Machine Learning for Design and Manufacturing Automation

Li-C. Wang
University of California, Santa Barbara
Background – Our Experience (since 2003)

- **Supervised learning**
  - Classification
  - Regression

- **Unsupervised learning**
  - Transformation
  - Clustering
  - Outlier
  - Rule Learning

**Apply**

- Delay test
- Fmax
- Test cost reduction
- Po-Si Validation
- Yield
- Customer return
- Design-silicon timing correlation
- Layout hotspot
- Functional verification

**Pre-silicon | Post-silicon | Post-shipping**

**Practical | Academic | InBetween**

AI Seminar 9/11/2017

— “Experience of Data Analytics in EDA and Test – Principles, Promises, and Challenges”
Outline

- Basic Concepts (1 hour)
  - Machine learning basic
  - Key points to apply machine learning

- Application examples (45 minutes)
  - Functional verification
  - Yield optimization

- Summary (2 minutes)
  - Autonomous learning system
Outline

- **Basic Concepts**
  - Machine learning basic
    - Key points to apply machine learning

- **Application examples**
  - Functional verification
  - Yield optimization

- **Summary**
  - Autonomous learning system
Machine Learning

Artificial Intelligence (Acting humanly – The Turing Test Approach)

- 1950 “Computing Machinery and Intelligence” – The Turing Test
- Other AI: Thinking humanly, Thinking rationally, Acting rationally
Training from “Data”

- Machine learns from training data to build a model
- The model is used to predict future “unseen” data
Two Practitioners’ Questions

- What machine learning algorithm (tool) to use?
- How can I use the Model?
ML Python Lib: http://scikit-learn.org/

Classification
- Identifying to which category an object belongs to.
- **Applications**: Spam detection, image recognition.
- **Algorithms**: SVM, nearest neighbors, random forest, ...

Regression
- Predicting a continuous-valued attribute associated with an object.
- **Applications**: Drug response, Stock prices.
- **Algorithms**: SVR, ridge regression, Lasso, ...

Clustering
- Automatic grouping of similar objects into sets.
- **Applications**: Customer segmentation, Grouping experiment outcomes.
- **Algorithms**: K-Means, spectral clustering, mean-shift, ...

Dimensionality reduction
- Reducing the number of random variables to consider.
- **Applications**: Visualization, increased efficiency.
- **Algorithms**: PCA, feature selection, non-negative matrix factorization.

Model selection
- Comparing, validating and choosing parameters and models.
- **Goal**: Improved accuracy via parameter tuning.
- **Modules**: grid search, cross validation, metrics.

Preprocessing
- Feature extraction and normalization.
- **Application**: Transforming input data such as text for use with machine learning algorithms.
- **Modules**: preprocessing, feature extraction.
A learning tool usually takes the dataset as above

- **Samples**: examples to be reasoned on
- **Features**: aspects to describe a sample
- **Vectors**: resulting vector representing a sample
- **Labels**: care behavior to be learned from (optional)
Supervised Learning

- **Classification**
  - Labels represent classes (e.g. +1, -1: binary classes)

- **Regression**
  - Labels are some numerical values (e.g. frequencies)
Unsupervised Learning

(Features) \( f_1 \quad f_2 \quad ... \quad f_n \)

\[
X = \begin{bmatrix}
\vec{x}_1 \\
\vec{x}_2 \\
\vdots \\
\vec{x}_m \\
\end{bmatrix} = \begin{bmatrix}
x_{11} & x_{12} & ... & x_{1n} \\
x_{21} & x_{22} & ... & x_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
x_{m1} & x_{m2} & ... & x_{mn} \\
\end{bmatrix} \quad \rightarrow \quad \vec{y} = \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_m \\
\end{bmatrix}
\]

No y’s

- Work on features
  - Transformation
  - Dimension reduction

- Work on samples
  - Clustering
  - Novelty detection
  - Density estimation

AI Seminar 9/11/2017
Two Most Common Questions

- Which tool should I use?
- Do you have a better tool?

Years ago I would probably say ...
Well ... it is application dependent
Two Most Common Questions

- Which tool should I use?
- Do you have a better tool?

Now I will tell you this ... It is not about the ML tool!
Outline

- **Basic Concepts**
  - Machine learning basic
  - Key points to apply machine learning

- **Application examples**
  - Functional verification
  - Yield optimization

- **Summary**
  - Autonomous learning system
What? It Is NOT About ML Tool (Algorithm)?

➢ Why you said that?

➢ Because there is NO “Machine Learning!”
What?

- NO Machine earning?
- Then, why am I here?
Quotes From The Vajra Prajna Paramita Sutra

- 金剛般若波羅蜜經
  - 思辨型式: 肯定→否定→肯定 (Define ⇒ Undefine ⇒ ReDefine)

- “So as it is called All Dharmas; There is No Dharmas; That is why it is called All Dharmas”
  - “言一切法者, 即非一切法, 是故名一切法”

- “So as it is called All Thought; There is No Thought; That is why it is called All Thought”
  - “如來說諸心, 皆為非心, 是名為心”

- “So as it is called All Living; There is No Living; That is why it is called All Living
  - “眾生眾生者, 如來說非眾生, 是名眾生”
Thought Process of The Talk

Machine Learning

No Machine Learning

Machine Learning

Loop
No Machine Learning – Two Views

- In general, even with unlimited data there is no one ML algorithm that is better than another
  - “General” – The hypothesis space comprises all possible functions based on \( n \) variables

- With limited/small dataset, likely no ML algorithm can give you the correct model
  - “Limited/small” – the data is constrained
No Free Lunch for ML – Wolpert 1996

- In general, there is no one ML algorithm that is better than another (e.g. random guess)

- How ML community counter-argues NFL?
  - Theory: Bayesian Inference + Kolmogorov Complexity + Occam razor principle + As data grows to infinity
  - Occam razor: Make a “smooth” assumption

- Wolpert 2011 – Occam’s razor would provide a free lunch if there is a reason to believe that your model becomes more accurate when fed with more data
“Smooth” Assumption

- ML would say the color of the area is **blue**
- NFL would say “**No, it can be any color**”
Another Example of “Smooth” Assumption

- Occam’s razor would give you a simpler model and conclude “It is red”
- NFL would say “can be either green or red”
  - See IEEE TCAD 36(6) 2017
    “Experience of Data Analytics in EDA and Test”
Five Key Assumptions To Prove “Learnable”

- A restriction on $F$ (otherwise, NFL)
- An assumption on $D$ (i.e. not time-varied)
- Assuming size $m$ is in order $O(poly(n))$, $n$: # of features
- Making sure an efficient algorithm $L$ exists
- Assuming a way to measure error $Err(f(x), h(x))$
VC Dimension measures $F$’s “complexity”

=> The minimum # of samples required is ~ VC-D (Sample Complexity)
Theory – Learnability

- L. G. Valiant, 1984
- Efficiently PAC-Learnable
  - Algorithm $L$ runs in $\text{Poly}(n, \text{sizeof}(f), \frac{1}{\delta}, \frac{1}{\epsilon})$
  - $L$ output hypothesis $h$ with probability $> 1 - \delta$
    - Probably
  - $h$ has an average error $< \epsilon$
    - Approximately Correct

- The PAC model can be used to prove that a class of $F$
  - Is learnable ($=\text{VC-D is poly}(n)$)
  - Is learnable but does not have an efficient algorithm
  - Is efficiently learnable
To learn, $F$ cannot be too complex e.g. VC Dimension($F$) $\sim$ Poly($n$)
Learning Representation $F$ Can Be Hard

Sample Generator $G$ \[ \rightarrow \] Function $f$ \[ f(x) \rightarrow \] Functional Class Representation $F$

Learning Algorithm $L$ \[ \rightarrow \] Hypothesis $h$

Assume $D$ is optimal for learning even though $F$ is learnable, learning $f$ can still be NP-Hard
Representation Independent Learning Can be Hard

Assume \( D \) is optimal for learning and \( F \) is learnable, Representation-Independent Learning of \( F \) can be Crypto-Hard
Learning Complexity Hierarchy

No Free Lunch

K-term DNF formula
Poly-size circuit

Poly-size Neural Net
Depth-2 circuit

Efficiently Learnable

Learnable

NP-Hard

Crypto-Hard

Poly VC-D

All Possible Function Classes
But, ML Is So Successful, Isn’t It?

- Speech recognition
- Language translation
- Computer vision
- Autonomous vehicle
- ...

- If there is no ML, how could they be so successful?
The Common Approach To Succeed in ML

- No Free Lunch
- Learnable
- NP-Hard
- Poly VC-D
- Crypto-Hard
- All Possible Functions

Assume the largest Deep Neural Network possible

- Demand huge data for training!
- Demand special HW for speed!

Assume your target is here
How ML Learns A Subject? (e.g. recognize a tree)

After loop $\infty$ times on the same subject, the robot will eventually pass the test

- This is how in principle, today’s ML passes the test
- In other words, BIG DATA saves the day!
Does ML Really Work? (Blind Spot in Perception)

- Blind spot – examples that can trick your machine learning model

Does ML Really Work? (Blind Spot in Perception)

- Clean image (b) is recognized correctly as a “washer” through the camera, while adversarial images (c) and (d) are misclassified.

One Reason For Not Being Able To Learn

No Free Lunch

Learnable

NP-Hard

Cryptographic Hard

Poly VC-D

All Possible Functions

Your target is here

But you apply a learning method assuming here
Other Reasons of Bad Learning Result

No Free Lunch

Learnable

NP-Hard

Poly VC-D

Crypto-Hard

All Possible Functions

But your learning Algorithm is only a heuristic, or you have not seen enough training samples

Your target is here
What We Stand in Theory

- **There is Machine Learning**
  - You know the VC Dimension is low &&
  - You know you have sufficient data &&
  - You know there is an efficient algorithm, or **you believe your heuristic will work well**
    - OR NP=RP=P
    - AND There is a poly algorithm to break a Crypto function

- **There is no Machine Learning**
  - You don’t make an assumption on the function class, and VC Dimension is not Poly(n), or
  - You don’t know an efficient algorithm (and P <> NP), or
  - You don’t have enough samples

- **Caveat: Complexity theory is based on** $n \to \infty$
  - How about practical learning problems?
Efficiently Learnable Analogy to SAT Solver

- **Boolean Satisfiability is NP-Complete**
  - Practical SAT solver continues to improve its performance
  - More benchmarks can be solved every year
  - Can we solve all practical problems one day?

- **Machine Learning for Perception**
  - No one knows the true complexity of the problem
  - We know learning a poly-size Neural Net is Crypto-Hard
  - Practical Neural Net model continues to improve
  - More benchmarks can be learned every year
  - Can we achieve Human Perception one day?
There is no machine learning
   – (if we don’t mention anything else in our assumption)

There is machine learning if we know a certain class of functions to be learned, or P=NP

Otherwise, in practice it is usually unknown if we can have meaningful machine learning!
Do I Apply Deep Learning Network?

- Do you know your network (capacity) is large enough to capture the complexity of the target?
- Do you have large enough dataset?
- Do you have enough computation power (HW)?
- What if I don’t?

- You need to restrict the learning complexity by selecting a small enough \( n \) to enable learning!
Considering Efficiently PAC-Learnable Only
Learning a Boolean Monomial

- Given $n$ Boolean variables $x_1, x_2, \cdots, x_n$, the class $F$ contains all possible monomials
  - e.g. $x_2x_5x_8$, or $x'_3x_6x'_10x_{12}x_{20}$

- For sample size $m > \frac{2n}{\varepsilon} \left( \ln(2n) + \ln \left( \frac{1}{\delta} \right) \right)$, the functional is efficiently PAC-learnable

- To learn, only positive samples ($f(x)=1$) are needed
  - Negative samples provide little or no power for learning
Monomial Learning Can be The Basis For Many Problems Encountered in Design & Manufacturing

- **Supervised learning**
  - Classification
  - Regression

- **Unsupervised learning**
  - Transformation
  - Clustering
  - Outlier
  - Rule Learning

- **Apply**
  - Functional verification
  - Layout hotspot
  - Design-silicon timing correlation
  - Yield
  - Pre-silicon
  - Post-silicon
  - Post-shipping

- **AI Seminar**
  - 9/11/2017

- **AI Problems**
  - Delay test
  - Fmax
  - Test cost reduction
  - Po-Si Validation
  - Customer return

- **AI Knowledge**
  - Practical
  - Academic
  - InBetween
Functional Verification

- $x_1, x_2, \cdots, x_n$ are controllable signals
- What combination of signals can activate coverage point A (e.g. $A=z_1 z_2 =01$)?
  - monomial learning problem
A snippet of layout can be described with $n$ features $x_1, x_2, \cdots, x_n$

What combination of features explain a failure (or hot spot) at the layout location?

– monomial learning problem
A path can be described with $n$ features $x_1, x_2, \cdots, x_n$

What combination of features explain the observed unexpected silicon timing behavior?

– monomial learning problem
There are $n$ factors $x_1, x_2, \ldots, x_n$ that can affect yield.

What combination of factors cause the yield drop?

- At least as hard as monomial learning problem.
Customer Return Modeling

- **A customer return** passes all test individually, but fails at customer site
  - Suppose there are $n$ tests $x_1, x_2, \ldots, x_n$

- **What combination of tests can be used to screen the return as an outlier?**
  - At least as hard as monomial learning problem
So this is a good news because Design/Manufacturing learning problems are easier?

Not Quite! There are other constraints making it hard
Things Making Monomial Learning Hard

- You don’t have large enough samples
- Even you do, you only have very few or even no positive samples at all
- You need to know the true answer, not just the “average prediction error rate”
- You have little control of your sample generator $G$ – the sample distribution can be highly biased so some samples never got generated
How Big The $m$ Required For An $n$

- The number of features $n$ decides the required $m$
  - Assuming $0.1\% \epsilon$ error bound, and $99\% \delta$ probably correct

$$m > \frac{2n}{0.001} \left( \ln(2n) + \ln\left(\frac{1}{0.01}\right) \right)$$

![Graph showing the relationship between $m$ and $n$ features](image)
Now, Considering Your Problem

➢ What if your don’t have the require $m$ number of samples for your selected $n$ features?

➢ You need to select a small enough $n$ to enable learning!
What If I Have 1 (or no) Positive Sample?

- Assuming that we want to learn the true answer with high probability ($\epsilon = 0$)
- Let us see how much power negative samples can provide with a fixed $n$
What If I Have 1 (or no) Positive Sample?

- Assuming that we want to learn the true answer with high probability ($\epsilon = 0$)
- Let us see how much power negative samples can provide with a fixed $n$

*With 0 positive sample, $n = 12$*

<table>
<thead>
<tr>
<th>Number of negative samples</th>
<th>Percentage of removed monomials</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0%</td>
</tr>
<tr>
<td>68</td>
<td>20%</td>
</tr>
<tr>
<td>135</td>
<td>40%</td>
</tr>
<tr>
<td>202</td>
<td>60%</td>
</tr>
<tr>
<td>269</td>
<td>80%</td>
</tr>
<tr>
<td>336</td>
<td>100%</td>
</tr>
<tr>
<td>403</td>
<td></td>
</tr>
<tr>
<td>470</td>
<td></td>
</tr>
<tr>
<td>537</td>
<td></td>
</tr>
<tr>
<td>604</td>
<td></td>
</tr>
<tr>
<td>671</td>
<td></td>
</tr>
<tr>
<td>738</td>
<td></td>
</tr>
<tr>
<td>805</td>
<td></td>
</tr>
<tr>
<td>872</td>
<td></td>
</tr>
<tr>
<td>939</td>
<td></td>
</tr>
</tbody>
</table>

88.2% remains
What Can You Do?

- e.g. You can limit the hypothesis space by limiting the size of the hypothesis to be \( \leq 3 \) variables

% of all \( k \leq 3 \) monomials are filtered

Number of variables \( n = 30 \)
Target monomial length \( k \leq 3 \)

- Experiment
- Theoretical
- Lower Bound

Number of negative samples
The Required $m$ Is Sensitive To $k$

- The required $m$ is sensitive to your size assumption.

The graph shows the number of negative samples ($m$) required to have a 99% chance of filtering out all other monomials with length $\leq k$. The x-axis represents the number of variables ($n$), and the y-axis represents the number of negative samples ($m$). The graph includes curves for $k=3$ (blue), $k=4$ (red), and $k=5$ (green) for different values of $n$.
Biased distribution:
There are 5 pairs of variables. Two variables in a pair has the same value with probability 0.875
Now, Considering Your Problem Again

- What if your don’t have the require number of positive samples for your selected \( n \) features?

- And you don’t have uniformly random generation of samples?

- You need **Domain Knowledge**
  - To select a small enough \( n \)
  - Assume a small enough hypothesis size
  - To iteratively search for answer based on a small subset of features where uniformly random assumption is likely applicable
Limited/Small Data Problems

- Applying ML tools becomes an iterative search process
- The analyst
  - (1) Prepare the datasets to be analyzed (selecting features)
  - (2) Determine if the results are meaningful
- The effectiveness depends on how the analyst conducts these two steps – not about the tool in use!
Learn = Data + Knowledge

- Assume little knowledge
  - Machine Learning for efficiently learnable class
- Assume some knowledge
  - Limited data problems
- Demand more knowledge
  - Small-data problems

Data Size (Cost) => Demand large data

Very limited data
=> Demand more knowledge

Domain knowledge
Outline

- Basic Concepts
  - Machine learning basic
  - Key points to apply machine learning

- Application examples
  - Functional verification
  - Yield optimization

- Summary
  - Autonomous learning system
Generating Tests to Improve Coverage

- **Context:** To learn how to hit a target (coverage point)
- **Inputs:** Traces from positive/negative tests
- **Outputs:** rules (e.g. monomials)
- **Usage:** Use the rules to produce tests
Example #1

- Assertion A depends on filling up a queue A
  - Assertion A is hit by 3 tests out of 1000

- Learn a rule based on these 3 novel tests
One example rule can be illustrated as the following
- There is a `lmw` (load-multiple-word) instructions
- The page associated with the `lmw` address is not guarded
- The destination register of `lmw` is prior to G20

Improve coverage on assertion A
- Also cover related assertions B and C
Example #2

- Initially only assertion IV are covered
- After learning, all five are covered
- Results are improved further iteratively
Real Challenges

Testbench

Positive tests (traces)

Negative tests (traces)

Learning

Candidate rules

Tests are encoded in variables

C1: How to START with relevant variables?
Key variables must be included while keeping the number of variables small

C2: how to produce desired tests?
Learning Domain Knowledge from Documents

- Applying text mining techniques
- Extract signals from design Documents

Diagram:

1. Document → Text mining → Signal mapping → Signals
2. Tokenization
3. POS tagging
4. Filtering
5. Word → Name matching → Filter signals
6. Signal list → Partial matching
7. Filtering → signal(s)
Level 1: The set of variables discussed

Level 2: The subsets of variables that are related among themselves

Level 3: For each subset of variables, what their actual relationship is

- Level 1: Extract the set based on recognition of words used as variable names
- Level 2: Variables described together in a sentence, a paragraph, or related paragraph
- Level 3: Require natural language processing

*Feature extraction from design documents to enable rule learning for improving assertion coverage.*

*Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific, IEEE, 2017.*
Effectiveness of The Extracted Signals

- Use the extracted signals for rule learning
- Obtained 100%-accurate rules for 64% cov. points
Learning from
A Verification Engineer
(DAC 2017)
Learning from Test Examples

- Learning software to learn from test examples
- Use the learning software to generate new tests

Test \( t_1 \)
Test \( t_2 \)
...  
Test \( t_n \)

Learning Software

Learn from direct tests
Generate new tests

Test \( t'_1 \)
Test \( t'_2 \)
...
Avoid The Program Synthesis Problem

- Each direct test is a C program

- In general, the learning software can be thought of as solving a “Program Synthesis” problem
  - Teach a machine how to write C program

- Program Synthesis is a very difficult problem

- Avoid solving the Program Synthesis problem
Key Idea

- A test is represented by a sequence of primitives

\[ [A\ B\ C] \rightarrow \begin{array}{c}
\text{test} \\
\hline \\
\text{Prim. A} \\
\text{Prim. B} \\
\text{Prim. C}
\end{array} \]

- In this view, a test is represented as a sentence in an unknown formal language

- The learning software then tries to learn the unknown formal language from samples
Relevant Works

- **Grammatical Inference**
  - Learn an automaton from samples
  - It has been shown that if only in-model samples are available, the only learnable language is finite-length language
  - DFA is capable of describing finite-length languages
    - Our work uses DFA learning

- **Process Discovery**
  - An emerging field that learns process models, e.g. Petri nets, from process instances
  - Not able to apply this techniques directly to our applications
    - We focus on test generation, not process analysis
    - Need to ensure the generated tests are valid
Our New Learning Approach (DAC 2017)

- We call it **Constrained Process Discovery**
- Split the learning into two parts
  - Learning an upper-bound process model (a DFA)
  - Learn constraints into a database
- Using a SAT solver to generate tests that comply with both the model and constraints
Three tests: [A,B,C,D,H], [B,C,E,F,D,H], and [A,B,C,E,G,D,H]

New test is: [B, C, E, G, D, H]

This graph is built with the so-called “1-prefix” rule

A stronger prefix rule produces a less-generalized graph
Based on 30 direct tests
  – Test length is between 46 to 63 primitives

The upper-bound model
Experiment Results

- System-level RTL verification environment for a commercial dual-core microcontroller SoC
- 194 primitives, each a block of C code
- 22 constraints added while tests were developed

![Graph showing the increase in newly-covered CPs over time.](chart.png)

Legend:
- 500 new tests
- # of new tests generated

AI Seminar 9/11/2017
## More Results

<table>
<thead>
<tr>
<th># manual tests</th>
<th># originally covered CPs</th>
<th># generated tests</th>
<th># newly covered CPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>167</td>
<td>500</td>
<td>85</td>
</tr>
<tr>
<td>35</td>
<td>216</td>
<td>500</td>
<td>64</td>
</tr>
<tr>
<td>40</td>
<td>244</td>
<td>100</td>
<td>64</td>
</tr>
</tbody>
</table>

New tests produced from the process model can always improve the coverage.
Outline

- Basic Concepts
  - Machine learning basic
  - Key points to apply machine learning

- Application examples
  - Functional verification
  - Yield optimization

- Summary
  - Autonomous learning system
Learning From A Yield Analysis Engineer (VTS 2017)
An automotive SoC product
Yield fluctuated over time
Went through one design revision, multiple test revisions, and process adjustment experiments

Our result would provide “added value” to these efforts
Objective: Discover Process Adjustments

- **Task:** Discover strong correlations between a test fallout and a process parameter

- **Hope:** Adjustment of the process parameter leads to improved yield and reduced yield fluctuation

The desired outcome with yield optimization (what we want)

Distribution estimated based on 2000+ wafers

Yield

AI Seminar 9/11/2017
Basic Question

The correlation can exist in many different ways.

- This essentially is a search process.
Found several recommended process changes
- 3 process changes based on 5 process parameters
- 1 change applied to all experimental lots

Result: Significant yield improvement and reduction of fluctuation
Recall The “Iterative Search Process”

- In the yield example, we and the product teams had access to the same set of tools
- Why we succeeded and they did not?
  - Because we look at aspects they never did
  - We constructed and analyzed datasets in ways that they never thought about
How About Next Time?

Suppose a yield problem on another product line

**Same person:** What if the solution demands looking at the data from a different aspect?

What if the task is given to a different person?
Next, we will show how this can be done
To learn from analyst’s experience, we need to have a way to model the experience.

Our approach

- Define a set of operators
- Model experience as “an execution path” following a sequence of operators
- Record execution paths in a log file
- Apply process mining to learn from the log file
- Obtain a WF-net model as shown above
An Example Path

- Canonical correlation to show that the test result and the parameter are correlated

C-Correlation = 0.84
Determine that the 4th category of fails on the test is the one actually correlated to the parameter.
Further Optimization

- Improve correlation by considering temporal effects

Correlation without separation = -0.86

Average E-test value for PP1

Corr = -0.85

Corr = -0.91

# of 4th category fails
Yield Issue on a 2\textsuperscript{nd} Product Line

- Another automotive product line
- Operates in the 76-77 GHz band allocated for vehicular radars on an unlicensed basis (FCC)
- Unexpected yield loss at 77 GHz test in hot
Discover trim count is relevant to hot fails
This specific path was generalized by the learning and not stored in the log file
Determine that parameter C affects the frequency test value which decides the trim count.
Finding which process parameters are relevant is not enough – The next question:
- Which manufacturing recipes/tools I need to change?

There are three basic types of data to analyze:

- **THL**: Travel history (equipment) for each lot (recipes)
- **Chamber**: When each wafer went to which chamber in each tool
- **ED**: Equipment sensor data
Discovering A Meaningful Plot

- Combine E-test data and THL data enables discovery of a potential explanation of yield excursion

Example Removed
Outline

➢ Basic Concepts
  – Machine learning basic
  – Key points to apply machine learning

➢ Application examples
  – Functional verification
  – Yield optimization

➢ Summary
  – Autonomous learning system
For limited/small (D&M) data application context, we have to implement an Autonomous System to effectively apply Machine Learning

- i.e. Applying Machine Learning = Autonomous System
Key Take-Away Messages

- Machine learning is part of AI

- **BIG DATA** is the key for modern ML successes

- For various theoretical reasons, for machine learning in Design/Manufacturing **Domain Knowledge** is critical
  - To prepare the “right” datasets in iterative search
  - To determine meaningfulness of learning result

- Hence, “**Learning Domain Knowledge**” is the key

- Applying Machine learning = Autonomous system (AI)
THANK YOU!

Questions?