ORIE 4741: Learning with Big Messy Data

Automated Machine Learning

Professor Udell

Operations Research and Information Engineering
Cornell

November 18, 2021

Announcements 11/18/21

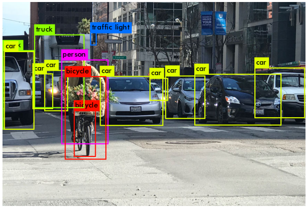
- hw5 due this morning, hw6 out this weekend
- extra project office hours tonight

Poll

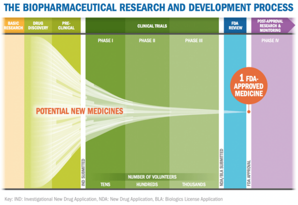
- I am traveling the whole week of Thanksgiving
- I am leaving for Thanksgiving after my classes end Tuesday
- I am staying in Ithaca for Thanksgiving

Outline

So many machine learning problems. . .

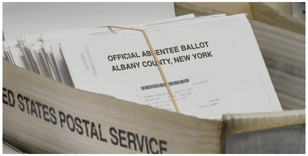


object detection



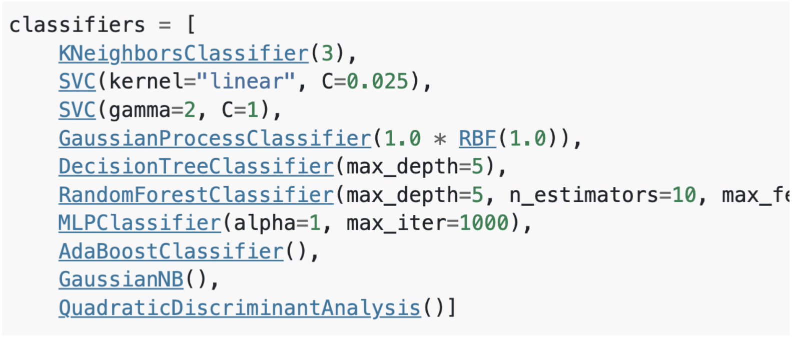
drug discovery

speech recognition



social science

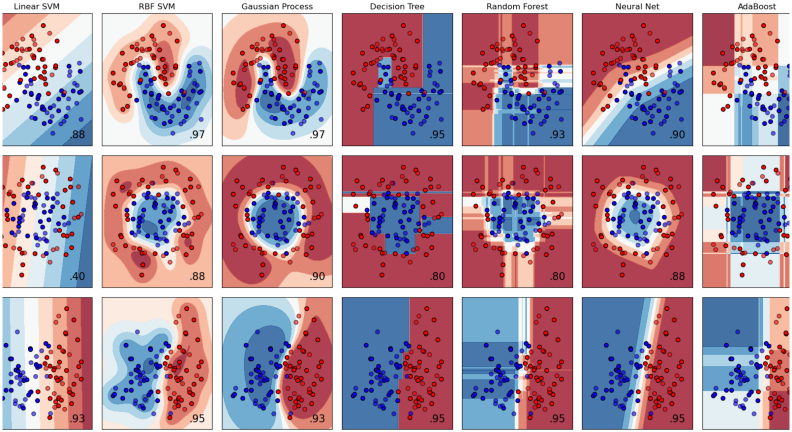
. . . so little time



```
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`

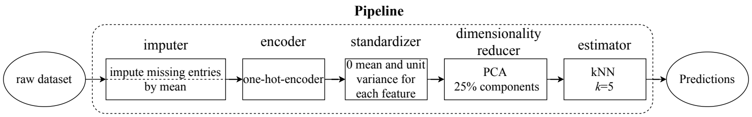
Different models perform differently



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?

Poll

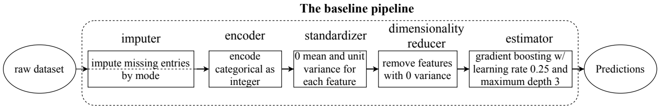
Which of these estimators do you think performs best most often for classification?

- logistic regression
- decision tree
- gradient boosting
- multilayer perceptron
- SVM

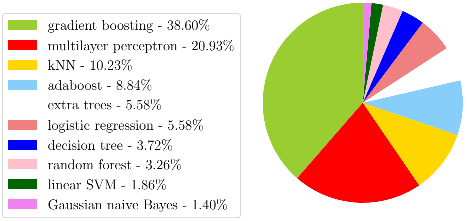
No Free Lunch

On 215 midsize OpenML classification datasets:

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


The baseline pipeline
raw dataset
imputer
impute missing entries by mode
encoder
encode categorical as integer
standardizer
0 mean and unit variance for each feature
dimensionality reducer
remove features with 0 variance
estimator
gradient boosting w/ learning rate 0.25 and maximum depth 3
Predictions

- The best estimator for each dataset:

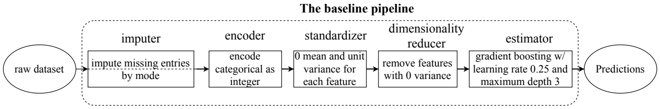


source: [Yang et al., 2020]

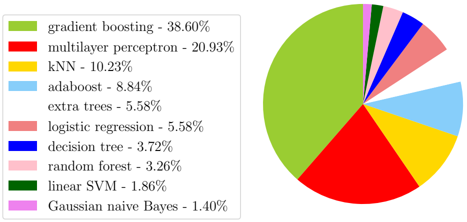
No Free Lunch

On 215 midsize OpenML classification datasets:

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


The baseline pipeline
raw dataset
imputer
impute missing entries by mode
encoder
encode categorical as integer
standardizer
0 mean and unit variance for each feature
dimensionality reducer
remove features with 0 variance
estimator
gradient boosting w/ learning rate 0.25 and maximum depth 3
Predictions

- The best estimator for each dataset:

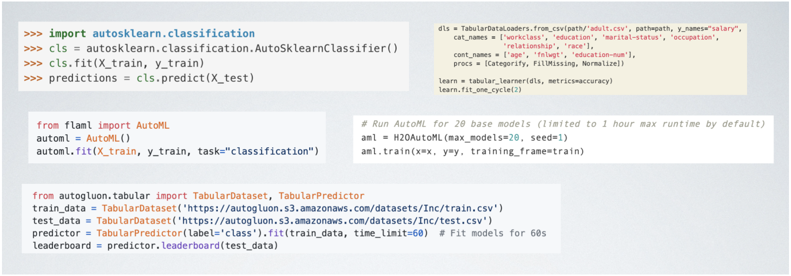


source: [Yang et al., 2020]

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.

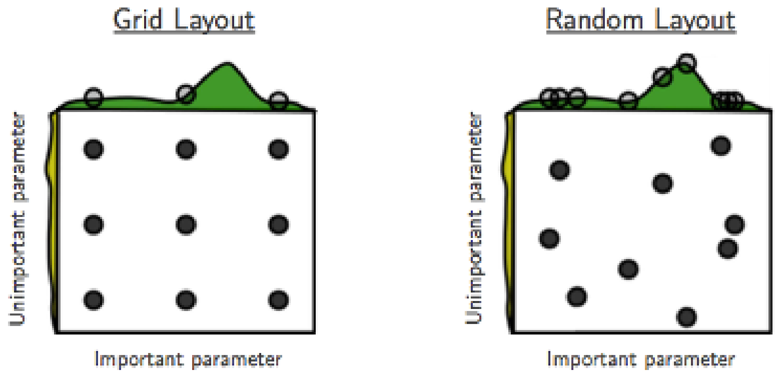
types of AutoML:

- **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

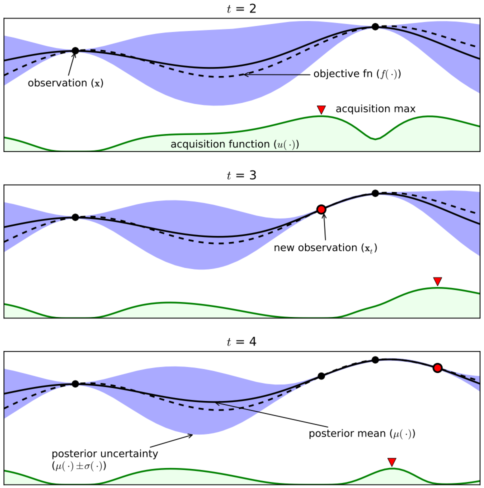
Grid search vs random search



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)

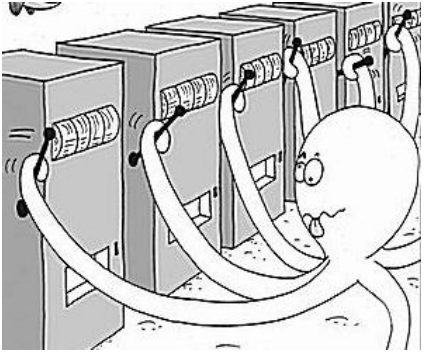


source: Brochu et al, 2010 [Brochu et al., 2010]

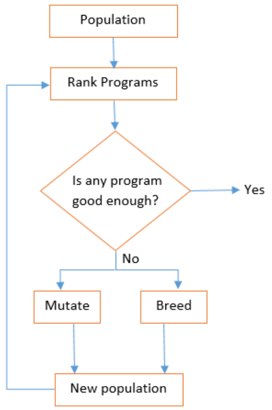
Multi-armed bandit

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



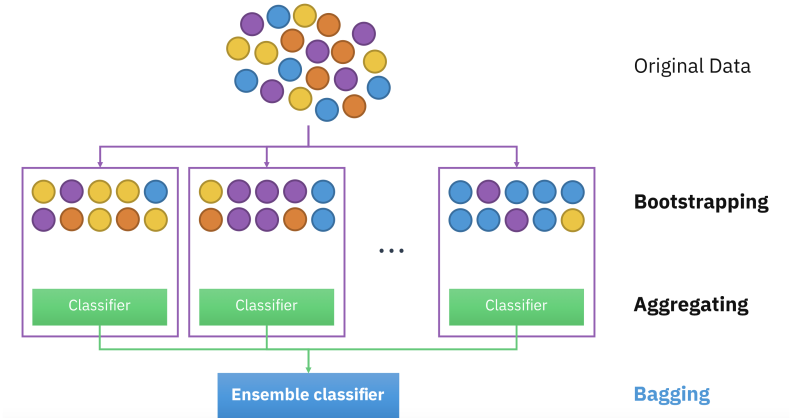
Genetic programming



source: dotnetlovers.com

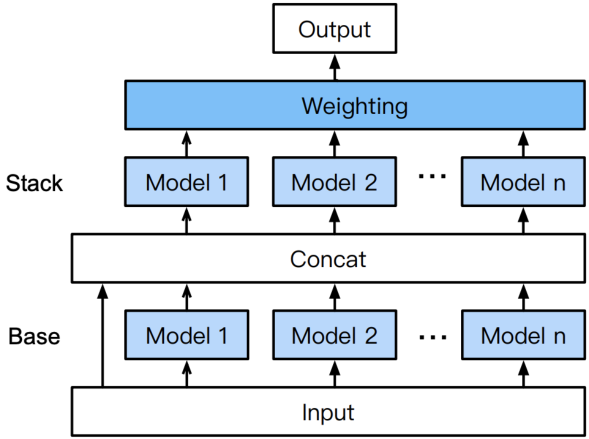
"Survival of the fittest": Automatically explore numerous possible pipelines to find the best for the given dataset

Ensemble



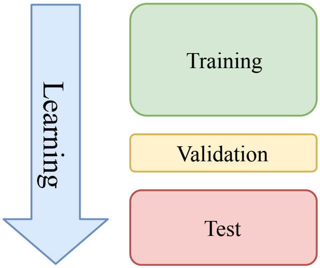
source: Sirakorn - CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=85888768

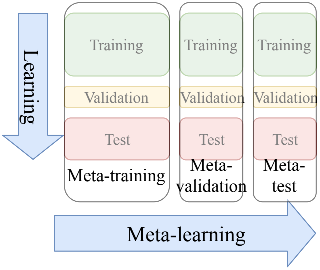
Stacking



source: AutoGluon Tabular [Erickson et al., 2020]

meta-learning





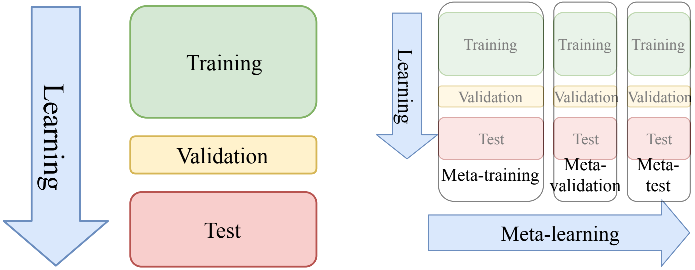
Learning meta-learning

source: OBOE [Yang et al., 2019]

can use meta-learning to

- generalize across datasets
- generalize across models
- pick a model on a new dataset without *any* expensive function evaluations

meta-learning



Learning   meta-learning

source: OBOE [Yang et al., 2019]

can use meta-learning to

- generalize across datasets
- generalize across models
- pick a model on a new dataset without *any* expensive function evaluations

but how can we featurize a dataset, or featurize a model?

Dataset meta-features

Meta-feature name	Explanation
number of instances	number of data points in the dataset
log number of instances	the (natural) logarithm of number of instances
number of classes	
number of features	
log number of features	the (natural) logarithm of number of features
number of instances with missing values	
percentage of instances with missing values	
number of features with missing values	
percentage of features with missing values	
number of missing values	
percentage of missing values	
number of numeric features	
number of categorical features	
ratio numerical to nominal	the ratio of number of numerical features to the number of categorical features
ratio numerical to nominal	
dataset ratio	the ratio of number of features to the number of data points
log dataset ratio	the natural logarithm of dataset ratio
inverse dataset ratio	
log inverse dataset ratio	
class probability (min, max, mean, std)	the (min, max, mean, std) of ratios of data points in each class
symbols (min, max, mean, std, sum)	the (min, max, mean, std, sum) of the numbers of symbols in all categorical features
kurtosis (min, max, mean, std)	
skewness (min, max, mean, std)	
class entropy	the entropy of the distribution of class labels (logarithm base 2)
landmarking meta-features [?]	
LDA	
decision tree	decision tree classifier with 10-fold cross validation
decision node learner	10-fold cross-validated decision tree classifier with `criterion="entropy"`, `max_depth=1`, `min_samples_split=2`, `min_samples_leaf=1`, `max_features=None`
random node learner	10-fold cross-validated decision tree classifier with `max_features=1` and the same above for the rest
1-NN	
PCA fraction of components for 95% variance	the fraction of components that account for 95% of variance
PCA kurtosis first PC	kurtosis of the dimensionality-reduced data matrix along the first principal component
PCA skewness first PC	skewness of the dimensionality-reduced data matrix along the first principal component

A simple meta-learning system: Auto-sklearn

offline, for all training datasets:

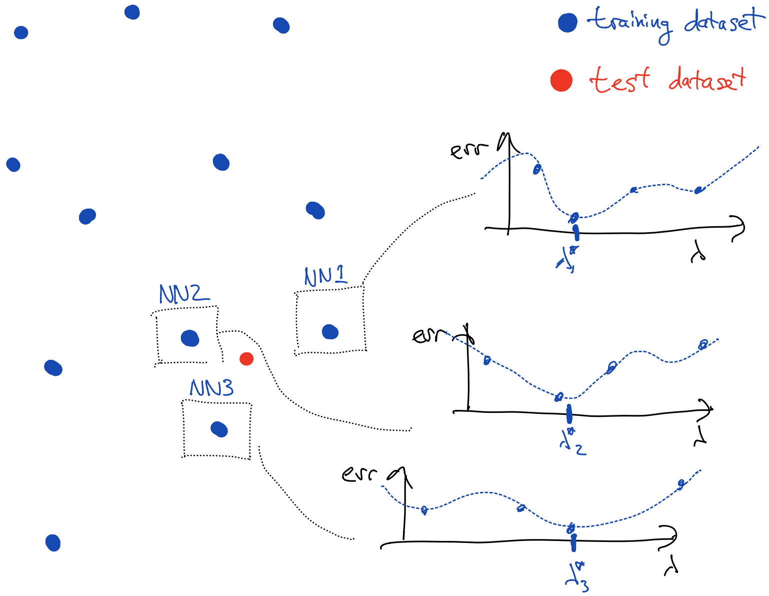
- compute dataset meta-features
- use Bayesian optimization to find the best model + hyperparameters

online, for test dataset:

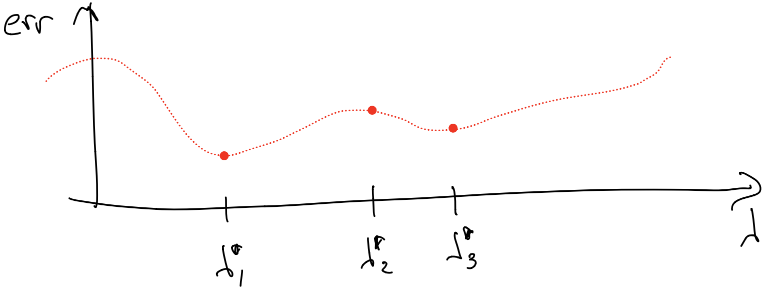
- compute dataset meta-features
- consider the best model + hyperparameters for *k* most similar datasets
- (optionally) tune hyperparameters further with Bayesian optimization
- fit models; form ensemble

source: Simplified from Auto-sklearn [Feurer et al., 2015]

A simple meta-learning system: Auto-sklearn



A simple meta-learning system: Auto-sklearn



source: Simplified from Auto-sklearn [Feurer et al., 2015]

Low-rank metalearning

our thesis: you can and should metalearn from the task itself

- run experiments on other datasets and fast-to-train models
- use low rank structure to metalearn

a similar approach to low-rank metalearning using Bayesian optimization: [Fusi et al., 2018]

OBOE: low rank autoML

given: n datasets, d machine learning models

OBOE: low rank autoML

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

OBOE: low rank autoML

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

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

source: OBOE [Yang et al., 2019]

OBOE: low rank autoML

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

$$Y \quad \approx \quad XW$$

$$\text{datasets}\left\{ \overbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}^{\text{models}} \right. \approx \begin{bmatrix} -x_1^T- \\ \vdots \\ -x_n^T- \end{bmatrix} \begin{bmatrix} | & & | \\ w_1 & \dots & w_d \\ | & & | \end{bmatrix}$$

source: OBOE [Yang et al., 2019]

OBOE: low rank autoML

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

$$Y \approx XW$$

$$\text{datasets}\left\{ \overbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ ? & ? & ? & ? & ? \end{bmatrix}}^{\text{models}} \right. \approx \begin{bmatrix} -x_1^T- \\ \vdots \\ -x_n^T- \\ ? \quad ? \end{bmatrix} \begin{bmatrix} | & & | \\ w_1 & \ldots & w_d \\ | & & | \end{bmatrix}$$

source: OBOE [Yang et al., 2019]

OBOE: low rank autoML

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

$$Y \approx XW$$

$$\text{datasets} \left\{ \overbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ ? & \times & \times & ? & \times \end{bmatrix}}^{\text{models}} \right. \approx \begin{bmatrix} -x_1^T- \\ \vdots \\ -x_n^T- \\ ? \quad ? \end{bmatrix} \begin{bmatrix} | & & | \\ w_1 & \dots & w_d \\ | & & | \end{bmatrix}$$

source: OBOE [Yang et al., 2019]

OBOE: low rank autoML

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

$$Y \approx XW$$

$$\text{datasets} \left\{ \overbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ ? & \times & \times & ? & \times \end{bmatrix}}^{\text{models}} \right. \approx \begin{bmatrix} -x_1^T- \\ \vdots \\ -x_n^T- \\ -x_{n+1}^T- \end{bmatrix} \begin{bmatrix} | & & | \\ w_1 & \dots & w_d \\ | & & | \end{bmatrix}$$

source: OBOE [Yang et al., 2019]

OBOE: low rank autoML

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

$$Y \approx XW$$

$$\text{datasets} \left\{ \overbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \cdot & \times & \times & \cdot & \times \end{bmatrix}}^{\text{models}} \right. \approx \begin{bmatrix} -x_1^T- \\ \vdots \\ -x_n^T- \\ -x_{n+1}^T- \end{bmatrix} \begin{bmatrix} | & & | \\ w_1 & \dots & w_d \\ | & & | \end{bmatrix}$$

OBOE: low rank autoML

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

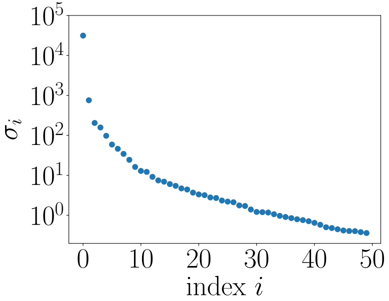
$$Y \approx XW$$

$$\text{datasets}\left\{\overbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \cdot & \times & \times & \cdot & \times \end{bmatrix}}^{\text{models}}\right. \approx \begin{bmatrix} -x_1^T- \\ \vdots \\ -x_n^T- \\ -x_{n+1}^T- \end{bmatrix} \begin{bmatrix} | & & | \\ w_1 & \dots & w_d \\ | & & | \end{bmatrix}$$

- rows $x_i \in \mathbf{R}^k$ of X are *dataset metafeatures*
- columns $w_j \in \mathbf{R}^k$ of W are *model metafeatures*
- $x_i^T w_j \approx Y_{ij}$ are *predicted model performance*

source: OBOE [Yang et al., 2019]

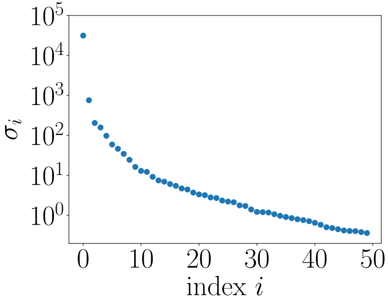
Is AutoML really low rank?



tradeoff:

- model improves with higher rank
- required experiments increase with higher rank

Is AutoML really low rank?

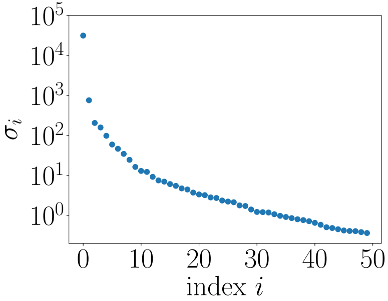


tradeoff:

- model improves with higher rank
- required experiments increase with higher rank

our approach: increase rank until you run out of time

Is AutoML really low rank?



tradeoff:

- model improves with higher rank
- required experiments increase with higher rank

our approach: increase rank until you run out of time

(most square-ish data matrices are approximately low rank [Udell and Townsend, 2018])

Value of information

- want to find unknown vector $x \in \mathbf{R}^k$

Value of information

- want to find unknown vector $x \in \mathbf{R}^k$
- pick set of measurements $y_j \in \mathbf{R}^k$, $j \in S \subseteq [1, \ldots, n]$

Value of information

- want to find unknown vector $x \in \mathbf{R}^k$
- pick set of measurements $y_j \in \mathbf{R}^k$, $j \in S \subseteq [1, \ldots, n]$
- measure $a_j = x^T y_j + \epsilon_j$ where $\epsilon_j \sim \mathcal{N}(0, 1)$ iid

Value of information

- want to find unknown vector $x \in \mathbf{R}^k$
- pick set of measurements $y_j \in \mathbf{R}^k$, $j \in S \subseteq [1, \ldots, n]$
- measure $a_j = x^T y_j + \epsilon_j$ where $\epsilon_j \sim \mathcal{N}(0, 1)$ iid
- estimate x via least squares:

$$\hat{x} = (YY^T)^{-1} Ya = (YY^T)^{-1} Y(Y^T x + \epsilon)$$

where $Y = [y_j]_{j \in S}$

Value of information

- want to find unknown vector $x \in \mathbf{R}^k$
- pick set of measurements $y_j \in \mathbf{R}^k$, $j \in S \subseteq [1, \dots, n]$
- measure $a_j = x^T y_j + \epsilon_j$ where $\epsilon_j \sim \mathcal{N}(0, 1)$ iid
- estimate x via least squares:

$$\hat{x} = (YY^T)^{-1} Ya = (YY^T)^{-1} Y(Y^T x + \epsilon)$$

  where $Y = [y_j]_{j \in S}$
- hence

$$\begin{aligned} \mathbf{E}(\hat{x}) &= x \\ \mathbf{var}(\hat{x}) &= (YY^T)^{-1} = \left(\sum_{j \in S} y_j y_j^T \right)^{-1} \end{aligned}$$

Experiment design for timely model selection

Which algorithms to use to predict performance?

$$\begin{aligned} \underset{v_j}{\text{minimize}} \quad & -\log\det\left(\sum_{j=1}^{n} v_j y_j y_j^T\right) \\ \text{subject to} \quad & \sum_{j=1}^{n} v_j \hat{t}_j \leq \tau \\ & v_j \in \{0,1\} \qquad \forall j \in [n]. \end{aligned}$$

- $\hat{t}_j$: estimated runtime of each machine learning model
- τ: runtime budget

Experiment design for timely model selection

Which algorithms to use to predict performance?

$$\begin{array}{ll} \underset{v_j}{\text{minimize}} & -\log\det\left(\sum_{j=1}^{n} v_j y_j y_j^T\right) \\ \text{subject to} & \sum_{j=1}^{n} v_j \hat{t}_j \leq \tau \\ & v_j \in \{0,1\} \qquad \forall j \in [n]. \end{array}$$

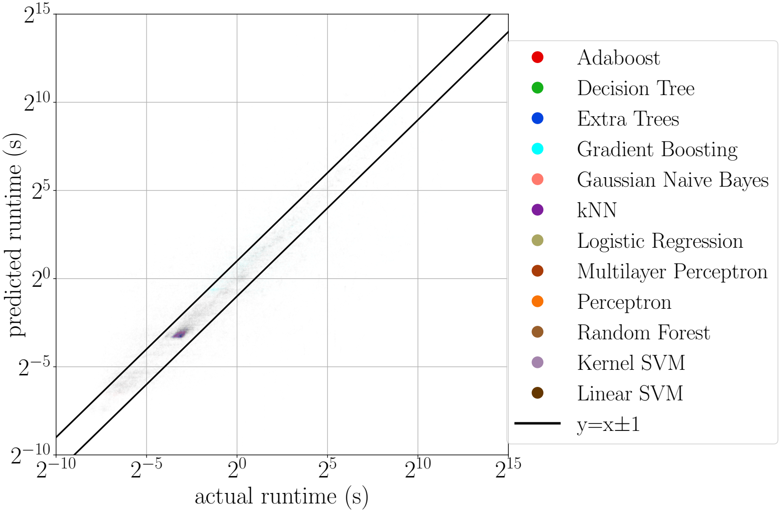
- $\hat{t}_j$: estimated runtime of each machine learning model
- τ: runtime budget

to solve:

- relax to semidefinite program [Yang et al., 2019]
- use greedy algorithm; submodularity guarantees good performance [Yang et al., 2020]

Estimated runtime

estimate runtime using polynomial regression on (# datapoints, # features)



Oboe: Does it work?

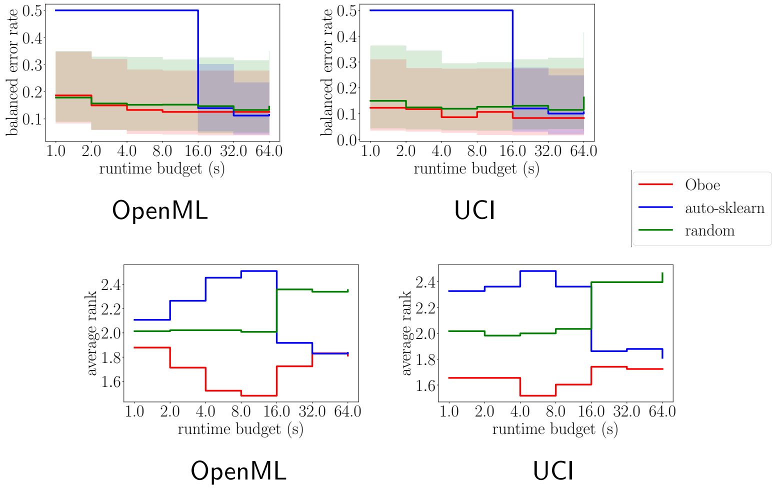
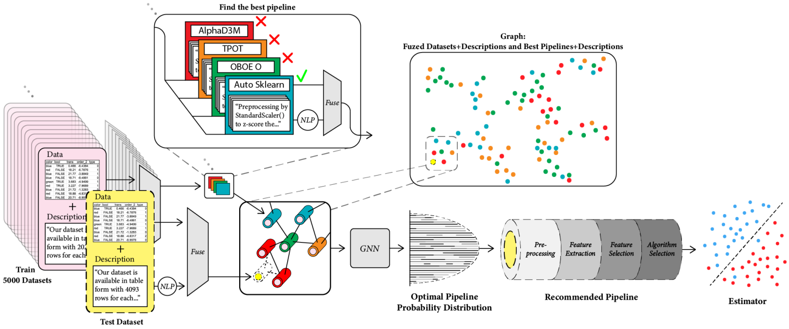


Figure: In 3a and 3b, shaded area = 75th–25th percentile.
In 3c and 3d, rank 1 is best and 3 is worst.

Metalearning with NLP and GNNs



source: Real-time AutoML [Drori et al., 2020]

Outline

AutoML systems

Optimizing over scikit-learn style models:

- **Auto-WEKA** [Thornton et al., 2013]: BO on conditional search space
- **auto-sklearn** [Feurer et al., 2015]: meta-learning + BO
- **TPOT** [Olson et al., 2016]: genetic programming
- **Hyperband** [Li et al., 2018]: multi-armed bandit
- **PMF** [Fusi et al., 2018]: matrix factorization + BO
- **Oboe** [Yang et al., 2019]: matrix factorization + experiment design
- **AutoGluon** [Erickson et al., 2020]: ensembling, stacking
- **FLAML** [Wang et al., 2020]: multi-armed bandit
- ...

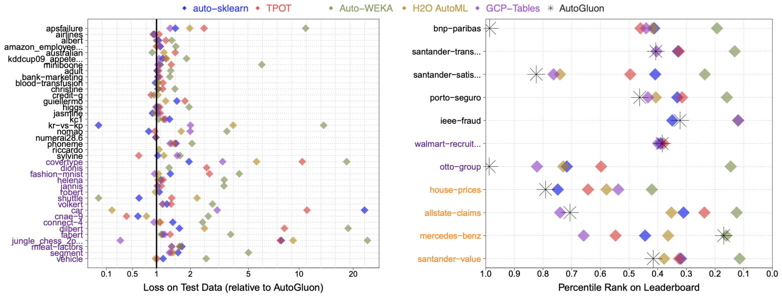
commercial tools:

- Google AutoML Tabular
- Microsoft Azure AutoML
- Amazon AutoGluon on SageMaker
- H2O AutoML

Neural architecture search (NAS)

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

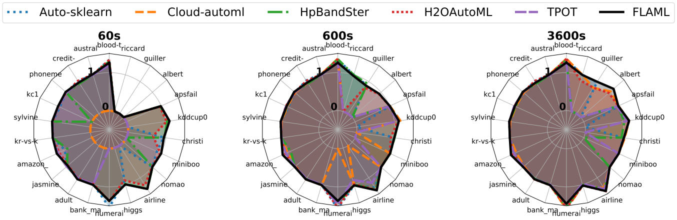
Lots of good options!



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

source: AutoGluon Tabular [Erickson et al., 2020]

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]

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

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

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

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

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

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

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

References II

Jin, H., Song, Q., and Hu, X. (2019).
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. ACM.

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

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2018).
Hyperband: A novel bandit-based approach to hyperparameter optimization.
Journal of Machine Learning Research, 18(185):1–52.

Olson, R. S., Urbanowicz, R. J., Andrews, P. C., Lavender, N. A., Kidd, L. C., and Moore, J. H. (2016).
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.

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

Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2013).
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.

References III

Udell, M. and Townsend, A. (2018).
Why are big data matrices approximately low rank?
SIAM Mathematics of Data Science (SIMODS), to appear.

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

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

Yang, C., Akimoto, Y., Kim, D. W., and Udell, M. (2019).
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.

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

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