Automatic Synthesis of statistical and machine learning Algorithms

Johann Schumann
SGT, Inc., NASA ARC
Example: Analysis of terminal traffic pattern

- Given trajectory data for landing and takeoff from the airport
- Clustering of trajectories to find structure in the data

Let’s use our favorite “multivariate clustering” software/tool
- Matlab
- R
- Weka
- AutoBayes
- ...
Example: Analysis of terminal traffic pattern

Result:
Num_classes = 5

Uups... expecting only 4 classes

Need a special PDF to handle angles
Declarative problem specification for the task

Model cluster1 as 'simple cluster analysis'.
const nat N as 'data points'. where 0 < N.
const nat C. where 0 < C. where C <= N.
double µ(0..C-1), σ(0..C-1).

double phi(0..C-1) as 'class frequency'.
   where 0 = sum(_i := 0 .. C-1, phi(_i))-1.
output double c(0..N-1).
c(_i) ~ discrete(vector(_i := 0 .. C-1, phi(_i))).
data double x(0..N-1).
x(_i) ~ gauss(µ(c(_i)), σ(c(_i))).

• declare constants and parameters
• the class frequency is a probability, i.e. adds up to 1
• c is the class assignment
• the data are Gaussian distributed
• What do we want?
  • estimates of the parameters
  • return the class assignment
model ac_track as 'AC track analysis'.
const nat N as '# tracks'. where 0 < N.
const nat C. where 0 < C. where C << N.
double \( \lambda_x(0..C-1), m_x(0..C-1) \).
double phi(0..C-1) as 'class frequency'.
    where 0 = sum(_i := 0 .. C-1, phi(_i))-1.
output double c(0..N-1).
c(_) ~ discrete(vector(_i := 0 .. C-1, phi(_i))).

data double x(0..N-1).
x(_i) ~ vonmises(\( \lambda_x(c(_i)), m_x(c(_i)) \)).

max pr({x | {phi, \_hd, m_hd } }
    for {phi, \_hd, m_hd }).

This is the actual input of the AutoBayes tool
Program Synthesis

- AutoBayes generated an entirely different algorithm
- 880 lines of C++ versus 582 lines
- Highly documented code
- Interfaces to Matlab or octave
The AutoBayes Synthesis tool

• Tool for the automated generation of customized data analysis and machine learning algorithms
• Bayesian approach for problem break-down and solution
• Full declarative input specifications
• Target languages: C, C++
• Schema-based program synthesis
Schema-based synthesis: Example

• We need an algorithm to maximize

\[ \text{max } -5x^2 + 3x + 7 \text{ for } x. \]

the we can do the text-book style technique:

1. calculate the derivative: \( \frac{df}{dx} \)

2. set it to 0:

   \[ 0 = \frac{df}{dx} \]

3. solve that equation for \( x \)

4. the solution is the desired maximum
Schema for univariate optimization

\[\text{schema}(\text{max } F \text{ for } X, C) :-
\]

\%
\text{INPUT (Problem), OUTPUT (Code fragment)}

\text{length}(X, 1),

\%
\text{calculate the first derivative}
\text{simplify}(\text{deriv}(F, X), DF),

\%
\text{solve the equation}
\text{solve}(\text{true}, 0 = DF, X, S),

C = S

1. build the derivative: \( \frac{df}{dx} \)
2. set it to 0: \( 0 = \frac{df}{dx} \)
3. solve that equation for \( x \) and generate code for that
4. the solution is the desired maximum
schema(max F for X, C) :-
    % INPUT (Problem), OUTPUT (Code fragment)
    % guards
    length(X, 1),

    % calculate the first derivative
    simplify(deriv(F, X), DF),
    % solve the equation
    solve(true, x, 0 = DF, S),
        % is that really a maximum?
    simplify(deriv(DF, X), DDF),
    solve(true, x, 0 < DDF, _),
    C = S
Schema for univariate optimization

\[
\text{schema}(\text{max } F \text{ for } X, C) :-
\]

\[
\begin{align*}
% \text{ INPUT (Problem), OUTPUT (Code fragment)} \\
% \text{ guards} \\
\text{length}(X, 1), \\
\text{ % calculate the first derivative} \\
\text{simplify(deriv(F, X), DF),} \\
\text{ % solve the equation} \\
\text{solve(true, x, 0 = DF, S),} \\
\text{ % possibly more checks} \\
\text{ % is that really a maximum?} \\
\text{simplify(deriv(DF, X), DDF),} \\
\text{(solve(true, x, 0 > DDF, _)} \\
\rightarrow \text{true ;} \\
\text{writeln(‘Proof obligation not solved’)}, \\
C = S
\end{align*}
\]
Schema for univariate optimization

schema(max F for X, C) :-
   % INPUT (Problem), OUTPUT (Code fragment)
   % guards
   length(X, 1),

   % calculate the first derivative
   simplify(deriv(F, X), DF),
   % solve the equation
   solve(true, x, 0 = DF, S),
   % possibly more checks
   % is that really a maximum?
   simplify(deriv(DF, X), DDF),
   (solve(true, x, 0 > DDF, _)
    -> true ;
   writeln('Proof obligation not solved automatically')), XP = ['The maximum for', expr(F), 'is calculated ...'],
   V = pv_fresh,
   C = let(assign(V, C, [comment(XP)]), V).
education

AutoBayes Applications

Astrophysics

Software Testing

Air Traffic Control

Earth Science/Geodata

• NASA Open Source: https://software.nasa.gov/software/ARC-16276-1
• Manual: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20080042409.pdf
Algorithm Design via Synthesis

• An appropriate *machine-learning algorithm* is the basis for each application

• Some domains require specialized algorithms or algorithm variants for the specific task at hand. Examples:
  • Clustering, Optimization, Filtering

• Some program synthesis systems (like AutoBayes) can generate a *customized algorithm* from scratch or schemas

• Algorithm variants can be generated easily for the same task, e.g.
  • K-means, EM, kd-trees, ... for clustering
  • Gradient descend, Nelder-Mead, BGFS, Levenberg-Margquart, ... for optimization
  • Matrix-inverse, Schmidt, Bierman update, etc. for Kalman filter
Algorithm Instantiation via Synthesis

• Many ML tasks require substantial mathematical pre-c calculations
  • E.g., discretization and linearization of the state-transition matrix in a Kalman filter.

• Such calculations need to be done before the results can be “plugged” into the ML algorithm

• Algorithms often can be improved such results (e.g., matrix is diagonal)

• Examples of such systems:
  • AutoFilter: synthesis of KF and EKF algorithms
  • *kf: synthesis of Kalman filter software [1]

Many machine-learning tasks require a number of individual activities:
- Reading and parsing of data
- Data transformation and preprocessing
- Training and testing
- Assessment of results
- Visualization
- ...

Work-flow tool for ML
- Reduce amount of custom code
- Increase flexibility

Example:
- Weka Knowledge-flow
  (not synthesis per se)
Architecture by Synthesis

• Modern hardware architectures enables the fast and efficient execution of machine-learning algorithms
  • GPU, MPI cluster, Epiphany, FPGA, Tensor flow, neuromorphic, ...

• Very high effort to implement a ML algorithms for a specific hardware architecture

• Very easy to get things wrong

• Program synthesis tools can generate efficient and customized algorithms for a number of different target architectures
  • From a single high-level algorithm description

• Example:
  • SpiralGen (spiralgen.com)
Algorithm Optimization by Synthesis

• System requirements can constrain time, power, and hardware alternatives

• What is the “best” algorithm and hardware architecture for the given task under the given requirements?

• Synthesis techniques can help to solve that problem:
  • Fast and effort-less generation of multiple algorithms and architectures for the same ML task
  • Automated generation of test data, evaluation and test environments
What else can synthesis do for ML?

• Automatic generation of algorithm design documentation and detailed mathematical derivations

• Automatic generation of multiple artifacts for debugging, testing, verification and validation

Algorithm derivation generated by AutoBayes (excerpt)
The Joy of Generating C Code from MATLAB

By Bill Chou, MathWorks

Engineers have translated low-level languages like C into machine code for decades using compilers. But is it possible to translate a high-level language like MATLAB® to C using coders? Most engineers would agree that it's possible in theory—but does it work in practice? Is the generated code readable or spaghetti? Efficient or bloated? Fast or slow? And does it support industrial workflows, or just R&D?
Synthesis for ML: All problems solved?

Synthesis systems for ML applications have huge advantages, but
• Very hard and costly to develop and maintain
• Steep learning curve for the user, tools often too researchy
• Difficult integration, understanding, and modifying of generated code

Future
• Strongly increased need for ML algorithms on specialized hard/software platforms (UAS, (Cube)Sat, Rovers,...),
• Availability of special hardware for ML (Tensor flow, neuromorphic, highly parallel, low power),
• Rad-hard by algorithm,
• Advanced, fast changing ML tasks during the mission
makes synthesis for Machine-Learning algorithms and software an exciting prospect