AI-Augmented Algorithms
How I Learned to Stop Worrying and Love Choice

Lars Kotthoff
University of Wyoming
larsko@uwyo.edu

Warsaw, 17 April 2019
Outline

▷ Big Picture
▷ Motivation
▷ Choosing Algorithms
▷ Tuning Algorithms
▷ Applications
▷ Outlook and Resources
Big Picture

▷ advance the state of the art through meta-algorithmic techniques
▷ rather than inventing new things, use existing things more intelligently – automatically
▷ invent new things through combinations of existing things
Motivation – What Difference Does It Make?
Prominent Application

Performance Differences

Performance Improvements

Common Theme

Performance models of black-box processes

▷ also called surrogate models
▷ substitute expensive underlying process with cheap approximate model
▷ build approximate model using machine learning techniques based on results of evaluations of the underlying process
▷ no knowledge of what the underlying process is required (but can be helpful)
▷ may facilitate better understanding of the underlying process through interrogation of the model
Choosing Algorithms
Algorithm Selection

Given a problem, choose the best algorithm to solve it.

Algorithm Selection

Portfolio
- Algorithm 1
- Algorithm 2
- Algorithm 3

Training Instances
- Instance 1
- Instance 2
- Instance 3

Algorithm Selection

Performance Model

Instance 4: Algorithm 2
Instance 5: Algorithm 3
Instance 6: Algorithm 3

Feature Extraction
Algorithm Portfolios

▷ instead of a single algorithm, use several complementary algorithms
▷ idea from Economics – minimise risk by spreading it out across several securities
▷ same for computational problems – minimise risk of algorithm performing poorly
▷ in practice often constructed from competition winners or other algorithms known to have good performance

Algorithms

“algorithm” used in a very loose sense
▷ algorithms
▷ heuristics
▷ machine learning models
▷ software systems
▷ machines
▷ ...

Parallel Portfolios

Why not simply run all algorithms in parallel?
▶ not enough resources may be available/waste of resources
▶ algorithms may be parallelized themselves
▶ memory/cache contention
Building an Algorithm Selection System

▷ requires algorithms with complementary performance
▷ most approaches rely on machine learning
▷ train with representative data, i.e. performance of all algorithms in portfolio on a number of instances
▷ evaluate performance on separate set of instances
▷ potentially large amount of prep work
Key Components of an Algorithm Selection System

- feature extraction
- performance model
- prediction-based selector/scheduler

optional:
- presolver
- secondary/hierarchical models and predictors (e.g. for feature extraction time)
Types of Performance Models

- **Regression Models**
  - A1: 1.2
  - A2: 4.5
  - A3: 3.9

- **Classification Model**
  - A1
  - A2
  - A3

- **Pairwise Classification Models**
  - A1 vs. A2
  - A1 vs. A3
  - A1: 1 vote
  - A2: 0 votes
  - A3: 2 votes

- **Pairwise Regression Models**
  - A1 - A2
  - A1 - A3
  - A1: -1.3
  - A2: 0.4
  - A3: 1.7

- **Instance 1, Instance 2, Instance 3...**
  - Instance 1: Algorithm 2
  - Instance 2: Algorithm 1
  - Instance 3: Algorithm 3...
Tuning Algorithms
Algorithm Configuration

Given a (set of) problem(s), find the best parameter configuration.
Parameters?

▷ anything you can change that makes sense to change
▷ e.g. search heuristic, optimization level, computational resolution
▷ not random seed, whether to enable debugging, etc.
▷ some will affect performance, others will have no effect at all
Automated Algorithm Configuration

- no background knowledge on parameters or algorithm – black-box process
- as little manual intervention as possible
  - failures are handled appropriately
  - resources are not wasted
  - can run unattended on large-scale compute infrastructure
Algorithm Configuration

Parameter domains & starting values

Calls with different parameter settings

Configuration scenario

Target algorithm

Solves

Problem instances

Returns solution cost

General Approach

▷ evaluate algorithm as black-box function
▷ observe effect of parameters without knowing the inner workings, build surrogate model based on this data
▷ decide where to evaluate next, based on surrogate model
▷ repeat
When are we done?

▷ most approaches incomplete, i.e. do not exhaustively explore parameter space
▷ cannot prove optimality, not guaranteed to find optimal solution (with finite time)
▷ performance highly dependent on configuration space

→ How do we know when to stop?
How much time/how many function evaluations?

▷ too much $\rightarrow$ wasted resources
▷ too little $\rightarrow$ suboptimal result
▷ use statistical tests
▷ evaluate on parts of the instance set
▷ for runtime: adaptive capping
▷ in general: whatever resources you can reasonably invest
Grid and Random Search

▷ evaluate certain points in parameter space

Model-Based Search

▷ evaluate small number of configurations
▷ build model of parameter-performance surface based on the results
▷ use model to predict where to evaluate next
▷ repeat
▷ allows targeted exploration of new configurations
▷ can take instance features into account like algorithm selection

Iter = 1, Gap = 1.9909e−01

Model-Based Search Example
Model-Based Search Example

Iter = 2, Gap = 1.9909e−01

- $y$ and $y_{hat}$
- $e_i$
Model-Based Search Example

Iter = 3, Gap = 1.9909e−01
Model-Based Search Example

Iter = 4, Gap = 1.9992e−01

-1.0 -0.5 0.0 0.5 1.0

0.0
0.4
0.8

0e+00
2e−04
4e−04
6e−04
8e−04

x
typp
● init
● prop
● seq

Iter = 4, Gap = 1.9992e−01
Model-Based Search Example

Iter = 5, Gap = 1.9992e−01

Graph showing the iterative process with points marked as follows:
- **init** (red circle)
- **prop** (blue triangle)
- **seq** (green square)
- **ei** (black dot)

Graph axes:
- X-axis: \(-1.0\) to \(1.0\)
- Y-axis: \(0.0\) to \(0.8\)
- \(e_i\) axis: \(0.0\) to \(2.0\times10^{-4}\)

Legend:
- **y** (solid black line)
- **yhat** (dashed black line)
- **ei** (black dot)

Initial point marked as **init** (red circle) at \((-1.0, 0.0)\)
Model-Based Search Example

Iter = 6, Gap = 1.9996e−01

-1.0 -0.5 0.0 0.5 1.0
0.0
0.4
0.8
0.00000
0.00003
0.00006
0.00009
0.00012

0.00012
0.00009
0.00006
0.00003
0.00000
Model-Based Search Example

Iter = 7, Gap = 2.0000e−01

- $y_i$ vs $x$
- Type: $y$, $yhat$, $init$, $prop$, $seq$
- $ei$ vs $x$
- $x$ range from $-1.0$ to $1.0$
Model-Based Search Example

Iter = 8, Gap = 2.0000e−01

-1.0 -0.5 0.0 0.5 1.0

0.0e+00 5.0e−06 1.0e−05 1.5e−05 2.0e−05
Model-Based Search Example

Iter = 9, Gap = 2.0000e−01
Model-Based Search Example

Iter = 10, Gap = 2.0000e−01
Selected Applications
Compiler Parameter Tuning

▷ pre-defined optimization levels offer not much flexibility
▷ improvements possible by tuning full compiler parameter space
▷ tuned compute-intensive AI algorithms
▷ up to 40% runtime improvement over gcc -02/-03

---

Application – Optimizing Graphene Oxide Reduction

▷ reduce graphene oxide to graphene through laser irradiation
▷ allows to create electrically conductive lines in insulating material
▷ laser parameters need to be tuned carefully to achieve good results
From Graphite/Coal to Carbon Electronics

The process starts with graphite and coal, which are oxidized and exfoliated. The graphite oxide is then obtained by pyrolysis and oxidation. The graphite oxide is dispersed in a solvent, and the supernatant is obtained through sonication and centrifugation. The graphene oxide film is then deposited using ultrasonic spray deposition. Finally, the 2-D laser patterning is used to create carbon electronics.
Evaluation of Irradiated Material

$\frac{I_G}{I_D} = 1.2$

$\frac{I_G}{I_D} = 6.8$
Morphology of Irradiated Material
Surrogate-Model-Based Optimization
Surrogate-Model-Based Optimization

During Training

• Predictions work even with small training dataset (19 points)
• AI Model achieved $I_G/I_D$ ratio (>6) after 1st prediction

After 1st prediction
Explored Parameter Space
Outlook
Quo Vadis, Software Engineering?


https://larskotthoff.github.io/assurvey/
Tools and Resources

LLAMA  https://bitbucket.org/lkotthoff/llama
SATzilla  http://www.cs.ubc.ca/labs/beta/Projects/SATzilla/
  iRace  http://iridia.ulb.ac.be/irace/
mlrMBO  https://github.com/mlr-org/mlrMBO
SMAC  http://www.cs.ubc.ca/labs/beta/Projects/SMAC/
Spearmint  https://github.com/HIPS/Spearmint
TPE  https://jaberg.github.io/hyperopt/
autofolio  https://bitbucket.org/mlindauer/autofolio/
Auto-WEKA  http://www.cs.ubc.ca/labs/beta/Projects/autoweka/
Auto-sklearn  https://github.com/automl/auto-sklearn
Algorithm Selection  choose the best *algorithm* for solving a problem

Algorithm Configuration  choose the best *parameter configuration* for solving a problem with an algorithm

▷ mature research areas
▷ can combine configuration and selection
▷ effective tools are available
▷ COnfiguration and SElection of ALgorithms group COSEAL
  http://www.coseal.net

Don’t set parameters prematurely, embrace choice!
I’m hiring!

Several funded graduate positions available.