Knowledge-driven AutoML

...A future current-trend in AI?

by Corneliu Cătălin

10th of May 2024

VUB Artificial Intelligence Research Group
Outline
Outline

> AutoML - short introduction

> Knowledge-driven AutoML
  - The "Omini" project
  - Architecture & Motivation
  - Pipeline generation flow

> Open research challenges
  - Mining & sharing experiments
  - Wrapping ML-libs - inventing dependencies
  - Optimization & evaluation
Auto M.L.
Auto ML

- The process of generating and tuning ML pipelines automatically
Auto ML

- The process of generating & tuning ML pipelines automatically

- Can be split into two parts:
Auto ML

- The process of generating & tuning ML pipelines automatically

- Can be split into two parts:
  1. Pipe synthesis

![Diagram showing a pipeline structure]
Auto ML

- The process of generating and tuning ML pipelines automatically

- Can be split into two parts:
  1. pipe synthesis
  2. hyperparameter optimization

[Diagram of a pipeline with labeled sections]
Auto ML

- Most approaches consider the pipe structure fixed (auto-weka, auto-sklearn)
Auto ML

- Most approaches consider the pipe structure fixed (auto-Weka, auto-sklearn)

- New research (~2016) considers pipe structure flexible: trees, rules, planning
  - Evolutionary logic
  - Ordering of blocks
Auto ML

- Most approaches consider the pipe structure fixed (auto-Weka, auto-sklearn)

- New research (~2016) considers pipe structure flexible: trees, rules, planning
  - evolutionary logic
  - ordering of blocks

- Sophisticated and flexible use of knowledge structures still not explored!
AutoML

- **Note**: Neural networks case i.e. neural arch. search - not covered here :)

Knowledge-driven AutoKL
Knowledge-driven AutoML

- Data-driven technique for AutoML
- Available at:
  - Experiments: https://github.com/zgornei/Knowledge-driven-AutoML
  - Library: https://github.com/zgornei/KdautoML
The Omina project
The Omina project

- Omina Technologies: AI company based in Antwerp developing ML solutions
The Omina project

- Omina Technologies: AI company based in Antwerp developing ML solutions

- Moving from manual to automatic pipeline building
The Omina project

- Omina Technologies: AI company based in Antwerp developing ML solutions

- Moving from manual to automated pipeline building
  - Why?
    - Improve development quality
    - Adapt to limited data access
    - Scale: performance, use-cases etc.
The Omina project

- The goal of the project: use knowledge to optimize & speed up ML pipe building
The Omina project

- The goal of the project: use knowledge to 
  advance and speed up ML pipe building

- Knowledge

  - Data science: code, pipes etc.
  - Processes: how entities intermediate and generate data
  - World: open data, knowledge bases, wikis...
Architecture & Motivation
Architecture & Motivation

- Two approaches become apparent:
Architecture & Motivation

- Two approaches become apparent:
  - Generate complete pipelines first
    (a bundle of them, test, optimize, etc.)
Architecture & Motivation

- Two approaches become apparent:
  - generate complete pipelines first (a bunch of them, test, optimize, etc.)

Knowledge used for:
- generating pipes
- post-processing
  - combine them, i.e., optimization
Architecture & Motivation

- Two approaches become apparent:
  - generate and execute dynamically pipes
Architecture & Motivation

- Two approaches become apparent:
  - generate and execute dynamically pipelines

Knowledge used to
- decide what ops.
  - feasible based on:
    - current operation
    - potential pipe(s)
    - execution results

pipeline 

pale science knowledge
Architecture & Motivation

- The latter "dynamic" approach chosen:
  - better control over program space
  - similarity in knowledge application
  - more explanatory power
  - allows for interactivity i.e. human intervention
    - delays, partial execution etc.
    - [potentially] more resource friendly
  - the first one was already explored (by MIT)
Architecture & Motivation
Architecture & Motivation

Program space i.e. all possible pipes

how to construct it?
Architecture & Motivation

OP1 i.e. load data

OP2 i.e. process data

OP3 i.e learn model

↓ flow of operations

↑ automation
Architecture & Motivation

decisions on what blocks/ops are feasible

Knowledge

Graph of

OP1 i.e. load data

OP2 i.e. process data

OP3 i.e. learn model

Flow of operations

Automation
Architecture & Motivation

- Decisions on what blocks/ops are feasible
- Knowledge
  - Graph DB

Code:
- \( a = \text{foo}(\ldots) \)
- \( b = \text{bar}(a, \ldots) \)
- \( c = \text{bar}(b, \ldots) \)
- Return \( c \)

Flow of operations: automation

Program structure (object)
- Symbolic → code + exec.
Architecture & Motivation

Automaton
- controls build flow
- communicates w. KB
- sends code / gets data from prep-structure
Architecture & Motivation

Automaton
- controls build flow
- communicates with KB
- sends code/gets data from prep structure

Kb
- stores code, actions hierarchically
  * biases hierarchically
  * biases structure

bias = condition over an action/block/step
Architecture & Motivation

Program Structure:
- contains prep. space
- can build code
- can execute code
- holds data/results
- depends on programming language/library

Automaton:
- controls build flow
- communicates w. KB
- sends code/gets data from prep. structure

KB:
- stores code, actions hierarchically,
  * bias hierarchically,
  * bias structure

bias = condition over an action/block/op
Pipeline generation flow
Let's start from the beginning...
Pipeline generation flow

// no program.
Pipeline generation flow

1. Read first op (load)

// no program.
Pipeline generation flow

1. Read first op (load)
2. Query KB what operations are available
Pipeline generation flow

1. Read first op (load)
2. Query KB what ops and biases are available
3. Get list of (ops, biases)
Pipeline generation flow

1. Read first op (load)
2. Query KB what ops + biases are available
3. Get list of (ops, biases)
4. Execute noise functions, return feasible ops
Pipeline generation flow:

1. Read first op (load)
2. Query KB what ops + biases are available
3. Get list of (ops, biases)
4. Execute bias functions, retain feasible ops
5. Construct code
Pipeline generation flow:

1. Read first op (load)
2. Query KB what ops + biases are available
3. Get list of (ops/biases)
4. Execute bias functions, retain feasible ops
5. Construct code
6. Execute pipeline
1. Read first op (load)
2. Query KB what ops + biases are available
3. Get list of (ops, biases)
4. Execute bias functions, retain feasible ops
5. Construct code
6. Execute pipeline
7. Next op
Open research challenges
Open research challenges

- Mining & sharing experiments
Open research challenges

Mining & sharing experiments

Data science experiment

Some fancy parser

Knowledge base

Abstract representation of the experiment using KB-concepts
Open research challenges

> Mining & sharing experiments

Data science experiment

Some fancy parser

Knowledge base

Abstract/Symbolic representation of the experiment using KB-concepts
Open research challenges

> Mining & sharing experiments

Data science experiment

Knowledge base

Some fancy parser

Discovered using

Knowledge-driven A* search

Exploration of the experimental space
Open research challenges
Open research challenges

wrapping ML libs
Open research challenges

> Currently, each AutoML library depends on a single ML library

AutoML1, AutoML2, ..., AutoMLn

/depends \depends \depends
ML1, ML2, ML3, ..., MLn

> Changes in ML (frequent) influence AutoML
Open research challenges

> wrapping ML Libs

What if a single AutoML library supports multiple ML libraries

AutoML, ML1, ML2, ..., MLn
Open research challenges

Swapping ML libs

What if a single AutoML library expects AutoML to multiple ML libraries?

---

Dependency inversion?

ML1, ML2, .., MLn

"Bob Martin - Clean Architecture"
Open research challenges

>Autokl-apl

>Autokl

>ML, kl, klz, klz

>What if a mix of multiple ML libraries?
Open research challenges

Wrapping ML libs

What if a single AutoML library supports multiple ML libraries?

AutoML

ML1, ML2, ..., MLn

An auto-task similar to program synthesis

i.e., intelligent code wrapping ML1

AutoML-API
Open research challenges
Open research challenges

Optimization & Evaluation
Open research challenges

Optimization & Evaluation

Currently that's done using some form of metric involving accuracy & time/resources
Open research challenges

Optimization & Evaluation

Currently that's done using some form of metric involving accuracy & time/resources

Interpretability?

Explainability?

Ease of use?

Scalability?

How to best integrate the qualitative?
Thank you.

Q & A Time ...