Automating Machine Learning

Madeleine Udell

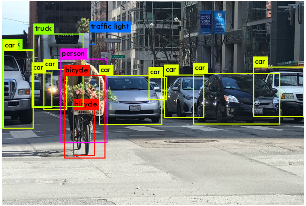
Operations Research and Information Engineering
Cornell University

Based on joint work with Chengrun Yang (Cornell)

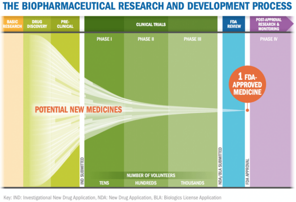
WIDS workshop, March 2021

Outline

So many machine learning problems. . .

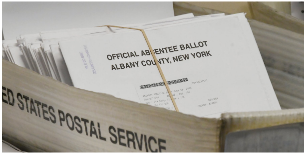


object detection



drug discovery

speech recognition



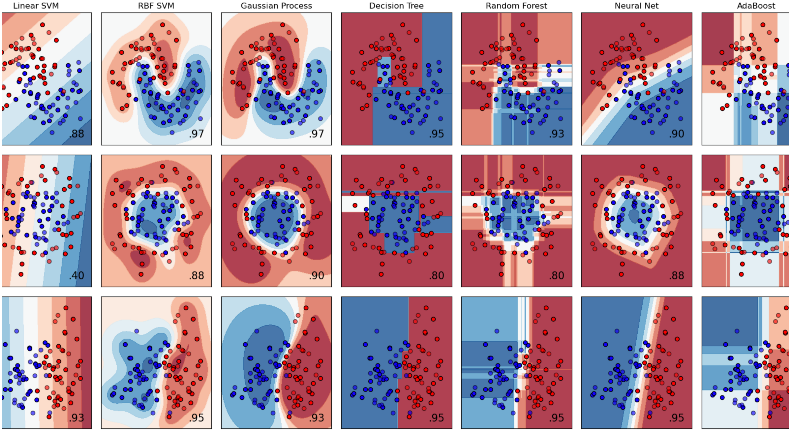
social science

. . . so little time

```
classifiers = [
    KNeighborsClassifier(3),
    SVC(kernel="linear", C=0.025),
    SVC(gamma=2, C=1),
    GaussianProcessClassifier(1.0 * RBF(1.0)),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=10, max_fe
    MLPClassifier(alpha=1, max_iter=1000),
    AdaBoostClassifier(),
    GaussianNB(),
    QuadraticDiscriminantAnalysis()]
```

source: `https://scikit-learn.org`

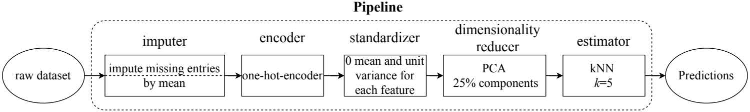
Different models perform differently



source: `https://scikit-learn.org`

Decisions, decisions...

a **pipeline**: a directed graph of learning components



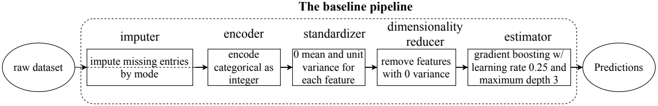
so many choices to make:

- data imputer: fill in missing values by median? ...
- encoder: one-hot encode? ...
- standardizer: rescale each feature? ...
- dimensionality reducer: PCA, or select by variance? ...
- estimator: use decision tree or logistic regression? ...
- hyperparameters: depth of decision tree?

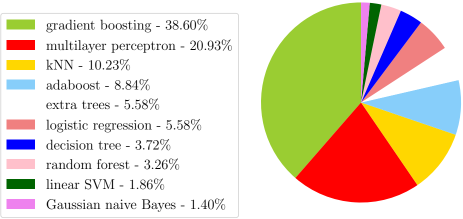
No Free Lunch

On 215 midsize OpenML classification datasets:

- The best-on-average pipeline (highest average ranking):



- The best estimator for each dataset:

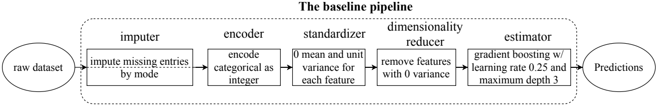


source: [Yang et al.(2020)Yang, Fan, Wu, and Udell]

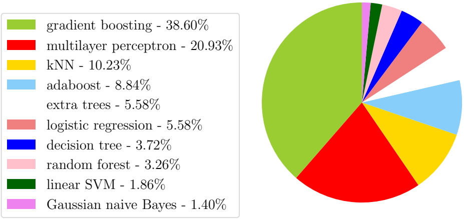
No Free Lunch

On 215 midsize OpenML classification datasets:

- The best-on-average pipeline (highest average ranking):



- The best estimator for each dataset:



source: [Yang et al.(2020)Yang, Fan, Wu, and Udell]

Theorem (No free lunch [Wolpert(1996)])

There is no one model that works best for every problem.

Problem solved!

```
>>> import autosklearn.classification
>>> cls = autosklearn.classification.AutoSklearnClassifier()
>>> cls.fit(X_train, y_train)
>>> predictions = cls.predict(X_test)
```

```
dls = TabularDataLoaders.from_csv(path/'adult.csv', path=path, y_names="salary",
    cat_names = ['workclass', 'education', 'marital-status', 'occupation',
                 'relationship', 'race'],
    cont_names = ['age', 'fnlwgt', 'education-num'],
    procs = [Categorify, FillMissing, Normalize])

learn = tabular_learner(dls, metrics=accuracy)
learn.fit_one_cycle(2)
```

```
from flaml import AutoML
automl = AutoML()
automl.fit(X_train, y_train, task="classification")
```

```
# Run AutoML for 20 base models (limited to 1 hour max runtime by default)
aml = H2OAutoML(max_models=20, seed=1)
aml.train(x=x, y=y, training_frame=train)
```

```
from autogluon.tabular import TabularDataset, TabularPredictor
train_data = TabularDataset('https://autogluon.s3.amazonaws.com/datasets/Inc/train.csv')
test_data = TabularDataset('https://autogluon.s3.amazonaws.com/datasets/Inc/test.csv')
predictor = TabularPredictor(label='class').fit(train_data, time_limit=60)  # Fit models for 60s
leaderboard = predictor.leaderboard(test_data)
```

Definitions

automated machine learning (AutoML) chooses a ML model + hyperparameters so you don't have to.

types of AutoML:

Definitions

automated machine learning (AutoML) chooses a ML model + hyperparameters so you don't have to.

types of AutoML:

- **hyperparameter tuning** chooses the best hyperparameters for the model

Definitions

automated machine learning (AutoML) chooses a ML model + hyperparameters so you don't have to.

types of AutoML:

- **hyperparameter tuning** chooses the best hyperparameters for the model
- **combined algorithm and hyperparameter search (CASH)** chooses an estimator and hyperparameters

Definitions

automated machine learning (AutoML) chooses a ML model + hyperparameters so you don't have to.

types of AutoML:

- **hyperparameter tuning** chooses the best hyperparameters for the model
- **combined algorithm and hyperparameter search (CASH)** chooses an estimator and hyperparameters
- **neural architecture search (NAS)** chooses a deep learning architecture
  e.g., number of layers, type of layer, width, learning rate

Definitions

automated machine learning (AutoML) chooses a ML model + hyperparameters so you don't have to.

types of AutoML:

- **hyperparameter tuning** chooses the best hyperparameters for the model
- **combined algorithm and hyperparameter search (CASH)** chooses an estimator and hyperparameters
- **neural architecture search (NAS)** chooses a deep learning architecture
  e.g., number of layers, type of layer, width, learning rate
- **metalearning**, or learning to learn, uses information gleaned from a corpus of datasets to choose a better model on a new dataset

Definitions

automated machine learning (AutoML) chooses a ML model + hyperparameters so you don't have to.

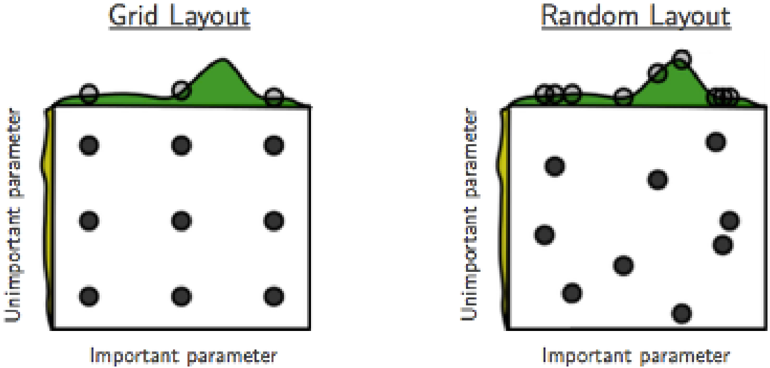
types of AutoML:

- **hyperparameter tuning** chooses the best hyperparameters for the model
- **combined algorithm and hyperparameter search (CASH)** chooses an estimator and hyperparameters
- **neural architecture search (NAS)** chooses a deep learning architecture
  e.g., number of layers, type of layer, width, learning rate
- **metalearning**, or learning to learn, uses information gleaned from a corpus of datasets to choose a better model on a new dataset

kinds of datasets: **tabular**, timeseries, image, text, video, genomics, . . .

Outline

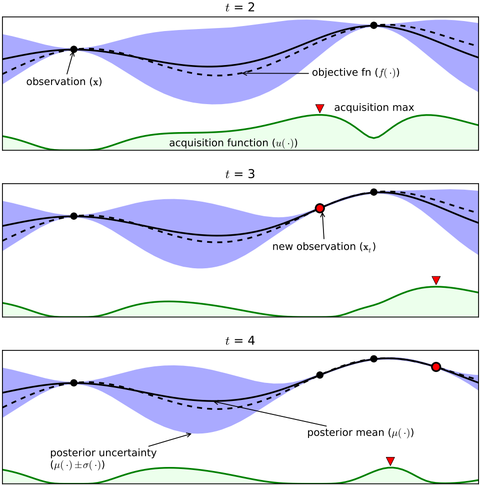
Grid search vs random search



source: Bergstra & Bengio 2012 [Bergstra and Bengio(2012)].

- grid search is more well-known
- random search samples more distinct values of each hyperparameter
- random search is more efficient when only some hyperparameters are important

Bayesian optimization (BO)

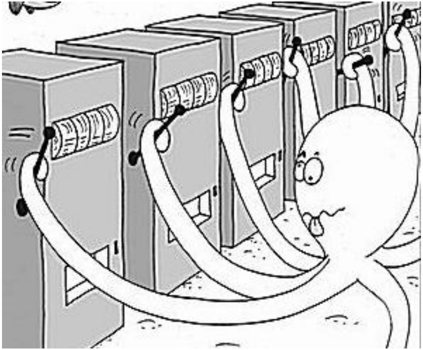


source: Brochu et al, 2010
[Brochu et al.(2010)Brochu, Cora, and De Freitas]

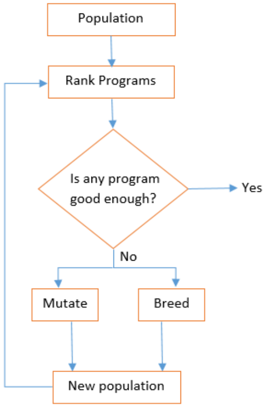
Multi-armed bandit

How long to spend evaluating each pipeline?

- Budget: training examples or training time
- Estimate performance of each pipeline with small budget
- Allocate budget to promising pipelines



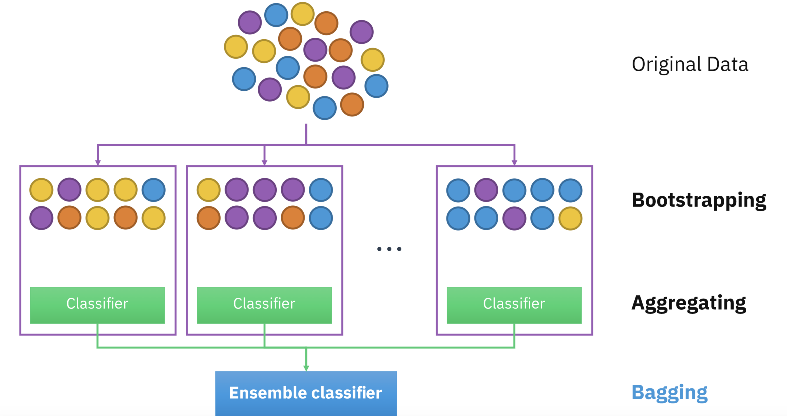
Genetic programming



"Survival of the fittest": Automatically explore numerous possible pipelines to find the best for the given dataset

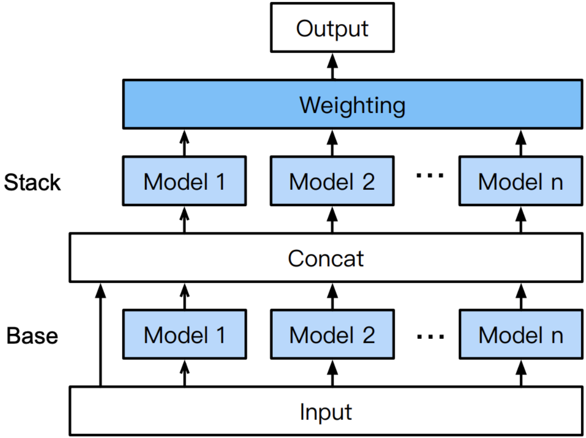
source: dotnetlovers.com

Ensemble



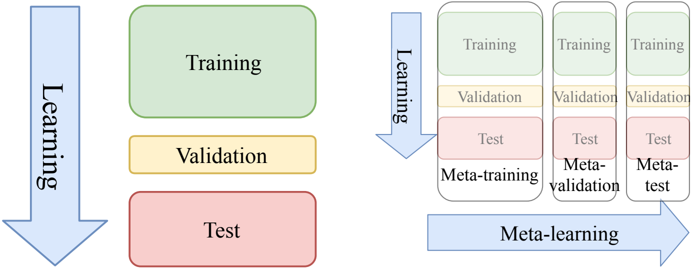
source: By Sirakorn - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=85888768

Stacking



source: AutoGluon Tabular
[Erickson et al.(2020)Erickson, Mueller, Shirkov, Zhang, Larroy, Li, and Smola]

Metalearning



Learning

Metalearning

- learning splits datasets
- metalearning splits learning instances:
  - same model, different datasets ("sets of datasets")
    e.g., stock market data on different days
  - different models, same dataset
    e.g., performance of ridge regression at different λ's

source: OBOE [Yang et al.(2019)Yang, Akimoto, Kim, and Udell]

OBOE: low rank autoML

given: m datasets, n machine learning models

OBOE: low rank autoML

given: m datasets, n machine learning models
measure: error of each model on each dataset

OBOE: low rank autoML

given: m datasets, n machine learning models
measure: error of each model on each dataset
form: $m \times n$ data table A

$$A = \text{datasets}\left\{ \overbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}^{\text{models}} \right.$$

source: OBOE [Yang et al.(2019)Yang, Akimoto, Kim, and Udell]

OBOE: low rank autoML

given: m datasets, n machine learning models
measure: error of each model on each dataset
form: $m \times n$ data table A
find: $X \in \mathbf{R}^{m\times k}$, $Y \in \mathbf{R}^{k\times n}$ for which

$$A \approx XY$$

$$\text{datasets}\left\{\overbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}^{\text{models}}\right. \approx \begin{bmatrix} -x_1- \\ \vdots \\ -x_m- \end{bmatrix} \begin{bmatrix} | & & | \\ y_1 & \dots & y_n \\ | & & | \end{bmatrix}$$

source: OBOE [Yang et al.(2019)Yang, Akimoto, Kim, and Udell]

OBOE: low rank autoML

given: m datasets, n machine learning models
measure: error of each model on each dataset
form: $m \times n$ data table A
find: $X \in \mathbf{R}^{m \times k}$, $Y \in \mathbf{R}^{k \times n}$ for which

$$A \approx XY$$

$$\text{datasets} \left\{ \overbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ ? & ? & ? & ? & ? \end{bmatrix}}^{\text{models}} \right. \approx \begin{bmatrix} -x_1- \\ \vdots \\ -x_m- \\ ? \quad ? \end{bmatrix} \begin{bmatrix} | & & | \\ y_1 & \dots & y_n \\ | & & | \end{bmatrix}$$

source: OBOE [Yang et al.(2019)Yang, Akimoto, Kim, and Udell]

OBOE: low rank autoML

given: m datasets, n machine learning models
measure: error of each model on each dataset
form: $m \times n$ data table A
find: $X \in \mathbf{R}^{m \times k}$, $Y \in \mathbf{R}^{k \times n}$ for which

$$A \approx XY$$

$$\text{datasets}\left\{\overbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ ? & \times & \times & ? & \times \end{bmatrix}}^{\text{models}}\right. \approx \begin{bmatrix} -x_1- \\ \vdots \\ -x_m- \\ ? \quad ? \end{bmatrix} \begin{bmatrix} | & & | \\ y_1 & \dots & y_n \\ | & & | \end{bmatrix}$$

source: OBOE [Yang et al.(2019)Yang, Akimoto, Kim, and Udell]

OBOE: low rank autoML

given: m datasets, n machine learning models
measure: error of each model on each dataset
form: $m \times n$ data table A
find: $X \in \mathbf{R}^{m\times k}$, $Y \in \mathbf{R}^{k\times n}$ for which

$$A \approx XY$$

$$\text{datasets}\left\{\begin{array}{c} \overbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ ? & \times & \times & ? & \times \end{bmatrix}}^{\text{models}} \end{array}\right. \approx \begin{bmatrix} -x_1- \\ \vdots \\ -x_m- \\ -x_{m+1}- \end{bmatrix} \begin{bmatrix} | & & | \\ y_1 & \dots & y_n \\ | & & | \end{bmatrix}$$

source: OBOE [Yang et al.(2019)Yang, Akimoto, Kim, and Udell]

OBOE: low rank autoML

given: m datasets, n machine learning models
measure: error of each model on each dataset
form: $m \times n$ data table A
find: $X \in \mathbf{R}^{m \times k}$, $Y \in \mathbf{R}^{k \times n}$ for which

$$A \approx XY$$

$$\text{datasets}\left\{ \overbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \cdot & \times & \times & \cdot & \times \end{bmatrix}}^{\text{models}} \right. \approx \begin{bmatrix} -x_1- \\ \vdots \\ -x_m- \\ -x_{m+1}- \end{bmatrix} \begin{bmatrix} | & & | \\ y_1 & \cdots & y_n \\ | & & | \end{bmatrix}$$

OBOE: low rank autoML

given: m datasets, n machine learning models
measure: error of each model on each dataset
form: $m \times n$ data table A
find: $X \in \mathbf{R}^{m\times k}$, $Y \in \mathbf{R}^{k\times n}$ for which

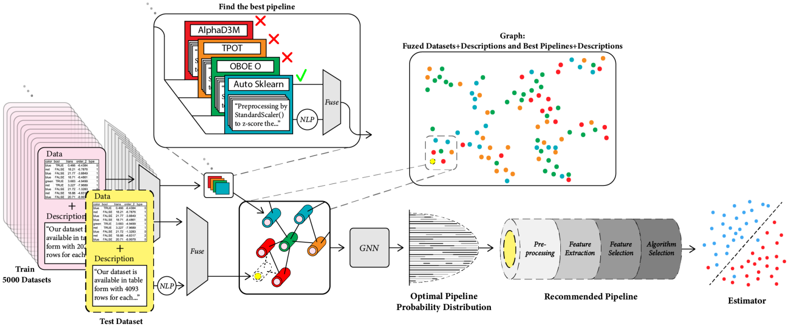
$$A \approx XY$$

$$\text{datasets}\left\{\overbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \cdot & \times & \times & \cdot & \times \end{bmatrix}}^{\text{models}}\right. \approx \begin{bmatrix} -x_1- \\ \vdots \\ -x_m- \\ -x_{m+1}- \end{bmatrix} \begin{bmatrix} | & & | \\ y_1 & \dots & y_n \\ | & & | \end{bmatrix}$$

- rows $x_i \in \mathbf{R}^k$ of X are *dataset metafeatures*
- columns $y_j \in \mathbf{R}^k$ of Y are *model metafeatures*
- $x_i y_j \approx A_{ij}$ are *predicted model performance*

source: OBOE [Yang et al.(2019)Yang, Akimoto, Kim, and Udell]

Metalearning with NLP and GNNs



source: Real-time AutoML
[Drori et al.(2020)Drori, Liu, Ma, Deykin, Kates, and Udell]

Outline

AutoML systems

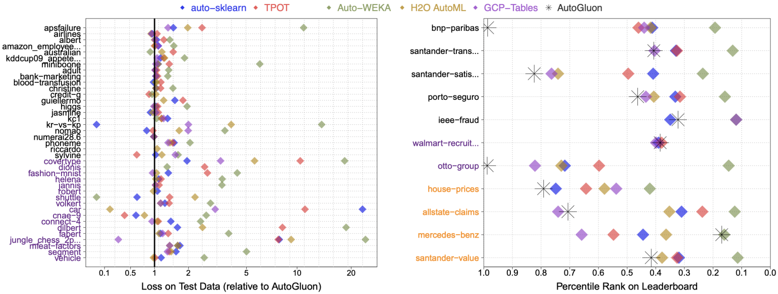
Optimizing over scikit-learn style models:

- **Auto-WEKA** [Thornton et al.(2013)Thornton, Hutter, Hoos, and Leyton-Brown]: BO on conditional search space
- **auto-sklearn** [Feurer et al.(2015)Feurer, Klein, Eggensperger, Springenberg, Blum meta-learning + BO
- **TPOT** [Olson et al.(2016)Olson, Urbanowicz, Andrews, Lavender, Kidd, ar genetic programming
- **Hyperband** [Li et al.(2018)Li, Jamieson, DeSalvo, Rostamizadeh, and Talwalkar multi-armed bandit
- **PMF** [Fusi et al.(2018)Fusi, Sheth, and Elibol]: matrix factorization + BO
- **Oboe** [Yang et al.(2019)Yang, Akimoto, Kim, and Udell]: matrix factorization + experiment design

Neural architecture search (NAS)

- **Google NAS** [Zoph and Le(2016)]: reinforcement learning
- **NASBOT** [Kandasamy et al.(2018)Kandasamy, Neiswanger, Schneider, Poczo
  BO + optimal transport
- **Auto-Keras** [Jin et al.(2019)Jin, Song, and Hu]: BO + network morphism
- **AutoML-Zero** [Real et al.(2020)Real, Liang, So, and Le]: genetic programming
- ...

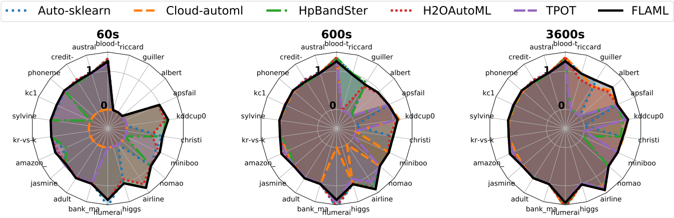
Lots of good options!



(A) AutoML Benchmark (1h) **(B)** Kaggle Benchmark (4h)

source: AutoGluon Tabular
[Erickson et al.(2020)Erickson, Mueller, Shirkov, Zhang, Larroy, Li, and Smola]

Fast and slow options



Binary classification datasets ordered by size counter clockwise, from smallest (blood-transfusion) to largest (riccardo). Metric: AUC.

source: FLAML [Wang et al.(2020)Wang, Wu, Weimer, and Zhu]

Outline

Challenges

Challenges

- interpretability: can we find good, interpretable models? when is interpretability necessary?

Challenges

- interpretability: can we find good, interpretable models? when is interpretability necessary?
- feature engineering

Challenges

- interpretability: can we find good, interpretable models? when is interpretability necessary?
- feature engineering
- overfitting

Challenges

- interpretability: can we find good, interpretable models? when is interpretability necessary?
- feature engineering
- overfitting
- cost:
  e.g., Google RL-based NAS [Zoph and Le(2016)]: 1k GPU days
  ($>$ \$70k on AWS)

Summary

- AutoML automatically picks a good ML pipeline for your problem
- lots of easy-to-use packages
- lots of interesting ideas

References I

J. Bergstra and Y. Bengio.
Random search for hyper-parameter optimization.
Journal of Machine Learning Research, 13(Feb):281–305, 2012.

E. Brochu, V. M. Cora, and N. De Freitas.
A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning.
arXiv preprint arXiv:1012.2599, 2010.

I. Drori, L. Liu, Q. Ma, J. Deykin, B. Kates, and M. Udell.
Real-time AutoML.
Submitted, 2020.

N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola.
Autogluon-tabular: Robust and accurate automl for structured data.
arXiv preprint arXiv:2003.06505, 2020.

M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter.
Efficient and robust automated machine learning.
In *Advances in Neural Information Processing Systems*, pages 2962–2970, 2015.

N. Fusi, R. Sheth, and M. Elibol.
Probabilistic matrix factorization for automated machine learning.
In *Advances in Neural Information Processing Systems*, pages 3352–3361, 2018.

References II

H. Jin, Q. Song, and X. Hu.
Auto-keras: An efficient neural architecture search system.
In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 1946–1956, New York, NY, USA, 2019. ACM.
ISBN 978-1-4503-6201-6.
doi: 10.1145/3292500.3330648.
URL http://doi.acm.org/10.1145/3292500.3330648.

K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. Xing.
Neural Architecture Search with Bayesian Optimisation and Optimal Transport.
arXiv preprint arXiv:1802.07191, 2018.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar.
Hyperband: A novel bandit-based approach to hyperparameter optimization.
Journal of Machine Learning Research, 18(185):1–52, 2018.
URL http://jmlr.org/papers/v18/16-558.html.

R. S. Olson, R. J. Urbanowicz, P. C. Andrews, N. A. Lavender, L. C. Kidd, and J. H. Moore.
Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I, chapter Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, pages 123–137.
Springer International Publishing, 2016.
ISBN 978-3-319-31204-0.
doi: 10.1007/978-3-319-31204-0_9.
URL http://dx.doi.org/10.1007/978-3-319-31204-0_9.

E. Real, C. Liang, D. R. So, and Q. V. Le.
Automl-zero: Evolving machine learning algorithms from scratch.
arXiv preprint arXiv:2003.03384, 2020.

References III

C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown.
Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms.
In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 847–855. ACM, 2013.

C. Wang, Q. Wu, M. Weimer, and E. Zhu.
FLAML: A fast and lightweight AutoML library, 2020.

D. H. Wolpert.
The lack of a priori distinctions between learning algorithms.
Neural Computation, 8(7):1341–1390, 1996.

C. Yang, Y. Akimoto, D. W. Kim, and M. Udell.
Oboe: Collaborative filtering for automl model selection.
In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1173–1183. ACM, 2019.

C. Yang, J. Fan, Z. Wu, and M. Udell.
AutoML pipeline selection: Efficiently navigating the combinatorial space.
In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2020.
URL `http://arxiv.org/abs/2006.04216`.

B. Zoph and Q. V. Le.
Neural architecture search with reinforcement learning.
arXiv preprint arXiv:1611.01578, 2016.