Machine Learning in Formal Verification

Manish Pandey, PhD
Chief Architect, New Technologies
Synopsys, Inc.

June 18, 2017
Build Better Formal Verification Tools?

Software that learns from ‘experience’ and enables users to become more productive?
A Machine Learning System

Source: https://m.xkcd.com/1838/
What is Machine Learning?

“Learning is any process by which a system improves performance from experience”
Herbert Simon

“The complexity in traditional computer programming is in the code (programs that people write). In machine learning, algorithms (programs) are in principle simple and the complexity (structure) is in the data. Is there a way that we can automatically learn that structure? That is what is at the heart of machine learning.”
Andrew Ng
What is Machine Learning?

• Algorithms that can improve performance using training data

• Applicable to situations where challenging to define rules manually

• Typically, a large number of parameter values learned from data
How many variables are we talking about?

- Tens to millions of variables
- Learn a complex multi-dimensional function that captures a solution to the problem
Basics
Machine Learning Example

- Each character is represented by a 20x25 pixels. $x \in \mathbb{R}^{500}$
- Character recognition machine learning task:
  Find a classifier $y(x)$ such that
  $$y : x \rightarrow \{a, b, c, ..., z\}$$
Example Details

• Each character is represented by a 20x25 pixels. $x \in \mathbb{R}^{500}$
• Character recognition machine learning task:
  Find a classifier $y(x)$ such that

$$y : x \rightarrow \{a, b, c, ..., z\} \quad y( ) = \text{[image]}$$
Example Details Cont’d

• Each character is represented by a 20x25 pixels. \( x \in \mathbb{R}^{500} \)

• Character recognition machine learning task:

Find a classifier \( y(x) \) such that \( y : x \to \{a, b, c, \ldots, z\} \quad y(\ ) =  \)

\[
Wx + b = y
\]

500-dimension Input

13026 variable function to model the mapping of pixels to characters
Training: Solving for W and b

Given input \( x \), and associated label \( L \)

- Compute \( y = Wx + b \)
- Compute \( S(y) \)
  \[
  S(z)_j = \frac{e^{a_j}}{\sum_{k=1}^{K} e^{a_k}} \quad \text{for } j = 1, \ldots, K.
  \]
- Cross entropy is
  \[
  D(S, L) = -\sum_i L_i \log(S_i)
  \]
- Loss function
  \[
  \text{Loss} = \frac{1}{N} \sum_i D(S(Wx_i + b), L_i)
  \]
- Compute derivative of \( W \) and \( b \) w.r.t. Loss = \( \nabla_w \)
- Adjust \( W \) and \( b \)
  \[
  W = W - \nabla_w \cdot \text{step\_size}
  \]

\[
\text{x} = \begin{bmatrix}
0, 0, 0, \ldots, 0, 1, 0, 0, 0, 0
\end{bmatrix}
\]

\[
\text{L} = \begin{bmatrix}
0, 0, 0, \ldots, 0, 1, 0, 0, 0, 0
\end{bmatrix}
\]
Gradient Decent

$$L(w_1, w_2)$$

$$\text{step size} \times dL(w_1, w_2)$$

All operating in 13026 variable space
ML Process Flow

Training

Data Repository -> Data Normalization, Random Sampling -> 90% Training Dataset -> Machine Learning -> ML Model

10% Test Dataset -> Model Validation

90% Validation Outcome

Prediction

ML Model -> New Dataset

Prediction -> Prediction Outcome
Multi-layer Networks

\[ y = Wx + b \]
\[ y = W_2(W_1x + b_1) + b_2 \]
\[ y = W_2(\max(W_1x + b_1, 0) + b_2) \]

Machine Learning Model

527000 variables!
Convolution Neural Networks

20 x 25 image
5x5x1 filter $w$

1 number:
the result of taking a dot product between the
filter and a small 5x5x1 chunk of the image
(i.e. $5 \times 5 \times 1 = 25$-dimensional dot product + bias)

$$w^T x + b$$

activation maps
Multi-Layer Convolutional Neural Networks

LeNet (1998)  5 Layers
AlexNet (2012)  8 Layers
VGGNet (2014)  19 Layers
GoogLeNet (2014)  22 Layers
Recurrent Neural Networks

- Vanilla Neural Network
- Image Captioning
- Sentiment Classification
- Machine Translation
- Frame-level Video Classification

Wx+b
Infrastructure
Data Pipelines

- Data Repository
- Data Normalization
- Training Dataset
- Test Dataset
- Machine Learning
  - Model
  - Model Validation
  - % Error
- Validation Outcome

FV Tool

New Dataset

ML Model

Prediction Outcome

Prediction

Coverage DB

Testbench/Trace DB

Coverage DB

Testbench/Trace DB

Training Dataset

Machine Learning

90%

10%

Prediction Outcome

Validation Outcome
On-line vs Off-line

• Tool choices
  – Learning – On-line or Off-line
  – Prediction – On-line

• Choices to be made at every phase of the tool operation
  – Compilation/Model Creation
  – Sequential Analysis/Solver
  – Debug
Machine Learning at Scale

• Off-line and on-line machine learning
  – Data volume
  – Learning speed
  – Prediction speed

• Managing data at scale is hard
  – Distributed data storage
  – Distributed computation
  – Deployment and Operational considerations
Apache Spark

- Distributed in-memory computation platform
- Underlying distributed storage
- Key idea – compute pipelines with
  - Parallel computation model
  - In-memory parallelization support
  - Checkpointing
- MLlib -- Parallel Machine Learning Library implements most common ML algorithms
Apache Spark for In-memory computation at scale

RDDs track *lineage* info to rebuild lost data

- `file.map(record => (record.type, 1)).reduceByKey((x, y) => x + y).filter((type, count) => count > 10)

[Zaharia et.al. 2013]
Fault Tolerance

RDDs track *lineage* info to rebuild lost data

- `file.map(record => (record.type, 1))
  .reduceByKey((x, y) => x + y)
  .filter((type, count) => count > 10)`
Mllib Example: Logistic Regression

Goal: find best line separating two sets of points

[Zaharia et al. 2013]
Mllib Example: Logistic Regression

data = spark.textFile(...).map(readPoint).cache()

w = Vector.random(D)

for (i <- 1 to iterations) {
  gradient = data.map(p =>
    (1 / (1 + exp(-p.y * w.dot(p.x)))) * p.y * p.x
  ).reduce((x, y) => x + y)
  w -= gradient
}

println("Final w: " + w)
Logistic Regression Results

- Running Time (s)
  - Number of Iterations: 1, 5, 10, 20, 30
  - Hadoop: first iteration 80 s, further iterations 1 s
  - Spark: 110 s / iteration

[Zaharia et al. 2013]
Problems
Formal Application Areas

• Broad Categories
  – Enhance current formal verification tools
  – Enable development of new tools

• Application Areas
  – Specification/Constraint Mining
  – Automate Troubleshooting – Expert helper
  – Debugging and root cause identification
  – Aid theorem proving - Develop Solving Strategy
  – Learning Theory -- Learn a concept from a concept class using positive and negative instances of the concept
FV Debugging Session

Describe your problem: I have having a failure in assertion I0/A1. The values of the Ack signal seems to be invalid in the displayed waveform.

**Machine:** hi
**Human:** hello
**Machine:** could you please let me know what is your run-time session?
**Human:** 200451
**Machine:** ok, I can access your FV session. What is the signal name?
**Human:** I0/Ack
**Machine:** Please specify the time at which you see the Ack signal is invalid.
**Machine:** What is value you expect?
**Human:** At time $t = 105$, I expect that the value to be 1.
**Machine:** Can you specify why this is expected?
**Human:** My constraints imply that a request should be successfully accepted. I0/REQ is high.
**Machine:** Upon analysis of your design and constraints, Ack can be 1, if reset is 0 in your constraints and I0/REQ stays high from time 25 to 55? Would you like me to modify your constraints and re-run the design?
**Human:** Yes, thanks!
Debugging and Root Cause Identification
Multiplier Recognition with CNNs (Dai 2016)

- Perform circuit-convolution and pooling to generate Boolean matrices for circuits
  - Fixed size representation of design regardless of the original circuit size.
  - Includes both functional and structural information
  - Circuits with distinct properties have different representations
- Use CNN to perform predictions.
Specification Mining (Fellner 2015)

- Manually writing specifications is complicated and error prone
- Learn specifications from runtime traces
  - Specification as probabilistic finite automata
  - Learn with similarity version of k-tails Algorithm
Machine Learning aided Theorem Proving (Bridge 2014)

• ML applied to the automation of heuristic selection in a first order logic theorem prover.
  – Heuristic selection based on features of the conjecture to be proved and the associated axioms is shown to do better than any single heuristic.

• Heuristic selection amenable to machine learning.
  – The connection between input feature values and the associated preferred heuristic is too complex to be derived manually
  – For any given sample problem the preferred heuristic may be found by running all heuristics. Obtaining labelled training data is simple.
  – thus straightforward given a good selection of trial problems. The approach taken is to

• Demonstrates ML techniques should be able to find a more sophisticated functional relationship between the conjecture to be proved and the best method to use for the proof search.
  – Theorem proving more accessible to non-specialists
Computation Learning Theory (Madhusudan 2007)

- Generic theme: Learn a concept from a concept class using positive and negative instances of the concept.
  - Can we learn a Boolean function given sample evaluations?
  - Learning in presence of noise
- Probably Approximately Correct Learning (Valiant’84)
  - For any concept $\delta, \epsilon$ we can, with probability $1-\delta$, efficiently learn using samples an $\epsilon$-approximation of the concept.
  - Conjunctions of Boolean literals is PAC-learnable.
- Learn to mine - Examples: simple loop invariants; simple predicates that control flow; simple agreements between components; simple concurrency conventions.
- Active learning [Angluin’86, Rivest’93]
  - Learner allowed to ask questions:
    - Membership questions: Is $w \in T$?
    - Equivalence question: Is $T = L(C)$?
Inductive inference for environment modeling (Seshia 2011)

• Program-specific timing model of system inferred from observations of the program’s execution automatically generated.
• Measure execution times of P along so-called basis paths, choosing amongst these uniformly at random over a number of trials.
• Timing model is inferred from the end-to-end
SAT Solver Parameter Tuning and Solver Selection for Formal Verification

• SAT is NP complete
  – Little hope we will find efficient solver that fits all problems
• Different solvers have strengths and weaknesses
  – MiniSat, MarchSAT, ...
• Each solver has a number of parameters that can perform well on certain types of problems
Parameter Tuning and SAT Solver Selection

Features

1. Property circuit level
2. SCOAP cycles
3. Number of flops uninitialized after RESET
4. Circuit Testability Index
5. Property Testability Index
6. SCOAP adjusted flops
7. SCMax
8. Number of flops
9. Number of gate bits
10. Number of free variables
11. Number of bits directly affected by constraints
12. Number of counters flops
13. Number of FSM flops
14. Number of memory array flops

We know the problem is NP complete, but different engines may affect differently by the features, some polynomially and some exponentially.

We attempt to optimize how many instances we can run to reduce the risk of a property not being proven.

Penido et al, STTT 2010
A Historical Aside - Knowledge Representation and Learning

- Origins of Boolean satisfiability techniques lie in early artificial intelligence approaches to represent knowledge and reason from it.

Determine ways to solve the problem of whether or not there is an assignment of truth values to the variables in a set of clauses -- “SAT”.
Thank You