An Autonomous System View To Apply Machine Learning

Li-C. Wang
University of California, Santa Barbara

***Abstract*—Applying machine learning in design and test has been a growing field of interest in recent years. Many potential applications have been demonstrated and tried. This tutorial paper provides a review of author's experience in this field, in particular how the perspective for applying machine learning evolved from one stage to another where the latest perspective is based on an autonomous system view to apply machine learning. The theoretical and practical barriers leading to this view are highlighted with selected applications.**

1. Introduction

Applying machine learning in design and test has attracted much interests in recent years. Tremendous amounts of simulation and measurement data are generated and collected in design and test processes. These data offer opportunities for applying machine learning [1].

As a start, applying machine learning could be seen as selecting a tool from a machine learning toolbox (e.g. [2]) and running the tool with data collected in an application context. The goal could be for predicting a behavior in the future data, or for understanding the current data at hand. A machine learning tool takes a dataset and produces a *learning model*. Depending on the underlying algorithm, a learning model, represented in a particular form, might or might not be interpretable by a person.

In many applications, applying machine learning is not just about choosing a particular tool and obtaining a particular learning model. Often, domain knowledge is required to enable the learning because the data is limited, and applying machine learning becomes more like following an iterative search process [1]. The focus of the process is on learning about the *features* rather than building a model [3].

The search process iterates between two agents, an analyst and a toolbox. In each iteration, the analyst decides an *analytic tool* to run and the data to run with. The analyst prepares the dataset and invokes the tool. The result is then examined by the analyst to decide what to do next. Figure 1 illustrates this view with the three components: input preparation, tool invocation, and result evaluation.

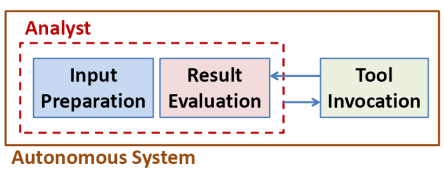

Figure 1. An autonomous system view to apply machine learning

In Figure 1, input preparation and result evaluation are largely domain knowledge driven. An implication of taking this view is that in order to automate the search process, all three components need to be automated, resulting in an autonomous system view to apply machine learning.

At first glance, Figure 1 seems to suggest that "machine learning" takes place in the tool invocation box. In other words, "applying machine learning" essentially means to invoke a tool in a machine learning toolbox. If applying machine learning means running a machine learning tool, then what the tool is actually learning?

Take clustering as an example, which is a common problem considered in *unsupervised learning*. One of the popular clustering algorithms is K-means [2]. Suppose one prepares a dataset and runs the K-means tool with the dataset. Suppose $k = 3$, so the output is a partitioning of the samples into three groups. If this is "machine learning," what the K-means tool is learning?

Take regression as another example, which is a common problem considered in *supervised learning*. One of the popular regression algorithms is *ridge regression*. Again, suppose one prepares a dataset and runs the regression tool with the dataset. Is that "machine learning?" If it is, what is the difference between "machine learning" and model building taught in a traditional statistics textbook?

To define what "machine learning" means in general is beyond this paper. However, it is important to clarify where machine learning takes place in the view of Figure 1. To answer that question, an analogy can be drawn between Figure 1 and a system view for autonomous vehicle.

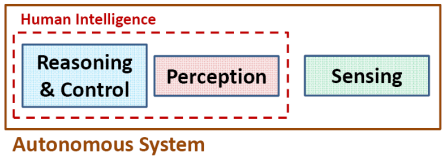

Figure 2. A system view for autonomous vehicle

Figure 2 shows a high-level system view with three components for an autonomous vehicle [4]. The sensing component can be based on a list of sensors such as short-range radar, long-range radar, LiDAR, ultrasonic, vision, stereo vision, etc. The perception component interprets the data from those sensors, for example to perform object recognition and lane detection, etc. The reasoning and control component calculates the free space for moving the vehicle, plans its path, and controls such as speed/brake/wheel.

978-1-5386-8382-8/18/$31.00

Essentially, the sensing component collects data from the environment. The perception component recognizes what the data mean. The reasoning and control component decides what to do next. The analogy between Figure 1 and Figure 2 is that the tool invocation corresponds to the sensing component, the result evaluation corresponds to the perception component, and the dataset preparation corresponds to the reasoning and control component.

The analogy might seem unintuitive at first. But it can become clear if we consider in Figure 1 where the human intelligence is involved. In Figure 1, the tool in a toolbox follows a fixed algorithm. It does not capture the intelligence of a person, in this case the analyst's. The analyst's intelligence is involved in the input preparation and result evaluation. Hence, if we consider "machine learning" as for learning about human intelligence, then the learning should take place in these two components.

Similarly, in Figure 2 the intelligence of a driver is in the perception and reasoning components. The sensing component essentially serves as a mechanical interface to the real-world environment. In this view, for example it is intuitive to see that a deep neural network can be employed in the perception component, and *deep learning* [5] is widely accepted as a form of machine learning.

1.1. Machine learning and AI

The analogy between Figure 1 and Figure 2 becomes clear if one sees "machine learning" as "taking a data-driven approach to model human intelligence." With this interpretation, the link between the tool invocation component and the sensing component is clear, i.e. they both are interfaces to the environment and they both are not where the analyst's intelligence is employed. In Figure 2, the environment is the outside world where the car is driving. In Figure 1, the environment can be thought of as the software infrastructure used by the analyst to perform an engineering task.

It is important to note that in Figure 1 and Figure 2, machine learning *can* be applied to model human intelligence in the left two components, but is not necessary. For example, if the human intelligence can be modeled in terms of a set of *rules*, then there might not be a need to take a *data-driven* approach. Based on the views presented in those two figures, suppose the left two components are implemented entirely with rules. Then such a system can still simulate human intelligence and hence, can be called an artificial intelligence system (e.g. an expert system). However, it does not involve machine learning and hence, would not be called a machine learning based system.

2. Intelligent Engineering Assistant (IEA)

In a more general sense, Figure 1 tries to capture the process carried out by a person in performing an engineering task. In this sense, the tools involved by the person are not limited to those tools from a particular machine learning toolbox. They can include other tools, for example a plotting tool, a data processing tool, etc. Therefore, the autonomous system illustrated in Figure 1 can be thought of as for performing the task autonomously. Such a system can be called an Intelligent Engineering Assistant (IEA) [6].

As an example, Figure 3 illustrates the scope of an engineering task in the production yield application context. In production, every wafer goes through a sequence of tools (manufacturing equipments). There could be more than hundreds of tools involved. Data collected in the manufacturing process is organized in three levels [7]. Suppose these data are stored in a production database. After a wafer is produced, the test process comprises multiple stages of testing, including such as class probe (e-test), wafer probe, burn-in, and final test. Suppose data collected in those stages are stored in a test database.

Both databases can be a virtual database and suppose there is an interface to access them. An engineer accesses data from those databases to perform a task. For example, the task is triggered by observing a potential place to improve yield, i.e. a yield issue. Then, the engineer uses various analytic tools to analyze the data. The aim is to uncover some interesting findings [7], such as a correlation between an e-test and a type of fails, or a particular pattern on a wafer map. The outcome of this analysis is summarized in a PPT presentation. The engineer brings this presentation to a meeting and discusses possible actions. In Figure 3, input to the autonomous system is the data accessed from the interface, and output is a PPT presentation.

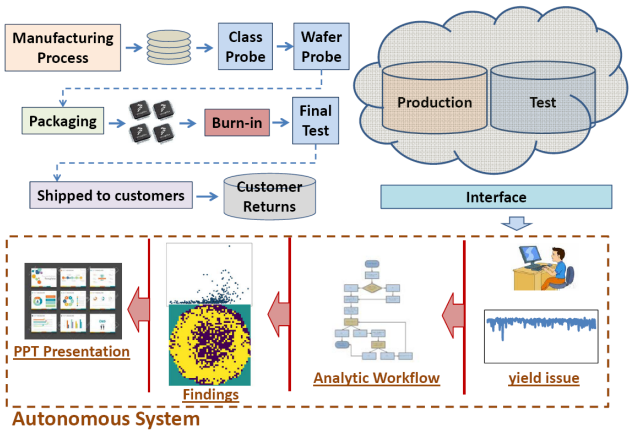

Figure 3. Tasks performed in the production yield context

In Figure 3, one can think about the analytics performed by the engineer in terms of a workflow (let it be called an *analytic workflow*). It is important to point out that this workflow may exist implicitly in the person's mind, rather than explicitly be written in a document (or captured by a software script). In essence, building an autonomous system is to automate the respective analytic workflow. This automation conceptually is similar to the common practice in a company where a software script is used to automate an established methodology (or flow). The key question is, whether a traditional software script is sufficient to automate the underlying analytic workflow or not, and if not where machine learning can help.

2.1. Concept-Based Workflow Programming

To illustrate where machine learning can help, consider an analytic workflow example depicted in Figure 4. This workflow segment is triggered by observing a "yield excursion." Then, the "low-yield" lots are selected for further analysis. The analysis looks for a "correlation trend" between an e-test and the fails from a test bin. If such a "correlation trend" is found, the fails are used to generate wafer maps and the analysis looks for certain patterns such as "grid pattern" and "edge failing pattern" on those maps.

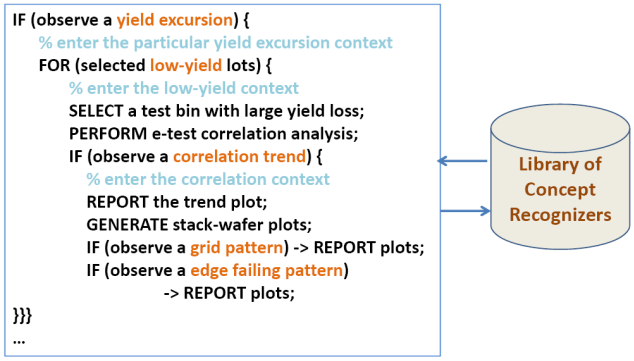

Figure 4. Illustration of a concept-based workflow

In such a workflow segment, the engineer (or analyst) has a specific perception for what "yield excursion," "low-yield," "correlation trend," "grid pattern," and "edge failing pattern" mean. Let each of them be called a *concept*. To automate the workflow segment, automatic recognition (or determination of the existence) of each concept is needed. Recognition of each concept can be accomplished by writing a software script. However, in some cases it might be challenging to capture a concept perceived by a person in terms of fixed *rules*. It is for those cases, a data-driven machine learning model can become useful [7].

Take the concept "correlation trend" as an example. One can capture this concept as "having a Pearson correlation ≥ 0.8." With this definition, one can write a software script to call the Pearson correlation tool in a toolbox and check if the correlation coefficient is ≥ 0.8 or not. This approach captures the concept with a fixed rule.

The limitation of using a fixed rule is that it might not capture the concept entirely as perceived by a person [7]. For example, The top plot in the Findings stage of Figure 3 shows that as the x value becomes larger, the y value also drifts larger. The plot has a x-y correlation coefficient much smaller than 0.8, but it has a trend interesting to a person's perception. Hence, an alternative to recognize the concept can be based on a neural network model to capture the perception [7]. A neural network model can be used to enhance the recognition of a particular concept, in addition to using a fixed rule.

To enable autonomous execution of a workflow segment as illustrated in Figure 4, what we need is a library of *concept recognizers*. Conversely, if a library of concept recognizers are provided, this library enables the construction of an automatic workflow based on those concepts. From this perspective, the paradigm to support building an autonomous system in view of Figure 1 can be called *Concept-Based Workflow Programming* (CBWP).

2.2. Where machine learning is applied

In view of CBWP, it is then intuitive to see that machine learning is applied for building a concept recognizer to support the workflow programming. Consequently, Figure 3 can be re-drawn into a new view as that in Figure 5. In this new view, the workflow is captured in a software program written in a traditional programming language such as Python, except that it can make a function call to a concept recognizer in the concept recognition library. The library comprises a set of concept recognizers which may be implemented with a machine learning approach and/or a fixed-rule approach. The workflow can also make a call to invoke a tool in a toolbox which can be a machine learning toolbox, a plotting toolbox, a data processing toolbox, etc.

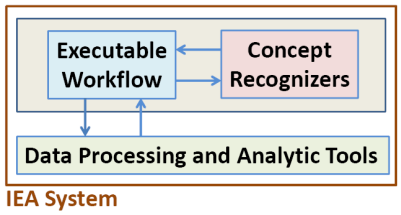

Figure 5. Illustration of the three components in an IEA system

In Figure 5, machine learning is employed to enhance the capabilities of the workflow. An executable workflow does not have to involve any machine learning model. However, the capabilities of such a workflow can be limited because of its limitations for capturing human perception.

2.3. Workflow construction

One may ask the question: Can machine learning be used to learn a workflow model as well? The answer is yes. An earlier work in [8] utilizes *process learning* to learn a workflow based on the usage log of analytic tools, which tracks how an engineer carries out a particular analytic process. Learning a workflow can be seen as a form of *knowledge acquisition* which can be quite important for application contexts where direct construction of a workflow is difficult to practice. Moreover, knowledge acquisition can be essential for implementing a scheme to support knowledge accumulation over time and/or across different people. However, implementing those knowledge acquisition functions adds another layer of complexity to an IEA system. If it is not required by the application context, the workflow in Figure 5 can be constructed manually to speed up the development of the IEA system.

In this work, knowledge acquisition is treated as a follow-up work for building an IEA system. A more critical issue is the availability of the concept recognizers that one desires to use for constructing a workflow. In view of building a practical IEA system, the development of the concept recognition library is more essential than implementing a knowledge acquisition layer.

3. Evolution To IEA

The IEA system view in Figure 5 is an intuitive way to think about where to apply machine learning. Each concept provides a functional specification for what is supposed to be accomplished. The applicability of a machine learning technique and a learning model can therefore be assessed in terms of the functional specification. Moreover, the goodness of a concept recognizer can be evaluated in the context of the workflow, i.e. the workflow context provides a way to define the requirements for a recognizer such as its accuracy and robustness. Without a workflow the added value of employing a learning model in practice can be unclear and consequently, justification for "applying machine learning" can become somewhat debatable.

In retrospect, the IEA system view is rather intuitive. However, along the research path leading to this view, it was not always the case. From the original motivation of "applying machine learning in design and test" to the current IEA view, the research path actually went through multiple stages where the perspective for applying machine learning evolved from one stage to another.

Along the path, a change of perspective often took place after realizing the difficulty to overcome a theoretical challenge, a practical constraint, or both, and such realization was often triggered after a better understanding of some key questions (not necessarily after finding a clear answer, because for some questions finding those answers could also be hard). These questions include such as:

- When can we say that a machine has learned?
- Does the No-Free-Lunch theorem [9] matter?
- Why cross-validation is needed?
- How to interpret overfitting?
- Why the Occam's Razor principle does not work?
- Can machine learn with "small" data?
- When can we call it a machine learning solution?
- What is the added value of employing such a solution in practice?

It is interesting to note that the list misses the obvious question: What is the best machine learning algorithm to use? This is because this algorithmic question did not drive the change of a perspective along the research path.

In the rest of the paper, let the discussion start with the first question: When can we say that a machine has learned?

4. The Theoretical Questions

Take supervised learning as an example. Figure 6 illustrates a theoretical setup for supervised learning where relative issues regarding the meaning of "learning" can be explained. Note that this explanation is for highlighting some important issues from a practitioner's perspective and in no way, trying to be mathematically rigorous. In this setup, there are five areas to make an assumption in order to ensure learning. These five areas are labeled in the figure.

First, there is the hypothesis space H. H is a set of functions (hypotheses) and one of them f is the true answer

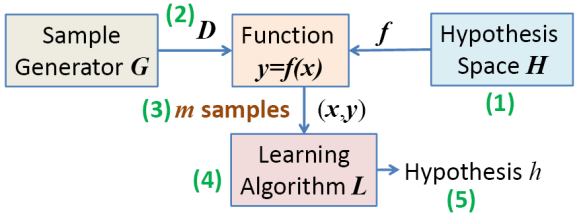

Figure 6. Five areas of assumptions for supervised learning

to be learned. Of course this function f is unknown to the learner. Further, H is also unknown so one needs to make an assumption regarding H. Once assumed, the learning algorithm operates based on this assumed H.

To learn, it is assumed that a sample generator G produces a set of m samples $\vec{x}_1, \ldots, \vec{x}_m$ according to an unknown but fixed distribution D. For each sample $\vec{x}_i$, its label y_i is given, as calculated by $f(\vec{x}_i)$. Then, the dataset comprising the m pairs $(\vec{x}_i, y_i)$, is provided to the learner.

The learner employs a learning algorithm L to learn. The algorithm L outputs its answer h. Ideally, if the answer is correct, we would have $\forall \vec{x}$ generated from $G, f(\vec{x}) = h(\vec{x})$.

In theory, f has to be *learnable* [10][11][12] in order for a learning algorithm to achieve some sort of learning. To ensure *learnability*, assumptions need to be made in view of the setup, and there are five areas to make an assumption.

The first assumption concerns the hypothesis space H. It is intuitive to think that learnability depends on the complexity of the *assumed* H (call it H_L to differentiate it from the original unknown H), i.e. the more complex the H_L is, the more difficult the learning is. However, to formally articulate this intuition, one needs a formal way to define what "complexity" means.

If H_L is finite and enumerable, then its complexity can be measured more easily. For example, if H_L is the set of all Boolean functions based on n variables, then H_L contains 2^{2^n} distinct functions. Hence, its complexity can be characterized as $O(2^{2^n})$, i.e. its size.

The difficulty is when the assumed H_L is not restricted to be finite and enumerable, i.e. it can be infinite and/or uncountable. In this case, one cannot rely on counting to say about its complexity. One theory to measure the complexity of an assumed H_L is based on its ability to fit the data. This concept is called *capacity* of the hypothesis space which can be characterized in terms of the *VC dimension* [12].

The VC dimension (VC-D) also represents the minimum number of samples required to identify a f randomly chosen from H_L. Therefore, in order to ensure learning one needs to make an assumption on the VC-D, for example VC-D should be on the order of $poly(n)$ (polynomial in n, the number of features). Otherwise, the number of required samples can be too large for the learning to be practical.

The second assumption concerns the sample generator G. The common assumption is that G produces its samples by drawing a sample randomly according to a fixed distribution D. Hence, as far as the learning concerns, all future samples are generated according to the same distribution D.

The third assumption concerns the number of samples (m) available to the learning algorithm. This m has to be at

least as large as the VC-D. Otherwise, the samples are not enough for learning the answer f.

Even though the samples are sufficient (because we assume the samples are generated randomly, this is the same as saying that m is sufficiently large), we still need to concern if there exists a computationally efficient learning algorithm L to learn it. In most cases, learning the function f can be computationally hard [11].

The computational hardness can be characterized in terms of the traditional NP-Hardness notion [11] or the hardness for breaking a cryptography function [13]. For example, learning a 3-term DNF (Disjunctive Normal Form) formula using the DNF representation is hard [11]. In fact, except for a few special cases, learning based on a Boolean functional class is usually hard [11]. Moreover, learning based on a simple neural network is hard [14]. The computational hardness implies that in practice for most of the interesting learning problems, the learning algorithm can only be a *heuristic*. Consequently, its performance cannot be guaranteed for all problem instances.

The last assumption concerns how the answer h is evaluated. In the discussion of the other assumed areas above, we implicitly assume that the "closeness" of h to f is evaluated through an error function $Err()$, for example $Err(h, f) = Prob(h(\vec{x}) \neq f(\vec{x}))$ for a randomly drawn $\vec{x}$. Notice that with such an $Err()$, an acceptable answer does not require $h = f$. As far as the learning concerns, as long as their outputs are the same with a high probability, h can be an acceptable answer for f.

Such an error function is widely acceptable because in most of the applications the purpose of the learning is for prediction and the goal is to have a predictor whose accuracy is high enough. However, for applications in design and test such a view for evaluating the learning result can be misleading, for example when the use of a learning model is not for prediction, but for interpretation.

To see the intuition, consider that f is the AND function of 100 Boolean variables, i.e. $f(\vec{x}) = x_1 x_2 \cdots x_{100}$. For a randomly generated input $\vec{x}$, it is almost for sure that $f(\vec{x}) = 0$. As a result, for a given set of m samples it is likely that all we see in the dataset is that $y = 0$. The learner would output the hypothesis "$h = 0$" as the answer. How good is this answer? From the error function perspective, this answer is very accurate because the probability, $Prob(h(\vec{x}) \neq f(\vec{x}))$ for a randomly drawn $\vec{x}$, is very small, i.e. the probability is only $\frac{1}{2^{100}}$. On the other hand, "$h = 0$" is certainly not the correct answer and perhaps not a desirable answer in practice. In fact, the example f is called a "needle-in-a-haystack" function and the No-Free-Lunch [9] holds for learning such a function [15] (based on the demand to find the exact answer).

4.1. Learning can be hard in practice

In theory, learning is hard [15]. In practice, learning can be even harder. For a practitioner, the first challenge faced is to make a proper assumption regarding the hypothesis space H to begin with. This assumption essentially limits what f could be learned from the data. In most of the practical cases, there is no way to know if an assumed H_L is proper or not. Hence, in most cases the approach is to make an assumption that is as conservative as possible, for example to assume a H_L whose capacity is as large as possible.

However, in practice the capacity of the assumed H_L is also constrained by two other factors, the data availability and the computational power to execute the learning algorithm. If H_L is too complex, one might not have sufficient data to learn about f. Even with sufficient data, one might not have the computational resource to run the learning algorithm that can get to f. For example, this second constraint can be observed in the recent trend of hardware acceleration for deep learning neural networks [5].

The second challenge is that it is difficult to know if the data is sufficient. In theory the capacity of an assumed H_L can be measured in terms of its VC dimension [12]. In practice the effective capacity of a hypothesis space is also limited by the learning algorithm [5], making its estimation quite difficult. Consequently, unless the assumed H_L is simple enough to allow enumeration of all its hypotheses, its capacity is usually unknown, making it difficult to determine if the data is sufficient or not.

The third challenge comes from the learning algorithm. As mentioned above, a learning algorithm is likely to be just a heuristic. Its performance is not guaranteed. Also, the behavior of an algorithm can be parametrized and choosing the parameter values can further add to the challenge.

On top of all these challenges, it is also difficult to ensure that the generator indeed follows the same fixed distribution in the future when a learning model is applied.

4.2. Learning theories and the No-Free-Lunch

When can we say that a machine has learned? Answering the question can be quite challenging. One one hand, there are four theoretical frameworks trying to answer the question in supervised learning: the PAC (Probably Approximately Correct) [10][11] framework, the VC framework (the Statistical Learning Theory) [12], the SP framework (the Statistical Physics) (e.g. see [16]), and the Bayesian framework (e.g. see [17]). On the other hand, there is the No-Free-Lunch (NFL) theorem for learning [9] and NFL holds even in view of the four learning frameworks [18].

As discussed in [18], the main reason why none of the four frameworks can prove learning is because in order to do so, one has to allow all of the four things to vary (in contrast to fixing or discard any of them) in the formalism. These four things are, the hypothesis space H (and the f), the data, the the assumed H_L (and the h), and the cost measuring the goodness of h for f. In view of [18], the most general framework among the four is the Bayesian framework. However, it does not explicitly separate the assumed H_L from the unknown H in the formalism.

On the surface, NFL says that in general there is no learning, if learning means to learn from the samples at hand to predict about the *unseen* samples. More specifically, there is no one learning algorithm whose performance on average

is better than another, i.e. no algorithm is better than random guessing (and hence no learning). In view of Figure 6, one can think that NFL holds in general when one fails to justify the assumption made on H. Moreover, a variant of NFL holds if H satisfies a certain property [19].

In practice, practitioners often dismiss NFL based on the thinking that in a real world problem the H to be learned on cannot include all possible functions. To see this, assume that H is the set of all Boolean functions based on n variables. The size of H is 2^{2^n}. However, implementing most of these functions requires a circuit of $O(2^n)$ size which are unlikely to exist in the real world.

Such a thinking might have a point. However, it also points out that there is a need for making a realistic assumption regarding H. Once a practitioner makes an assumption on H to avoid NFL, how does one know the assumption is indeed realistic? This is a fundamental question that makes answering the original question hard.

4.3. NFL in practice

To illustrate an intuition behind NFL, consider the simple setup in Figure 7. Suppose the generator G produces the samples to cover 7 points in the Boolean space based on the three variables x_1, x_2, x_3. The learning might be seen as uncovering the true Boolean function that governs the input-output relationship.

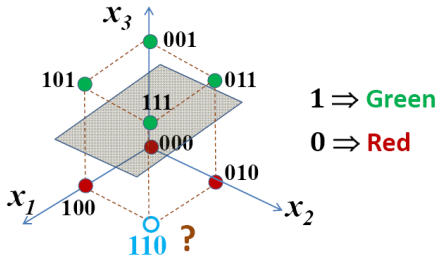

Figure 7. An example to illustrate NFL

Because there is only one unknown sample "110" left, the true answer can be either $f_a(\vec{x}) = x_3$ or $f_b(\vec{x}) = x_3 + x_1x_2x_3'$, i.e. $f_a(110) = 0$ or $f_b(110) = 1$. Suppose a learning algorithm chooses f_a as the answer because it is *simpler*. Then, based on the common error function mentioned before, we have $Err(f_a, f) \leq \frac{1}{16}$, i.e. at least a 93.75% accuracy. But is this learning? After all, if f_b is chosen we also have $Err(f_b, f) \leq \frac{1}{16}$.

In view of NFL, the error should be based on only the *unseen* sample 110. Because $f(110)$ can be either 0 or 1 and because seeing other samples provides no information to judge which answer is more likely to be true, there should be no learning here. Indeed, the conclusion that f_a is the answer is based on *logic minimization*, not learning.

Now consider the case where each input and the output in the setup is corrupted with a small noise, i.e. the input variables are no longer Boolean and the output value can be non-binary. In this case, one can have a large training dataset appear differently from a large validation dataset. Suppose the training dataset cover around each of the 7 points many times, except for around the point 110. A learning algorithm produces the answer: $f_a'(\vec{x}) = 1$ if $x_3 > 0.5$; otherwise $f_a'(\vec{x}) = 0$. This model also shows high accuracy, i.e. above or about 93.75% on the validation dataset. Is that learning or is that simply minimization plus noise filtering?

The simple example illustrates that it might be difficult to claim that a machine has learned in practice. The common *cross-validation* approach does not guarantee that. Cross-validation might be more meaningful if there is no overlap between the training dataset and the validation dataset after "noise filtering." But it can be challenging to draw the boundary between what is "noise" and what is not.

5. Machine Learning To A Practitioner

When applying a machine learning algorithm, a practitioner is often instructed to pay attention to the notion of *overfitting*. In practice, one is often instructed to observe overfitting in cross-validation. Let D_T and D_V denote the training and validation datasets. Let $EmErr(h, D)$ be an error function to report an *empirical error rate* by applying a model h onto the dataset D. Let the training error rate be $e_T = EmErr(h, D_T)$ and validation error rate be $e_V = EmErr(h, D_V)$. In learning, a learning algorithm has only D_T to work on. Hence, the algorithm tries to improve on e_T without knowing e_V.

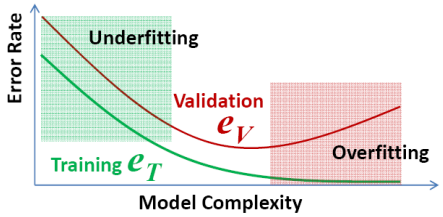

Figure 8. Overfitting Vs. Underfitting (Traditional View)

Traditionally, overfitting means that the learning algorithm continues to improve on e_T, but e_T and e_V deviate from each other. In contrast, *underfitting* means that e_T is high and hence there is still room for improvement. Fig. 8 illustrates these concepts where the x-axis can be thought of as a choice of model complexity.

Refer back to Figure 6. Because a learning algorithm does not know H, it assumes a hypothesis space H_L to begin with, for example assuming a particular neural network architecture. When the assumption of H_L is over-constrained (i.e. its capacity overly limited), H_L might not contain a hypothesis h that is close enough to f. As a result, $Err(h, f)$ does not approach zero. This can be considered as another perspective to understand the concept of underfitting.

In practice, to avoid underfitting the learning begins with an assumed H_L that is as less constrained as possible, i.e. with a capacity as large as possible. Assuming a more complex H_L can mean more samples needed to cover the hypothesis space. As a result, obtaining sufficient data samples to achieve a good coverage on H_L might become practically impossible.

For example, Figure 9 depicts such a situation. After a learning algorithm checks the hypotheses in H_L against

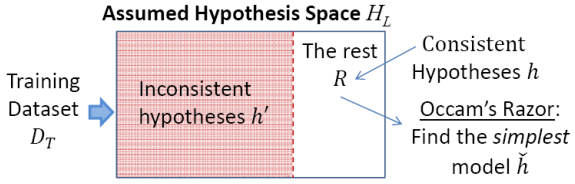

Figure 9. Occam's Razor in learning

all samples in the training dataset D_T, the space can be divided into two subsets. The first includes all hypotheses h' that are inconsistent with the samples, for example $EmErr(h', D_T) \neq 0$. Then, the rest R (called the *version space*) includes all *consistent* hypothesis h where $EmErr(h, D_T) = 0$.

In view of Figure 9, overfitting can be thought of as when R contains two or more distinct answers. Here "distinct" means existing one $\vec{x}$ where $\vec{x} \notin D_T$ and the two hypotheses h_1, h_2 result in two different values, i.e. $h_1(\vec{x}) \neq h_2(\vec{x})$. In other words, there exists a sample to differentiate h_1 and h_2 but this sample is not in D_T. In practice R can contain a large number of distinct hypotheses after learning on D_T.

5.1. Occam's Razor

In machine learning, a common strategy to pick a model in the version space R is based on the Occam's Razor principle, i.e. picking the "simplest" model as the answer. Applying the Occam's Razor principle requires a measure for model simplicity. Suppose this measure is defined. Then picking the model in R becomes an optimization problem, i.e. optimizing the model according to the simplicity measure. Because such an optimization problem can be computationally hard, a heuristic is usually developed to tackle the problem and such a heuristic does not guarantee always finding the optimal model.

In theory, there is also some subtlety to apply the Occam's Razor principle in learning [20]. Furthermore, the definition of a simplicity measure might or might not have a physical meaning in an application context. For all those reasons, an "optimized" model reported by a learning tool might provide little assurance on e_V. Consequently, cross-validation is required to evaluate the model even though it is considered as an optimized model by a learning algorithm.

5.2. A just-fit approach

Ideally, to avoid overfitting one desires that two conditions are met: (1) The assumed H_L contains the true answer f; (2) There are enough samples in the training dataset to filter out all hypotheses in H_L except one. In a recent work [3], the second condition is called the *just-fit*.

The just-fit can be taken as a requirement to be verified in the learning algorithm box in Figure 6. In this case, the algorithm essentially is performing a *coverage check*. This just-fit check ensures that the data is sufficient for a given H_L. With the just-fit check, the remaining issue in learning is to search for a proper H_L.

The work in [3] proposes such a learning approach: Instead of finding an optimal model for a given dataset based on a given H_L, the goal is to search for a proper H_L that satisfies the just-fit check. The approach is computationally more expensive than a traditional optimization-based learning algorithm. So far it can only be implemented for a limited scope of Boolean learning [3].

6. Machine Learning For Design and Test

As discussed in [1][3], many applications in design and test do not have a "big-data" problem. On the contrary, in some applications the data can be very limited such that running a reliable cross-validation is not practically feasible. This is a key reason to cast the learning into the view of the iterative search process in [1], and to motivate the development of the *just-fit* approach in [3].

The discussion so far suggests that it can be difficult to claim a data-driven solution as a machine learning solution, especially if one follows the perspective of NFL. Regardless of the difficulty, a practitioner can take a middle ground toward that issue. In view of Figure 6 one can take the perspective that if an approach concerns only the learning algorithm box, then it is for model building. For a machine learning approach, it should also include learning regarding the hypothesis space H. Loosely speaking, this can mean that the approach has to include learning about the *features* (and the parameters) used to define a hypothesis space [3].

This perspective for machine learning is consistent with the recent development in *deep learning* [5]. For a deep neural network, most of its layers are dedicated for learning the features. However, learning the features can also mean more data is required and in design and test, an application typically does not have the data big enough to employ such a "single-model" approach like deep learning.

The IEA system view in Figure 5 therefore provides an alternative to overcome the limited-data issue. This is because an IEA system relies on domain knowledge to build a workflow and consequently to break a high-level learning problem into a collection of small learning problems, i.e. recognition of concepts. Each concept poses a much easier learning problem. As a result, learning a reasonably good concept recognizers does not require big data [7]. Of course, the data requirement depends on the measure to assess what "reasonably good" means in view of the workflow.

6.1. The research path to IEA

As mentioned in Section 3, the research path leading to the current IEA view evolved through multiple stages. Four stages are highlighted below and their periods could overlap.

- In stage one, the research was algorithmic focus and centering on the question: What is the best learning algorithm to apply in a problem context?
- In stage two, it was methodology centric, focusing on developing a methodology to enable application of a machine learning algorithm.

- In stage three, it was application driven, focusing on determining which applications have a better chance to deploy a machine learning based solution.
- In stage four, the focus was on observing an added value over existing practices.

In the rest of the paper, selected past results are used to review the evolution in these four stages.

6.1.1. Algorithmic focus. The research path started with an early work in [21]. The choice of the algorithm for the study was Support Vector Machine (SVM) [22]. The paper stated: "The primary purpose of this work has twofold: to demonstrate that path-based learning is indeed feasible and to show how it can be applied in test and diagnosis applications." In other words, in the start the focus was on how a problem originated from design and test could be formulated in such a way that it is feasible to employ a machine learning algorithm. SVM was chosen because the algorithm had a solid theoretical foundation [12]. More importantly, the critical path selection problem could be cast into the support vector formulation [21].

In the next few years, the research continued to be algorithmic focus. For example, a later work in [23] provides a study of three algorithms in the context of delay-based outlier detection. They are: the traditional PCA [24], the random forests [25], and the one-class SVM [22].

In [26], the application is for Fmax prediction, i.e. predicting functional Fmax based on structural timing measurements. Two advanced regression methods, SVR (support vector regression) and Gaussian Process [27], were compared to three traditional methods: nearest neighbor, linear regression, and ridge regression [28].

By 2009, several other problems had also been studied in the areas such as functional verification, layout verification, and design-silicon timing correlation (see [1] for more detailed discussion). The challenges for developing a practical machine learning solution gradually became clear.

Take Fmax prediction as an example. Usually, because characterizing the Fmax was expensive, only a small number of chips (samples) were characterized in the early post-silicon stage. On the other hand, the number of *features* (the structural delay variables) could be large. Without sufficient samples, the problem was more on selecting an optimal small set of features than building an optimal model.

More importantly, Figure 10 illustrates a fundamental issue in Fmax prediction. The experiment was based on high-performance processor chips at the time. There were 1443 delay variables to be chosen for building a Fmax model. Two lots of chips were studied. As shown, for chips from Lot 1 the best selection includes 23 delay variables with prediction accuracy at 98%. For chips from Lot 2 the best selection includes 46 variables with accuracy at 80%. The most noticeable issue is plotted on the right where the locations of the best 23 delay variables selected were shown – they are quite different between the two lots.

Results in Figure 10 show that if a Fmax model is built on one Lot, the model might not be applicable on another Lot. Keep in mind that the study was based on high-performance chips and hence, the precision requirement for the Fmax prediction was high. In view of Figure 6, this can be seen as a violation of the assumption that G always follows the same fixed distribution D in training and in application of a learning model.

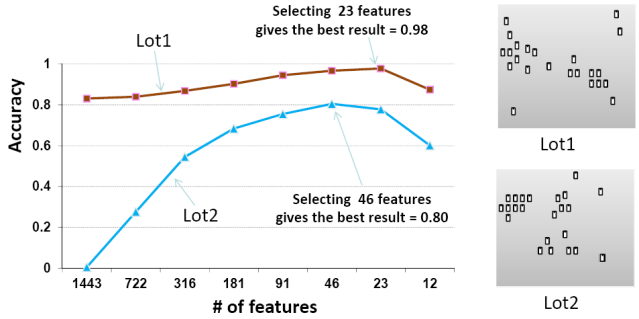

Figure 10. Issue for deploying a Fmax model

It is interesting to note that in a recent work [29], five outlier analysis methods are studied. However, the focus is no longer to see which one is the best. Rather, it is observed that on one is always better than another (i.e. the *seemingly NFL* notion). Hence, the focus is to develop an *applicability* measure to assess *when* one algorithm is the best to apply.

6.1.2. Methodology centric. Between 2007 and 2010, one focused research area was design-silicon timing correlation [30]. Figure 11 shows the application context in [31].

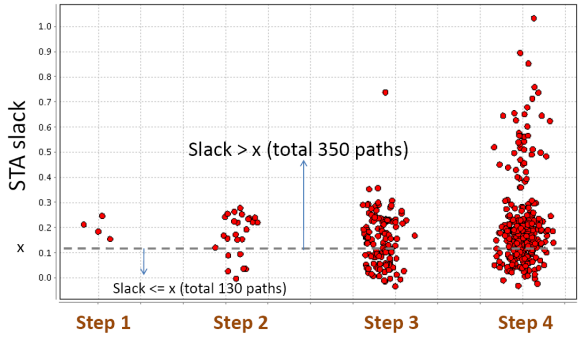

Figure 11. An application context for design-silicon correlation

In the figure, each red dot is a timing path. There are 480 dots shown. The 130 dots below the x dash line represent critical paths determined by the static timing analyzer (STA). The rest 350 paths above are non-critical. The Step 1 to Step 4 are frequency steps used to measure the path delays on silicon. Dots shown above Step 1 are the slowest paths, and all dots shown are considered as silicon-timing critical.

The question is, why the 350 STA-non-critical paths are silicon-timing critical? In fact, based on the given x value, the STA reports 21,589 STA-critical paths. Hence, there are 21,459 STA-critical paths do not show up in the figure. To enable the analysis, a set of design-related features were developed for describing a path. A *cause* to explain the data is represented as a combination of features [31].

While some non-obvious causes were found in [31] and verified to be a real issue, it was unclear if the added value was mostly due to the learning or due to the design features

manually developed. The learning methodology demands the set of features to start. However, if important features are already manually included, it is questionable if solving the remaining problem can be called "machine learning." For example, if the analysis is based on a simple *rule learning*, is that "machine learning?"

6.1.3. Application driven. From 2011, the research began to focus more on test data analytics. Two noticeable applications were: test cost reduction [32][33] and customer return modeling [34]. Take brun-in cost reduction as an example. The basic thinking is to build an outlier model to predict a burn-in fail in order to select (or skip) chips for burn-in. For example, Figure 12 shows an outlier model that projects a known burn-in fail (red dot) as an "outlier."

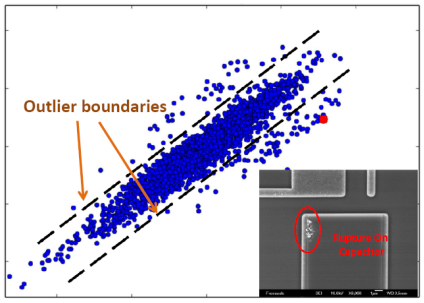

Figure 12. An outlier model for predicting a burn-in fail

However, the difficulty for building a burn-in fail prediction model is illustrated in Figure 13. Each dot represents a part. The x,y of a part represent their measured value based on the same test, before and after the burn-in. The plot shows that there are many parts (within the big red circle) whose test values before the burn-in are very normal, but those values after the burn-in become abnormal. This shows that the burn-in did alter the characteristics on those parts. Then, how does one know that the test data before the burn-in has sufficient information to allow the prediction at all?

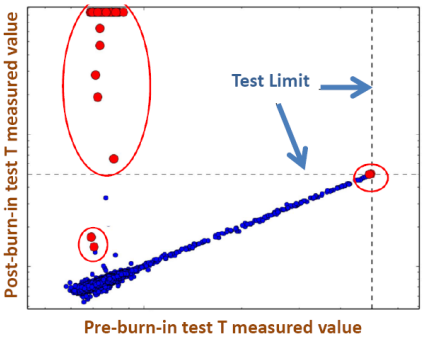

Figure 13. Difficulty for predicting burn-in fails

The discussion so far highlights various challenges for developing a machine learning solution in practice. Data availability and domain knowledge availability are two important considerations when assessing if an application is ready to employ such a solution. Impact to the existing infrastructure is another important consideration.

6.1.4. Added value. Since 2013, the research has converged into two application areas, functional verification [35] and production yield optimization [36].

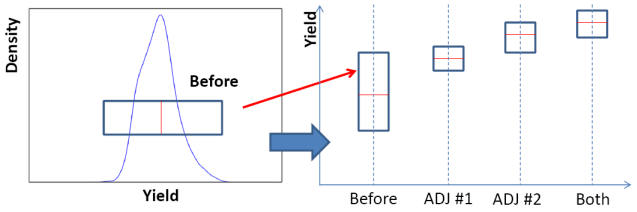

Figure 14. Added-value in terms of yield improvements

The work in [36] marks a turning point for the research. The work started when we were given the data to resolve a production yield issue. The product is a SoC for the automotive market. The production had been in progress for more than a year. However, a significant yield fluctuation was observed in the production. This fluctuation is illustrated in the left plot in Figure 14. The plot is based on about 2000 wafers. Prior to our engagement, both product team and design team had worked on the problem for many months. One design revision, multiple test revisions, and several process adjustments had been implemented to improve the yield, but none of those attempts succeeded. We were tasked by the company to investigate if "advanced machine learning" techniques could help discover a new solution.

After working on the problem for several weeks, we discovered enough evidences to recommend 5 process parameters for tuning. The contracted foundry accepted and implemented them as two adjustments, ADJ#1 and ADJ#2. Wafers were produced based on applying each adjustment individually and both together ("Both" in Figure 14), resulting in significant yield improvements as shown.

While the added value was observed, it was also realized that the main reason we were able to uncover a solution to resolve the yield issue while the product team could not, was not because we had a better machine learning toolkit. Rather, it was because we followed an analytic workflow (in our mind) that performed more comprehensive analytics than what the product team had done. Our workflow also included two novel aspects: (1) We observed that a concept (such as "strong correlation") could appear in a production period but might not always be there. Hence, a search along the temporal line was required. (2) We developed a way to recognize the concept "no correlation." This turned out to be quite useful for demonstrating minimal risk associated with a recommended process parameter tuning.

In the next two years, we continued to work with the company on other product lines. After learning enough experience ourselves, we were able to improve the workflow and also articulate it. It was the articulation of the workflow, which led to the IEA system view shown in Figure 5.

In retrospect, the algorithmic-focus stage started with the motivation to apply machine learning in design and test, and ended with the realization that algorithm was not the most critical issue for developing a machine learning solution in practice. While understanding machine learning algorithms are essential, it is far from being sufficient.

The methodology-centric stage started with the feature-based learning methodology [1], and ended with the realization that the demand for a feature set to start learning is an important barrier to prevent practical use of the methodology. During the application-driven stage the research converged into two applications, functional verification and production yield optimization.

Finally, in the fourth stage practical added value was observed, but it was also realized that most of the added value was due to the domain knowledge of an analyst (i.e. the analyst box in Figure 1). Hence, if one desires to provide a solution with the entirety of the added value, one has to build an autonomous system as illustrated in Figure 5.

7. Final Remarks

It is important to note that all the perspectives described in this review regarding "applying machine learning" are based on author's own experience in a limited set of application contexts. The field is much bigger than what has been reviewed. Hence, one should not take the perspectives as they are generally applicable. Every perspective has its background context and it is important to consider the perspective with the context together. Can machine learning be applied not following the IEA view? Of course it can. Is finding the best machine learning algorithm an important question? Of course there can be application contexts where this question is the most important one.

This review is not meant to be a survey. The discussion might have focused more on the negative side of the results and the limitations for applying machine learning. This is because those aspects are usually less emphasized in a regular paper, but to author's experience are important for seeing the big picture of a work. This does not imply that the positive side of the results is less important. Also, some questions are given without a clear answer. This is because either the author does not have one or discussing the answer further might take too much space. Nevertheless, it is still important to state those questions because sometime, a question itself might be more important than the answer.

Acknowledgment: This work is supported in part by National Science Foundation Grant No. 1618118. The author would also like to thank his doctoral student Chuanhe (Jay) Shan for his invaluable help on preparing this paper and its presentation at the conference, including the prototype of an IEA system to enable a demo during the presentation.

References

[1] L.-C. Wang, "Experience of data analytics in EDA and test - principles, promises, and challenges," *IEEE Trans. on CAD*, vol. 36, no. 6, pp. 885–898, 2017.

[2] Pedregosa and et al., "Scikit-learn: Machine learning in python," *JMLR*, pp. 2825–2830, 2011.

[3] L.-C. Wang, "Chapter 10: Learning from Limited Data in VLSI CAD," *Machine Learning in VLSI Computer–Aided Design*, 2018.

[4] ed. by Azim Eskandarian, "Section 10 fully autonomous driving," *Handbook of Intelligent Vehicles*, 2012.

[5] I. Goodfellow, Y. Benjio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[6] [Online]. Available: https://iea.ece.ucsb.edu/

[7] M. Nero, J. Shan, L. Wang, and N. Sumikawa, "Concept recognition in production yield data analytics," *International Test Conference*, 2018.

[8] S. Siatkowski, L.-C. Wang, N. Sumikawa, and L. Winemberg, "Learning the process for correlation analysis," *VLSI Test Symposium*, 2017.

[9] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Compt.*, vol. 8, no. 7, pp. 1341–1390, 1996.

[10] L. G. Valiant, "A theory of learnable," *Communications of ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[11] M. J. Kearns and U. Vazirani, *An Introduction to Computational Learning Theory*. The MIT Press, 1994.

[12] V. Vapnik, *The Nature of Statistical Learning Theory*, 2000.

[13] M. J. Kearns and U. Vazirani, "Cryptographic limitations on learning boolean formulae and finite automata," *JACM*, vol. 14, no. 1, pp. 67–95, 1994.

[14] A. Daniely, N. Linial, and S. Shaleve-Shwartz, "From average case complexity to improper learning complexity," *ACM Symposium on Theory of Computing*, pp. 441–448, 2014.

[15] T. M. English, "Optimization is easy and learning is hard in the typical function," *Proceedings of the CEC*, pp. 924–931, 2000.

[16] A. Engel and C. V. den Broeck, *Statistical Mechanics of Learning*. Cambridge University Press, 2001.

[17] D. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[18] D. H. Wolpert, "The Relationship Between PAC, the Statistical Physics framework, the Bayesian framework, and the VC framework," *Technical Report, SFI-TR-03-123*, 2003.

[19] D. Whitley, "Chapter 16: Sharpened and Focused No Free Lunch and Complexity Theory," *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, 2014.

[20] D. H. Wolpert, "The relationship between Occam's Razor and convergent guessing," *Complex System*, vol. 4, pp. 319–368, 1990.

[21] L.-C. Wang and et al., "On path-based learning and its applications in delay test and diagnosis," in *Design Automation Conference*, 2004.

[22] B. Schlkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2001.

[23] S. H. Wu, and et al., "A study of outlier analysis techniques for delay testing," *International Test Conference*, 2008.

[24] I. Jolliffe, *Principal Component Analysis*. Springer, 1986.

[25] L. Breiman, "Random forests," *Machine Learning Journal*, vol. 45, pp. 5–32, 2001.

[26] J. Chen and et al., "Data learning techniques and methodology for fmax prediction," *International Test Conference*, 2009.

[27] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[28] T. Hastie and et al., *The Elements of Statistical Learning - Date Mining, Inference, and Prediction*. Springer Series in Statistics, 2001.

[29] L.-C. Wang and et al., "Some considerations on choosing an outlier method for automotive product lines," *International Test Conference*, 2017.

[30] L.-C. Wang and et al., "Design-silicon timing correlation — a data mining perspective," *Design Automation Conference*, 2007.

[31] J. Chen and et al., "Mining ac delay measurements for understanding speed-limiting paths," *International Test Conference*, 2010.

[32] D. G. Drmanac, L.-C. Wang, and M. Laisne, "Wafer probe test cost reduction of an rf/a device by automatic testset minimization: A case study," *International Test Conference*, 2011.

[33] N. Sumikawa, Li-C.Wang, and M. S. Abadir, "An experiment of burn-in time reduction based on parametric test analysis," *International Test Conference*, 2010.

[34] N. Sumikawa and et al., "Screening customer returns with multivariate test analysis," *International Test Conference*, 2013.

[35] W. Chen, L.-C. Wang, and J. Bhadra, "Simulation knowledge extraction and reuse in constrained random processor verification," *Design Automation Conference*, 2013.

[36] J. Tikkanen, S. Siatkowski, N. Sumikawa, L.-C. Wang, and M. S. Abadir, "Yield optimization using advanced statistical correlation methods," *IEEE International Test Conference*, 2014.