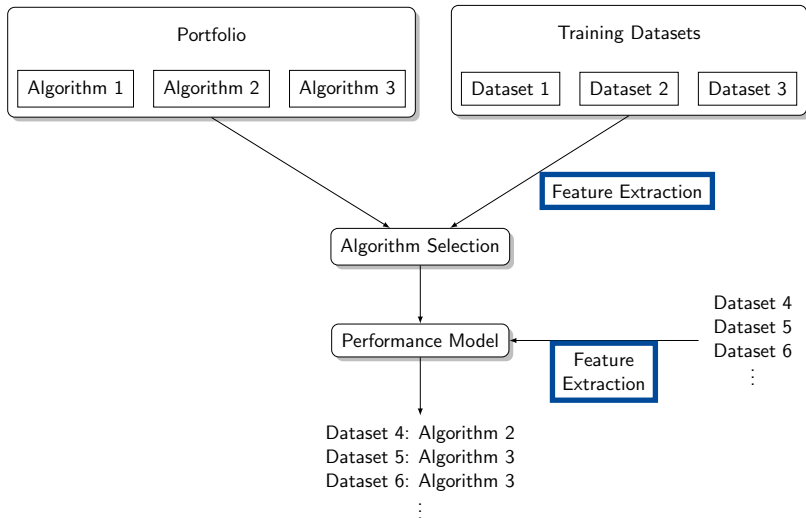


# AutoML: Algorithm Selection

## 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
- ...

→ usually leads to much better results than 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

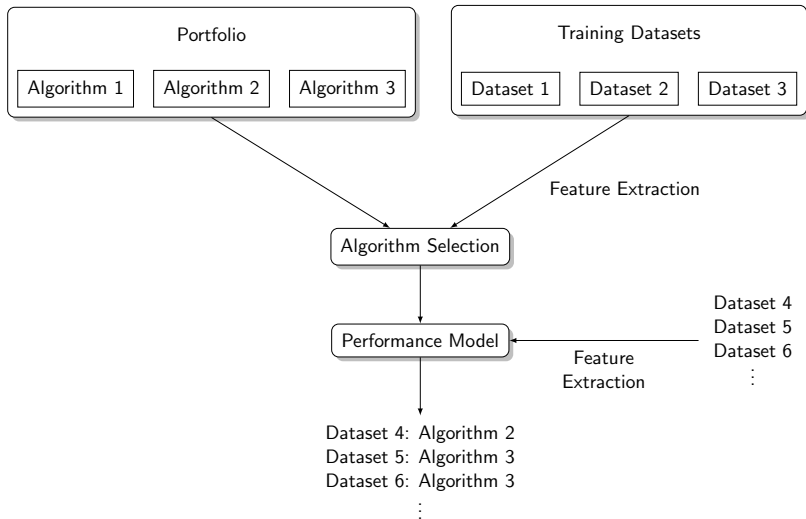


# AutoML: Algorithm Selection

## 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

“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

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

## Overview and Motivation

Bernd Bischl   Frank Hutter   Lars Kotthoff  
Marius Lindauer   Joaquin Vanschoren

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

# Algorithm Selection

## More formally

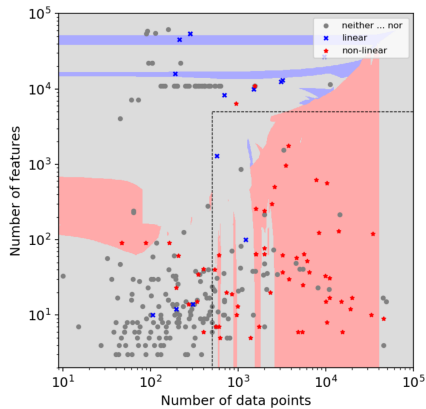
Let

- $p(\mathcal{D})$  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

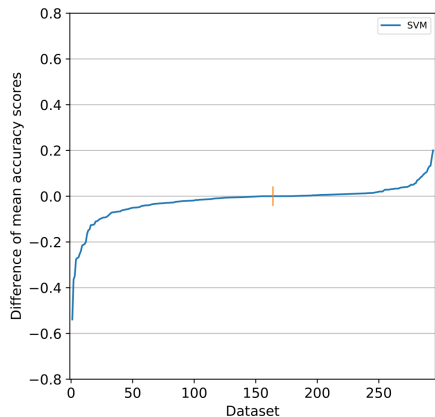
the *per-instance algorithm selection problem* is to obtain a mapping  $s : \mathcal{D} \mapsto \mathcal{A}$  such that

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

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



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

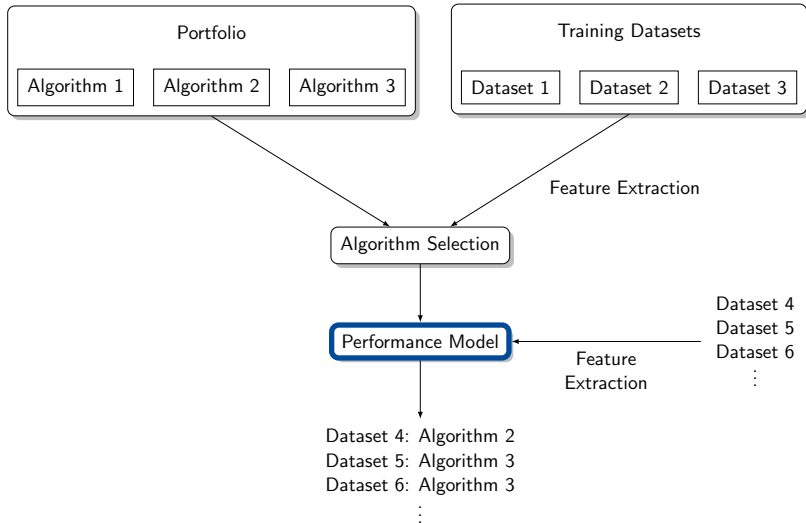


# AutoML: Algorithm Selection

## 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)

→ 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

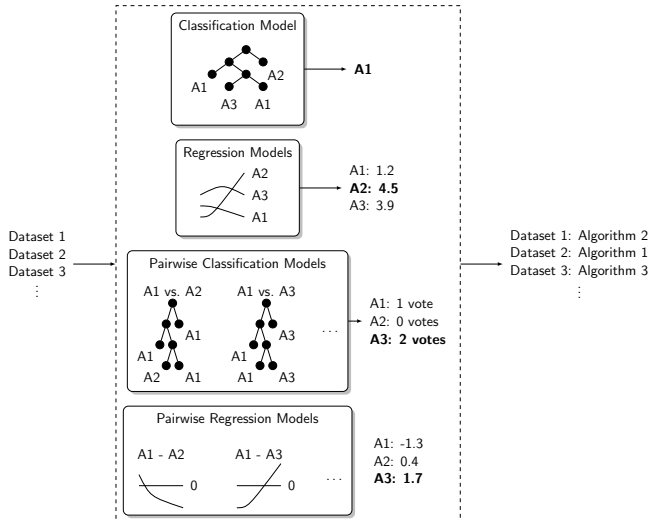
# Models for Individual Algorithms

- 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

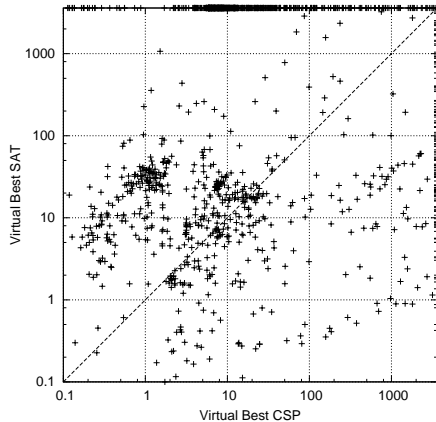
Bernd Bischl   Frank Hutter   Lars Kotthoff  
Marius Lindauer   Joaquin Vanschoren



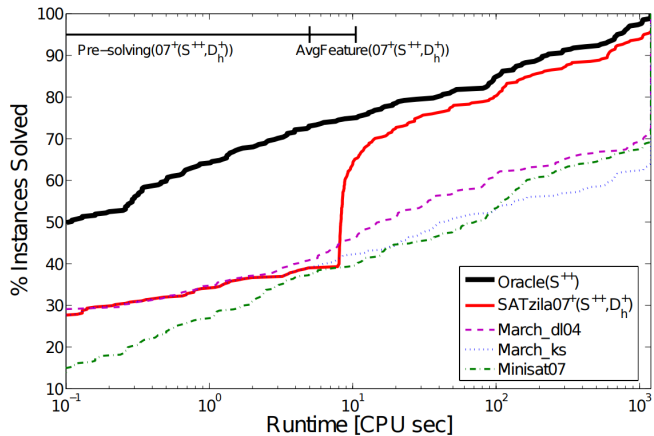
# Motivation

- 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] |



# 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

- [https://github.com/coseal/aslib\\_data](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>

**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?  
[Lars.Kotthoff](mailto:Lars.Kotthoff)

## Algorithm Selection Literature Summary

Last update 21 November 2018

click headings to sort  
 click columns to expand



citation	dataset	features	predict what	predict how	predict where	portfolio	year
Largely 1983a, Largely 1983a	search	past performance	algorithm	hand-crafted and learned rules	offline and online	dynamic	1983
Calverton et al. 1981	planning	problem domain features, search statistics	search rules	explication based rule construction	online	dynamic	1981
Gratch and DeJong 1992	planning	problem domain features, search statistics	control rules	probabilistic rule construction	online	dynamic	1992
Smith and Smith 1992	software design	features of abstract representation	algorithms and data structures	circulated annealing	offline	static	1992
Aha 1992	machine learning	instance features	algorithm	learned rules	offline	static	1992
Broley 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
Milson 1993a, Milson 1993a, Milson 1995	CSP	runtime performance	algorithm	hand-crafted and learned rules	offline	dynamic	1993
Cahil 1984	software design	instance features	algorithms and data structures	frame-based knowledge base	offline	static	1984
Tsang et al. 1985	CSP	instance features	-	-	offline	static	1985
Brewer 1985	software design	runtime performance	algorithms, data structures and their parameters	statistical model	offline	static	1985
Wierwille et al. 1995, Joehi et al. 1995	differential equations	instance features	runtime performance	Bayesian belief propagation, neural nets	offline	static	1995
Bonnet et al. 1996	CSP	search statistics	switch algorithms?	hand-crafted rules	online	static, static order	1996
Allen and Hömmer 1996	SAT, CSP	problem	runtime performance	hand-crafted rules	online	static	1996
Sakkout et al. 1996	CSP	search statistics	switch algorithms?	hand-crafted rules	online	static	1996
Huelsenman et al. 1997	graph colouring	past performance	resource allocation	statistical model	offline	static	1997
Gomes and Selman 1997a, Gomes and Selman 1997a	CSP	problem size and past performance	algorithm	statistical model	offline	static	1997
Cook and Vareli 1997	parallel search	problem	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
Lichten and Lemstra 1998	branch and bound	problem	runtime performance	hand-crafted rules	online	static	1998
Gomes et al. 1999	vehicle routing problem	runtime performance	algorithm	genetic algorithms	offline	static	1999
Hose et al. 1999	planning	instance features	resource allocation	linear regression	offline	static	1999
Tepstrima-Mann et al. 1999	scheduling	instance and search features	algorithm	genetic algorithms	offline	dynamic	1999
Wolpin 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
Braxell and Soares 2000	classification	past performance	tasking	distribution model	offline	static	2000
Lapoutsakis and Litzman 2000	order selection, sorting	instance features	estimating cost for each sub-problem	NECP	online	static	2000
Silko 2000	CSP	problem	cost of solving problem	statistical model	offline	static	2000
Plattinger et al. 2000	classification	instance features, problem	algorithm	9 different classifiers	offline	static	2000
Fuhrmann 2000	CSP	past performance	resource allocation	performance simulation for different allocations	offline	static	2000
Soares and Braxell 2000	machine learning	instance features	tasking	nearest neighbour	offline	static	2000
Gomes and Selman 2001	CSP, mixed integer programming	past performance	algorithm	statistical model	offline	dynamic	2001
Eysenck and Preuder 2001, Eysenck et al. 2002, Eysenck et al. 2008, Eysenck and Petrovic 2011	CSP	variable characteristics	algorithm	weights, hand-crafted rules	offline and online	dynamic	2001
Lapoutsakis and Litzman 2001	DPLL branching rules	instance features	estimating cost for each sub-problem	NECP	online	static	2001
Rosenfeld 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, model parameters	Bayesian model	offline and online	static	2001

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