

[41] N. Viswanathan, M. Pan, and C. Chu, "FastPlace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2007, pp. 135–140.

[42] D. Papa, C. Alpert, C. Sze, Z. Li, N. Viswanathan, G.-J. Nam, and I. L. Markov, "Physical synthesis with clock-network optimization for large systems on chips," *IEEE Micro*, vol. 31, no. 4, pp. 51–62, 2011.

[43] M. Cho, H. Xiang, H. Ren, M. M. Ziegler, and R. Puri, "Latchplanner: latch placement algorithm for datapath-oriented high-performance VLSI designs," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 342–348.

[44] S. Held and U. Schorr, "Post-routing latch optimization for timing closure," in *IEEE/ACM Design Automation Conference (DAC)*, 2014, pp. 1–6.

[45] S. I. Ward, N. Viswanathan, N. Y. Zhou, C. C. Sze, Z. Li, C. J. Alpert, and D. Z. Pan, "Clock power minimization using structured latch templates and decision tree induction," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 599–606.

[46] "Calibre pattern matching," http://www.mentor.com/products.

[47] "Synopsys IC Validator," http://www.synopsys.com.

[48] "Cadence Virtuoso DFM," http://www.cadence.com.

[49] "Anchor Semiconductor NanoScope," http://www.anchorsemi.com.

[50] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences (Calcutta)*, vol. 2, pp. 49–55, 1936.

[51] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning from Data*. AMLBook, 2012.

[52] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Addison-Wesley, 2006.

[53] P. J. Bickel, B. Li, A. B. Tsybakov, S. A. van de Geer, B. Yu, T. Valdés, C. Rivero, J. Fan, and A. van der Vaart, "Regularization in statistics," *Test*, vol. 15, no. 2, pp. 271–344, 2006.

[54] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statistics Surveys*, vol. 4, pp. 40–79, 2010.