AutoML: Algorithm Selection

Overview and Motivation

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Algorithm Selection

Given a problem, choose the best algorithm to solve it. [Rice. 1975]

Algorithm Selection

More formally

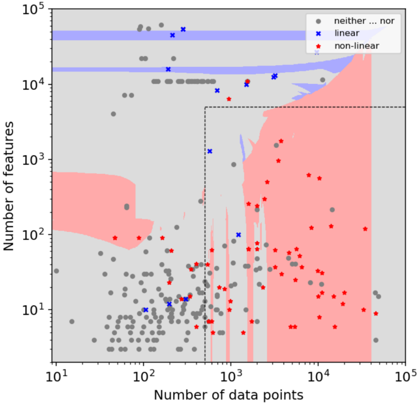
Let

- $p(\mathcal{D})$ be a probability distribution over datasets $\mathcal{D} \in \mathbf{D}$,
- $\mathbf{P}$ a portfolio of algorithms $\mathcal{A} \in \mathbf{P}$, and
- $c : \mathbf{P} \times \mathbf{D} \rightarrow \mathbb{R}$ be a cost metric

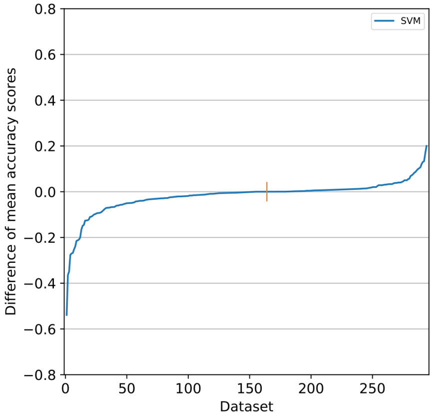
the *per-instance algorithm selection problem* is to obtain a mapping $s : \mathcal{D} \mapsto \mathcal{A}$ such that

$$\arg\min_{s} \int_{\mathbf{D}} c(s(\mathcal{D}), \mathcal{D}) p(\mathcal{D}) \, \mathrm{d}\mathcal{D}$$

Motivation: Performance Differences [Strang et al. 2018] I



Motivation: Performance Differences [Strang et al. 2018] II

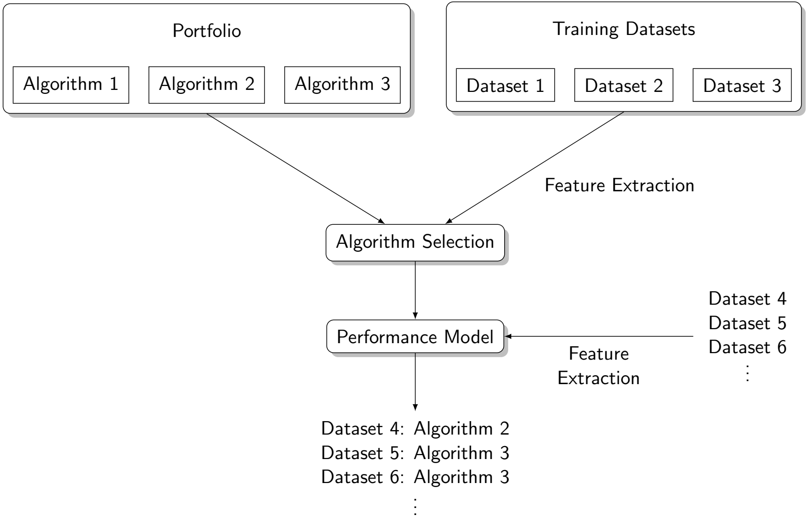


AutoML: Algorithm Selection

Algorithm Selection

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Algorithm Selection



Algorithm Portfolios

- instead of a single algorithm, use several (hopefully complementary) algorithms
- idea from Economics – minimize risk by spreading it out across several securities
- same here – minimize risk of algorithm performing poorly
- in practice often constructed from algorithms known to perform well
- idea similar to ensembles or boosting – leverage strengths and alleviate weaknesses, but learn which algorithm to choose for a particular dataset

Algorithms

"algorithm" used in a very loose sense

- different learners
- different parameterizations of the same learner
- different ensembles, boosted learners
- different machine learning workflows/pipelines
- ...

Evaluation of Portfolios

- single best algorithm
  - algorithm with the best performance across all datasets
  - lower bound for performance of portfolio – hopefully we are better!
- virtual best algorithm
  - choose the best algorithm for each dataset
  - corresponds to oracle predictor or overhead-free parallel portfolio
  - upper bound on portfolio performance

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 contention
- ...

Building an Algorithm Selection System

- most approaches rely on (meta-)machine learning
- train with representative data, i.e. performance of all algorithms in portfolio on representative datasets
- evaluate performance on separate set of datasets
- potentially large amount of prep work
- existing repositories of machine learning performances (e.g. OpenML) can help

Choosing Datasets

- we want selectors that generalize, i.e. good for more than one dataset
- split datasets into training set (which we learn a selector on) and test set (which we only evaluate performance on)
- need to balance easy/hard datasets in both sets
- may need a lot of data

Key Components of an Algorithm Selection System

- feature extraction
- performance model
- prediction-based selector

optional:

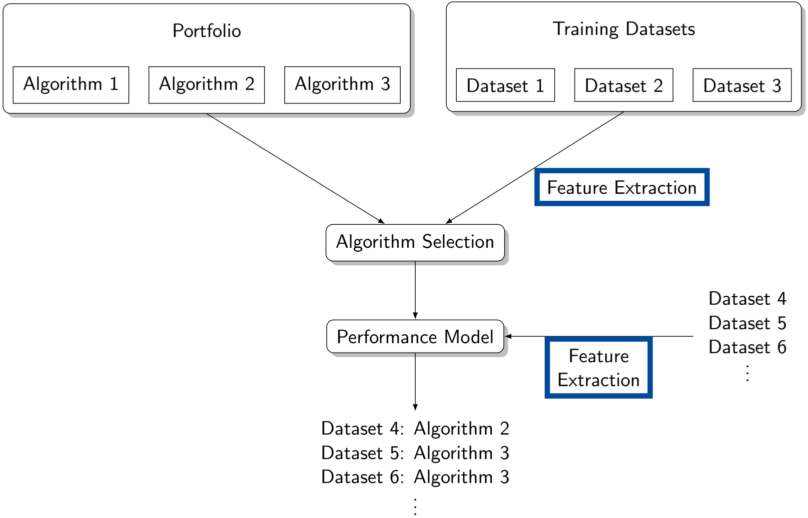
- presolver
- secondary/hierarchical models and predictors (e.g. for feature extraction time to avoid spending a long time for small performance gains)

AutoML: Algorithm Selection

Features

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Algorithm Selection



Features

- relate properties of datasets to algorithm performance
- relatively cheap to compute – must be cheaper than running the algorithm to see what its performance is
- often specified by domain expert
- syntactic and information-theoretic – analyze dataset
- probing – run an algorithm for short time or on subset of data

Syntactic and Information-Theoretic Features

- number of binary/numeric/categorical features
- number of classes
- class entropy
- skewness of classes
- fraction of missing values
- correlation between features and target
- ...

Probing Features (Landmarkers)

- performance of majority class/mean value predictor
- decision stump performance
- simple rule model performance
- performance of algorithm of interest on 1% of data
- ...

$\rightarrow$ usually leads to much better results that using just syntactic and information-theoretic features

No Features

- use deep learning to process dataset or problem instance as-is
- no need for expert-designed features
- only preliminary applications so far, performance not good, no widespread adoption yet

Aside: Algorithm Features

- can characterize algorithm in addition to datasets
- allows to relate performance to specific aspects of an algorithm rather than black boxes
- for example size of code base, properties of abstract syntax tree. . .
- ongoing work

What Features Do We Need in Practice?

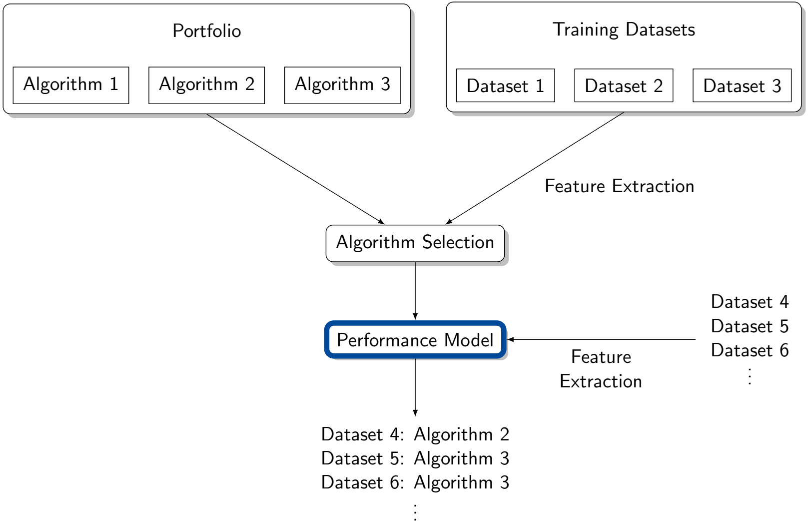
- trade-off between complex features and complex models
- in practice, very simple features can perform well
- often only few features of a set are needed (e.g. 5 out of $>$100)
- in the end, whatever works best

AutoML: Algorithm Selection

Performance Models

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Algorithm Selection



Types of Performance Models

- models for entire portfolios
- models for individual algorithms
- models that are somewhere in between (e.g. pairs of algorithms)

$\rightarrow$ for each of these, many different machine learning approaches are suitable

Models for Entire Portfolios

- predict the best algorithm in the portfolio (e.g. classifier to use)
- alternatively: cluster in meta-feature space and assign best algorithm to each cluster

optional (but important):

- attach a “weight” during learning (e.g. the difference between best and worst algorithm) to bias model towards the “important” datasets
- special loss metric

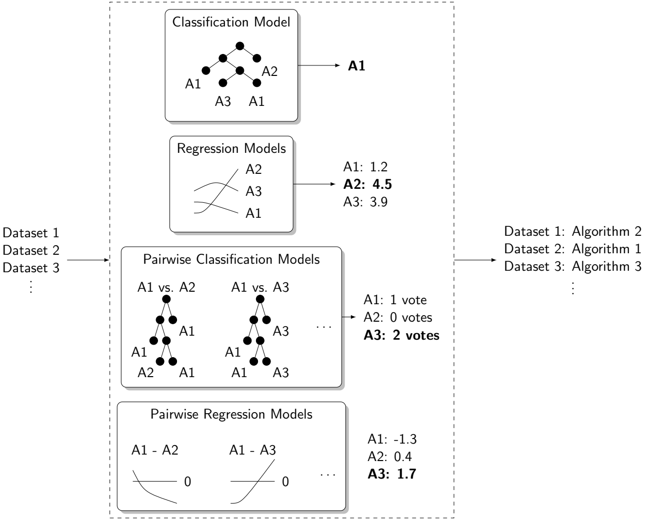
Models for Individual Algorithms

- predict the performance for each algorithm separately
- combine the predictions to choose the best one
- for example: predict accuracy, choose algorithm with highest predicted accuracy

Hybrid Models

- for example: consider pairs of algorithms to take relations between them into account
- for each pair of algorithms, learn model that predicts which one has better performance, or predicts performance difference
- . . . or collaborative filtering approaches

Types of Performance Models



Types of Predictions/Algorithm Selectors

- best algorithm (and its performance)
- n best algorithms ranked
- ensemble of n best algorithms

Time/Frequency of Prediction

- one-shot
  - select algorithm(s) once
  - want to process single dataset and choose the best approach
- multi-shot
  - continuously monitor dataset(s) features and/or performance
  - for example on data streams or to process sets of datasets

AutoML: Algorithm Selection

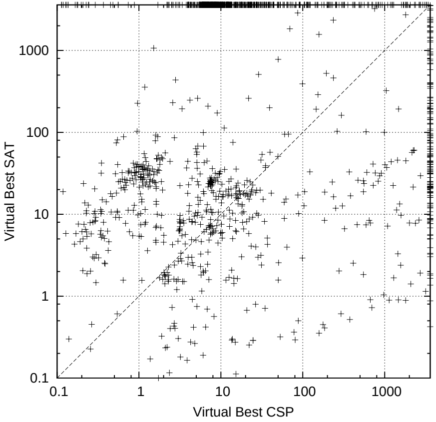
Bonus: Combinatorial Problems

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

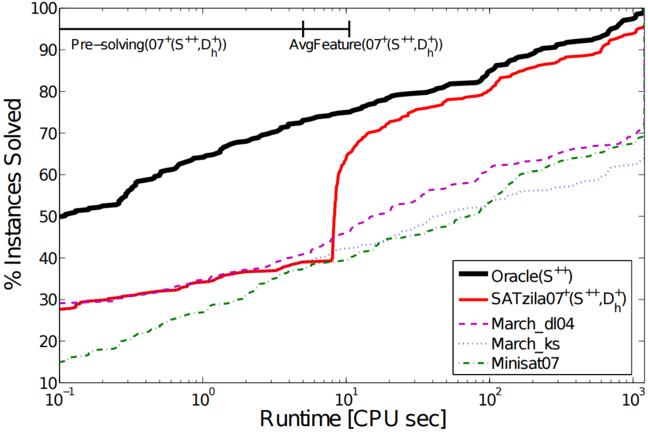
Motivation

- Algorithm Selection applied in many other domains
- success and performance improvements for combinatorial and optimization problems in AI dwarfs those in machine learning
- important application area of AI facilitating cross-disciplinary collaborations and advances

Motivation: Performance Differences [Barry et al. 2014] I



Motivation: Leveraging the Differences [Xu et al. 2008]



Algorithms [Huberman et al. 1997]

- constraint solvers
- search strategies
- modeling choices
- different types of consistency

Features

- number of variables, number of clauses/constraints/...
- ratios
- order of variables/values
- connectivity clause/constraints–variable graph or variable graph
- number of nodes/propagations within time limit
- estimate of search space size
- tightness of problem/constraints
- ...

Example System – SATzilla [Xu et al. 2008]

- portfolio of 7 SAT solvers, trained on 4811 problem instances
- syntactic (33) and probing features (15)
- ridge regression to predict log runtime for each solver, choose the solver with the best predicted performance
- later version uses random forests to predict better algorithm for each pair, aggregation through simple voting scheme
- pre-solving, feature computation time prediction, hierarchical model, selection of algorithms to include in portfolio based on overall performance
- won several competitions

Benchmark library – ASlib [Bischl et al. 2015]

- `https://github.com/coseal/aslib_data`
- SAT, CSP, QBF, ASP, MAXSAT, OR, ML...
- includes data used frequently in the literature that you may want to evaluate your approach on
- more scenarios in the pipeline
- `http://aslib.net`

Tools

autofolio	`https://bitbucket.org/mlindauer/autofolio/`
LLAMA	`https://bitbucket.org/lkotthoff/llama`
SATzilla	`http://www.cs.ubc.ca/labs/beta/Projects/SATzilla/`

(Much) More Information [Kotthoff. 2014]

Comments? Suggestions? Corrections?
Let me know!

Algorithm Selection Literature Summary

Last update 21 November 2018

click headings to sort
click citations to expand

Fork me on GitHub

citation	domain	features	predict what	predict how	predict when	portfolio	year
Langley 1983b, Langley 1983a	search	past performance	algorithm	hand-crafted and learned rules	offline and online	dynamic	1983
Carbonell et al. 1991	planning	problem domain features, search statistics	control rules	explanation-based rule construction	online	dynamic	1991
Gratch and DeJong 1992	planning	problem domain features, search statistics	control rules	probabilistic rule construction	online	dynamic	1992
Smith and Setliff 1992	software design	features of abstract representation	algorithms and data structures	simulated annealing	offline	static	1992
Aha 1992	machine learning	instance features	algorithm	learned rules	offline	static	1992
Brodley 1993	machine learning	instance and algorithm features	algorithm	hand-crafted rules	offline	static	1993
Kamel et al. 1993	differential equations	past performance, instance features	algorithm	hand-crafted rules	offline	static	1993
Minton 1993b, Minton 1993a, Minton 1996	CSP	runtime performance	algorithm	hand-crafted and learned rules	offline	dynamic	1993
Cahill 1994	software design	instance features	algorithms and data structures	frame-based knowledge base	offline	static	1994
Tsang et al. 1995	CSP	instance features	-	-	-	static	1995
Brewer 1995	software design	runtime performance	algorithms, data structures and their parameters	statistical model	offline	static	1995
Weerawarana et al. 1996, Joshi et al. 1996	differential equations	instance features	runtime performance	Bayesian belief propagation, neural nets	offline	static	1996
Borrett et al. 1996	CSP	search statistics	switch algorithm?	hand-crafted rules	online	static, static order	1996
Allen and Minton 1996	SAT, CSP	probing	runtime performance	hand-crafted rules	online	static	1996
Sakkout et al. 1996	CSP	search statistics	switch algorithm?	hand-crafted rules	online	static	1996
Huberman et al. 1997	graph colouring	past performance	resource allocation	statistical model	offline	static	1997
Gomes and Selman 1997b, Gomes and Selman 1997a	CSP	problem size and past performance	algorithm	statistical model	offline	static	1997
Cook and Varnell 1997	parallel search	probing	set of search strategies	decision trees, Bayesian classifier, nearest neighbour, neural net	online	static	1997
Fink 1997, Fink 1998	planning	past performance	resource allocation	statistical model, regression	offline	static	1997
Lobjois and Lemaître 1998	branch and bound	probing	runtime performance	hand-crafted rules	online	static	1998
Caseau et al. 1999	vehicle routing problem	runtime performance	algorithm	genetic algorithms	offline	static	1999
Howe et al. 1999	planning	instance features	resource allocation	linear regression	offline	static	1999
Terashima-Marín et al. 1999	scheduling	instance and search features	algorithm	genetic algorithms	offline	dynamic	1999
Wilson et al. 2000	software design	instance features	data structures	nearest neighbour	offline	static	2000
Beck and Fox 2000	job shop scheduling	instance feature changes during search	algorithm scheduling policy	hand-crafted rules	online	static	2000
Brazdil and Soares 2000	classification	past performance	ranking	distribution model	offline	static	2000
Lagoudakis and Littman 2000	order selection, sorting	instance features	remaining cost for each sub-problem	MDP	online	static	2000
Sillito 2000	CSP	probing	cost of solving problem	statistical model	offline	static	2000
Pfahringer et al. 2000	classification	instance features, probing	algorithm	9 different classifiers	offline	static	2000
Fukunaga 2000	TSP	past performance	resource allocation	performance simulation for different allocations	offline	static	2000
Soares and Brazdil 2000	machine learning	instance features	ranking	nearest neighbour	offline	static	2000
Gomes and Selman 2001	CSP, mixed integer programming	past performance	algorithm	statistical model	offline	dynamic	2001
Epstein and Freuder 2001, Epstein et al. 2002, Epstein et al. 2005, Epstein and Petrovic 2011	CSP	variable characteristics	algorithm	weights, hand-crafted rules	offline and online	dynamic	2001
Lagoudakis and Littman 2001	DPLL branching rules	instance features	remaining cost for each sub-problem	MDP	online	static	2001
Nareyek 2001	optimization	search statistics	expected utility of algorithm	reinforcement learning	offline and online	static	2001
Horvitz et al. 2001	CSP	instance and instance generator features, search statistics	runtime performance, restart parameters	Bayesian model	offline and online	static	2001

http://larskotthoff.github.io/assurvey/