Feast, AutoML and XAI

Przemysław Biecek

2019

Machine Learning:
The High-Interest Credit Card of Technical Debt

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov,
Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young
{dsculley,gholt,dgg,edavydov}@google.com
{toddphillips,ebner,vchaudhary,mwyoung}@google.com
Google, Inc

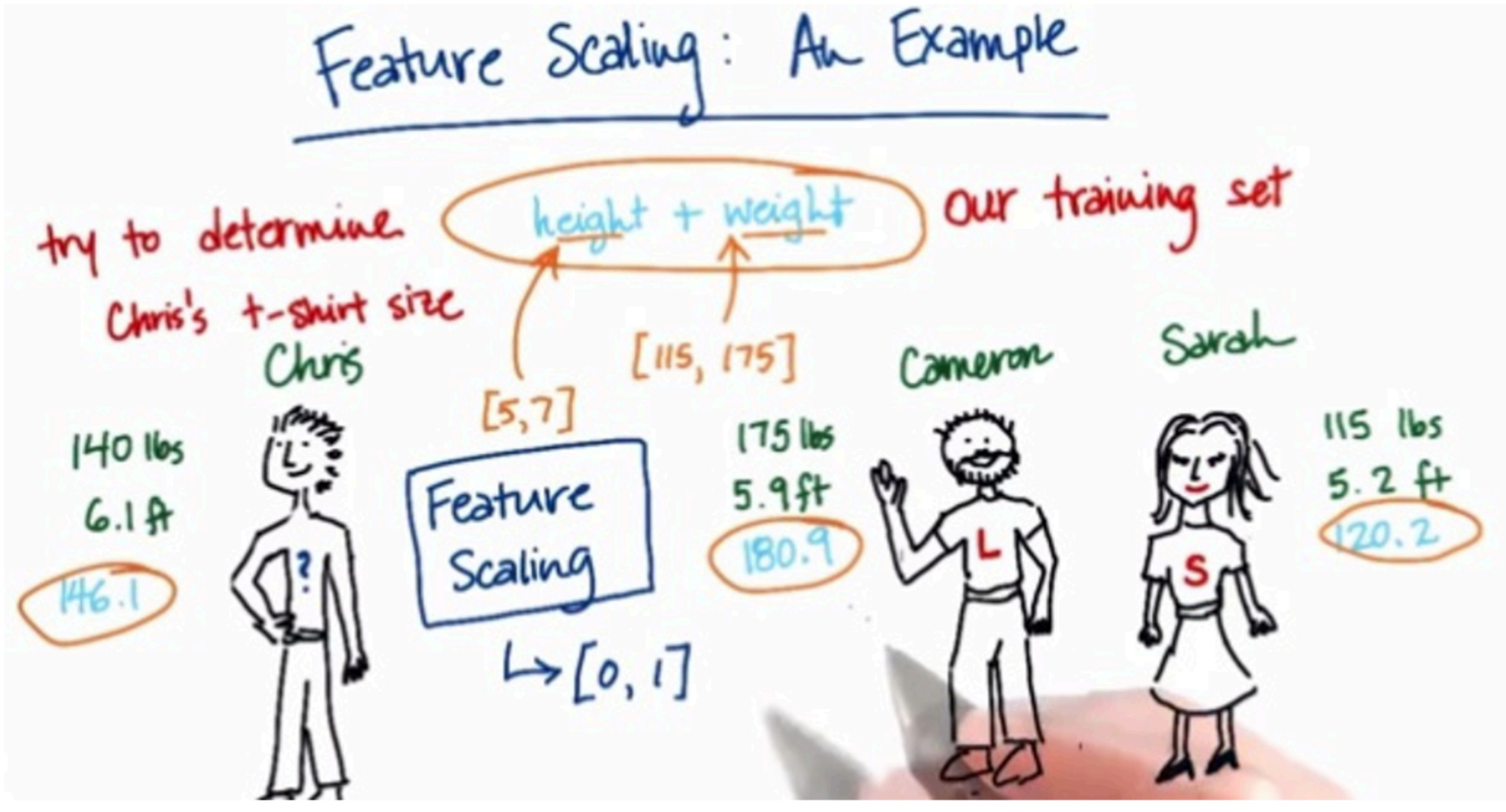
Abstract

Machine learning offers a fantastically powerful toolkit for building complex systems quickly. This paper argues that it is dangerous to think of these quick wins as coming for free. Using the framework of *technical debt*, we note that it is remarkably easy to incur massive ongoing maintenance costs at the system level when applying machine learning. The goal of this paper is highlight several machine learning specific risk factors and design patterns to be avoided or refactored where possible. These include boundary erosion, entanglement, hidden feedback loops, undeclared consumers, data dependencies, changes in the external world, and a variety of system-level anti-patterns.

Introducing Feast

Google's New Feature Store for Machine Learning Applications

Jesus Rodriguez Follow
Jan 23 · 5 min read

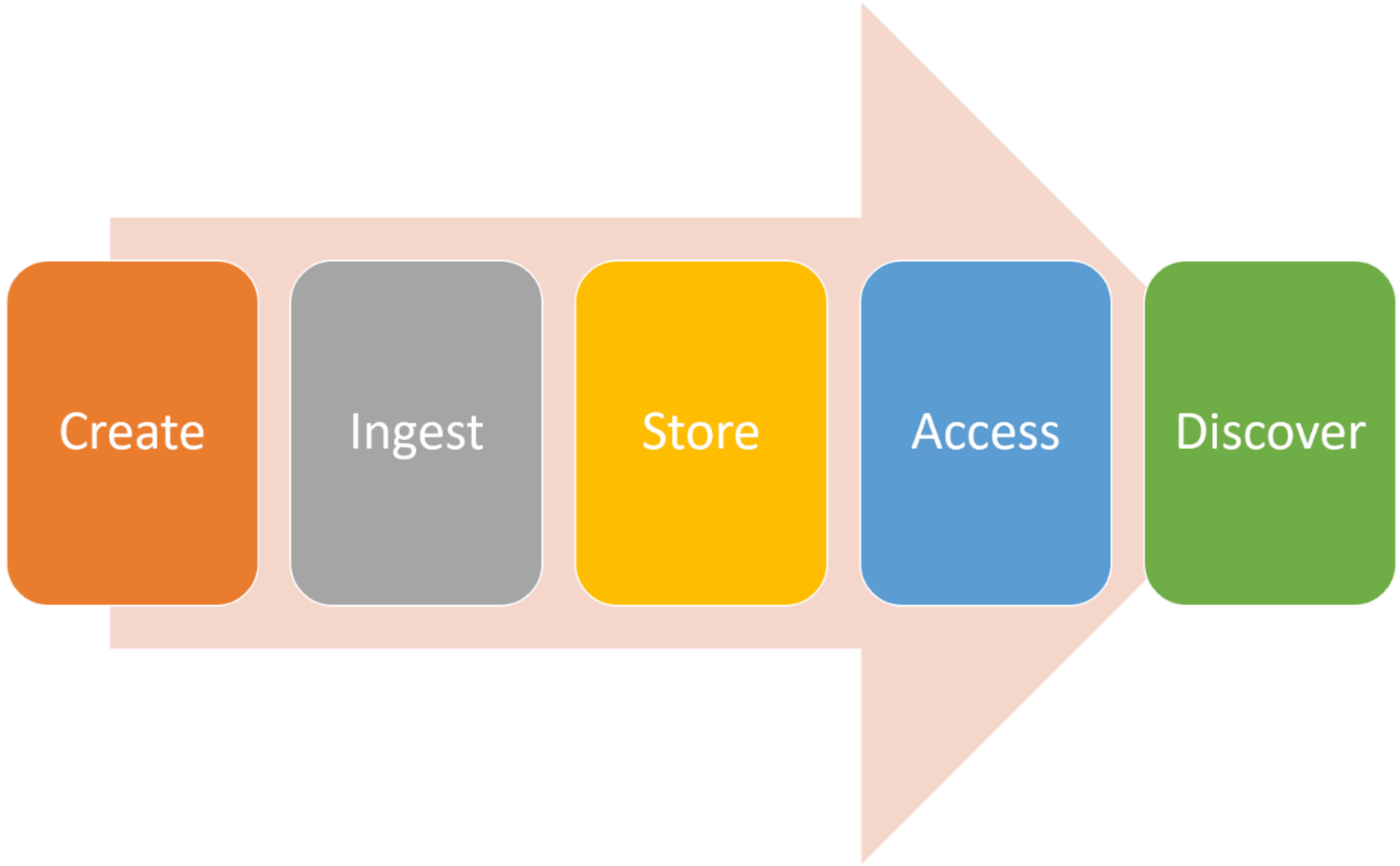


· **Feature Standardization:** Feast attempts to present a centralized repository for describing features of machine learning models. This provides structure to the way features are defined and allows teams to reuse features across different machine learning models.

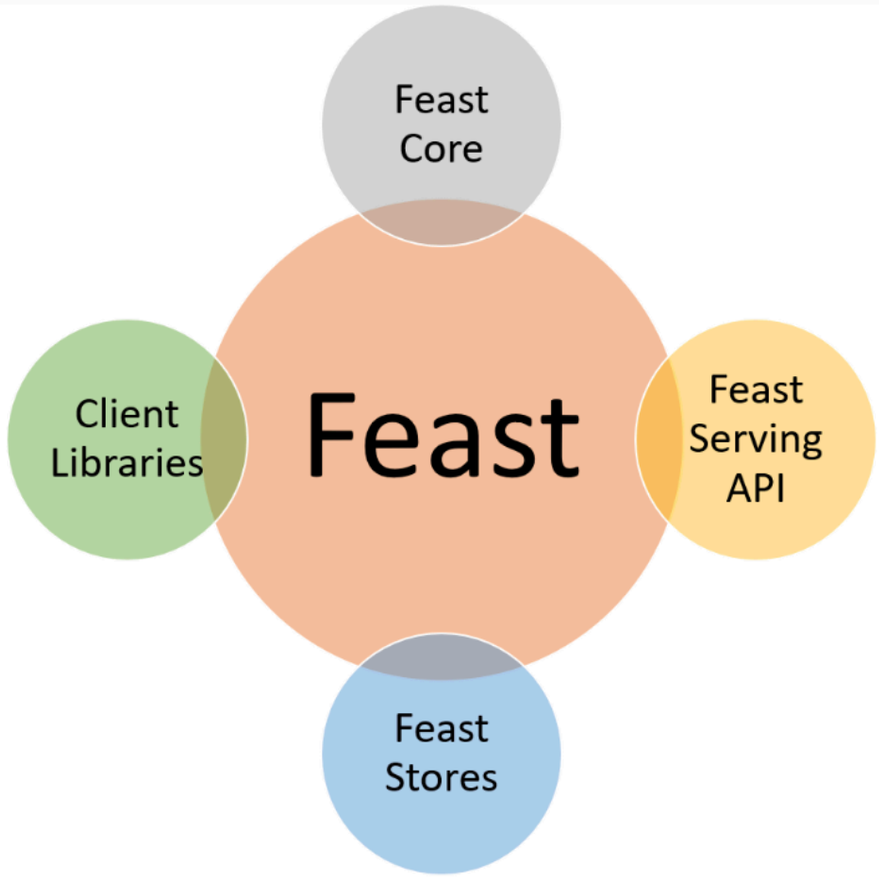
· **Feature Discovery:** Feast enables the exploration and discoverability of features and their associated information. This allows for a deeper understanding of features and their specifications, more feature reuse between teams and projects, and faster experimentation.

· **Model Training-Serving Consistency:** Feast's standard representations enables feature consistency between model training and serving. This addresses the constant mismatch between the development and production version of machine learning models.

· **Feature Infrastructure Management:** A pretty obvious benefit, Feast abstracts the infrastructure needed to extract, store and manage features across machine learning models. Although conceptually simple, feature extraction is one of those areas that ends up consuming incredibly large amounts of time in machine learning implementations.



- **Create:** features based on defined format and programming model
- **Ingest**: features via streaming input, import from files or BigQuery tables, and write to an appropriate data store
- **Store:** feature data for both serving and training purposes based on feature access patterns
- **Access:** features for training and serving
- **Discover:** information about entities and features stored and served by Feast



· **Feast Core:** The Core subsystem is responsible for managing the different components of Feast. For instance, Feast Core manages the execution of feature ingestion jobs from batch and streaming sources while also enabling the registration and management of entities, features, data stores, and other system resources.

· **Feast Store:** Feast supports two fundamental types of stores: warehouses and serving. Feast Warehouse Stores are based on Google BigQuery and maintain all historical feature data. The warehouse can be queried for batch datasets which are then used for model training. Serving Stores are responsible for maintaining feature values for access in a production serving environment.

· **Feast Serving API:** This API is responsible for the retrieval of feature values by models in production. Feast Serving API supports HTTP and gRPC models which allows for low latency and high throughput execution models.

Feature Store

Uber Michelangelo

https://eng.uber.com/michelangelo/

> Finding good features is often the hardest part of machine learning and we have found that building and managing data pipelines is typically one of the most costly pieces of a complete machine learning solution.
>
> A platform should provide standard tools for building data pipelines to generate feature and label data sets for training (and re-training) and feature-only data sets for predicting. These tools should have deep integration with the company's data lake or warehouses and with the company's online data serving systems. The pipelines need to be scalable and performant, incorporate integrated monitoring for data flow and data quality, and support both online and offline training and predicting. Ideally, they should also generate the features in a way that is shareable across teams to reduce duplicate work and increase data quality. They should also provide strong guard rails and controls to encourage and empower users to adopt best practices (e.g., making it easy to guarantee that the same data generation/preparation process is used at both training time and prediction time).
>
> (...) (a) feature store that allows teams to s learning problems.

Goals

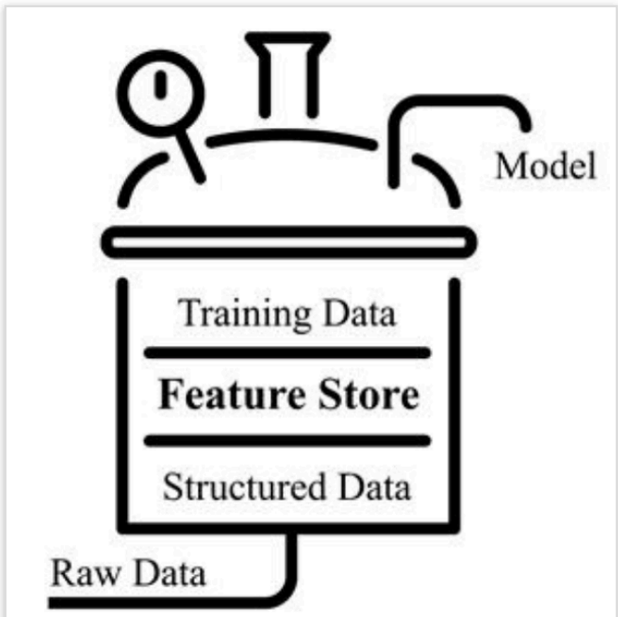
1. Consume data from sources and transform them into Features for ML experiments.
2. Being able to produce a Feature Set by consuming and mix Features from different sources (including public, curated data sources), with the help of Feature Discovery tooling.
3. Feature Intelligence:
    - Automated Feature extraction (featuretools.com)
    - Feature Visualization
    - Pre-trained model integration to generate new features
    - Any type of Feature Selection aid
4. Feature set management and versioning
5. Training Data generation and management
6. Prediction Endpoint generation
7. Availability of an SDK for rapid model training and deployment using existing solutions (Sagemaker, Comet ML, Valohai, others, ...)

Stockholm, Sweden, Jan. 21, 2019 (GLOBE NEWSWIRE) -- Logical Clocks, the enterprise vendor for Hopsworks - a data platform for scale-out data science and AI, today announced the release of the first Enterprise Feature Store for Machine Learning. The Feature Store solves the problem of ad-hoc and siloed machine learning pipelines, where features, the training data for such pipelines, tend to become disorganized, disjointed, and duplicated, leading to correctness problems and redundant work.

Today, Logical Clocks AB are announcing the release of a Feature Store as part of Hopsworks version 0.8.0. The Feature Store is a central vault for documented, curated, and access-controlled features. In-house Feature Stores are already successfully in production at companies such as Uber, LinkedIn, Airbnb, and Comcast. Now, for the first time, a Feature Store is available, as open-source, in an Enterprise Data platform, Hopsworks.

With the increasing adoption of machine learning in the Enterprise, organizations are looking to reduce the cost of developing and deploying AI by increasing the productivity of their Data Scientists. **According to Uber**, “dealing with data access, integration, feature management, and pipelines can often waste a huge amount of a data scientist’s time”. The Feature Store solves the data access and feature management problem for Data Science by removing the need for Data Scientists to constantly re-implement feature pipelines for collecting and transforming data to feed their machine learning models. Instead, Data Scientists can select features from the Feature Store to generate clean training data that can then be consumed directly by machine learning models. Hopsworks’ Feature Store builds on Apache Spark and Apache Hive to enable it to scale to massive data volumes.

Feature Store



Distill data into intelligence using a Feature Store

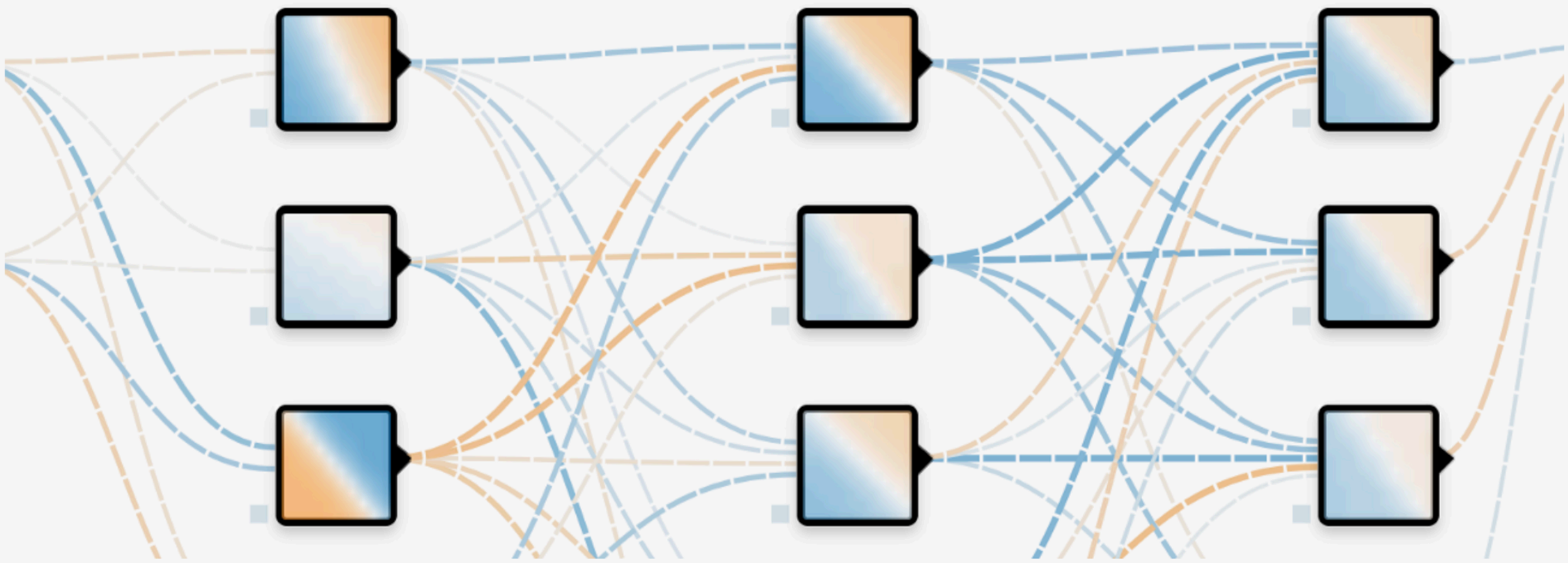
Logical Clocks AB

Machine Learning

AutoML and the Rise of Advanced Machine Learning Platforms

Google's Next 18 shows that machine learning is becoming more and more developer-friendly—all thanks to AutoML.

William Wen
September 14, 2018



Source: TensorFlow

TL;DR: Google is making its ML offerings more accessible and developer-friendly. Thanks to AutoML on G Cloud (and other machine learning platforms), big players like Google are taking huge steps towards popularizing and automating ML.

https://www.connected.io/post/automl-and-the-rise-of-advanced-machine-learning-platforms

Research Opportunities in AutoML - Rich Caruana

AutoML Tools to Better Understand Data

- Auto variable type determination
  - 0, 1, 2, 3, 4, 5: nominal, ordinal, integer, continuous?
  - Are there dates in fields?
  - Is a field a unique identifier or sequence number?
- Auto coding
  - Different coding needed for NNs, SVMs, KNN vs. decision tree-based method
- Auto missing value detector
  - 0, 1, 2
  - -1, 0, +1
  - Can't just try everything --- missing variables often cause leakage!
- Auto anomaly detection

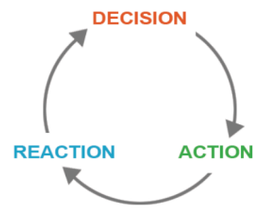
Want to do New Research that Gets Cited?

- Pick a part of the ML pipeline that's still largely manual
- Define what it would mean to make it (more) automatic
- Develop and publish methods
  - Fully-automatic "robot" that solves problem
  - Assistant that helps human recognize and solve problem
  - Tools that alert when problem (probably) exists
- Make data sets publicly available
- Make code available for use as a baseline (and possibly openSource)
- Organize a challenge competition on that part of the pipeline
- Good way to pick a thesis topic!

Importance of Hyper-Parameter Optimization

- Hyper-Parameter Optimization is most mature subarea in AutoML
  - Manual heuristic search: surprisingly sub-optimal
  - Grid search: effective with small number of parameters
  - Random search: better than grid with larger number of parameters
  - Bayesian Optimization: better than random with very large # parameters
  - …
- With modern algorithms (boosting, deep neural nets, …) parameter optimization is much more critical than you might think…
  - … because modern high-flying algorithms are all low-bias, high variance

Feedback --- the Future Curse of ML!



- If you train a model on patient data
- And model is used to change practice of medicine (intervention)
- Next time you collect data it is affected by model…
- …so how do you collect unbiased data 2nd time around?

- This is a deep, fundamental problem that in some domains is not easy to solve (ethically, or efficiently) --- problem with non-causal learning

https://indico.lal.in2p3.fr/event/2914/contributions/6481/attachments/6048/7173/CaruanaAutoMLWorkshopICML2015rev4.pdf

LITERATURE ON NEURAL ARCHITECTURE SEARCH

The following list considers papers related to neural architecture search. It is by no means a complete list. If you miss a paper on this list, please let us know.

- Architecture Search (and Hyperparameter Optimization):

  - Evolving Space-Time Neural Architectures for Videos (Piergiovanni et al. 2018)
    https://arxiv.org/abs/1811.10636
  - InstaNAS: Instance-aware Neural Architecture Search (Cheng et al. 2018)
    https://arxiv.org/abs/1811.10201
  - Evolutionary-Neural Hybrid Agents for Architecture Search (Maziarz et al. 2018)
    https://arxiv.org/abs/1811.09828
  - Joint Neural Architecture Search and Quantization (Chen et al. 2018)
    https://arxiv.org/abs/1811.09426
  - Transfer Learning with Neural AutoML (Wong et al. 2018)
    http://papers.nips.cc/paper/8056-transfer-learning-with-neural-automl.pdf
  - Evolving Image Classification Architectures with Enhanced Particle Swarm Optimisation (Fielding and Zhang 2018)
    https://ieeexplore.ieee.org/document/8533601
  - Deep Active Learning with a Neural Architecture Search (Geifman and El-Yaniv 2018)
    https://arxiv.org/abs/1811.07579
  - Stochastic Adaptive Neural Architecture Search for Keyword Spotting (Véniat et al. 2018)
    https://arxiv.org/abs/1811.06753
  - You only search once: Single Shot Neural Architecture Search via Direct Sparse Optimization (Zhang et al. 2018)
    https://arxiv.org/abs/1811.01567
  - Automatically Evolving CNN Architectures Based on Blocks (Sun et al. 2018)
    https://arxiv.org/abs/1810.11875
  - Training Frankenstein's Creature to Stack: HyperTree Architecture Search (Hundt et al. 2018)
    https://arxiv.org/abs/1810.11714
  - Fast Neural Architecture Search of Compact Semantic Segmentation Models
    via Auxiliary Cells (Nekrasov et al. 2018)
    https://arxiv.org/abs/1810.10804
  - Automatic Configuration of Deep Neural Networks with Parallel Efficient Global Optimization (van Stein et al. 2018)

https://www.ml4aad.org/automl/literature-on-neural-architecture-search/

Awesome-AutoML-Papers

A curated list of automated machine learning papers, articles, tutorials, slides and projects.

Automated Feature Engineering

- **Expand Reduce**
  - 2017 | AutoLearn — Automated Feature Generation and Selection | Ambika Kaul, et al. | ICDM | PDF
  - 2017 | One button machine for automating feature engineering in relational databases | Hoang Thanh Lam, et al. | arXiv | PDF
  - 2016 | Automating Feature Engineering | Udayan Khurana, et al. | NIPS | PDF
  - 2016 | ExploreKit: Automatic Feature Generation and Selection | Gilad Katz, et al. | ICDM | PDF
  - 2015 | Deep Feature Synthesis: Towards Automating Data Science Endeavors | James Max Kanter, Kalyan Veeramachaneni | DSAA | PDF
- **Hierarchical Organization of Transformations**
  - 2016 | Cognito: Automated Feature Engineering for Supervised Learning | Udayan Khurana, et al. | ICDMW | PDF
- **Meta Learning**
  - 2017 | Learning Feature Engineering for Classification | Fatemeh Nargesian, et al. | IJCAI | PDF
- **Reinforcement Learning**
  - 2017 | Feature Engineering for Predictive Modeling using Reinforcement Learning | Udayan Khurana, et al. | arXiv | PDF
  - 2010 | Feature Selection as a One-Player Game | Romaric Gaudel, Michele Sebag | ICML | PDF

Architecture Search

- **Evolutionary Algorithms**
  - 2017 | Large-Scale Evolution of Image Classifiers | Esteban Real, et al. | PMLR | PDF
  - 2002 | Evolving Neural Networks through Augmenting Topologies | Kenneth O.Stanley, Risto Miikkulainen | Evolutionary Computation | PDF
- **Local Search**
  - 2017 | Simple and Efficient Architecture Search for Convolutional Neural Networks | Thomoas Elsken, et al. | ICLR | PDF
- **Meta Learning**
  - 2016 | Learning to Optimize | Ke Li, Jitendra Malik | arXiv | PDF
- **Reinforcement Learning**
  - 2018 | Efficient Neural Architecture Search via Parameter Sharing | Hieu Pham, et al. | arXiv | PDF
  - 2017 | Neural Architecture Search with Reinforcement Learning | Barret Zoph, Quoc V. Le | ICLR | PDF
- **Transfer Learning**

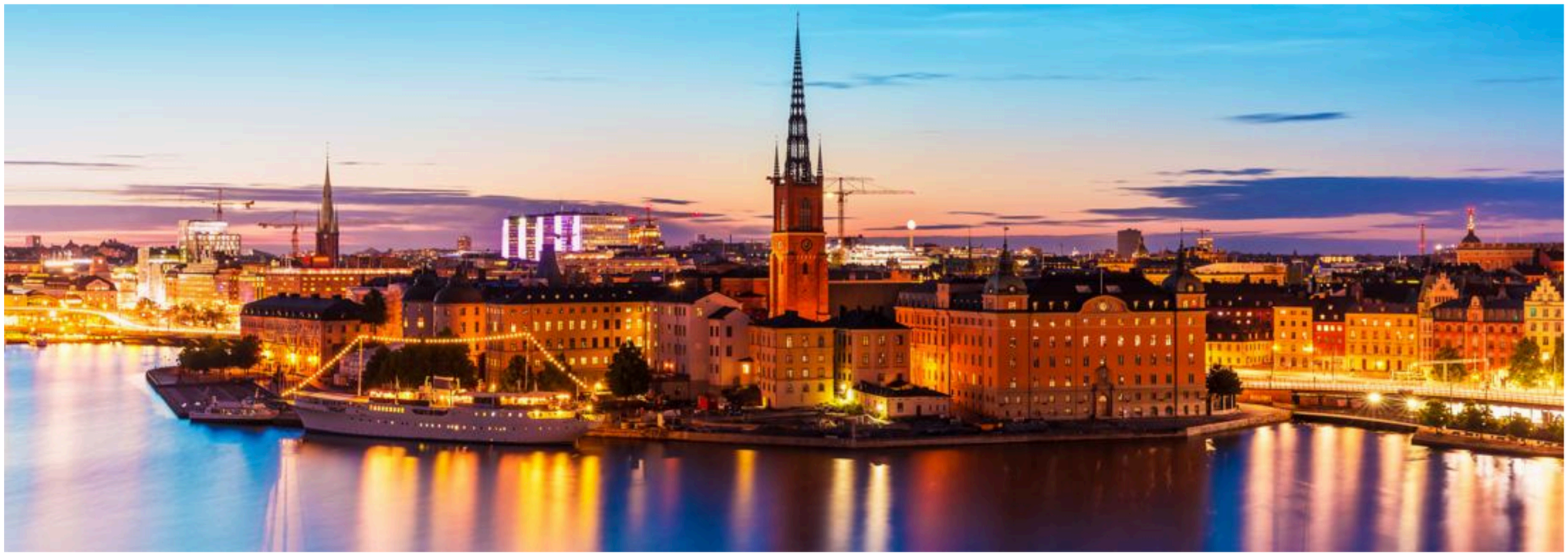
Home

International Workshop on Automatic Machine Learning

July 14th, 2017

Collocated with the Federated AI Meeting (ICML, IJCAI, AMAS, and ICCBR), Stockholm, Sweden

Ask a question for the panel discussion



https://sites.google.com/site/automl2018icml/accepted-papers

~ 40 accepted papers

37 Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph Gonzalez and Ion Stoica.
Demo Tune: A Research Platform for Distributed Model Selection and Training Paper

50 Jorge G Madrid, Hugo J Escalante, Eduardo Morales, Wei Wei Tu, Yang Yu, Lisheng Sun-Hosoya, Isabelle Guyo
Towards AutoML in the presence of Drift: first results *(Contributed Talk 3)* Paper

6 Casey Davis and Christophe Giraud-Carrier.
Annotative Experts for Hyperparameter Selection Paper

7 Kunkun Pang, Mingzhi Dong, Yang Wu and Timothy Hospedales.
Meta-Learning Transferable Active Learning Policies by Deep Reinforcement Learning *(Contributed Talk 2)*

12 Mark McLeod, Michael A. Osborne and Stephen J Roberts.
Adaptive Quadrature for Fast Sequential Hyperparameter Marginalization Paper

19 Esteban Real, Alok Aggarwal, Yanping Huang and Quoc V. Le.
Evolutionary Algorithms and Reinforcement Learning: **A Comparative Case Study for Architecture Search** *(C*

35 Iddo Drori, Yamuna Krishnamurthy, Remi Rampin, Raoni de Paula Lourenco, Jorge Piazentin Ono, Kyunghyun C
AlphaD3M: Machine Learning Pipeline Synthesis (*Contributed Talk 1)* *Paper*

43 Scott Langevin, David Jonker, Chris Bethune, Glen Coppersmith, Casey Hilland, Jonathon Morgan, Paul Azunre
Demo Distil: A Mixed-Initiative Model Discovery System for Subject Matter Experts Paper

8 Erin Grant, Ghassen Jerfel, Katherine Heller and Thomas L. Griffiths.
Augmenting Gradient-Based Meta-Learning with Latent Variables to Capture Task Heterogeneity Paper

17 Yolanda Gil, Ke-Thia Yao, Varun Ratnakar, Daniel Garijo, Greg Ver Steeg, Pedro Szekely, Robert Brekelmans, M
P4ML: A Phased Performance-Based Pipeline Planner for Automated Machine Learning Paper

4 Janek Thomas, Stefan Coors and Bernd Bischl.
Demo Automatic Gradient Boosting Paper

9 Haifeng Jin, Qingquan Song and Xia Hu.
Demo Efficient Neural Architecture Search with Network Morphism Paper

13 Matthias Feurer, Benjamin Letham and Eytan Bakshy.
Scalable Meta-Learning for Bayesian Optimization using Ranking-Weighted Gaussian Process Ensembles Paper

16 Prajit Ramachandran and Quoc Le.
Dynamic Network Architectures Paper

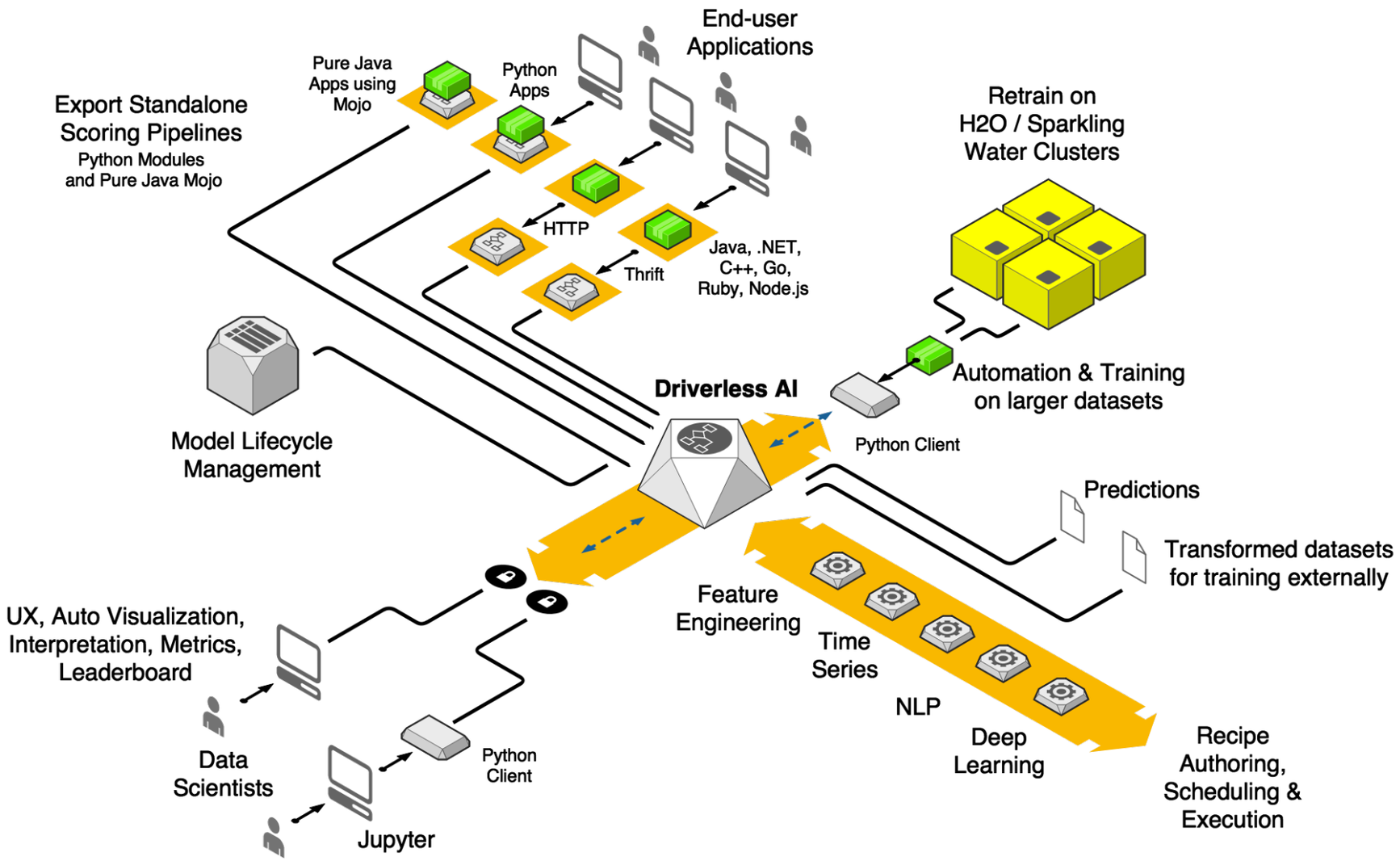
https://sites.google.com/site/automl2018icml/accepted-papers

H2O.ai
DRIVERLESSAI

Automated Modelling

H2O's AutoML can be used for automating the machine learning workflow, which includes automatic training and tuning of many models within a user-specified time-limit. Stacked Ensembles – one based on all previously trained models, another one on the best model of each family – will be automatically trained on collections of individual models to produce highly predictive ensemble models which, in most cases, will be the top performing models in the AutoML Leaderboard

In recent years, the demand for machine learning experts has outpaced the supply, despite the surge of people entering the field. (H2O)



H2O.ai DRIVERLESSAI

Automated Modelling

Driverless AI Roadmap

Feature	v1.0	v1.1	v1.2	v1.3	v1.4 (NOW)	v1.5	v2.0
Kaggle **Grandmaster Recipes** for i.i.d. data							
Automatic Visualization							
Machine Learning Interpretability							
GBM (XGBoost) for high accuracy incl. stacked ensembles (CPU/GPU)							
5-minute Install with **Docker** for Linux/Mac/Windows - **Cloud/OnPrem**							
Standalone **Python Scoring Pipeline**							
Hardware acceleration: **NVIDIA GPUs (DGX-1 etc.)**							
User Management and Security (LDAP/Kerberos)							
Data Connectors: NFS/HDFS/S3/GCS/BigQuery, CSV/Excel/Parquet/Feather							
GLM (Linear models) for high interpretability (CPU/GPU)							
Native Installer: **RPM/DEB**							
Cloud Neutral: Amazon/Microsoft/Google							
Kaggle Grandmaster Recipes for **Time-Series**							
IBM Power8/Power9							
AutoDoc							
Deep Learning **TensorFlow** Models (CPU/GPU)							
Standalone **Java Scoring Pipeline** (MOJO)							
Deep Learning for **NLP** / Text (CPU/GPU)							
LightGBM Models (CPU/GPU)							
Improved Time-Series Recipes (Multiple Windows, **MLI** for Time-Series							
Improved Final Ensemble							
Local Feature Brain							
C++ Scoring Pipeline							
FTRL Models							
Multi-Node Training and much more (based on customer demand)							

Required Data Parameters

- y: This argument is the name (or index) c
- training_frame: Specifies the training set

Required Stopping Parameters

One of the following stopping strategies (time or number-of-model based) must be specified. When both options are set, then the AutoML run will stop as soon as it hits one of either of these limits.

- max_runtime_secs: This argument controls how long the AutoML will run at the most, before training the final Stacked Ensemble models. Defaults to 3600 seconds (1 hour).
- **max_models**: Specify the maximum number of models to build in an AutoML run, excluding the Stacked Ensemble models. Defaults to `NULL/None`.

http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

- **Which models are trained in the AutoML process?**

  The current version of AutoML trains and cross-validates the following algorithms (in the following order): A default Random Forest (DRF), an Extremely Randomized Forest (XRT), three pre-specified XGBoost GBM (Gradient Boosting Machine) models, five pre-specified H2O GBMs, a near-default Deep Neural Net, a random grid of XGBoost GBMs, a random grid of H2O GBMs, and lastly if there is time, a random grid of Deep Neural Nets. AutoML then trains two Stacked Ensemble models. Particular algorithms (or groups of algorithms) can be switched off using the `exclude_algos` argument. This is useful if you already have some idea of the algorithms that will do well on your dataset. As a recommendation, if you have really wide or sparse data, you may consider skipping the tree-based algorithms (GBM, DRF, XGBoost).

  A list of the hyperparameters searched over for each algorithm in the AutoML process is included in the appendix below. More details about the hyperparamter ranges for the models in addition to the hard-coded models will be added to the appendix at a later date.

  Both of the ensembles should produce better models than any individual model from the AutoML run with the exception of some rare cases. One ensemble contains all the models, and the second ensemble contains just the best performing model from each algorithm class/family. The "Best of Family" ensemble is optimized for production use since it only contains six (or fewer) base models. It should be relatively fast to use (to generate predictions on new data) without much degredation in model performance when compared to the "All Models" ensemble.

http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

AutoML Output

The AutoML object includes a "leaderboard" of models that were trained in the process, including the 5-fold cross-validated model performance (by default). The number of folds used in the model evaluation process can be adjusted using the `nfolds` parameter. If the user would like to score the models on a specific dataset, they can specify the `leaderboard_frame` argument, and then the leaderboard will show scores on that dataset instead.

The models are ranked by a default metric based on the problem type (the second column of the leaderboard). In binary classification problems, that metric is AUC, and in multiclass classification problems, the metric is mean per-class error. In regression problems, the default sort metric is deviance. Some additional metrics are also provided, for convenience.

Here is an example leaderboard for a binary classification task:

model_id	auc	logloss	mean_p
StackedEnsemble_AllModels_AutoML_20181022_221411	0.7870176	0.5541308	0.3254
StackedEnsemble_BestOfFamily_AutoML_20181022_221411	0.7857408	0.5553949	0.3265
XGBoost_grid_1_AutoML_20181022_221411_model_3	0.7825571	0.5598532	0.3326
XGBoost_1_AutoML_20181022_221411	0.7810665	0.5601261	0.3312
XGBoost_3_AutoML_20181022_221411	0.7808475	0.5611616	0.3240
XGBoost_grid_1_AutoML_20181022_221411_model_4	0.7806241	0.5606613	0.3229
XGBoost_2_AutoML_20181022_221411	0.7800521	0.5613740	0.3361
GBM_5_AutoML_20181022_221411	0.7798300	0.5614880	0.3267
GBM_1_AutoML_20181022_221411	0.7772283	0.5628248	0.3408
GBM_2_AutoML_20181022_221411	0.7751517	0.5645617	0.3356
GBM_3_AutoML_20181022_221411	0.7712083	0.5688081	0.3413

http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

Optional Miscellaneous Parameters

- nfolds: Number of folds for k-fold cross-validation of the models i 5. Use 0 to disable cross-validation; this will also disable Stacked E overall best model performance).
- balance_classes: Specify whether to oversample the minority clas distribution. This option is not enabled by default and can increas option is only applicable for classification. Majority classes can be **max_after_balance_size** parameter.
- class_sampling_factors: Specify the per-class (in lexicographical or ratios. By default, these ratios are automatically computed during balance. Note that this requires `balance_classes=true`.
- max_after_balance_size: Specify the maximum relative size of the class counts (**balance_classes** must be enabled). Defaults to 5.0. (T
- stopping_metric: Specifies the metric to use for early stopping of individual models. Defaults to `"AUTO"`. The available options are:
  - `AUTO`: This defaults to `logloss` for classification, `deviance`
  - `deviance` (mean residual deviance)
  - `logloss`
  - `MSE`
  - `RMSE`
  - `MAE`
  - `RMSLE`
  - `AUC`
  - `lift_top_group`
  - `misclassification`
  - `mean_per_class_error`
- stopping_tolerance: This option specifies the relative tolerance for the metric-b criterion to stop a grid search and the training of individual models within the A value defaults to 0.001 if the dataset is at least 1 million rows; otherwise it defa value determined by the size of the dataset and the non-NA-rate. In that case, computed as 1/sqrt(nrows * non-NA-rate).
- stopping_rounds: This argument is used to stop model training when the stopp AUC) doesn't improve for this specified number of training rounds, based on a s average. In the context of AutoML, this controls early stopping both within the searches as well as the individual models. Defaults to 3 and must be an non-ne disable early stopping altogether, set this to 0.
- sort_metric: Specifies the metric used to sort the Leaderboard by at the end of Available options include:
  - `AUTO`: This defaults to `AUC` for binary classification, `mean_per_class_error` classification, and `deviance` for regression.
  - `deviance` (mean residual deviance)
  - `logloss`
  - `MSE`
  - `RMSE`
  - `MAE`
  - `RMSLE`
  - `AUC`
  - `mean_per_class_error`

http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

Appendix: Random Grid Search Parameters

AutoML performs hyperparameter search over a variety of best model. In AutoML, the following hyperparameters are Forest and Extremely Randomized Trees are not grid search so they are not included in the list below.

GLM Hyperparameters

- `alpha`
- `missing_values_handling`

XGBoost Hyperparameters

- `ntrees`
- `max_depth`
- `min_rows`
- `min_sum_hessian_in_leaf`
- `sample_rate`
- `col_sample_rate`
- `col_sample_rate_per_tree`
- `booster`
- `reg_lambda`
- `reg_alpha`

GBM Hyperparameters

- `histogram_type`
- `ntrees`
- `max_depth`
- `min_rows`
- `learn_rate`
- `sample_rate`
- `col_sample_rate`
- `col_sample_rate_per_tree`
- `min_split_improvement`

Deep Learning Hyperparameters

- `epochs`
- `adaptivate_rate`
- `activation`
- `rho`
- `epsilon`
- `input_dropout_ratio`
- `hidden`
- `hidden_dropout_ratios`

http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

Machine Learning Interpretability

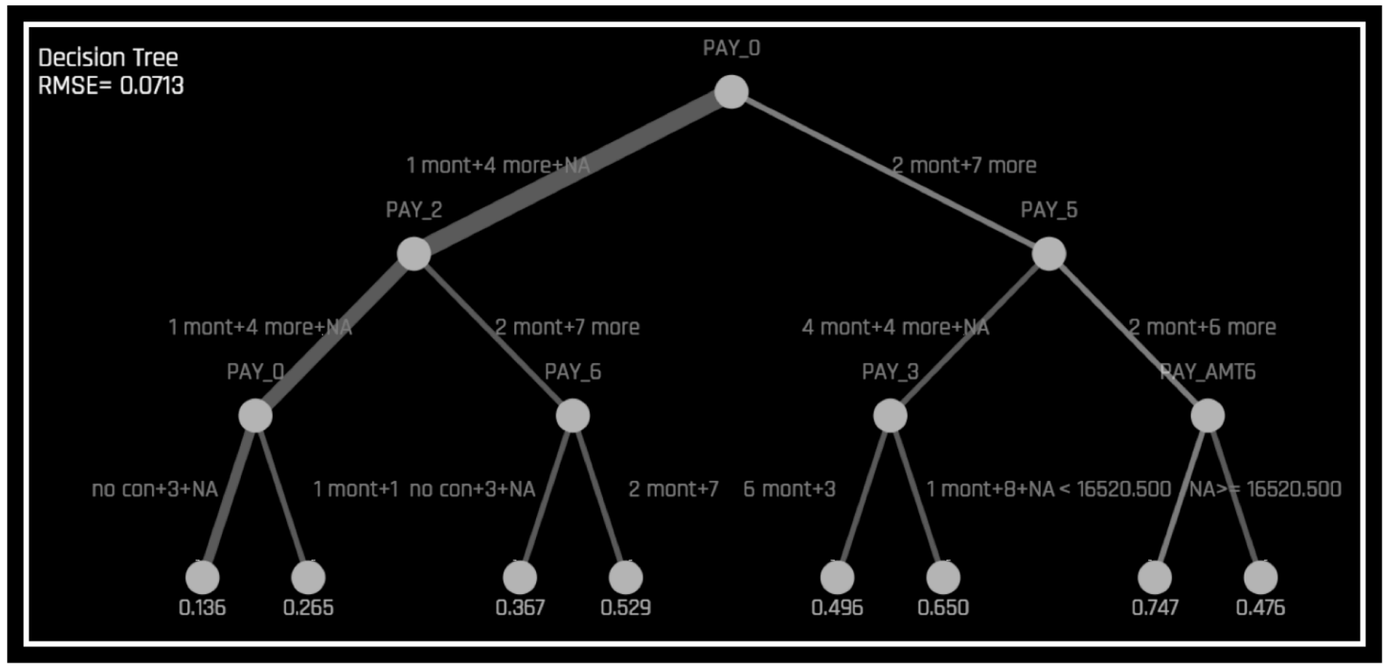


Figure 4: A summarization of a complex model's decision process as represented by a decision tree surrogate.

K-LIME

K-LIME is a variant of the LIME technique proposed by Ribeiro et al [8] With *K*-LIME, local generalized linear model (GLM) surrogates are used to explain the predictions of complex response functions, and local regions are defined by *K* clusters or user-defined segments instead of simulated, perturbed observation samples. Currently in Driverless AI, local regions are segmented with K-means clustering, separating the input training data into K disjoint sets: $\{\mathbf{X}_0 \cup \mathbf{X}_1 \cup \ldots \mathbf{X}_{K-1}\} = \mathbf{X}$.

http://docs.h2o.ai/driverless-ai/latest-stable/docs/booklets/MLIBooklet.pdf

H2O.ai DRIVERLESSAI

Automated Interpretability

http://docs.h2o.ai/driverless-ai/latest-stable/docs/booklets/MLIBooklet.pdf

Local Reason Codes

The baseline prediction for this cluster is: 0.699

For Row: 6 in Cluster: 5

Model Prediction Score: 0.892 k-LIME Prediction Score: 0.953

k-LIME predicts **"survived"** with 93.20% accuracy

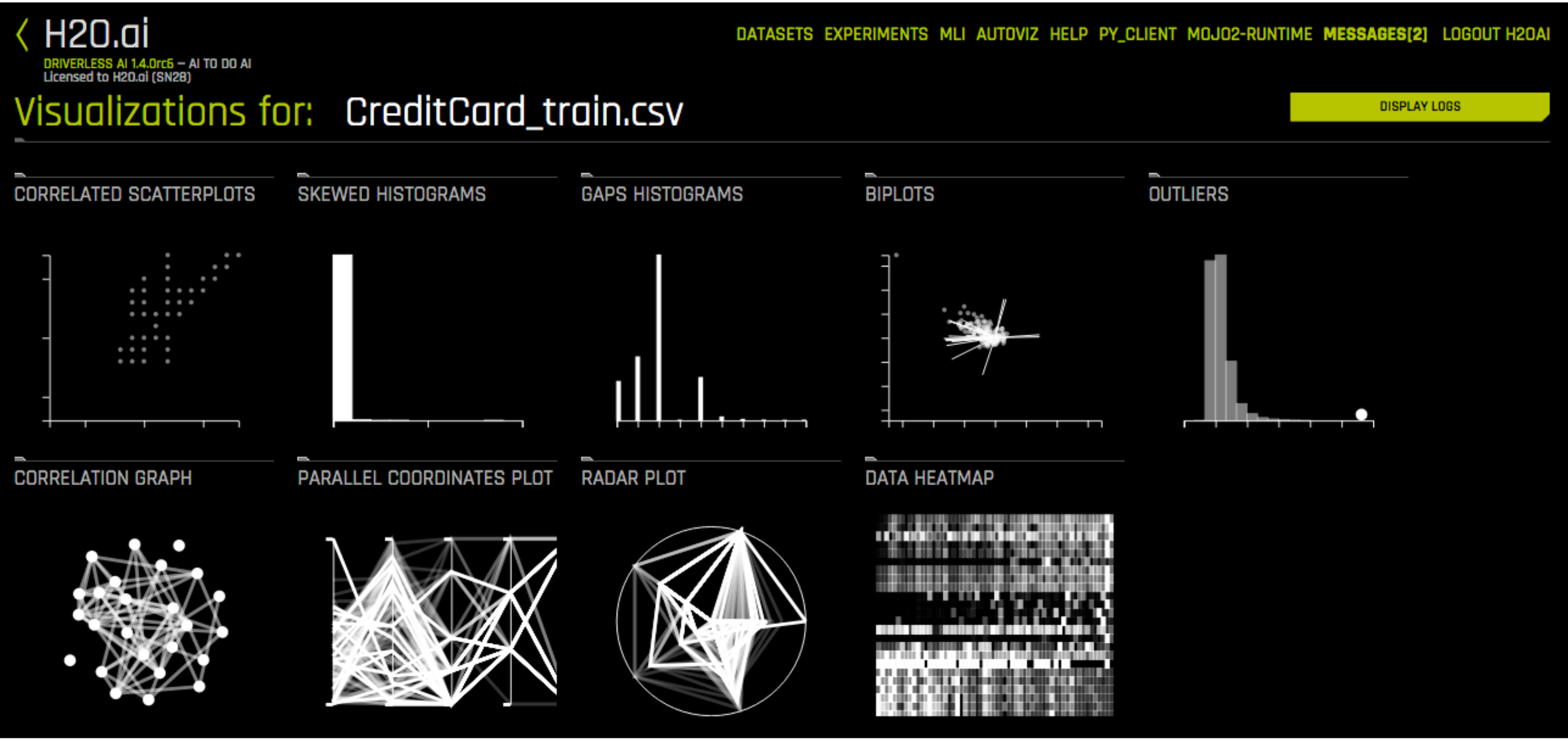
Variable	**with value**	**is associated with "survived" prediction's:**
pclass	1	increase of 1.13%
survived	1	target variable
sex	female	increase of 30.98%
age	63.000	decrease of 15.81%
sibsp	1	increase of 2.77%
fare	77.958	increase of 5.63%

Figure 19: Local reason codes for a single female passenger.

Leave-one-covariate-out (LOCO) provides a mechanism for calculating feature importance values for any model g on a per-observation basis $\mathbf{x}^{(i)}$ by subtracting the model's prediction for an observation of data, $g(\mathbf{x}^{(i)})$, from the model's prediction for that observation of data *without* an input feature X_j of interest, $g(\mathbf{x}^{(i)}_{(-j)}) - g(\mathbf{x}^{(i)})$ [4]. LOCO is a model-agnostic idea, and $g(\mathbf{x}^{(i)}_{(-j)})$ can be

H2O.ai
DRIVERLESSAI

Automated Data Vis



http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/datasets-viewing.html#visualize-datasets

H2O.ai DRIVERLESSAI

Automated Data Vis

- **Clumpy Scatterplots**: Clumpy scatterplots are 2D plots with evident clusters. These clusters are regions of high point density separated from other regions of points. The clusters can have many different shapes and are not necessarily circular. All possible scatterplots based on pairs of features (variables) are examined for clumpiness. The displayed plots are ranked according to the RUNT statistic. Note that the test for clumpiness is described in Hartigan, J. A. and Mohanty, S. (1992), "The RUNT test for multimodality," Journal of Classification, 9, 63–70. The algorithm implemented here is described in Wilkinson, L., Anand, A., and Grossman, R. (2005), "Graph-theoretic Scagnostics," in Proceedings of the IEEE Information Visualization 2005, pp. 157–164.
- **Correlated Scatterplots**: Correlated scatterplots are 2D plots with large values of the squared Pearson correlation coefficient. All possible scatterplots based on pairs of features (variables) are examined for correlations. The displayed plots are ranked according to the correlation. Some of these plots may not look like textbook examples of correlation. The only criterion is that they have a large value of Pearson's r. When modeling with these variables, you may want to leave out variables that are perfectly correlated with others.
- **Unusual Scatterplots**: Unusual scatterplots are 2D plots with features not found in other 2D plots of the data. The algorithm implemented here is described in Wilkinson, L., Anand, A., and Grossman, R. (2005), "Graph-theoretic Scagnostics," in Proceedings of the IEEE Information Visualization 2005, pp. 157–164. Nine scagnostics ("Outlying", "Skewed", "Clumpy", "Sparse", "Striated", "Convex", "Skinny", "Stringy", "Correlated") are computed for all pairs of features. The scagnostics are then examined for outlying values of these scagnostics and the corresponding scatterplots are displayed.
- **Spikey Histograms**: Spikey histograms are histograms with huge spikes. This often indicates an inordinate number of single values (usually zeros) or highly similar values. The measure of "spikeyness" is a bin frequency that is ten times the average frequency of all the bins. You should be careful when modeling (particularly regression models) with spikey variables.
- **Skewed Histograms**: Skewed histograms are ones with especially large skewness (asymmetry). The robust measure of skewness is derived from Groeneveld, R.A. and Meeden, G. (1984), "Measuring Skewness and Kurtosis." The Statistician, 33, 391-399. Highly skewed variables are often candidates for a transformation (e.g., logging) before use in modeling. The histograms in the output are sorted in descending order of skewness.
- **Varying Boxplots**: Varying boxplots reveal unusual variability in a feature across the categories of a categorical variable. The measure of variability is computed from a robust one-way analysis of variance (ANOVA). Sufficiently diverse variables are flagged in the ANOVA. A boxplot is a graphical display of the fractiles of a distribution. The center of the box denotes the median, the edges of a box denote the lower and upper quartiles, and the ends of the "whiskers" denote that range of values. Sometimes outliers occur, in which case the adjacent whisker is shortened to the next lower or upper value. For variables (features) having only a few values, the boxes can be compressed, sometimes into a single horizontal line at the median.
- **Heteroscedastic Boxplots**: Heteroscedastic boxplots reveal unusual variability in a feature across the categories of a categorical variable. Heteroscedasticity is calculated with a Brown-Forsythe test: Brown, M. B. and Forsythe, A. B. (1974), "Robust tests for equality of variances. Journal of the American Statistical Association, 69, 364-367. Plots are ranked according to their heteroscedasticity values. A boxplot is a graphical display of the fractiles of a distribution. The center of the box denotes the median, the edges of a box denote the lower and upper quartiles, and the ends of the "whiskers" denote that range of values. Sometimes outliers occur, in which case the adjacent whisker is shortened to the next lower or upper value. For variables (features) having only a few values, the boxes can be compressed, sometimes into a single horizontal line at the median.
- **Biplots**: A Biplot is an enhanced scatterplot that uses both points and vectors to represent structure simultaneously for rows and columns of a data matrix. Rows are represented as points (scores), and columns are represented as vectors (loadings). The plot is computed from the first two principal components of the correlation matrix of the variables (features). You should look for unusual (non-elliptical) shapes in the points that might reveal outliers or non-normal distributions. And you should look for purple vectors that are well-separated. Overlapping vectors can indicate a high degree of correlation between variables.
- **Outliers**: Variables with anomalous or outlying values are displayed as red points in a dot plot. Dot plots are constructed using an algorithm in Wilkinson, L. (1999). "Dot plots." The American Statistician, 53, 276–281. Not all anomalous points are outliers. Sometimes the algorithm will flag points that lie in an empty region (i.e., they are not near any other points). You should inspect outliers to see if they are miscodings or if they are due to some other mistake. Outliers should ordinarily be eliminated from models only when there is a reasonable explanation for their occurrence.
- **Correlation Graph**: The correlation network graph is constructed from all pairwise squared correlations between variables (features). For continuous-continuous variable pairs, the statistic used is the squared Pearson correlation. For continuous-categorical variable pairs, the statistic is based on the squared intraclass correlation (ICC). This statistic is computed from the mean squares from a one-way analysis of variance (ANOVA). The formula is (MSbetween - MSwithin)/(MSbetween + (k - 1)MSwithin), where k is the number of categories in the categorical variable. For categorical-categorical pairs, the statistic is computed from Cramer's V squared. If the first variable has k1 categories and the second variable has k2 categories, then a k1 x k2 table is created from the joint frequencies of values. From this table, we compute a chi-square statistic. Cramer's V squared statistic is then (chi-square / n) / min(k1,k2), where n is the total of the joint frequencies in the table. Variables with large values of these respective statistics appear near each other in the network diagram. The color scale used for the connecting edges runs from low (blue) to high (red). Variables connected by short red edges tend to be highly correlated.
- **Parallel Coordinates Plot**: A Parallel Coordinate
variables. Each variable has its own vertical axis
the axes for a single observation. If the data cor
their cluster number.
- **Radar Plot**: A Radar Plot is a two-dimensional g
variables. Each variable has its own axis that sta
standardized on each variable between 0 and 1
Each profile, which usually appears in the form
single observation. Multivariate outliers are rep
polar version of the popular Parallel Coordinate
more variables in a single plot.
- **Data Heatmap**: The heatmap graphic is constru
the heatmap represent variables, and columns r
standardized before display so that small values
and columns are permuted via a singular value
similar rows and similar columns are near each
- **Missing Values Heatmap**: The missing values he
transposed data matrix. Rows of the heatmap re
(instances). The data are coded into the values
colored red and nonmissing values are left blank
via a singular value decomposition (SVD) of the
columns are near each other.
- **Gaps Histogram**: The gaps index is computed u
on work by John Tukey. (Wainer, H. and Schach
with gaps can indicate a mixture of two or more
necessarily characterized in the dataset.

http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/datasets-viewing.html#visualize-datasets

H2O.ai DRIVERLESSAI

Automated Feature Engineering

Feature engineering is the secret weapon that advanced data scientists use to extract the most accurate results from algorithms. H2O Driverless AI employs a library of algorithms and feature transformations to automatically engineer new, high value features for a given dataset.

- Filter Transformer

  The Filter Transformer counts each numeric value in the dataset.

- Frequent Transformer

  The Frequent Transformer counts each categorical value in the dataset. This count can be either the raw count or the normalized count.

- Bulk Interactions Transformer

  The Bulk Interactions Transformer will add, divide, multiply, and subtract two columns in the data.

- Cluster Target Encoding Transformer

  Selected columns in the data are clustered, and target encoding is done on the cluster ID.

- Truncated SVD Numeric Transformer

  Truncated SVD trains on a selected numeric of columns in the data. The components of the truncated SVD will be new features.

- Cross Validation Target Encoding

  Cross validation target encoding is done on a categorical column.

- Cross Validation Categorical to Numeric Encoding

  This transformer converts a categorical column to a numeric column. Cross validation target encoding is done on the categorical column.

- Numeric to Categorical Target Encoding Transformer

  This transformer converts a numeric column to categorical by binning. Cross validation target encoding is done on the binned column.

- Numeric Categorical Target Encoding Transformer

  This transformer takes in both numeric and categorical columns and uses them together to make new features.

Dates Transformer

The Dates Transformer retrieves any date values, including:

- Year
- Quarter
- Month
- Day
- Day of year
- Week
- Week day
- Hour
- Minute
- Second

Text Transformer

The Text Transformer transforms a text column using TFIDF (term frequency-inverse document frequency) or count (count of the word). This may be followed by dimensionality reduction using truncated SVD.

Cluster Distance Transformer

Selected columns in the data are clustered, and the distance to a chosen cluster center is calculated.

Weight of Evidence

Creates likelihood type of features using the Weights Of Evidence (WOE) transformation method. The weight of evidence tells the predictive power of an independent variable in relation to the dependent variable, for example, the measurement of good customers in relations to bad customers.

$$WOE = \ln\left(\frac{\text{Distribution of Goods}}{\text{Distribution of Bads}}\right)$$

This only works with a binary target variable. The likelihood needs to be created within a stratified kfold if a fit_transform method is used. More information can be found here: http://ucanalytics.com/blogs/information-value-and-weight-of-evidencebanking-case/.

- Numeric To Categorical Weight of Evidence Transformer

  This transformer converts a numeric column to categorical by binning and the likelihood type of features using the WOE transformation method.

- Lags Transfomer

  Creation of target or feature lags.

- Lags Interaction Transfomer

  Creation of interactions between target/feature lags (lag2 - lag1, for instance

- Lags Aggregates Transformer

  Aggregations of target/feature lags like mean(lag7, lag14, lag21) with support max, median, sum, skew, kurtosis, std.

- Is Holiday Transformer

  Country-based detection for holidays; adds boolean as fetaure

- Numeric to Categorical Weight of Evidence Monotonic Transformer

  The Weight of Evidence or WoE measures the “strength” of a grouping for se and bad risk (default). Monotonic version ensures the bins of values are mono

- Text Linear Model Transformer

  TFIDF features put into linear Model to predict target

- Text CNN Transformer

  CNN/GRU Tensorflow model of text feature to predict target

- One Hot Encoding Transformer

  One-hot encoding of a feature into multiple boolean features

- Sorted Label Encoder Transformer

  Label encoding (numerical assignment of categorical), but sorted numeric ord value

http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/key-features.html#automatic-feature-engineering

NeurIPS | 2018

Thirty-second Conference on Neural Information Processing Systems

AutoML

AutoML1 (2015-2016) | AutoML2 (2017-2018) | AutoML3 (current)

Navigation

Home
How to cite us?
Winning Software
Data
Platform
_____EVENTS_____
Workshop@PRICAI2018
Workshop@ICML2018
Workshop@PAKDD2018
Workshop@ICML2017
Competitions@WCCI2016
Workshop@ICML2016
GPUtrack@ICML2016
Workshop@NIPS2015
Bootcamp@Stanford2015
MLIschool@Petersburg2...
Hackathon@ICML2015
Workshop@ICML2015
Hackathon@ESPCI2015
Workshop@ICML2014
BeatAutoSKLearn (v1)
AutoSKLearn worksheet
_____LINKS_____
CiML
Fact sheet
FORUM
ChaLearn
Sitemap

2
days since
NIPS compet. wshp

Home

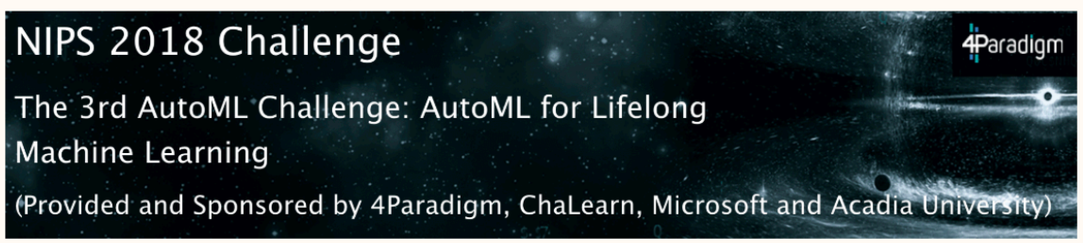
AutoML 3

Winners of the final AutoML phrase (see full-list):

First Place autodidact.ai
Second Place Meta_Learners
Third Place GrandMasters

Winners of Feedback phrase (see full-list):

First Place deepsmart
Second Place HANLAB
Third Place Fong



The competition is over, but there will be a re-match soon in conjunction with PAKDD 2019 and IJCNN 2019!

1) **NeurIPS competitions book.** The competition chairs are editing a volume of the Springer CiML series (https://www.springer.com/series/15602). Any participant of the challenge is eligible to submit a chapter, please note this chapters will be subject to a review process. Deadline: late January, more information during NeurIPS.

News:

Upcoming Spring 2019
AutoDL challenge [SIGN UP]

December 7, 2019
Competition Workshop at NeurIPS 2019

August 28, 2018
We had a workshop at PRICAI 2018.

July 14-15 2018:
We had a nice workshop at ICML 2018.

March 2018:
Our next competition on Life Long ML is accepted to NIPS 2018.

June 21, 2016: Microsoft published a BLOG post on AutoML 1.

We re-implemented old challenges in Codalab [NIPS 2003][KDDcup 2009, small, large] and created one for [MNIST data]

AutoML2

Only 2 phases:

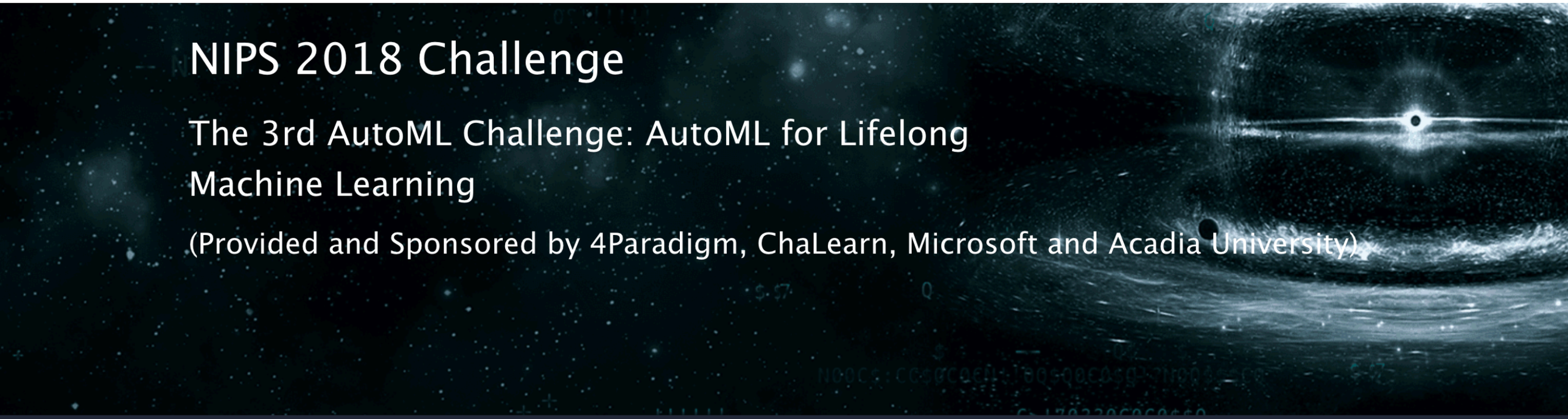
1. **Feed-back phase:** You get to practice (with code or result submission) on 5 datasets and get immediate feed-back on a leaderboard.
2. **AutoML phase:** The code gets blind tested on 5 new datasets.

The setting is similar to AutoML1, so you can get a head start by studying what was done.

AutoML1

The First ChaLearn AutoML challenge included 5 rounds, 30 datasets (available for download), and $30,000 in prizes. The goal was to design the perfect machine learning "black box" capable of performing all model selection and hyper-parameter tuning without any human intervention. Post-challenge submissions can be made on the **clone websites**:

- **Round 0**: Sample data (practice).
- **Round 1**: Novice level. binary classification problems, dense data (practice).
- **Round 2**: Intermediate level. Multi-class classification, dense data (practice).
- **Round 3**: Advanced level. Multi-class and multi-label classification, dense and sparse data (practice).
- **Round 4**: Expert. Regression + all the other cases seen so far.
- **Round 5**: Master. Final blind testing. We are not releasing the data, which will be used in further benchmarks.



NIPS 2018 Challenge

The 3rd AutoML Challenge: AutoML for Lifelong Machine Learning

(Provided and Sponsored by 4Paradigm, ChaLearn, Microsoft and Acadia University)

Overview

The final winners of the competition, see full-list:

First Place autodidact.ai

Second Place Meta_Learners

Third Place GrandMasters

Top-ranked participants (feedback phase), see full-list:

First Place deepsmart

Second Place HANLAB

Third Place Fong

The competition has been launched at CodaLab, please follow the link to participate:https://competitions.codalab.org/competitions/19836

In many real-world machine learning applications, AutoML is strongly needed due to the limited machine learning expertise of developers. Moreover, batches of data in many real-world applications may be arriving daily, weekly, monthly, or yearly, for instance, and the data distributions

will be used. Compared with previous AutoML competitions(http://automl.chalearn.org/), the focus of this competition is on **drifting concepts**, getting away from the simpler i.i.d. cases. Participants are invited to design a computer program capable of autonomously (without any human intervention) developing predictive models that are trained and evaluated in a lifelong machine learning setting.
Although the scenario is fairly standard, this challenge introduces the following difficulties:

• **Algorithm scalability**. We will provide datasets that are 10-100 times larger than in previous challenges we organized.

• **Varied feature types**. Varied feature types will be included (continuous, binary, ordinal, categorical, multi-value categorical, temporal). Categorical variables with a large number of values following a power law will be included.

• **Concept drift**. The data distribution is slowly changing over time.

• **Lifelong setting**. All datasets included in this competition are chronologically splitted into 10 batches, meaning that instance batches in all datasets are chronologically ordered (note that instances in one batch are not guaranteed to be chronologically ordered). The algorithms will be tested for their capability of adapting to changes in data distribution by exposing them to successive test batches chronologically ordered. After testing, the labels will be revealed to the learning machines and incorporated in the training data.
There're two phases of the competition:

The **Feedback phase** is a phase with **code submission**, participants can practice on 5 datasets that are of similar nature as the datasets of the second phase. Participants can make a limited number of submissions. Participants can download the labeled training data and the unlabeled test set. So participants can prepare their code submission at home and submit it later. The LAST code submission will be forwarded to the next phase for final testing.

The **AutoML phase** is the blind test phase with **no submission**. The last submission of the previous phase is blind tested on 5 new datasets. Participant's code will be trained and tested automatically, without human intervention. The final score will be evaluated by the result of the blind testing.

Data

Data Format

For each instance, we have the following 4 types of features split by blank in our instance files, i.e.,

- *Categorical* Feature: an integer describing which category the instance belongs to.
- *Numerical* Feature: a real value.
- *Multi-value Categorical* Feature: a set of integers, split by the comma. The size of the set is not fixed and can be different for each instance. For example, topics of an article, words in a title, items bought by a user and so on.
- *Time* Feature: an integer describing time information.

Note: Categorical/Multi-value Categorical features with a large number of values following a power law might be included.

Feedback Phase Public Datasets

5 public datasets are released, including their first 5 batches, another 5 batches are kept private.

DATASET	Budget	#Cat	#Num	#MVC	#Time	#Feature	#Instance
A	3600	51	23	6	2	82	~10 Million
B	600	17	7	1	0	25	~1.9 Million
C	1200	44	20	9	6	79	~2 Million
D	600	17	54	1	4	76	~1.5 Million
E	1800	25	6	1	2	34	~17 Million

Budget=Time budget(seconds). #Cat=Number of categorical features. #Num=Number of numerical features. #MVC=Number of multi-value categorical features. #Time=Number of time features. #Feature=Total number of features. #Instance=Total number of instances for all 10 batches.

Basic Tips

Here, we provide some basic tips for dealing with large datasets:

• Subsampling/multi-fidelity AutoML approaches might be needed for these datasets.

• Incremental learning might be needed for these datasets.

Some basic tips for handling difficult features:

• One-hot encoding for Categorical features.

• Hashing tricks might be used for Categorical and Multi-value Categorical features.

• Normalization triks for Numerical features.

• If the number of elements in a Categorical/Multi-value Categorical feature is too large, instead of hashing tricks, one might compute moving frequencies of these elements and always keep top ones.

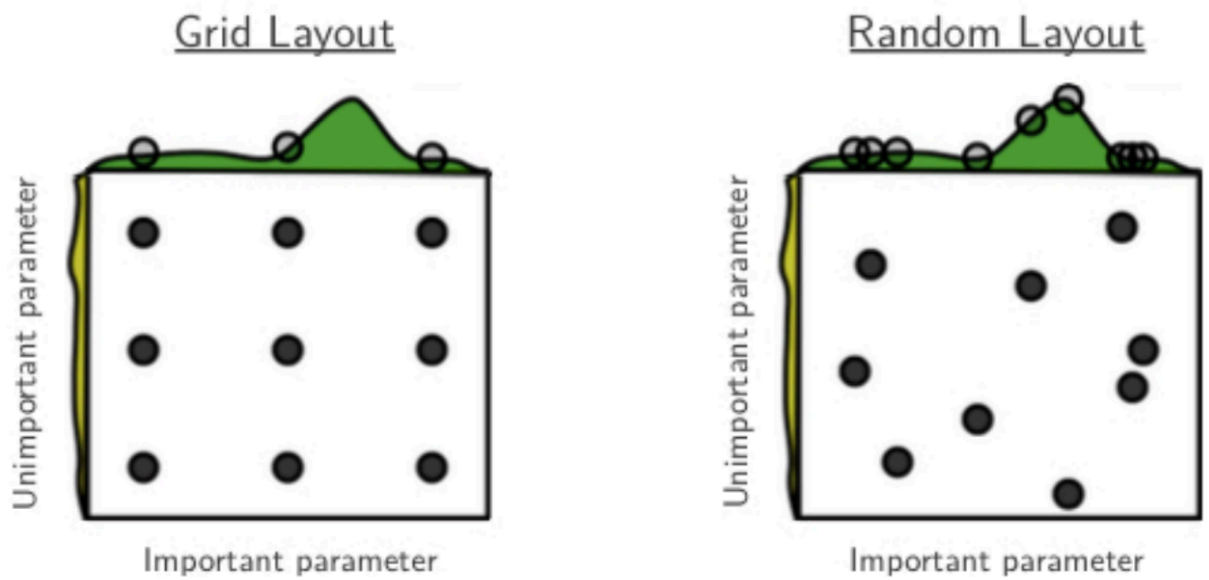
• For Time features, one might minus a fixed value from the original feature. When multiple Time features present, one might construct new features based on the difference of these Time features.

• For missing features, i.e., NaN, one might set it as a default value or replace it with another valid value indicating it is missing.

Note: The competition focuses on automatically combating with the drifting concepts. The processing of features might need to be adaptive over time. Automatic feature generation and selection methods or deep learning approaches might be important for these datasets.

Option 3: Random search?

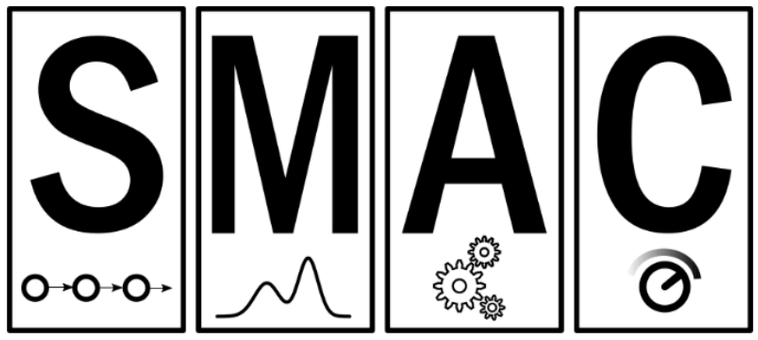
We can sample the space uniformly [Bergstra and Bengio 2012]



Better than grid search in various senses but still expensive to guarantee good coverage.

AutoML ...

provides methods and processes to make Machine Learning available for non-Machine Learning experts, to improve efficiency of Machine Learning and to accelerate research on Machine Learning. Machine learning (ML) has achieved considerable successes in recent years and an ever-growing number of disciplines rely on it. However, this success crucially relies on human machine learning experts to perform manual tasks. As the complexity of these tasks is often beyond non-ML-experts, the rapid growth of machine learning applications has created a demand for off-the-shelf machine learning methods that can be used easily and without expert knowledge. We call the resulting research area that targets progressive automation of machine learning AutoML.



SMAC

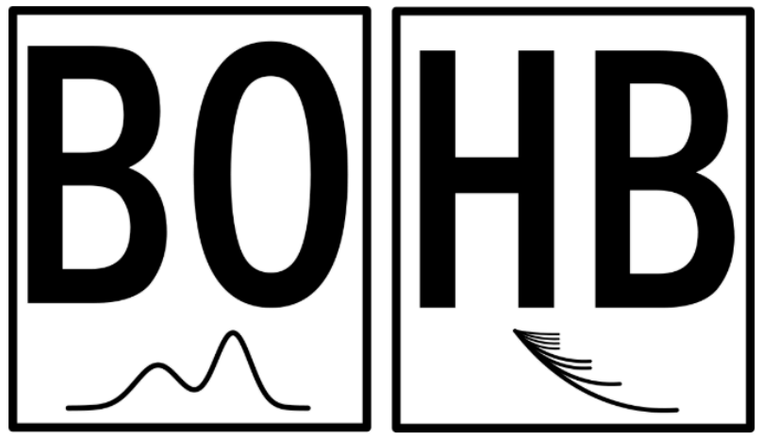
Sequential Model-based Algorithm Configuration is a state-of-the-art tool to optimize the performance of your algorithm by determining a well-performing parameter setting.

Read More »

Auto-Sklearn

Auto-Sklearn is an automated machine learning toolkit to automatically determine a well-performing machine learning pipeline. It is a drop-in replacement for a scikit-learn estimator

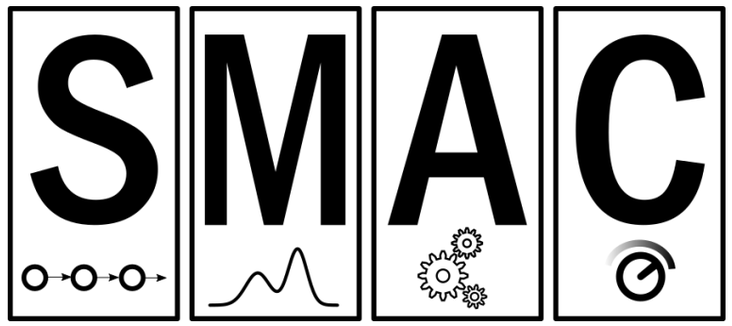
Read More »



BOHB

BOHB combines the benefits of both Bayesian Optimization and HyperBand, in order to achieve the best of both worlds: strong anytime performance and fast convergence to optimal configurations.

Read More »



Sequential Model-based Algorithm Configuration

AutoML systems

Throughout recent years several off-the-shelf packages have been developed which provide automated machine learning. While there are more packages than the one listed below, we restrict ourselves to a subset of the most well-known ones:

- AutoWEKA is an approach for the simultaneous selection of a machine learning algorithm and its hyperparameters; combined with the WEKA package it automatically yields good models for a wide variety of data sets.
- Auto-sklearn is an extension of AutoWEKA using the Python library scikit-learn which is a drop-in replacement for regular scikit-learn classifiers and regressors.
- TPOT is a data-science assistant which optimizes machine learning pipelines using genetic programming.
- H2O AutoML provides automated model selection and ensembling for the H2O machine learning and data analytics platform.
- TransmogrifAI is an AutoML library running on top of Spark.
- MLBoX is an AutoML library with three components: preprocessing, optimisation and prediction.

Keras + Hyperopt: A very simple wrapper for convenient hyperparameter optimization http://maxpumperla.github.io/hyperas/

Hyperas

A very simple convenience wrapper around hyperopt for fast prototyping with keras models. Hyperas lets you use the power of hyperopt without having to learn the syntax of it. Instead, just define your keras model as you are used to, but use a simple template notation to define hyper-parameter ranges to tune.

Installation

```
pip install hyperas
```

Quick start

Assume you have data generated as such

```
def data():
    x_train = np.zeros(100)
    x_test = np.zeros(100)
    y_train = np.zeros(100)
    y_test = np.zeros(100)
    return x_train, y_train, x_test, y_test
```

and an existing keras model like the following

```
def create_model(x_train, y_train, x_test, y_test):
    model = Sequential()
    model.add(Dense(512, input_shape=(784,)))
    model.add(Activation('relu'))
    model.add(Dropout(0.2))
```

https://github.com/maxpumperla/hyperas

DEvol - Deep Neural Network Evolution

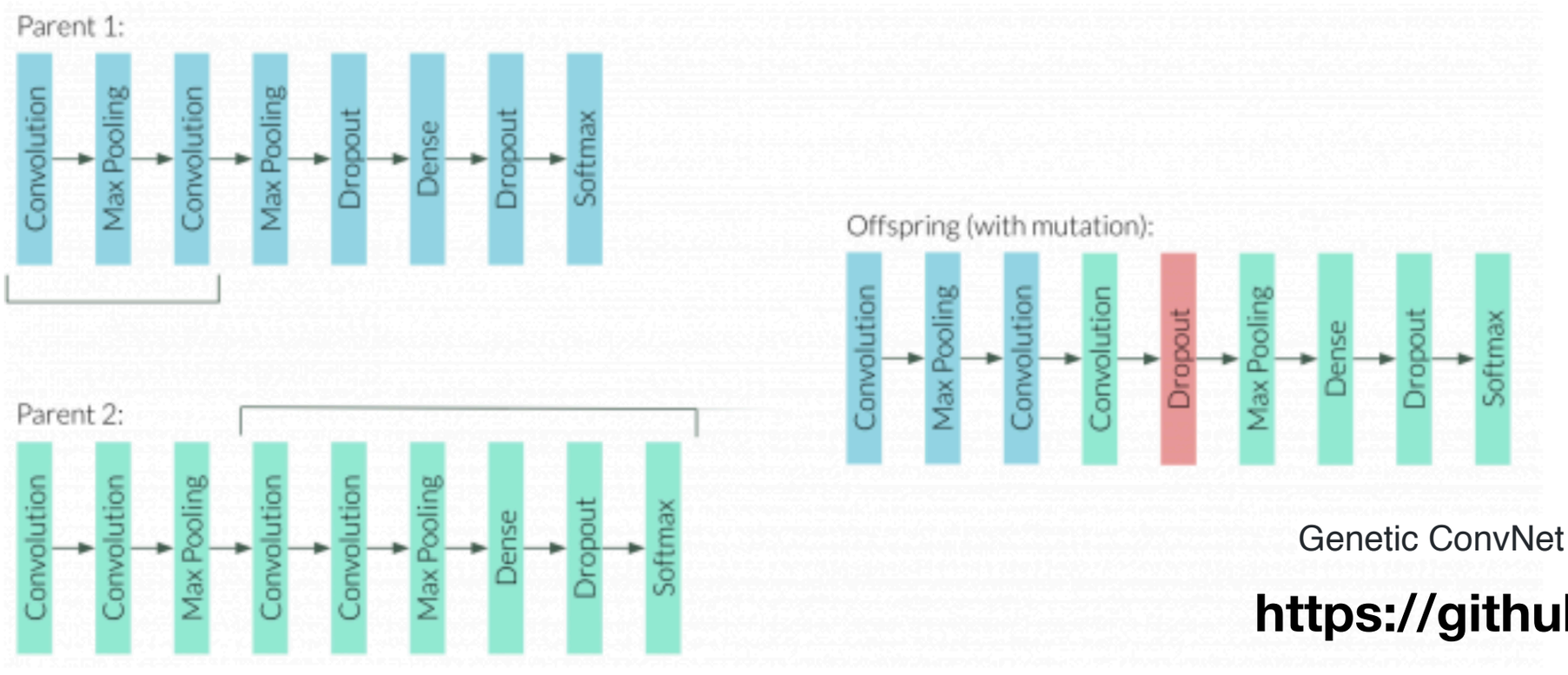
DEvol (DeepEvolution) is a basic proof of concept for genetic architecture search in Keras. The current setup is designed for classification problems, though this could be extended to include any other output type as well.

See `example/demo.ipynb` for a simple example.

Evolution

Each model is represented as fixed-width genome encoding information about the network's structure. In the current setup, a model contains a number of convolutional layers, a number of dense layers, and an optimizer. The convolutional layers can be evolved to include varying numbers of feature maps, different activation functions, varying proportions of dropout, and whether to perform batch normalization and/or max pooling. The same options are available for the dense layers with the exception of max pooling. The complexity of these models could easily be extended beyond these capabilities to include any parameters included in Keras, allowing the creation of more complex architectures.

Below is a highly simplified visualization of how genetic crossover might take place between two models.



Genetic ConvNet architecture search with Keras

https://github.com/joeddav/devol

Hyperparameter Scanning and Optimization for Keras

Tweet

build passing coverage 78%

Talos is a solution that helps finding hyperparameter configurations for Keras models. To perform hyperparameter optimization with Talos, there is no need to learn any new syntax, or change anything in the way Keras models are created. Keras functionality is fully exposed, and any parameter can be included in the scans.

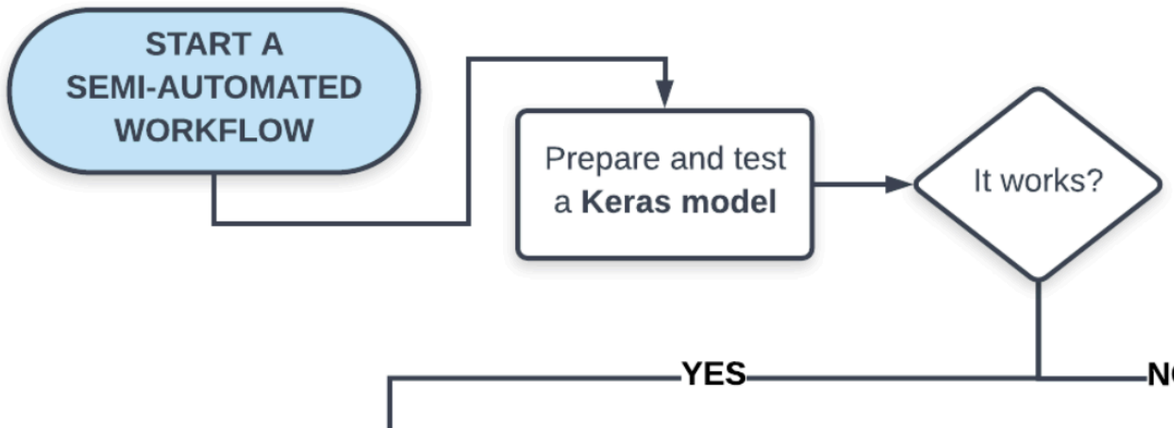
See a brief | concise | comprehensive example Notebook

Read the User Manual

Read a Report on Hyperparameter Optimization with Keras

Read the Roadmap

Install `pip install talos`



https://github.com/autonomio/talos

Welcome to Talos! You can use Talos for hyperparameter optimization with Keras models. Talos allows you to use Keras models exactly as you would otherwise, and is built for and tested on Python 2 and 3. Talos incorporates grid, random, and probabilistic hyperparameter optimization strategies, with focus on maximizing the flexibility, efficiency, and result of random strategy. Talos users benefit from access to pseudo, quasi, true, and quantum random methods. Talos provides a fully automated POD (Prepare, Optimize, Deploy) pipeline that consistently yields state-of-the-art prediction results in a wide-range of prediction problems. Talos is maintained by a non-profit foundation with 501(c)(3) status. The code is available on Github.

Random Optimizers

Random search is the recommended optimization strategy in Talos. This is invoked through the 'grid_downsample' argument in Scan() with a floating point value in the experiment options. For example, to randomly pick 10% of the permutations, a grid_downsample value of 0.1 is used.

Several pseudo, quasi, true, and quantum random methods are provided for random searches. These are controlled through the 'random_method' argument in Scan().

- Quantum
- Ambient sound
- Halton sequence
- Sobol sequence
- Korobov matrix
- Latin hypercube
- Latin hypercube with Sudoku style constraint
- Improved Latin hypercube (very slow)
- Uniform Mersenne
- Uniform cryptographically secure

Each random method results in a different degree of discrepancy; whereas uniform random methods tend to have higher discrepancy, as does quantum and ambient sound, hypercube and Korobov have lower.

```
Scan(x, y,
     params=p,
     model=input_model,
     grid_downsample=.1)
```

```
Scan(x, y,
     params=p,
     model=input_model,
     grid_downsample=.1,
     random_method=quantum)
```

Probabilistic reduction

The probabilistic reducers can be used together with a Grid search, or together with any of the Random methods. The reducer makes a stop between set number of rounds i.e. 'reduction_interval' and uses a probabilistic method to remove poorly performing parameter configurations from the remaining search space.

Several parameters are related with this:

Reduction Method: Currently only one reduction method is supported 'correlation'.

Reduction Interval: The number of rounds between each reduction stop.

Reduction Window: The number of rounds to look back for input signals (e.g. when reduction_window is 50, the results of the last 50 rounds are used for inference)

NeurIPS | 2018

Thirty-second Conference on Neural Information Processing Systems

Transfer Learning with Neural AutoML

Catherine Wong
MIT
catwong@mit.edu

Neil Houlsby
Google Brain
neilhoulsby@google.com

Yifeng Lu
Google Brain
yifenglu@google.com

Andrea Gesmundo
Google Brain
agesmundo@google.com

Abstract

We reduce the computational cost of Neural AutoML with transfer learning. AutoML relieves human effort by automating the design of ML algorithms. Neural AutoML has become popular for the design of deep learning architectures, however, this method has a high computation cost. To address this we propose Transfer Neural AutoML that uses knowledge from prior tasks to speed up network design. We extend RL-based architecture search methods to support parallel training on multiple tasks and then transfer the search strategy to new tasks. On language and image classification tasks, Transfer Neural AutoML reduces convergence time over single-task training by over an order of magnitude on many tasks.

https://papers.nips.cc/paper/8056-transfer-learning-with-neural-automl

NeurIPS | 2018

Thirty-second Conference on Neural Information Processing Systems

Transfer Learning with Neural AutoML

We propose Multitask Neural AutoML, that searches for model on multiple tasks simultaneously. It requires defining a generic search space that is shared across tasks. Many deep learning models require the same common design decisions, such as choice of network depth, learning rate, and number of training iterations. By defining a generic search space that contains common architecture and hyperparameter choices, the controller can generate a wide range of models applicable to many common problems. Multitask training allows the controller to learn a broadly applicable prior over the search space by observing shared behaviour across tasks. The proposed multitask controller has two key features: learned task representations, and advantage normalization.

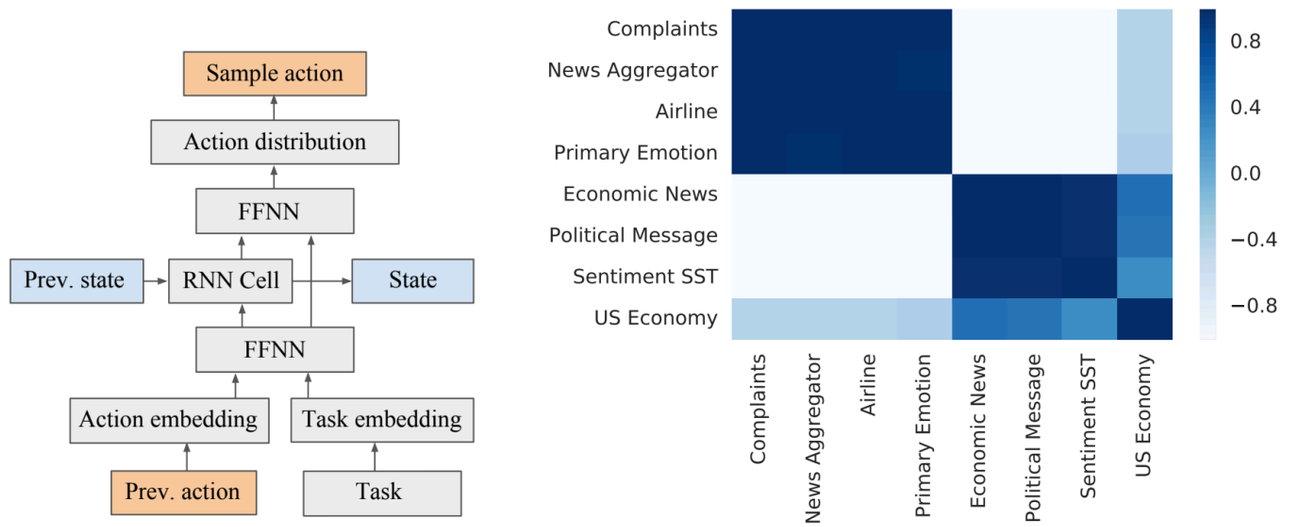


Figure 1: *Left:* A single time step of the recurrent multitask AutoML controller, in which a single action is taken. The task embedding is concatenated with the embedding of the action sampled at the previous timestep and passed into the controller RNN. All parameters, other than the task embeddings, are shared across tasks. *Right:* Cosine similarity between the task embeddings learned by the multitask neural AutoML model.

https://papers.nips.cc/paper/8056-t

NeurIPS | 2018

Thirty-second Conference on Neural Information Processing Systems

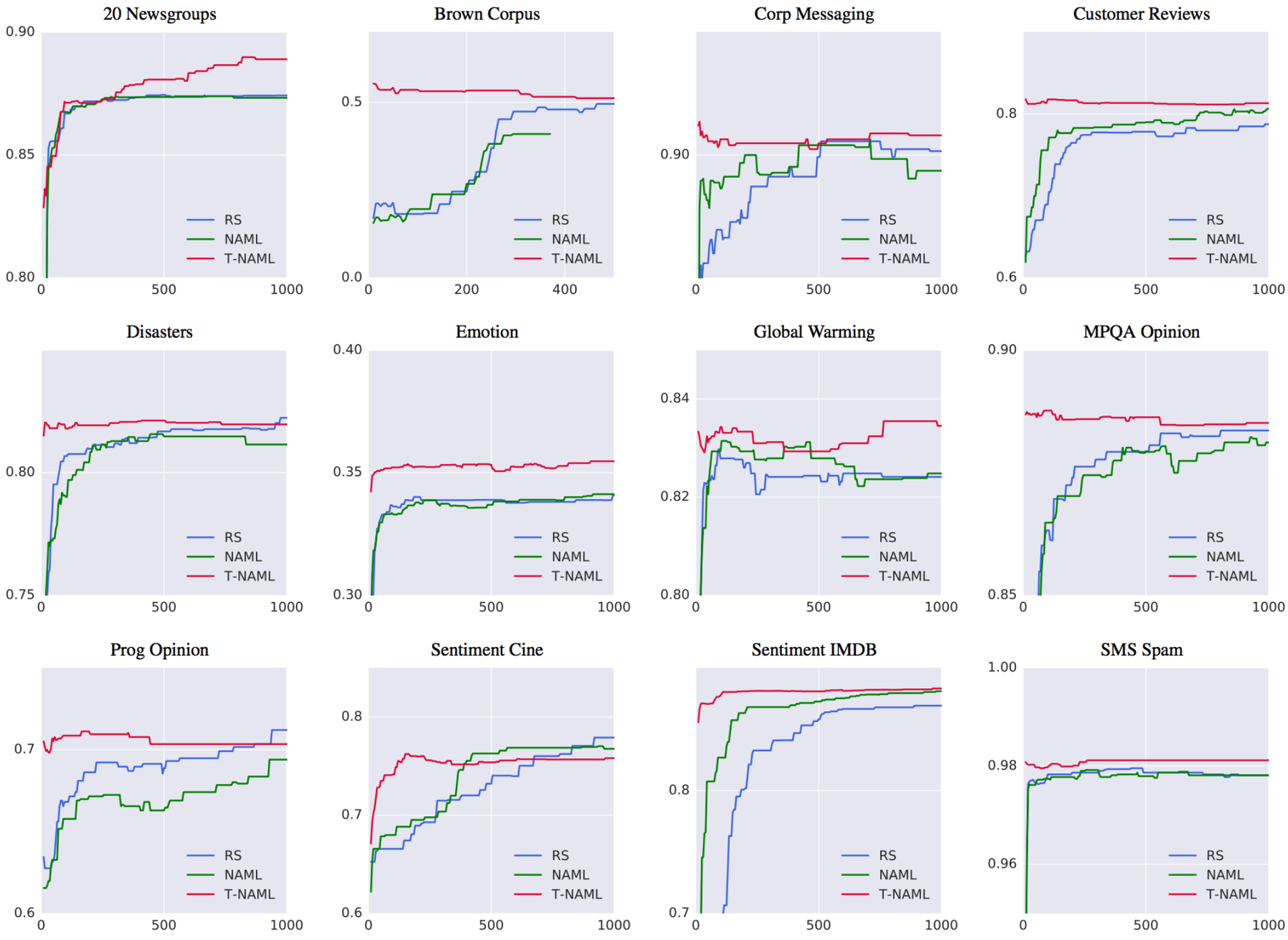


Figure 2: Learning curves for Random Search (RS), single-task Neural AutoML (NAML), and Transfer (T-NAML). *x-axis*: Number of trials (child model evaluations). *y-axis*: Average test set accuracy of the 10 models with best validation accuracy (test accuracy-top10) found up to each trial.

https://papers.nips.cc/paper/8056-transfer-learning-with-neural-automl.pdf

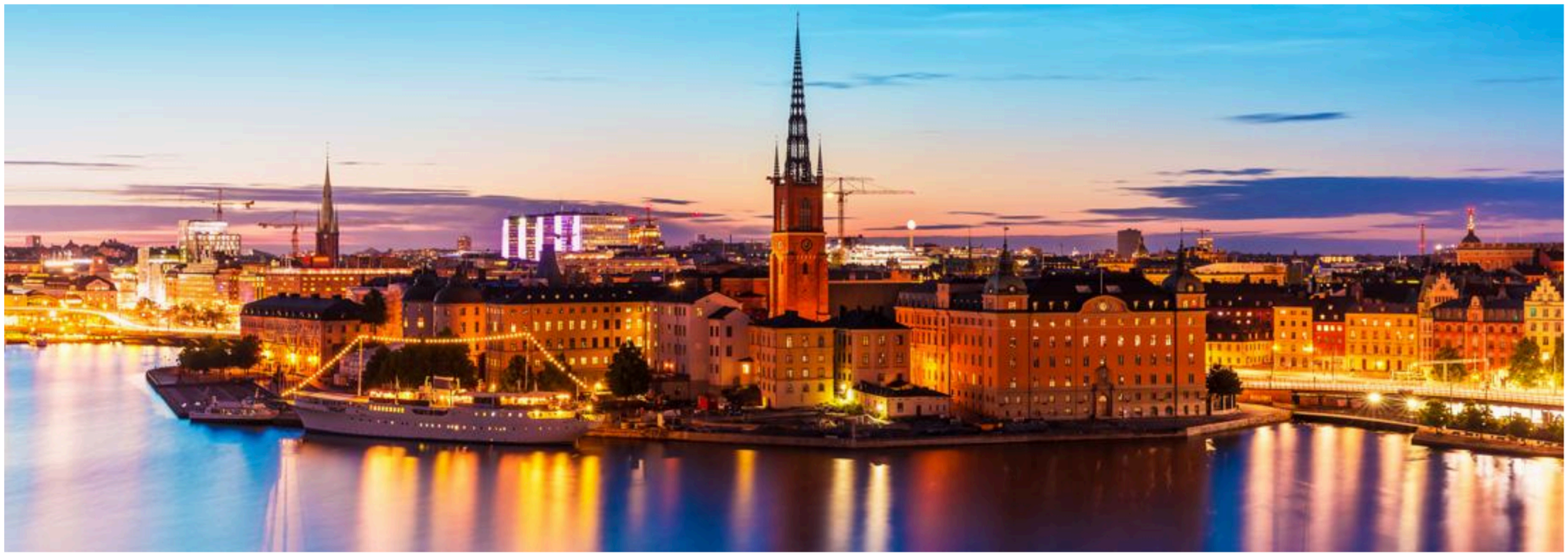
Home

International Workshop on Automatic Machine Learning

July 14th, 2017

Collocated with the Federated AI Meeting (ICML, IJCAI, AMAS, and ICCBR), Stockholm, Sweden

Ask a question for the panel discussion



https://sites.google.com/site/automl2018icml/accepted-papers

Towards AutoML in the presence of Drift: first results

Jorge G. Madrid JORGEGUS.93@GMAIL.COM
INAOE, Mexico
Hugo Jair Escalante HUGOJAIR@INAOEP.MX
INAOE, Mexico
Eduardo F. Morales EMORALES@INAOEP.MX
INAOE, Mexico
Wei-Wei Tu TUWWCN@GMAIL.COM
4Paradigm Inc., China
Yang Yu YUY@NJU.EDU.CN
Nanjing University, China
Lisheng Sun-Hosoya CECILE829@GMAIL.COM
UPSud, U. Paris-Saclay, France
Isabelle Guyon GUYON@CHALEARN.ORG
UPSud/INRIA, U. Paris-Saclay, France and ChaLearn, USA
Michele Sebag MICHELE.SEBAG@LRI.FR
CNRS, U. Paris-Saclay, France

Abstract

Research progress in AutoML has lead to state of the art solutions that can cope quite well with supervised learning task, e.g., classification with AutoSklearn. However, so far these systems do not take into account the changing nature of evolving data over time (i.e., they still assume i.i.d. data); even when this sort of domains are increasi applications (e.g., spam filtering, user preferences, etc.). We describe velop an AutoML solution for scenarios in which data distribution cha over time and in which the problem is approached in a lifelong learnin Auto-Sklearn with sound and intuitive mechanisms that allow it to c problems. The extended Auto-Sklearn is combined with concept drift that allow it to automatically determine when the initial models hav report experimental results in benchmark data from AutoML compet this scenario. Results demonstrate the effectiveness of the proposed r

Keywords: AutoML, Life Long Machine Learning, Concept Drift, A

In the rest of this section we describe the components of the proposed method, namely: the drift detector and the proposed adaptation mechanisms.

Data: $D(X, y)$ examples
Take a batch D'_t of size n, $D'_t(X', y') \in D(X, y)$; $T_t \leftarrow$ learn a model with auto-sklearn using D'_t
while *there is data in D* **do**
  Take next batch D'_{t+1}; $\hat{y} \leftarrow$ Make predictions with T_t; **for** $y_j \in \hat{y}$ **do**
    drift_detected = Detector($y_j == y'_j \in y'$) //drift will be detected with model performance
  end
  if *drif_detected* **then**
    $T_{t+1} = adapt(T, D'_{t+1})$; Detector.reset(); $t = t + 1$
  else
    $T_{t+1} = T_t$; $t = t + 1$
  end
end

Algorithm 1: Drift adaption schema for Auto-sklearn

3.2.1. Drift detector

Meta-Learning Transferable Active Learning Policies by Deep Reinforcement Learning

Kunkun Pang K.PANG@ED.AC.UK
Institute of Perception, Action and Behaviour, University of Edinburgh

Mingzhi Dong MINGZHI.DONG.13@UCL.AC.UK
Department of Statistics, University College London

Yang Wu YANGWU@RSC.NAIST.JP
Nara Institute of Science and Technology

Timothy M. Hospedales T.HOSPEDALES@ED.AC.UK
Institute of Perception, Action and Behaviour, University of Edinburgh

Abstract

Active learning (AL) aims to enable training high performance classifiers with low annotation cost by predicting which subset of unlabelled instances would be most beneficial to label. The importance of AL has motivated extensive research, proposing a wide variety of manually designed AL algorithms with diverse theoretical and intuitive motivations. In contrast to this body of research, we propose to treat active learning algorithm design as a meta-learning problem and learn the best criterion from data. We model an active learning algorithm as a deep neural network that inputs the base learner state and the unlabelled point set and predicts the best point to annotate next. Training this active query policy network with reinforcement learning, produces the best non-myopic policy for a given dataset. The key challenge in achieving a general solution to AL then becomes that of learner generalisation, particularly across heterogeneous datasets. We propose a multi-task dataset-embedding approach that allows dataset-agnostic active learners to be trained. Our evaluation shows that AL algorithms trained in this way can directly generalise across diverse problems.

Evolutionary Algorithms and Reinforcement Learning: A Comparative Case Study for Architecture Search

Esteban Real EREAL@GOOGLE.COM
Alok Aggarwal ALOKA@GOOGLE.COM
Yanping Huang HUANGYP@GOOGLE.COM
Quoc V. Le QVL@GOOGLE.COM
Google Brain, Mountain View, California, USA

Abstract

The effort devoted to hand-crafting image classifiers has motivated the use of architecture search to discover them automatically. Reinforcement learning and evolutionary algorithms have both shown promise for this purpose, but a controlled comparison has not yet been done. This study rigorously compares an evolutionary algorithm to a highly successful reinforcement learning baseline. Using the same hardware, compute effort and training code, we conduct repeated experiments side-by-side, exploring different datasets, search spaces and compute scales. We show that evolution consistently produces models with similar or higher accuracy, across a variety of contexts without need for re-tuning parameters. Moreover, evolution exhibits considerably better performance than reinforcement learning at early search stages, suggesting it may be the better choice when fewer compute resources are available. From these evolution experiments, a new architecture emerges, named *AmoebaNet-C*, which we present. It reaches a state-of-the-art top-1 accuracy on ImageNet of 83.1%.

Keywords: evolutionary algorithms, neural networks, neuro-evolution, neuroevolution, evolution, genetic algorithms, image classification, reinforcement learning, RL, architecture search, NAS, NASNet, meta learning, meta-learning, learn-to-learn, L2L

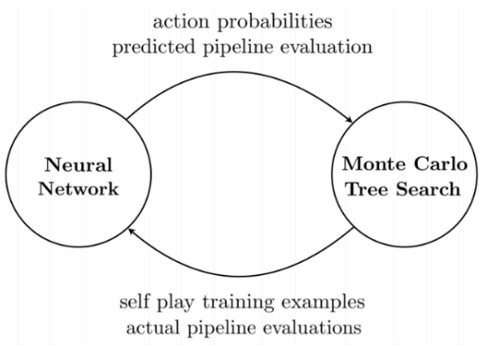
AlphaD3M: Machine Learning Pipeline Synthesis

Iddo Drori IDRORI@NYU.EDU
Yamuna Krishnamurthy YAMUNA@NYU.EDU
Remi Rampin REMI.RAMPIN@NYU.EDU
Raoni de Paula Lourenco RAONI@NYU.EDU
Jorge Piazentin Ono JORGEHPO@NYU.EDU
Kyunghyun Cho KYUNGHYUN.CHO@NYU.EDU
Claudio Silva CSILVA@NYU.EDU
Juliana Freire JULIANA.FREIRE@NYU.EDU
New York University, Tandon School of Engineering and Center for Data Science

Abstract

We introduce AlphaD3M, an automatic machine learni
reinforcement learning using sequence models with se
operations performed over machine learning pipeline
We compare AlphaD3M with state-of-the-art AutoML
and TPOT, on OpenML datasets. AlphaD3M achie
being an order of magnitude faster, reducing comput
is explainable by design.

Keywords: AutoML, Expert Iteration, Sequence Mc



	AlphaZero	AlphaD3M
Game	Go, chess	AutoML
Unit	piece	pipeline primitive
State	configuration	meta data, task, pipeline
Action	move	insert, delete, replace
Reward	win, lose, draw	pipeline performance

Figure 1: AlphaD3M iterative improvement (left); AlphaD3M game representation (right).

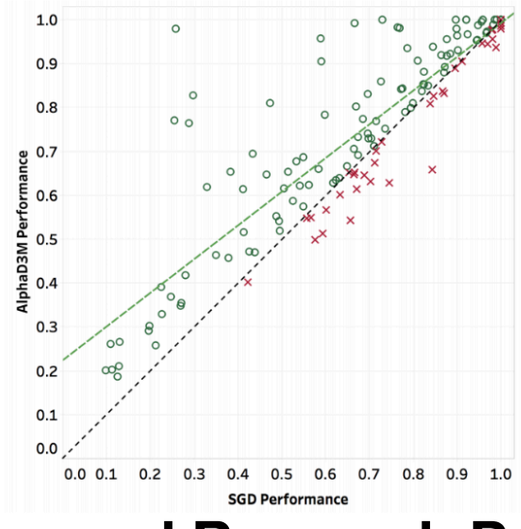


Figure 2: AlphaD3M vs. SGD performance on validation dataset for OpenML datasets

This work has been supported in part by the Defense Advanced Research Projects Agency (DARPA) Data-Driven Discovery of Models (D3M) Program.

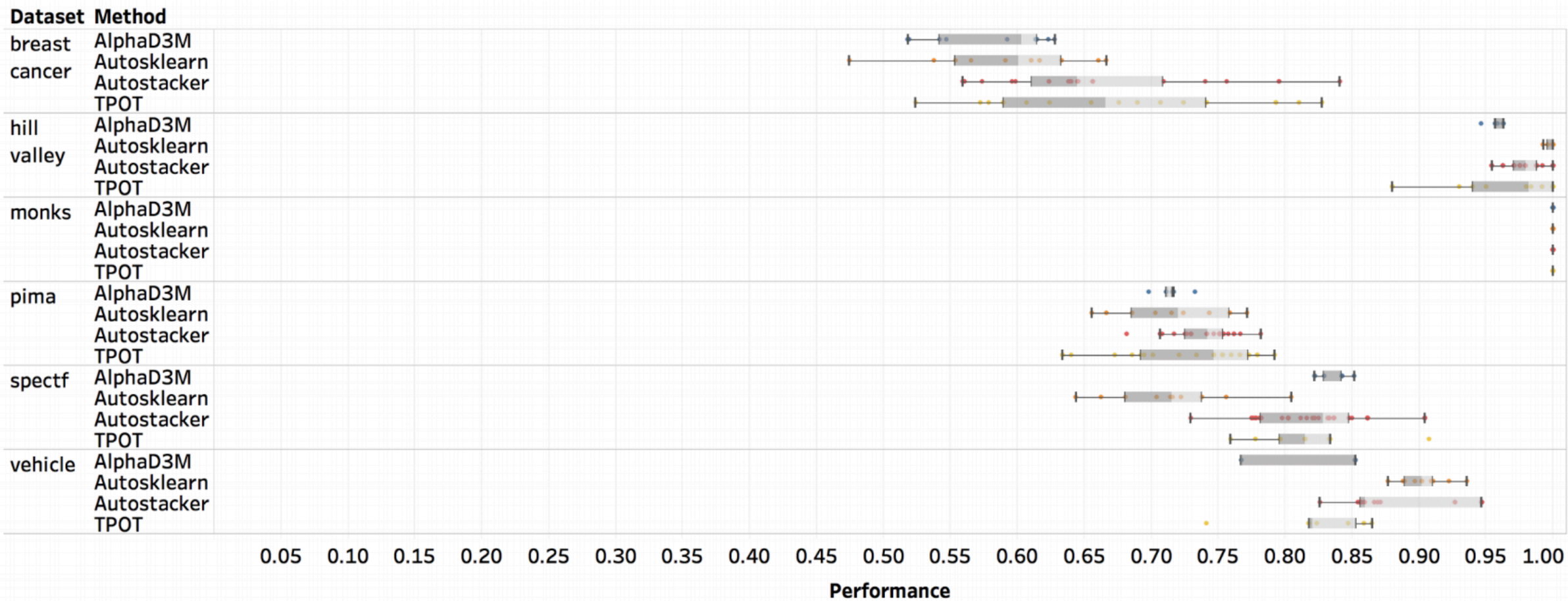
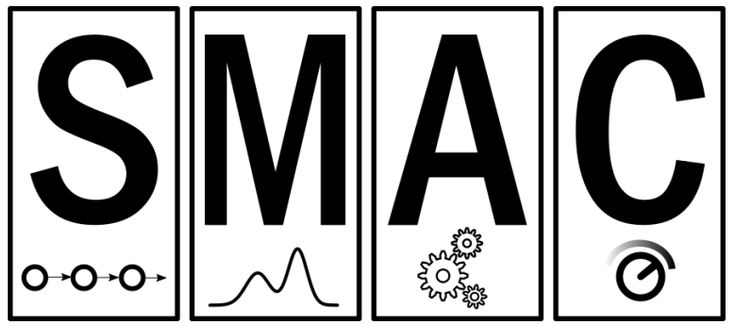


Figure 4: Comparing between performance of AutoML methods on OpenML datasets.

Table 1: Running time comparison (in seconds and speedup factors).

Dataset/Method	TPOT	Autostacker	AlphaD3M	Speedup vs TPOT	Speedup vs AS
breast cancer	3366	1883	460	7.3	4
hill valley	17951	8411	556	32.2	15.1
monks	1517	1532	348	4.3	4.3
pima	5305	1940	619	8.5	3.1
spectf	4191	1673	522	8	3.2
vehicle	16795	4010	531	31.6	7.5

This work has been supported in part by the Defense Advanced Research Projects Agency (DARPA) Data-Driven Discovery of Models (D3M) Program.



Sequential Model-based Algorithm Configuration

The open-source solution of AAD Freiburg uses a heterogeneous ensemble of learning machines (AutoSklearn (Feurer et al., 2015a,c)) combining the machine learning library scikit-learn (Pedregosa et al., 2011) with the state-of-the-art SMBO method SMAC to find suitable machine learning pipelines for a data set at hand. This is essentially a reimplementation of Auto-WEKA. To speed up the optimization process they employed a metalearning technique (Feurer et al., 2015b) which starts SMAC from promising configurations of scikit-learn. Furthermore, they used the outputs of all models and combined these into an ensemble using ensemble selection.

JMLR: Workshop and Conference Proceedings 1:1–10, 2016 ICML 2016 AutoML Workshop

A brief Review of the ChaLearn AutoML Challenge:

Any-time Any-dataset Learning without Human Intervention

Isabelle Guyon *U. Paris-Saclay, France, and ChaLearn, USA* GUYON@CHALEARN.ORG
Imad Chaabane *U. Paris-Saclay, France* IMAD-EDDINE.CHAABANE@U-PSUD.FR
Hugo Jair Escalante *INAOE, Puebla, Mexico* HUGO.JAIR@GMAIL.COM
Sergio Escalera *U. Barcelona and Computer Vision Center, Spain* SERGIO@MAIA.UB.ES
Damir Jajetic *IN2, Croatia* DAMIR.JAJETIC@IN2.HR
James Robert Lloyd *Qlearsite* JAMES.ROBERT.LLOYD@GMAIL.COM
Núria Macià *Universitat d'Andorra, Andorra* MACIA.NURIA@GMAIL.COM
Bisakha Ray *NYU School of Medicine, USA* BISAKHA.RAY@NYUMC.ORG
Lukasz Romaszko *University of Edinburgh, UK* LUKASZ.ROMASZKO@GMAIL.COM
Michele Sebag *U. Paris-Saclay, France* MICHELE.SEBAG@LRI.FR
Alexander Statnikov *NYU School of Medicine, USA* STATNIKOV@GMAIL.COM
Sébastien Treguer *Paris, France* STREGUER@GMAIL.COM
Evelyne Viegas *Microsoft Research, USA* EVELYNEV@MICROSOFT.COM

Abstract

The ChaLearn AutoML Challenge team conducted a large scale evaluation of fully automatic, black-box learning machines for feature-based classification and regression problems. The test bed was composed of 30 data sets from a wide variety of application domains and ranged across different types of complexity. Over six rounds, participants succeeded in

Sequential Model-based Algorithm Configuration

Algorithm configuration (AC) procedures automatically determine a parameter configuration with low cost (e.g., runtime) on a given benchmark set. General algorithm configuration procedures fall into two categories: model-free approaches, such as ParamILS (Hutter et al. 2009), irace (Lopez-Ibanez et al. 2016) or GGA (Ansotegui, Sellmann, and Tierney 2009), and model-based approaches, such as SMAC (Hutter, Hoos, and Leyton-Brown 2011) or GGA++ (Ansotegui et al. 2015).

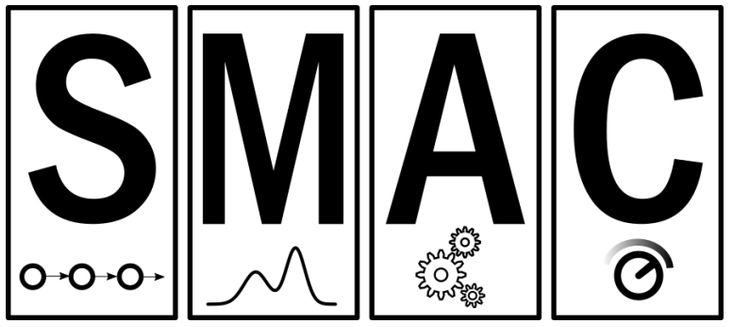
The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Warmstarting of Model-Based Algorithm Configuration

Marius Lindauer, Frank Hutter
University of Freiburg
{lindauer,fh}@cs.uni-freiburg.de

Abstract

The performance of many hard combinatorial problem solvers depends strongly on their parameter settings, and since manual parameter tuning is both tedious and suboptimal the AI community has recently developed several algorithm configuration (AC) methods to automatically address this problem. While all existing AC methods start the configuration process of an algorithm A from scratch for each new type of benchmark instances, here we propose to exploit information about A's performance on previous benchmarks in order to warmstart its configuration on new types of benchmarks. We introduce two complementary ways in which we can exploit this information to warmstart AC methods based on a predictive model. Experiments for optimizing a flexible modern SAT solver on twelve different instance sets show that our methods often yield substantial speedups over existing AC methods (up to 165-fold) and can also find substantially better configurations given the same compute budget.



Sequential Model-based Algorithm Configuration

An Evaluation of Sequential Model-Based Optimization for Expensive Blackbox Functions

Frank Hutter
Freiburg University
Department of Computer Science
fh@informatik.uni-freiburg.de

Holger Hoos
University of British Columbia
Department of Computer Science
hoos@cs.ubc.ca

Kevin Leyton-Brown
University of British Columbia
Department of Computer Science
kevinlb@cs.ubc.ca

ABSTRACT

We benchmark a sequential model-based optimization procedure, SMAC-BBOB, on the BBOB set of blackbox functions. We demonstrate that with a small budget of $10 \times D$ evaluations of D-dimensional functions, SMAC-BBOB in most cases outperforms the state-of-the-art blackbox optimizer CMA-ES. However, CMA-ES benefits more from growing the budget to $100 \times D$, and for larger number of function evaluations SMAC-BBOB also requires increasingly large computational resources for building and using its models.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*global optimization, unconstrained optimization*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

General Terms

Algorithms

Keywords

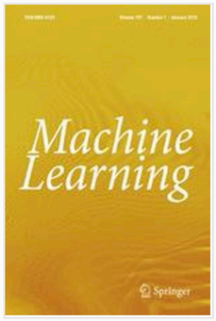
$\boldsymbol{\theta} \in \boldsymbol{\Theta}$ on instances $\pi \sim D$. Let $f(\boldsymbol{\theta}) = \mathbb{E}_{\pi \sim D}[m(\boldsymbol{\theta}, \pi)]$ denote the expected performance of A given parameter setting $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ (where the expectation is over instances π drawn from D; in the case of randomized algorithms, it would also be over random seeds). The problem is then to find a parameter setting $\boldsymbol{\theta}$ of A that minimizes $f(\boldsymbol{\theta})$.

SMAC is rooted in the statistics literature on sequential model-based optimization (SMBO) of expensive functions [13, 12]. To minimize functions $f : \boldsymbol{\Theta} \mapsto \mathbb{R}$ that do not have closed-form representations, are costly to evaluate, and do not allow the computation of gradients, SMBO first gathers initial data and then iterates over the following steps:

1. based on the data collected thus far, construct a model that predicts a probability distribution for f's value at arbitrary points $\boldsymbol{\theta} \in \boldsymbol{\Theta}$;
2. use the model to quantify the desirability $d(\boldsymbol{\theta})$ of learning $f(\boldsymbol{\theta})$ for each $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and to select $\boldsymbol{\theta} \in \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} d(\boldsymbol{\theta})$; and
3. evaluate $f(\boldsymbol{\theta})$, resulting in a new data point $\langle \boldsymbol{\theta}, f(\boldsymbol{\theta}) \rangle$.

The desirability function d serves to address the exploration/exploitation

SMAC Sequential Model-based Algorithm Configuration



Machine Learning
January 2018, Volume 107, Issue 1, pp 15–41 | Cite as

Efficient benchmarking of algorithm configurators via model-based surrogates

Authors | Authors and affiliations

Katharina Eggensperger ✉, Marius Lindauer, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown

Article
First Online: 22 December 2017

Part of the following topical collections:

- Special Issue on Metalearning and Algorithm Selection

Abstract

The optimization of algorithm (hyper-)parameters is crucial for achieving peak performance across a wide range of domains, ranging from deep neural networks to solvers for hard combinatorial problems. However, the proper evaluation of new algorithm configuration (AC) procedures (or *configurators*) is hindered by two key hurdles. First, AC scenarios are hard to set up, including the target algorithm to be optimized and the problem instances to be solved. Second, and even more significantly, they are computationally expensive: a single configurator

Efficient and Robust Automated Machine Learning

Matthias Feurer **Aaron Klein** **Katharina Eggensperger**
Jost Tobias Springenberg **Manuel Blum** **Frank Hutter**
Department of Computer Science
University of Freiburg, Germany
{feurerm,kleinaa,eggenspk,springj,mblum,fh}@cs.uni-freiburg.de

Abstract

The success of machine learning in a broad range of applications has led to an ever-growing demand for machine learning systems that can be used off the shelf by non-experts. To be effective in practice, such systems need to automatically choose a good algorithm and feature preprocessing steps for a new dataset at hand, and also set their respective hyperparameters. Recent work has started to tackle this *automated machine learning (AutoML)* problem with the help of efficient Bayesian optimization methods. Building on this, we introduce a robust new AutoML system based on scikit-learn (using 15 classifiers, 14 feature preprocessing methods, and 4 data preprocessing methods, giving rise to a structured hypothesis space with 110 hyperparameters). This system, which we dub AUTO-SKLEARN, improves on existing AutoML methods by automatically taking into account past performance on similar datasets, and by constructing ensembles from the models evaluated during the optimization. Our system won the first phase of the ongoing ChaLearn AutoML challenge, and our comprehensive analysis on over 100 diverse datasets shows that it substantially outperforms the previous state of the art in AutoML. We also demonstrate the performance gains due to each of our contri
insights into the effectiveness of the individual components of

	Abalone	Amazon	Car	Cifar-10	Cifar-10 Small	Convex	Dexter	Dorothea	German Credit	Gisette	KDD09 Appetency	KR-vs-KP	Madelon	MNIST Basic	MRBI	Secom	Semeion	Shuttle	Waveform	Wine Quality	Yeast
AS	**73.50**	**16.00**	0.39	**51.70**	**54.81**	**17.53**	**5.56**	**5.51**	**27.00**	**1.62**	**1.74**	0.42	**12.44**	2.84	**46.92**	**7.87**	**5.24**	**0.01**	14.93	33.76	40.67
AW	**73.50**	30.00	**0.00**	56.95	56.20	21.80	8.33	6.38	28.33	2.29	**1.74**	**0.31**	18.21	2.84	60.34	8.09	5.24	**0.01**	14.13	**33.36**	**37.75**
HS	76.21	16.22	0.39	-	57.95	19.18	-	-	27.67	2.29	-	0.42	14.74	**2.82**	55.79	-	5.87	0.05	**14.07**	34.72	38.45

Table 2: Test set classification error of AUTO-WEKA (AW), vanilla AUTO-SKLEARN (AS) and HYPEROPT-SKLEARN (HS), as in the original evaluation of AUTO-WEKA [2]. We show median percent error across 100 000 bootstrap samples (based on 10 runs), simulating 4 parallel runs. Bold numbers indicate the best result. Underlined results are not statistically significantly different from the best according to a bootstrap test with $p = 0.05$.

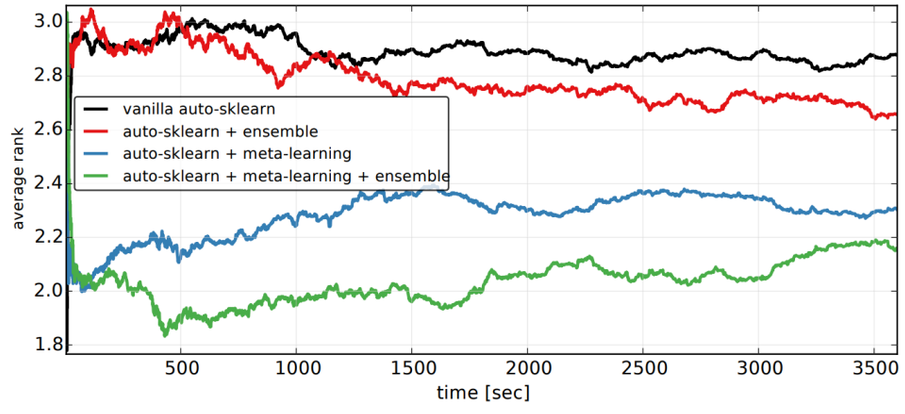


Figure 3: Average rank of all four AUTO-SKLEARN variants (ranked by balanced test error rate (BER)) across 140 datasets. Note that ranks are a relative measure of performance (here, the rank of all methods has to add up to 10), and hence an improvement in BER of one method can worsen the rank of another. The supplementary material shows the same plot on a log-scale to show the time overhead of meta-feature and ensemble computation.

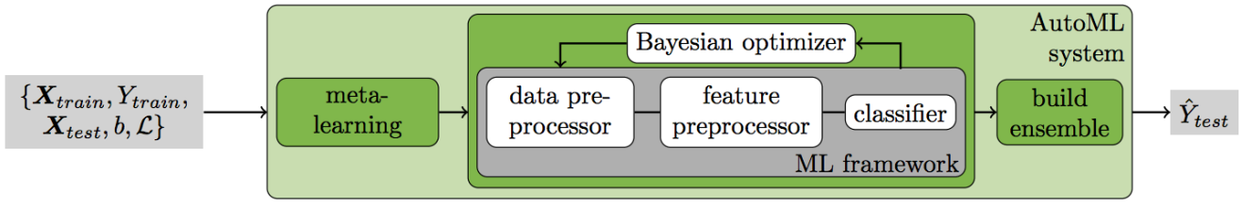


Figure 1: Our improved AutoML approach. We add two components to Bayesian hyperparameter optimization of an ML framework: meta-learning for initializing the Bayesian optimizer and automated ensemble construction from configurations evaluated during optimization.

https://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf

NeurIPS | 2018

Thirty-second Conference on Neural Information Processing Systems

Joint auto-encoders: a flexible multi-task learning framework

Baruch Epstein **Ron Meir** **Tomer Michaeli**

The Viterbi Faculty of Electrical Engineering
Technion - Israel Institute of Technology, Haifa, Israel
baruch.epstein@gmail.com, rmeir@ee.technion.ac.il, tomer.m@ee.technion.ac.il

Abstract

The incorporation of prior knowledge into learning is essential in achieving good performance based on small noisy samples. Such knowledge is often incorporated through the availability of related data arising from domains and tasks similar to the one of current interest. Ideally one would like to allow both the data for the current task and for previous related tasks to self-organize the learning system in such a way that commonalities and differences between the tasks are learned in a data-driven fashion. We develop a framework for learning multiple tasks simultaneously, based on sharing features that are common to all tasks, achieved through the use of a modular deep feedforward neural network consisting of shared branches, dealing with the common features of all tasks, and private branches, learning the specific unique aspects of each task. Once an appropriate weight sharing architecture has been established, learning takes place through standard algorithms for feedforward networks, e.g., stochastic gradient descent and its variations. The method deals with domain adaptation and multi-task learning in a unified fashion, and can easily deal with data arising from different types of sources. Numerical experiments demonstrate the effectiveness of learning in domain adaptation and transfer learning setups, and provide evidence for the flexible and task-oriented representations arising in the network.

NeurIPS | 2018

Thirty-second Conference on Neural Information Processing Systems

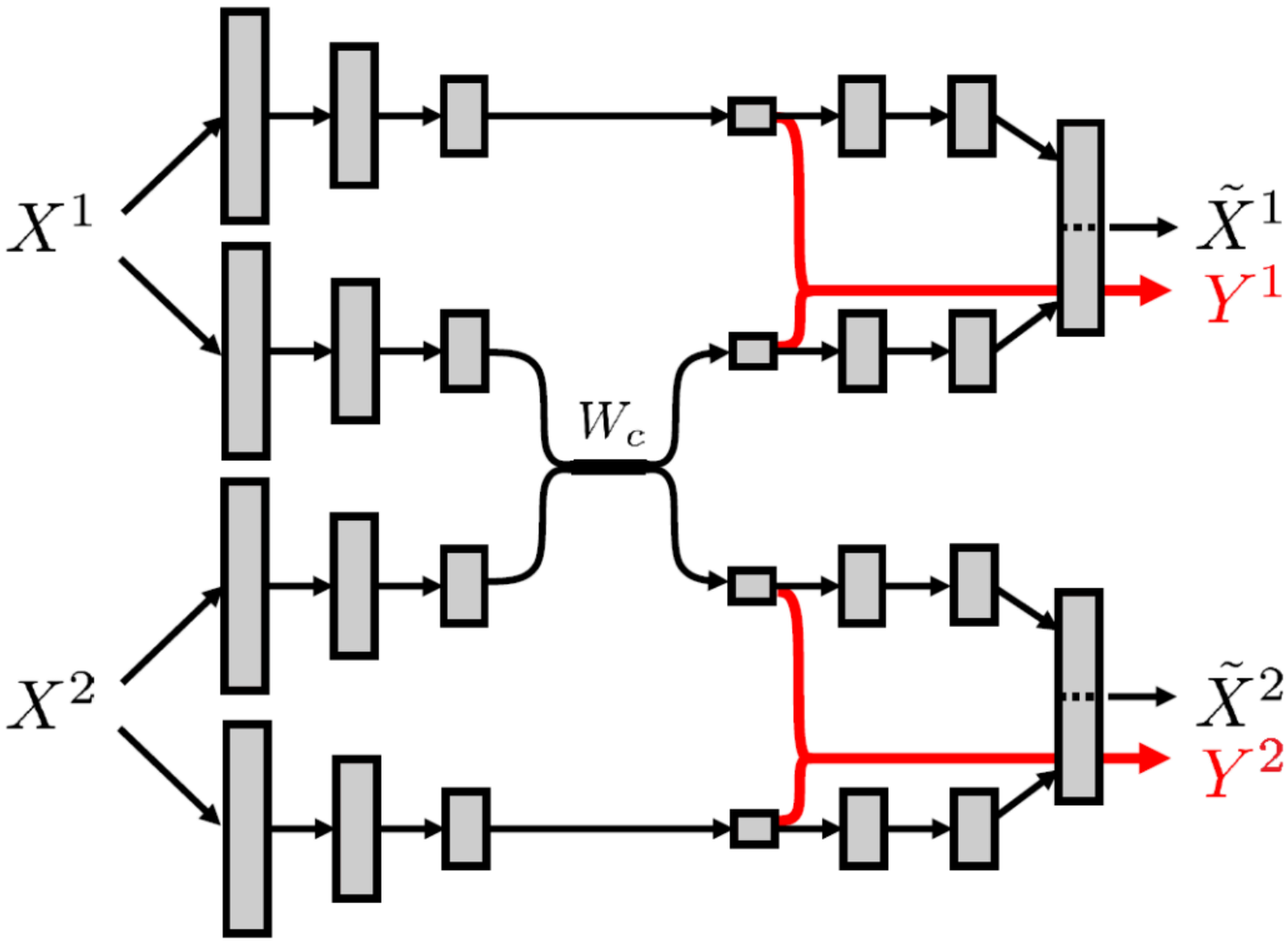


Figure 2: A schematic depiction of a joint autoencoder architecture extended for supervised learning.

https://arxiv.org/pdf/1705.10494.pdf

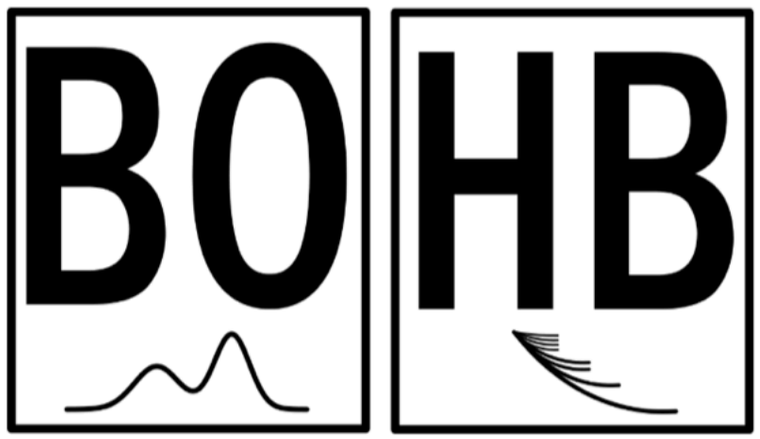
NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING

Barret Zoph,* **Quoc V. Le**
Google Brain
{barretzoph,qvl}@google.com

ABSTRACT

Neural networks are powerful and flexible models that work well for many difficult learning tasks in image, speech and natural language understanding. Despite their success, neural networks are still hard to design. In this paper, we use a recurrent network to generate the model descriptions of neural networks and train this RNN with reinforcement learning to maximize the expected accuracy of the generated architectures on a validation set. On the CIFAR-10 dataset, our method, starting from scratch, can design a novel network architecture that rivals the best human-invented architecture in terms of test set accuracy. Our CIFAR-10 model achieves a test error rate of 3.65, which is 0.09 percent better and 1.05x faster than the previous state-of-the-art model that used a similar architectural scheme. On the Penn Treebank dataset, our model can compose a novel recurrent cell that outperforms the widely-used LSTM cell, and other state-of-the-art baselines. Our cell achieves a test set perplexity of 62.4 on the Penn Treebank, which is 3.6 perplexity better than the previous state-of-the-art model. The cell can also be transferred to the character language modeling task on PTB and achieves a state-of-the-art perplexity of 1.214.

https://arxiv.org/pdf/1611.01578.pdf



BOHB

BOHB combines the benefits of both Bayesian Optimization and HyperBand, in order to achieve the best of both worlds: strong anytime performance and fast convergence to optimal configurations.

Read More »

PMLR Proceedings of Machine Learning Research

BOHB: Robust and Efficient Hyperparameter Optimization at Scale

[edit]

Stefan Falkner, Aaron Klein, Frank Hutter ; ***Proceedings of the 35th International Conference on Machine Learning, PMLR 80:1437-1446, 2018.***

Abstract

Modern deep learning methods are very sensitive to many hyperparameters, and, due to the long training times of state-of-the-art models, vanilla Bayesian hyperparameter optimization is typically computationally infeasible. On the other hand, bandit-based configuration evaluation approaches based on random search lack guidance and do not converge to the best configurations as quickly. Here, we propose to combine the benefits of both Bayesian optimization and bandit-based methods, in order to achieve the best of both worlds: strong anytime performance and fast convergence to optimal configurations. We propose a new practical state-of-the-art hyperparameter optimization method, which consistently outperforms both Bayesian optimization and Hyperband on a wide range of problem types, including high-dimensional toy functions, support vector machines, feed-forward neural networks, Bayesian neural networks, deep reinforcement learning, and convolutional neural networks. Our method is robust and versatile, while at the same time being conceptually simple and easy to implement.

Related Material

- Download PDF

https://github.com/automl/HpBandSter

Introduction to Bayesian Optimization

Javier González

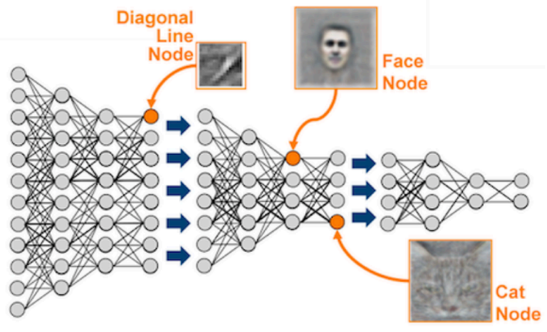
Masterclass, 7-February, 2107 @Lancaster University

"Civilization advances by extending the number of important operations which we can perform without thinking of them."

http://gpss.cc/gpmc17/slides/LancasterMasterclass_1.pdf

Expensive functions, who doesn't have one?

Parameter tuning in ML algorithms.



- Number of layers/units per layer
- Weight penalties
- Learning rates, etc.

General idea: surrogate modelling

1. Use a surrogate model of f to carry out the optimization.
2. Define an utility function to collect new data points satisfying some optimality criterion: *optimization* as *decision*.
3. Study *decision* problems as *inference* using the surrogate model: use a probabilistic model able to calibrate both, epistemic and aleatoric uncertainty.

Uncertainty Quantification

Bayesian Optimisation

[Mockus, 1978]

Methodology to perform global optimisation of multimodal black-box functions.

1. Choose some *prior measure* over the space of possible objectives f.
2. Combine prior and the likelihood to get a *posterior measure* over the objective given some observations.
3. Use the posterior to decide where to take the next evaluation according to some *acquisition/loss function*.
4. Augment the data.

Iterate between 2 and 4 until the evaluation budget is over.

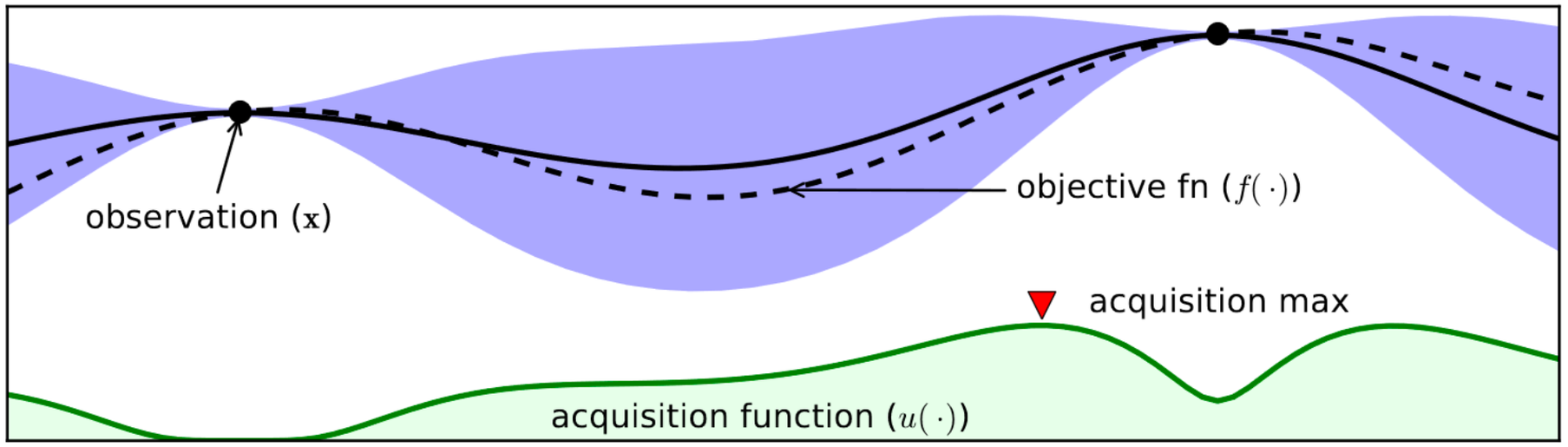
Took a while to start using these ideas in ML

Although in the stats community have been there for a while

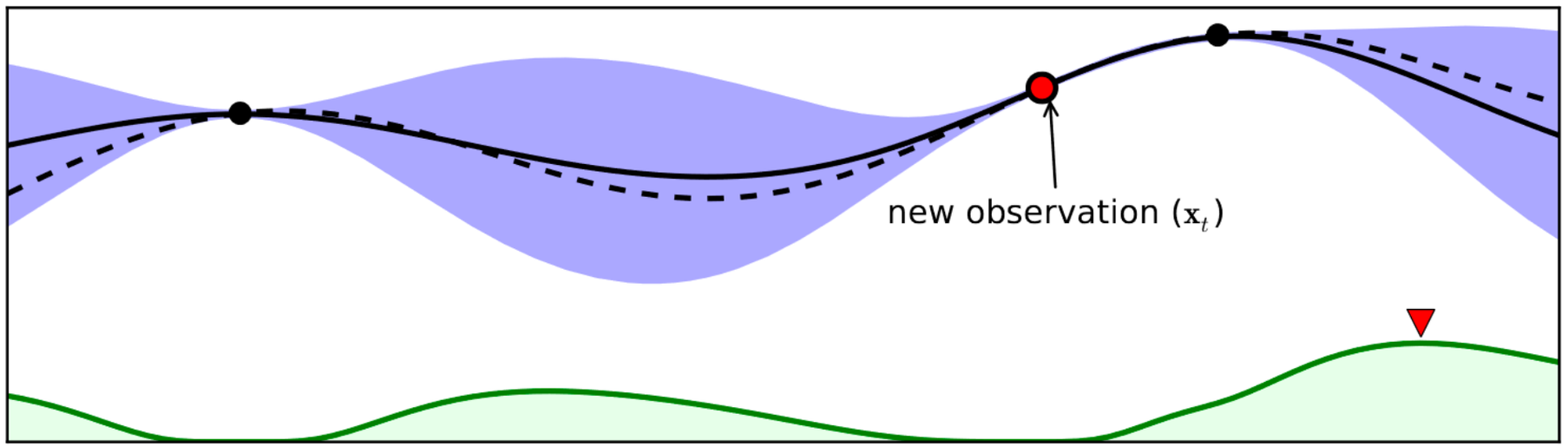
- BO depends on its own parameters.
- Lack of software to apply these methods as a black optimization boxes.
- Reduced scalability in dimensions and number of evaluations (this is still a problem).

Practical Bayesian Optimization of Machine Learning Algorithms. Snoek, Larochelle and Adams. NIPS 2012 (Spearmint)

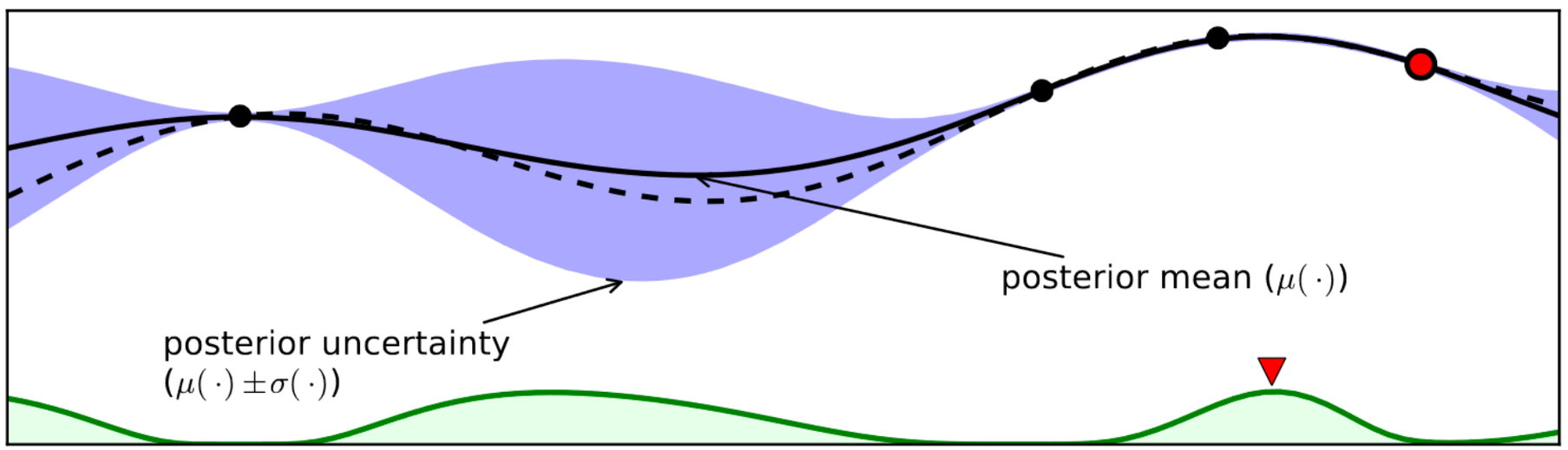
$t = 2$



$t = 3$



$t = 4$



https://towardsdatascience.com/shallow-understanding-on-bayesian-optimization-324b6c1f7083

Taking the Human Out of the Loop: A Review of Bayesian Optimization

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams and Nando de Freitas

***Abstract*—Big data applications are typically associated with systems involving large numbers of users, massive complex software systems, and large-scale heterogeneous computing and storage architectures. The construction of such systems involves many distributed design choices. The end products (e.g., recommendation systems, medical analysis tools, real-time game engines, speech recognizers) thus involves many tunable configuration parameters. These parameters are often specified and hard-coded into the software by various developers or teams. If optimized jointly, these parameters can result in significant improvements. Bayesian optimization is a powerful tool for the joint optimization of design choices that is gaining great popularity in recent years. It promises greater automation so as to increase both product quality and human productivity. This review paper introduces Bayesian optimization, highlights some of its methodological aspects, and showcases a wide range of applications.**

and the analytics company that sits between them. The analytics company must develop procedures to automatically design game variants across millions of users; the objective is to enhance user experience and maximize the content provider's revenue.

The preceding examples highlight the importance of automating design choices. For a nurse scheduling application, we would like to have a tool that automatically chooses the 76 CPLEX parameters so as to improve healthcare delivery. When launching a mobile game, we would like to use the data gathered from millions of users in real-time to automatically adjust and improve the game. When a data scientist uses a machine learning library to forecast energy demand, we would like to automate the process of choosing the best forecasting technique and its associated parameters.

Any significant advances in automated design can result in immediate product improvements and innovation in a wide

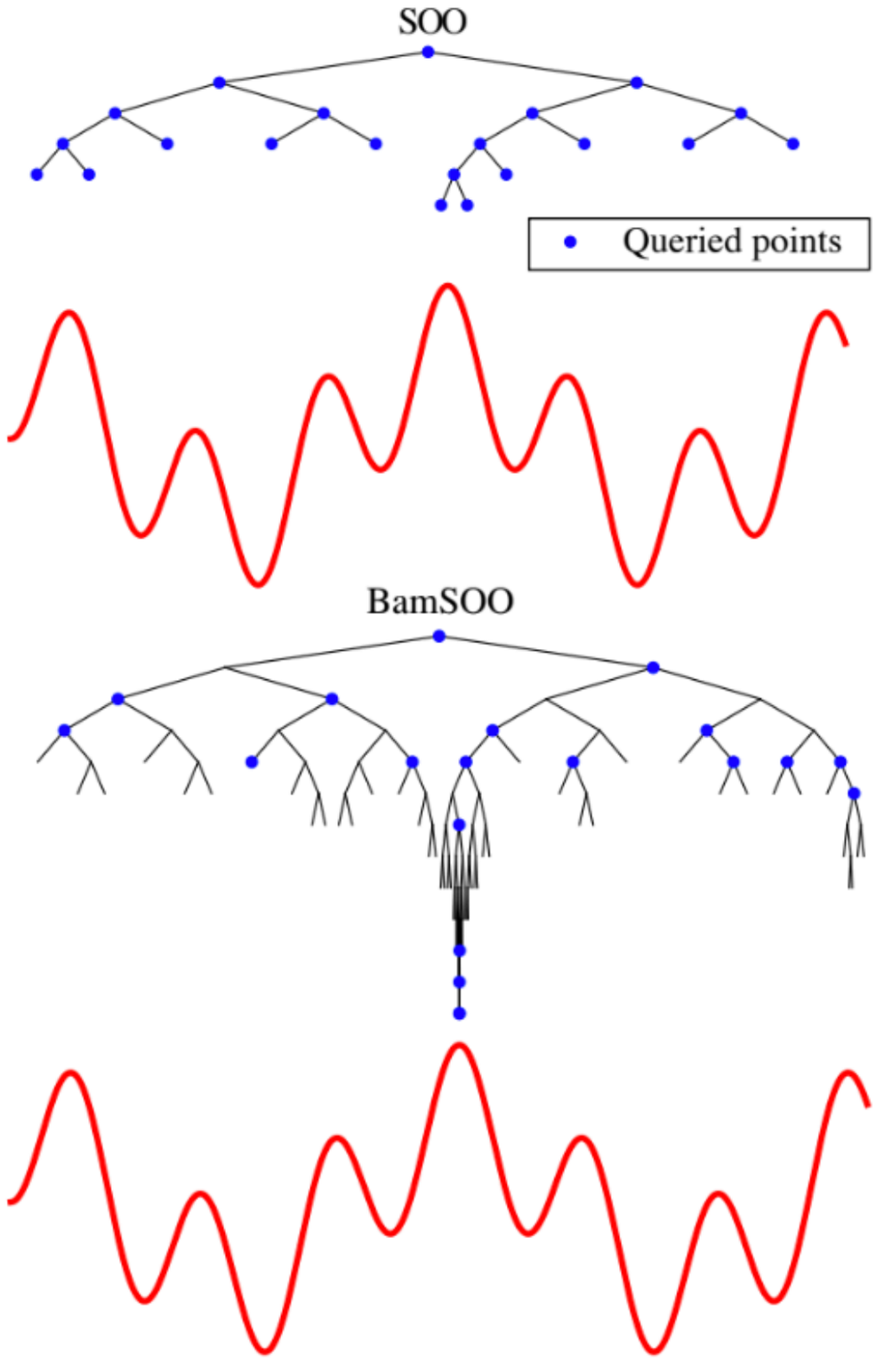


Fig. 8. Comparison of SOO (top) and BamSOO (bottom) on $f(x) = \frac{1}{2}\sin(15x)\sin(27x)$ in $[0, 1]$. Blue dots represent nodes where the objective was evaluated. BaMSOO does not evaluate f at points that are suboptimal with high probability under the surrogate model (not shown). Figure from [158].

https://towardsdatascience.com/shallow-understanding-on-bayesian-optimization-324b6c1f7083

A Tutorial on Bayesian Optimization

Peter I. Frazier

July 10, 2018

Abstract

Bayesian optimization is an approach to optimizing objective functions that take a long time (minutes or hours) to evaluate. It is best-suited for optimization over continuous domains of less than 20 dimensions, and tolerates stochastic noise in function evaluations. It builds a surrogate for the objective and quantifies the uncertainty in that surrogate using a Bayesian machine learning technique, Gaussian process regression, and then uses an acquisition function defined from this surrogate to decide where to sample. In this tutorial, we describe how Bayesian optimization works, including Gaussian process regression and three common acquisition functions: expected improvement, entropy search, and knowledge gradient. We then discuss more advanced techniques, including running multiple function evaluations in parallel, multi-fidelity and multi-information source optimization, expensive-to-evaluate constraints, random environmental conditions, multi-task Bayesian optimization, and the inclusion of derivative information. We conclude with a discussion of Bayesian optimization software and future research directions in the field. Within our tutorial material we provide a generalization of expected improvement to noisy evaluations, beyond the noise-free setting where it is more commonly applied. This generalization is justified by a formal decision-theoretic argument, standing in contrast to previous ad hoc modifications.

https://arxiv.org/pdf/1807.02811.pdf

AuDaS: Automated Data Scientist

AuDaS is a Machine Learning platform that allows anyone with or without a background in Data Science to automatically:

- Load and prepare data
- Build and operationalize business decisioning solutions
- Build and operationalize Time Series applications

AuDaS provides an extremely user-friendly, clutter free interface that guides you through the solution building process:

1. Uploading the data