Machine Learning For Feature-Based Analytics

Li-C. Wang
University of California, Santa Barbara
Machine Learning is supposed to construct an "optimal" model to fit the data (whatever "optimal" means)
ML Tools: e.g. http://scikit-learn.org/

scikit-learn
Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification
Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: SVM, nearest neighbors, random forest, ...

--- Examples

Regression
Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, ridge regression, Lasso, ...

--- Examples

Clustering
Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes.

Algorithms: K-Means, spectral clustering, mean-shift, ...

--- Examples

Dimensionality reduction
Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency.

Algorithms: PCA, feature selection, non-negative matrix factorization.

--- Examples

Model selection
Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning.

Modules: grid search, cross validation, metrics.

--- Examples

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)
Noticeable ML Applications In Recent Years

Self-Driving Car

Mobile Google Translation

Smart Robot

AlphaGo (Google)

*These images are found in public domain
Take Image Recognition As An Example

- **ImageNet: Large Scale Visual Recognition Challenge**
  - 1000 Object Classes, 1.4M Images

<table>
<thead>
<tr>
<th>Year</th>
<th>Top-5 error rate</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>28.20%</td>
<td>AlexNet</td>
</tr>
<tr>
<td>2011</td>
<td>25.80%</td>
<td>8-Layer</td>
</tr>
<tr>
<td>2012</td>
<td>16.40%</td>
<td>ZFNet</td>
</tr>
<tr>
<td>2013</td>
<td>11.70%</td>
<td>VGG</td>
</tr>
<tr>
<td>2014</td>
<td>7.30%</td>
<td>19-Layer</td>
</tr>
<tr>
<td>2014</td>
<td>6.70%</td>
<td>GoogleNet</td>
</tr>
<tr>
<td>2015</td>
<td>3.57%</td>
<td>22-Layer</td>
</tr>
<tr>
<td>2016</td>
<td>5.10%</td>
<td>152-Layer</td>
</tr>
</tbody>
</table>

Also see: O. Russakovsky et al. [rXiv:1409.0575v3](https://arxiv.org/abs/1409.0575) [cs.CV] 2014

[http://www.image-net.org/]
Deep Learning for Image Recognition

- ImageNet: Large Scale Visual Recognition Challenge (http://www.image-net.org/challenges/LSVRC/)
  - 1000 Object Classes, 1.4M Images

1st Enabler: The availability of a large dataset to enable the study of deeper neural network

http://www.image-net.org/
Also see: O. Russakovsky et al. rXiv:1409.0575v3 [cs.CV] 2014
Deep Learning for Image Recognition

- ImageNet: Large Scale Visual Recognition Challenge (http://www.image-net.org/challenges/LSVRC/)
  - 1000 Object Classes, 1.4M Images

Top-5 error rate

2nd Enabler: The availability of efficient hardware to enable training with such a large neural network

http://www.image-net.org/
Also see: O. Russakovsky et al. rXiv:1409.0575v3 [cs.CV] 2014

ISPD 2018 Monterey CA - Wang
Question Often Asked By A Practitioner

➢ Which tool is better?

In many EDA/Test applications, it is not just about the tool!
See: Li-C. Wang, “Experience of Data Analytics in EDA and Test – Principles, Promises, and Challenges,” TCAD Vol 36, Issue 6, June 2017
Challenges in Machine Learning for EDA/Test

- **Data**
  - Data can be rather limited
  - Data can be extremely unbalanced (very few positive samples of interest, many negative samples)
  - **Cross-validation** is not an option

- **Model Evaluation**
  - The meaningfulness of a model specific to the context
  - Model evaluation can be rather expensive
e.g. Functional Verification

- **Goal**: to achieve more coverage on CP
- **Approach**: Analyze simulation traces to find out
  - What combination of signals can activate CP?
- **Features**: $f_1, f_2, \cdots, f_n$ are testbench-controllable signals
- **Data**: Few or no samples that cover CP
  - Positive Samples: 0 to few
  - Negative Samples: 1K to few K’s
e.g. Physical Verification

- **Goal**: to model causes for an issue
- **Approach**: Analyze snippets of layout images to find out
  - What combination of features can cause an issue?
- **Features**: $f_1, f_2, \cdots, f_n$ are developed based on domain knowledge to characterize geometry or material properties
- **Data**: Few samples for a particular type of issue
  - Positive Samples: 1 to few
  - Negative Samples: many
e.g. Timing Verification

- **Goal**: to model causes for a miss-predicted silicon critical path
- **Approach**: Analyze unexpected silicon critical paths
  - What combination of design features can cause an unexpected critical path?
- **Features**: $f_1, f_2, \cdots, f_n$ are developed based on design knowledge to characterize a timing path
- **Data**: Few samples for a particular type of critical path
  - Positive Samples: 1 to few
  - Negative Samples: many (STA critical but not silicon critical – about $25K$ paths)
e.g. Yield

- **Goal**: to find a receipt to improve yield
- **Approach**: Analyze wafer yield data with process parameters
  - Tuning what combination of process parameters can improve yield?
- **Features**: $f_1, f_2, \cdots, f_n$ are tunable process parameters
- **Data**: Samples can be parts or wafers
  - Positive Samples: Failing parts or Low-yield wafers
  - Negative Samples: Others
### Feature-Based Analytics

#### Problem:
- Search for a **combination of features** or feature values among a large set of features

#### Data:
- Interested in **positive samples**
- **Extremely unbalanced** – Many more negative samples and very few positive samples

#### Not a traditional feature selection problem
- Insufficient data
- Cannot apply cross-validation to check a model
In Practice, This Is What Happens

- Learning from data becomes an **iterative search** process (usually run by a person)

$n$ features to consider

- Selected Features

- Run ML
  - Check Result

- Run ML
  - Check Result

- Run ML
  - Check Result

ISPD 2018 Monterey CA - Wang
Learning is an iterative search process

The analyst
   - (1) Prepare the datasets to be analyzed
   - (2) Determine if the results are meaningful

The effectiveness depends on how the analyst conducts these two steps – not just about the tool in use!
The effectiveness of the search largely depends on how the Analyst Layer is conducted.
The Analyst Layer demands a Machine Learning Toolbox where the model can be assessed WITHOUT cross-validation.
Implications

Automation requires automating both the Analyst Layer and the Machine Learning Toolbox.
Machine Learning Toolbox
Questions

- Recall main issue: We can’t apply cross-validation
- Why do we need cross-validation?
- Why can a machine learning algorithm guarantees the accuracy of its output model?
- What’s a machine learning algorithm trying to optimize anyway?
Five Assumptions To Machine Learning

1) A restriction on $H$ (otherwise, NFL)
2) An assumption on $D$ (i.e. not time-varied)
3) Assuming size $m$ is in order $O(poly(n))$, $n$: # of features
4) Making sure a practical algorithm $L$ exists
5) Assuming a way to measure error, e.g. $Err(f(x), h(x))$
Because we don’t know how complex $H$ should be, we assume the most complex $H$ we can afford in training.
As A Result, We Need Occam’s Razor Assumption

- Hypothesis space: e.g. all possible assignment of weight values in a neural network (can be infinite)

- Occam’s Razor (Regularization): Find the “simplest” hypothesis that fit the data
  - Hence, many machine learning algorithms solve a non-convex constrained minimization problem (NP-Hard or Harder)

- However, the simplicity measure might not be meaningful in an application context
The Learning Algorithm

Sample Generator $G$ \rightarrow \text{(2) } D \rightarrow \text{Function } y = f(x) \rightarrow f \rightarrow \text{Hypothesis Space } H

\text{(3) } m \text{ samples } \rightarrow (x,y) \rightarrow \text{Learning Algorithm } L \rightarrow \text{(5) Hypothesis } h

Because non-convex optimization is hard, some heuristic is used, and the solution is often a local minimum
In Practice, Many Things Are Not Ideal

- Your assumption of the hypothesis space might be too simple (underfitting) or too complex (overfitting)

- You may not have sufficient data to identify the exact answer from your assumed hypothesis space

- Your learning algorithm is only a heuristic and does not guarantee to find the “optimal” model

- As a result, you need cross-validation
Can we have a ML tool that can produce a model with some guarantee, without using Cross-Validation?
Alternative Machine Learning View

- Traditional machine learning: Find an optimal model based on the given dataset

- Alternative machine learning: Find an interpretable Hypothesis Space Assumption $H$ where a model can JUST-FIT the dataset but not overfitting

Search for a Model  
Search for An Assumption
Search for the “JUST-FIT” hypothesis space
- Such that the output model among the few answers consistent with all the samples

The JUST-FIT hypothesis space (if exists) can be a measure of quality for the model
VeSC-CoL:  
Our Concept Learning Tool
VeSC-CoL


- Handle binary-valued features

- Target (interpretable) concept: *k-term DNF, for small k*

- Designed to handle extremely-unbalanced dataset without cross-validation

- Two implementations: SAT-Based and OBDD-Based
K-term DNF – Terminology

1-term DNF or Monomial

\[ x_1 \overline{x}_2 x_4 \]

Length \( l = \) number of literals \( = 3 \)

2-term DNF or Monomial

\[ x_1 \overline{x}_2 x_4 + \overline{x}_4 x_6 \]

Length \( l = \) number of literals \( = 3+2 = 5 \)

\( n = \) number of features (variables)
VeSC-CoL’s Hypothesis Space Search

Given an upper bound on $k$ for $k$-term DNF

$H_l$ is the hypothesis space for all hypotheses with length $l$
Runtime Examples (k=1)

- Correct answer is with $l = 5$
- $n$ does not affect runtime much
- $l$ limits how far we can search
Interesting Finding

- As $n$ increases, you are likely to run out of time than to run out of data (assuming most are negative samples)

Number of features: $n$

![Graph showing runtime vs. number of features](image)
For BDD-based implementation, the runtime wall happens in the early processing of the negative samples.

Number of features: $n=100$
Guarantee by VeSC-CoL

- Assuming the correct answer can be represented as a $k$-term DNF for a selected $k$, then VeSC-CoL always find the correct answer (assuming runtime is allowed and data is sufficient)
  - Experimentally shown for $k$ up to 3, $l$ up to 8, negative sample size up to 10K

<table>
<thead>
<tr>
<th>VeSC-CoL</th>
<th>CART</th>
<th>ID3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{2}x_{63} \overline{x_{75}}x_{78}x_{80}$</td>
<td>$x_{3}x_{4}x_{28}x_{47} \overline{x_{55}}x_{58}$</td>
<td>$x_{2}x_{3}x_{4}x_{30}x_{47} \overline{x_{55}}x_{58}$</td>
</tr>
<tr>
<td>$x_{39}x_{45} \overline{x_{72}}x_{74}x_{95}$</td>
<td>$\overline{x_{5}}x_{16}x_{35} \overline{x_{45}}x_{55}x_{56}x_{59}$</td>
<td>$x_{8}x_{40} \overline{x_{45}}x_{64}x_{74}x_{87}$</td>
</tr>
<tr>
<td>$x_{2}x_{14}x_{52}x_{57}x_{87}$</td>
<td>$x_{11}x_{14}x_{24}x_{61}x_{64}x_{90}x_{92}$</td>
<td>$x_{5}x_{6}x_{16}x_{35} \overline{x_{45}}x_{56}x_{59}$</td>
</tr>
<tr>
<td>$x_{40}x_{45}x_{64}x_{74}x_{87}$</td>
<td>$\overline{x_{4}}x_{8}x_{45}x_{47}x_{64}x_{74}x_{89}$</td>
<td>$x_{2}x_{14}x_{24}x_{61}x_{64}x_{90}x_{92}$</td>
</tr>
<tr>
<td>$x_{57}x_{58}x_{77}x_{95}x_{98}$</td>
<td>$\overline{x_{5}}x_{29}x_{38}x_{43}x_{79}x_{99} + \overline{x_{3}}x_{5}x_{29}x_{38}x_{43}x_{49}x_{79}x_{99}$</td>
<td>$x_{5}x_{6}x_{11}x_{14}x_{18}x_{34}x_{45}$</td>
</tr>
</tbody>
</table>

Always Correct  Always Incorrect  Always Incorrect
Analyst Layer Automation
Before this example, we had done work for resolving another yield issue for another product line.

**Question:** Can we learn to model the experience from that work and automate the Analyst Layer to resolve this yield issue?
The Learning Objective

1st context 2nd context

Analytics Software
Plots

Generalized Experience
Analytics Software
Plots

ISPD 2018 Monterey CA - Wang
Modeling “Experience”

- To learn from analyst’s experience, we need to have a way to model the experience

- Knowledge acquisition
  - 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 learning to learn from the log file
- Obtain a Process Model as shown above
Discover trim count is relevant to hot fails
Obtain A Meaningful Result

- Determine that parameter C affects the frequency test value which decides the trim count
Summary: Three Observations

- The effectiveness of “Machine Learning” largely depends on how the Analyst Layer is conducted.

- Automation of “Machine Learning” needs to include automation of the Analyst Layer.

- Traditional machine learning tools are not designed to effectively support the Analyst Layer. Require an Alternative ML view and a learning tool designed to be used without Cross-Validation.
THANK YOU!