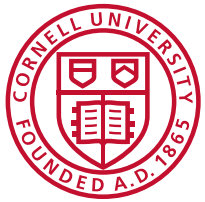


Introduction to Collaborative Filtering for Automated Machine Learning

Chengrun Yang (cy438)

ORIE 7391

March 16, 2022

Quiz

1. Meta-learning uses information from past datasets to guide learning on a new dataset.

a. True

b. False

2. Dataset similarity can be characterized by the closeness of dataset meta-features.

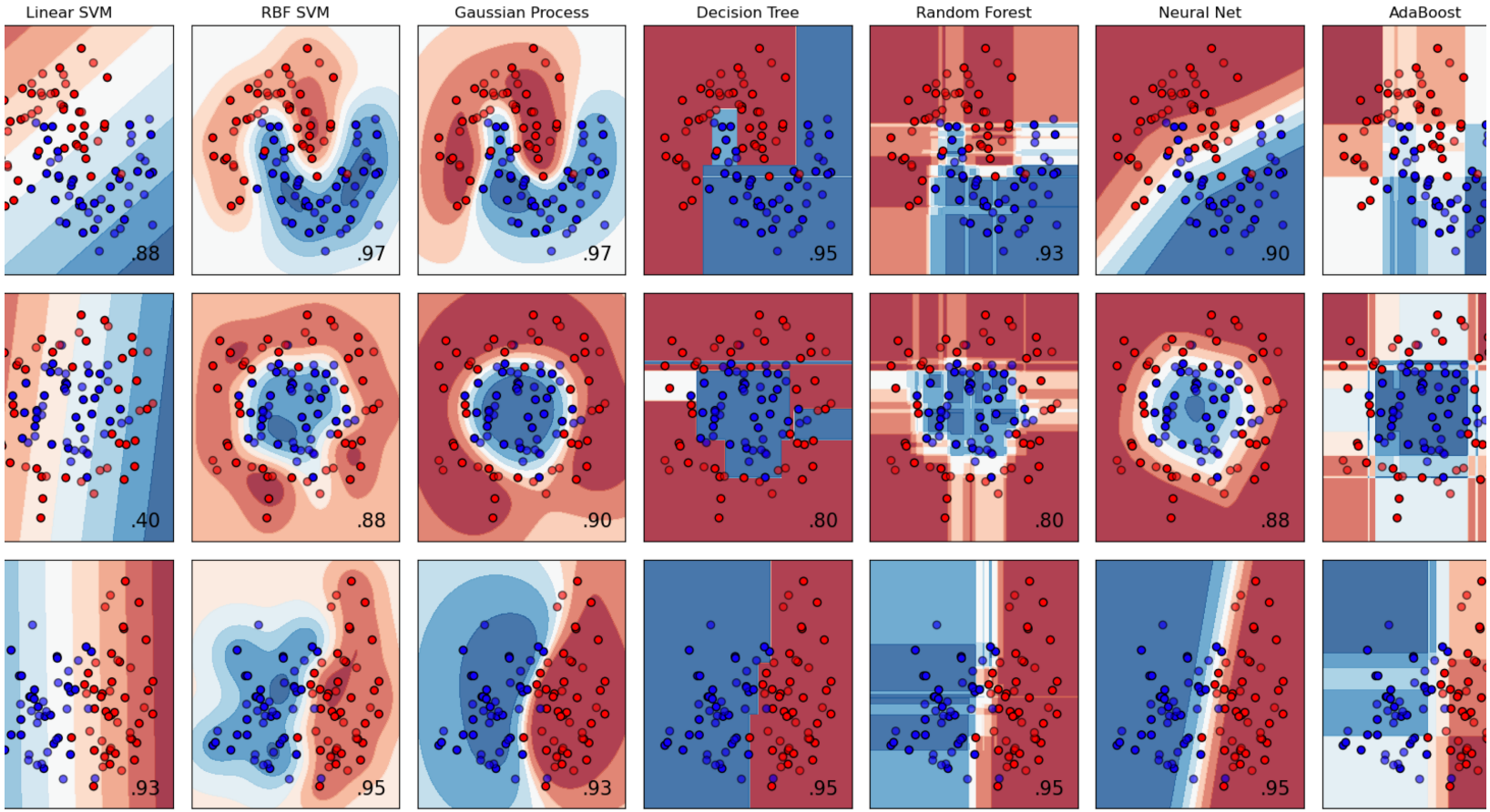
a. True

b. False

Outline

Best ML model depends on the dataset

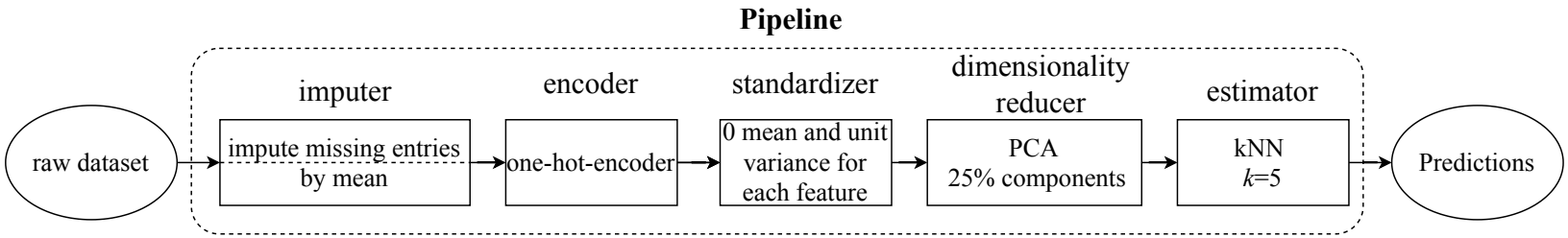
Decision boundaries of classifiers:



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

The machine learning pipeline space is huge

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

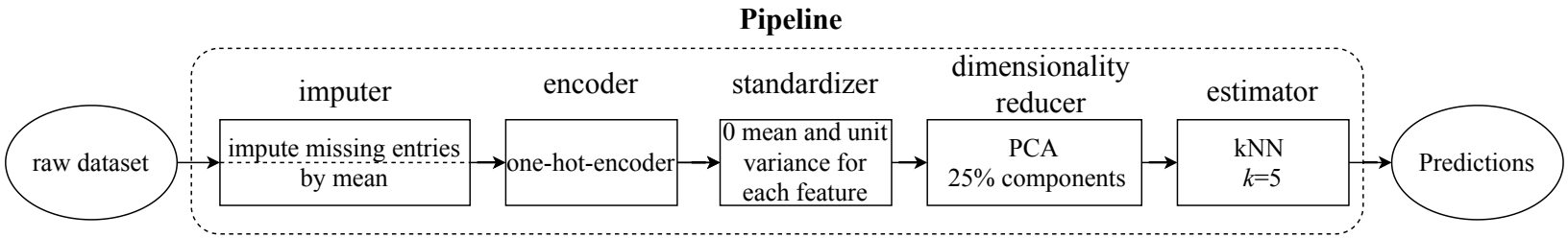


Data scientists have so many choices to make:

- data imputer: impute by mean, or median, or zero? ...
- encoder: one-hot encode? ...
- standardizer: rescale each feature? ...
- dimensionality reducer: PCA, or select by variance? ...
- estimator: use decision tree or logistic regression? ...

The machine learning pipeline space is huge

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



Data scientists have so many choices to make:

- data imputer: impute by mean, or median, or zero? ...
- encoder: one-hot encode? ...
- standardizer: rescale each feature? ...
- dimensionality reducer: PCA, or select by variance? ...
- estimator: use decision tree or logistic regression? ...

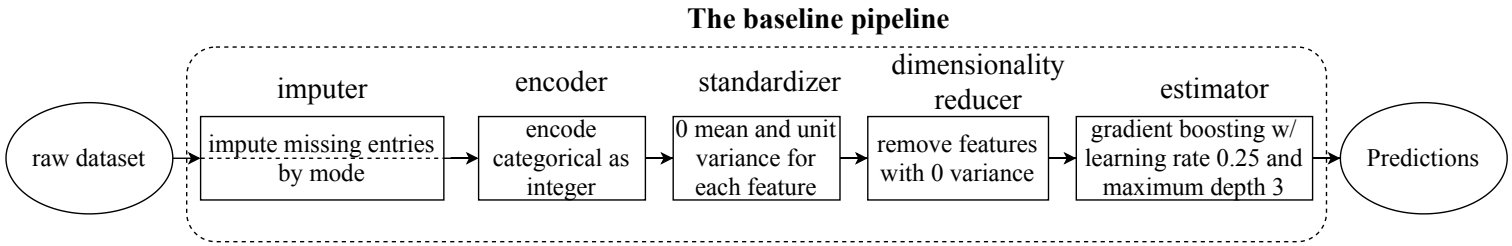
🤔 In this combinatorially large search space

1. impossible to enumerate all choices on large datasets
2. too expensive on small datasets

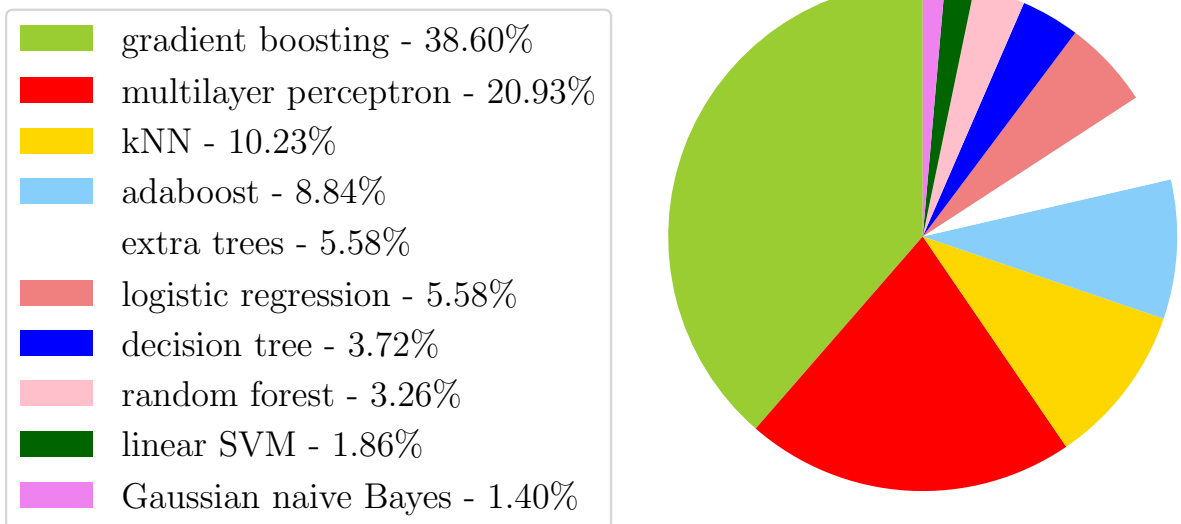
No Free Lunch

On 215 midsize OpenML classification datasets:

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



- the best estimator for each dataset:



Definitions

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

types of AutoML:

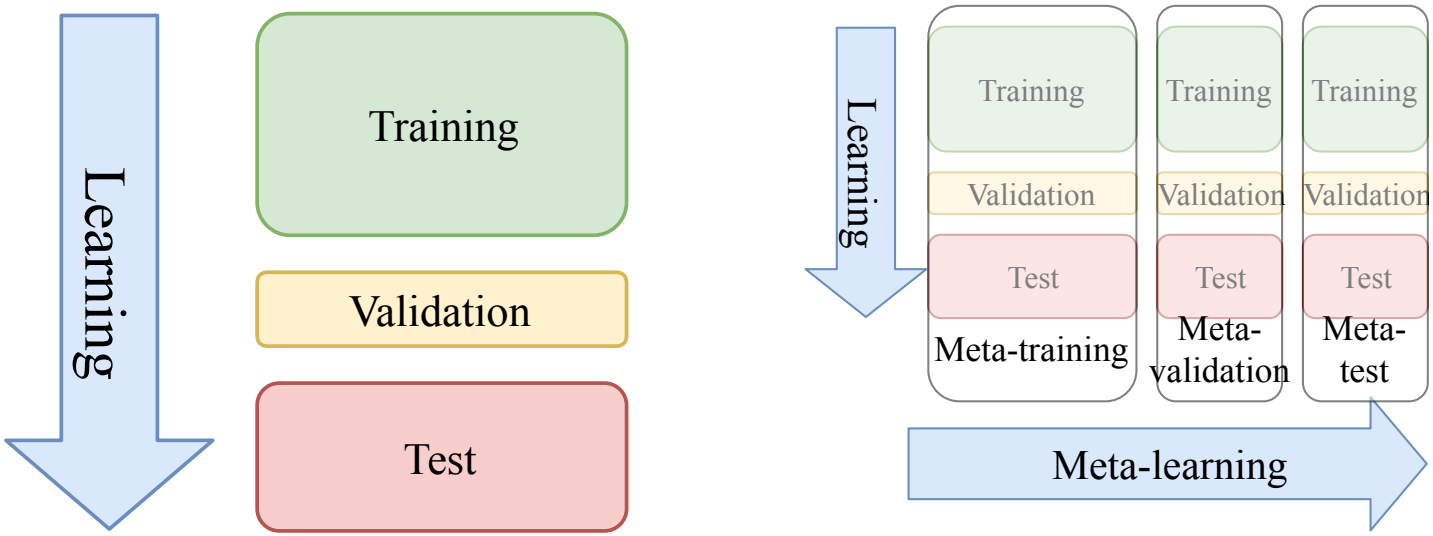
- **hyperparameter tuning** chooses the best hyperparameters for the model
- **combined algorithm and hyperparameter search (CASH)** chooses an estimator and hyperparameters
- **neural architecture search (NAS)** chooses a deep learning architecture
  e.g., number of layers, type of layer, width, activation type (will be discussed in a future class)
- **meta-learning**, or learning to learn, generalizes from a corpus of tasks to a new task

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

Techniques

- grid vs random search, *e.g.*, [BB12]
- bayesian optimization, *e.g.*, [BCDF10]
- genetic algorithms, *e.g.*, TPOT [OUA+16]
- multi-armed bandits, *e.g.*, Hyperband [LJD+18]
- ensembles, *e.g.*, random forest
- stacking, *e.g.*, [EMS+20]
- **meta-learning** *e.g.*, [FKE+15], [FSE18], [YAKU19]
- ...

Learning vs meta-learning



(a) Learning (b) Meta-learning

source: OBOE [YAKU19]

can use meta-learning to

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

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 [PBGC00]	
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

Outline

A simple meta-learning system: Auto-sklearn

offline, for all training datasets:

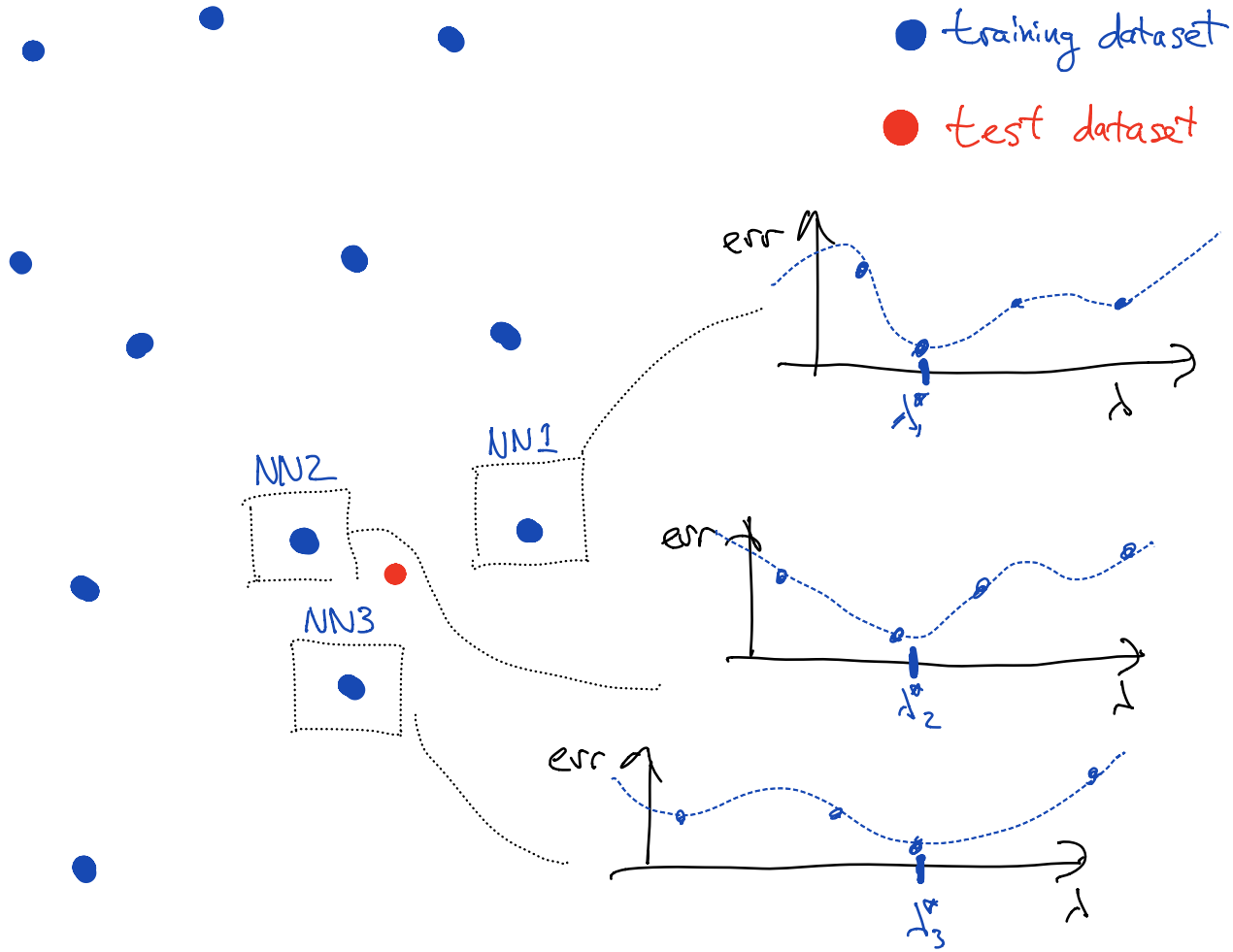
- compute dataset meta-features
- use Bayesian optimization to find the best model + hyperparameters

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

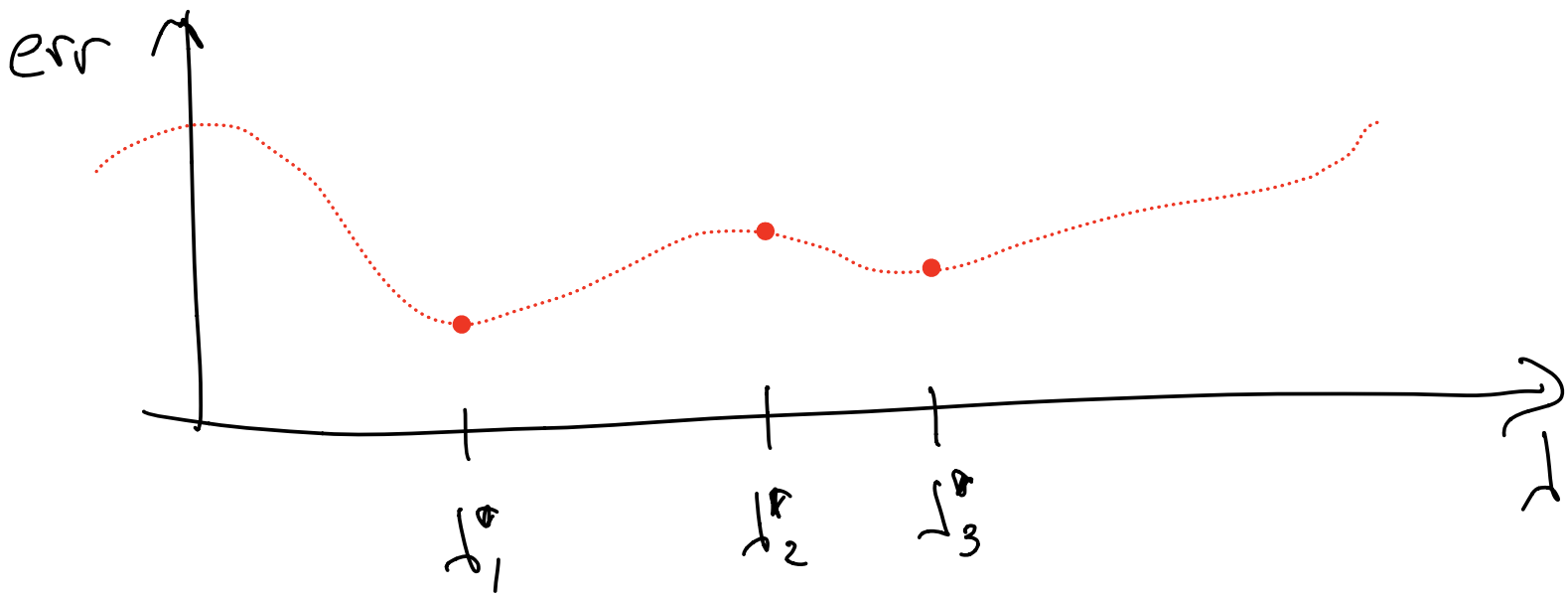
source: Simplified from Auto-sklearn [FKE+15]

A simple meta-learning system: Auto-sklearn



source: Simplified from Auto-sklearn [FKE+15]

A simple meta-learning system: Auto-sklearn



source: Simplified from Auto-sklearn [FKE+15]

Outline

Motivation

Latent spaces given by matrix factorization captures the structure in the space of ML pipelines.

given: m datasets, n machine learning pipelines
measure: cross-validation error of each pipeline on each dataset
form: $m \times n$ error matrix E
find: $X \in \mathbf{R}^{m \times k}$, $Y \in \mathbf{R}^{k \times n}$ for which

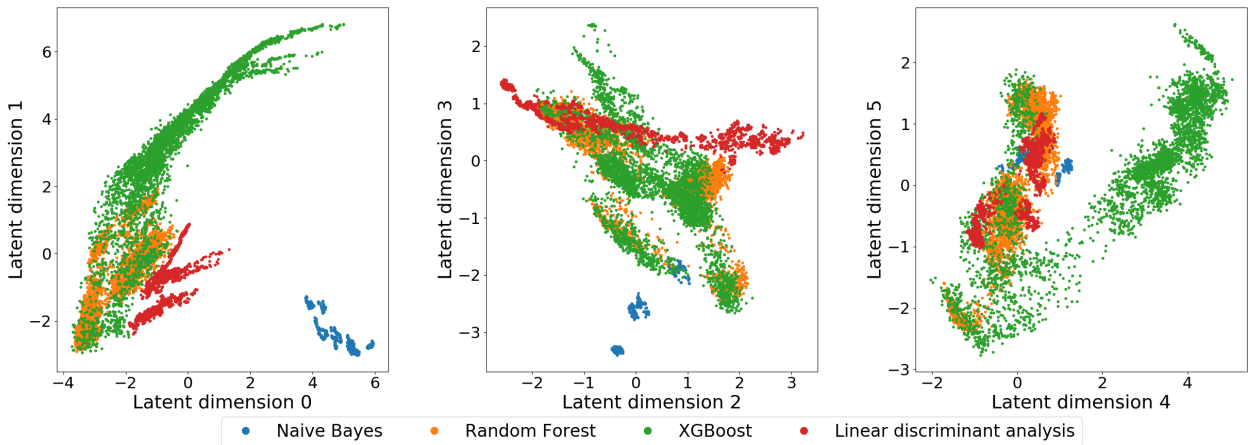
$$A \approx XY$$

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

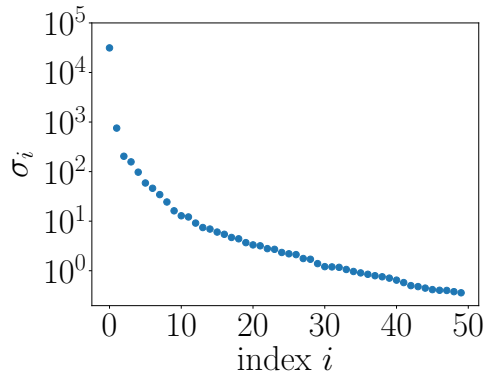
- rows $x_i \in \mathbf{R}^k$ of X are *dataset embeddings*
- columns $y_j \in \mathbf{R}^k$ of Y are *pipeline embeddings*

Low rank structures visualization

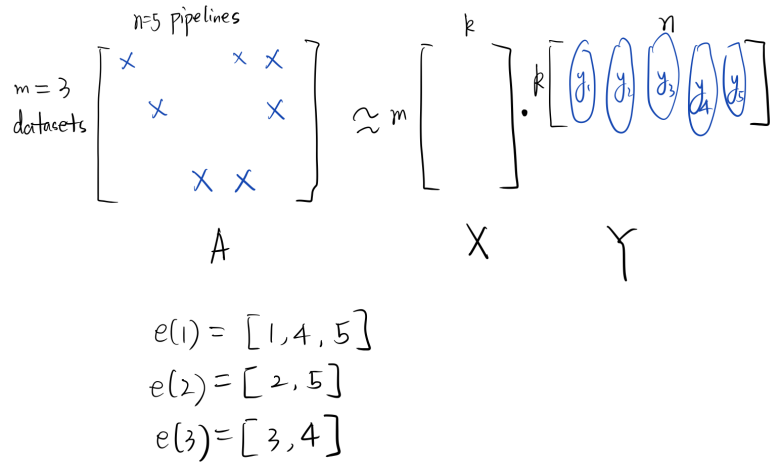
PMF [FSE18]: embeddings of 42,000 pipelines, colored by estimator type



Oboe [YAKU19]: singular value decay of the error matrix



PMF meta-training: probabilistic matrix factorization



With

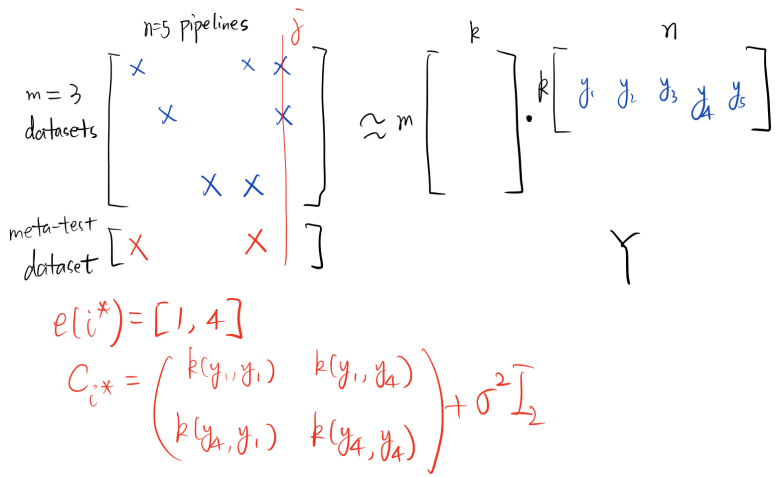
1. the RBF kernel over $\{v_i\}_{i\in[n]}$ being a matrix $K \in \mathbb{R}^{n\times n}$, with the (i,j)-th entry $K_{i,j} = k(v_i, v_j) := \alpha \exp\left(-\frac{\gamma_q}{2}||v_i - v_j||^2\right)$
2. $e(i)$ being the list of indices of evaluated pipelines on dataset i
3. dataset-wise covariance $C_i = K(Y_{e(i)}, Y_{e(i)}) + \sigma^2 I$, in which K is the RBF kernel over $\{Y_j\}_{j\in e(i)}$
4. RBF kernel hyperparameters $\theta = \{\alpha, \gamma_1, \ldots, \gamma_q\}$,

PMF meta-training **maximizes the posterior probability**

$$\mathbb{P}(A \mid Y, \theta, \sigma^2) = \int p(A \mid Y, f)\, p(f \mid Y)\, df = \prod_{i=1}^{m} \mathcal{N}(A_{i,e(i)} \mid 0,\ C_i),$$

over Y and θ.

PMF meta-test: Gaussian processes regression + expected improvement



On a new dataset i^*, with

1. posterior mean $\mu_{i^*j} = K_{e(i^*),j}^\top \, C_{i^*}^{-1} \, a_{i^*,e(i^*)}$
2. posterior variance $v_{i^*j} = K_{j,j} + \sigma^2 - K_{e(i^*),j}^\top C_{i^*}^{-1} K_{e(i^*),j}$
3. dataset-wise covariance $C_i = K(Y_{e(i)}, Y_{e(i)}) + \sigma^2 I$

the predicted performance of pipeline j is given by

$$\mathbb{P}(a_{i^*j} \mid Y, \theta, \sigma^2) = \mathcal{N}(a_{i^*j} \mid \mu_{i^*j}, v_{i^*j}).$$

Oboe meta-training: matrix factorization

given: m datasets, n machine learning pipelines
measure: cross-validation error of each pipeline on each dataset
form: $m \times n$ error matrix E
find: $X \in \mathbf{R}^{m \times k}$, $Y \in \mathbf{R}^{k \times n}$ for which

$$A \approx XY$$

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

- rows $x_i \in \mathbf{R}^k$ of X are *dataset embeddings*
- columns $y_j \in \mathbf{R}^k$ of Y are *pipeline embeddings*

Oboe meta-test: experiment design + linear regression

$$A \approx XY$$

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

- rows $x_i \in \mathbf{R}^k$ of X are *dataset embeddings*
- columns $y_j \in \mathbf{R}^k$ of Y are *pipeline embeddings*
- red crosses: pipelines to evaluate, selected by experiment design

Oboe meta-test: experiment design + linear regression (continued)

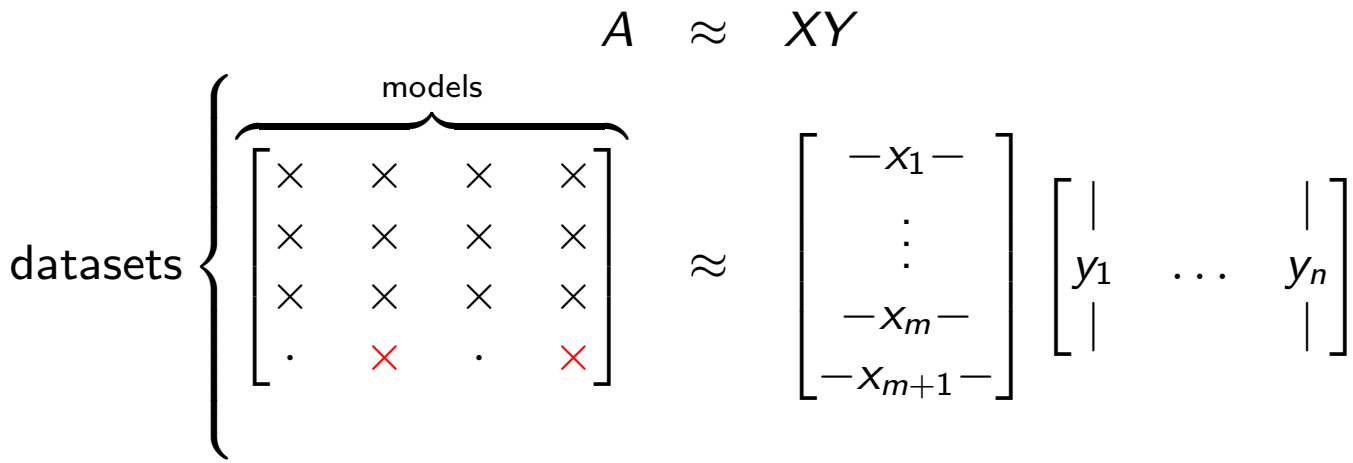
Experiment design: select a few **cheap but informative** pipelines to run on the new dataset

$$\begin{aligned} \text{maximize} \quad & \log\det \underbrace{\left(\sum_{j \in S} y_j y_j^\top\right)}_{\text{Fisher information of } x_{m+1}} \\ \text{subject to} \quad & \textstyle\sum_{j \in S} \hat{t}_j \leq \tau \\ & S \subseteq [n] \end{aligned}$$

- $\hat{t}_j$: estimated runtime of Model j
- τ: runtime budget

Oboe meta-test: experiment design + linear regression (continued)

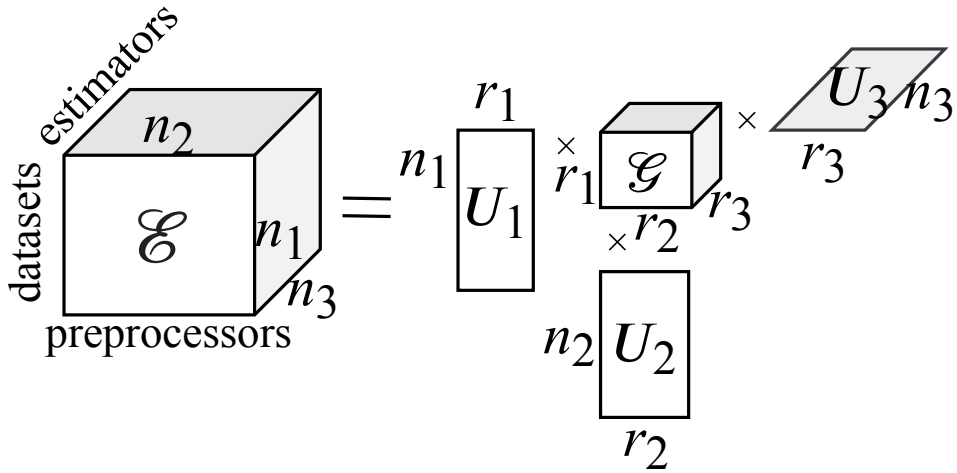
Linear regression: predict performance of other pipelines



- rows $x_i \in \mathbf{R}^k$ of X are *dataset embeddings*
- columns $y_j \in \mathbf{R}^k$ of Y are *pipeline embeddings*
- red crosses: pipelines to evaluate (selected by experiment design)
- $x_{m+1} y_j \approx A_{m+1,j}$ are *predicted model performance*

TensorOboe: tensors capture higher-order structures

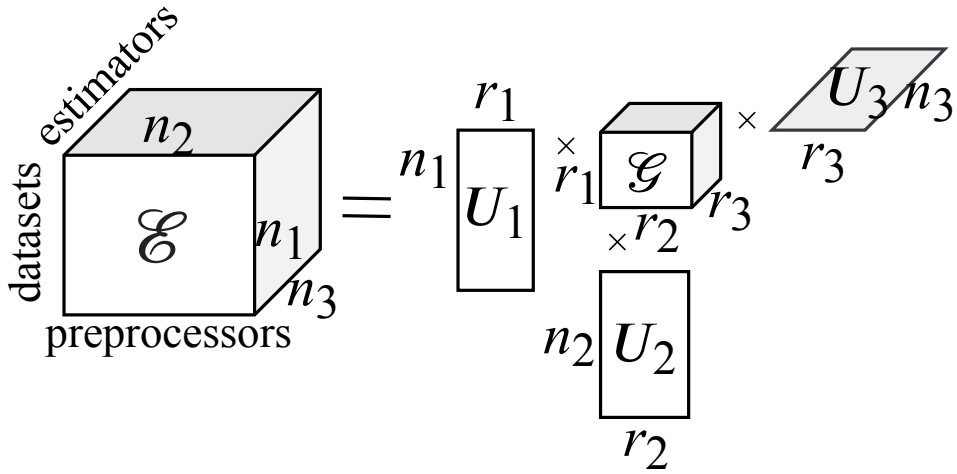
An example of a 3-dimensional error tensor:



- $\mathcal{E}$: error tensor $\mathbb{R}^{n_1 \times n_2 \times n_3}$
- $\mathcal{E}_{ijk}$: performance of the pipeline from Preprocessor j and Estimator k on Dataset i
- $[r_1, r_2, r_3]$: multilinear rank
- columns of U_1: dataset embeddings
- columns of U_2: preprocessor embeddings
- columns of U_3: estimator embeddings

TensorOboe: tensors capture higher-order structures

An example of a 3-dimensional error tensor:



- $\mathcal{E}$: error tensor $\mathbb{R}^{n_1 \times n_2 \times n_3}$
- $\mathcal{E}_{ijk}$: performance of the pipeline from Preprocessor j and Estimator k on Dataset i
- $[r_1, r_2, r_3]$: multilinear rank
- columns of U_1: dataset embeddings
- columns of $Y = (\mathcal{G} \times_2 U_2 \times_3 U_3)^{(1)}$: pipeline embeddings

Outline

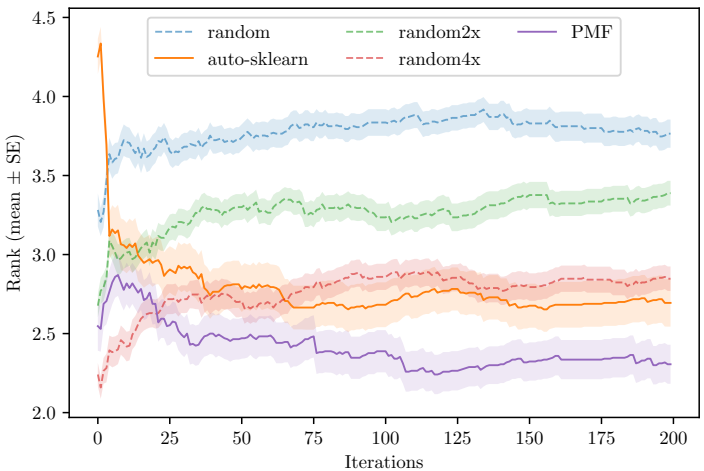
Experiment setup

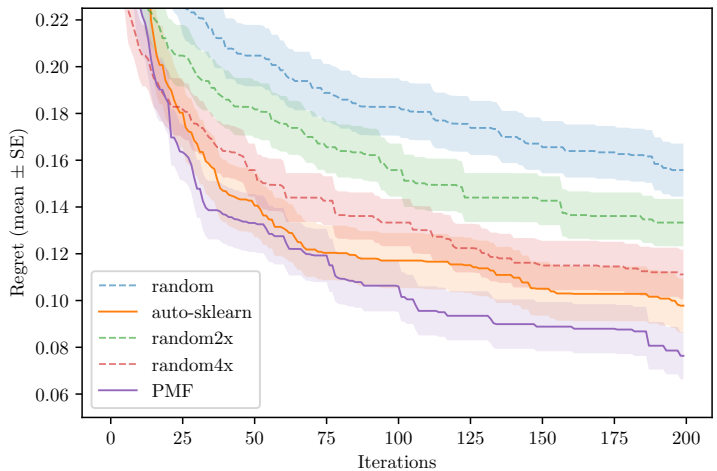
- datasets: hundreds, from public dataset repository
- pipelines: hundreds to tens of thousands
- evaluation:
  training-test split, or cross-validation across datasets

PMF experiments

Regret: accuracy of the true best pipeline - accuracy of the pipeline found by the method

Ranking and regret plots across auto-sklearn, random (2x and 4x) and PMF:



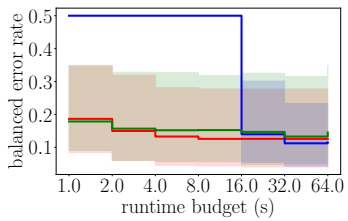


(42,000 pipelines in total)

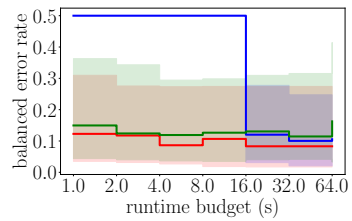
source: PMF [FSE18]

Oboe experiments

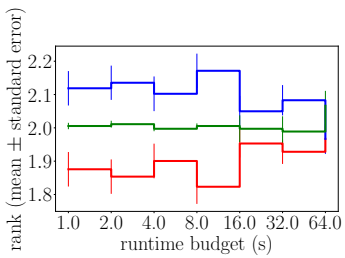
Experiment-design and linear regression in Oboe are much faster than meta-feature computation in auto-sklearn:



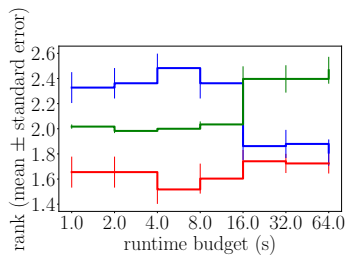
(a) OpenML (meta-LOOCV)



(b) UCI (meta-test)



(c) OpenML (meta-LOOCV)



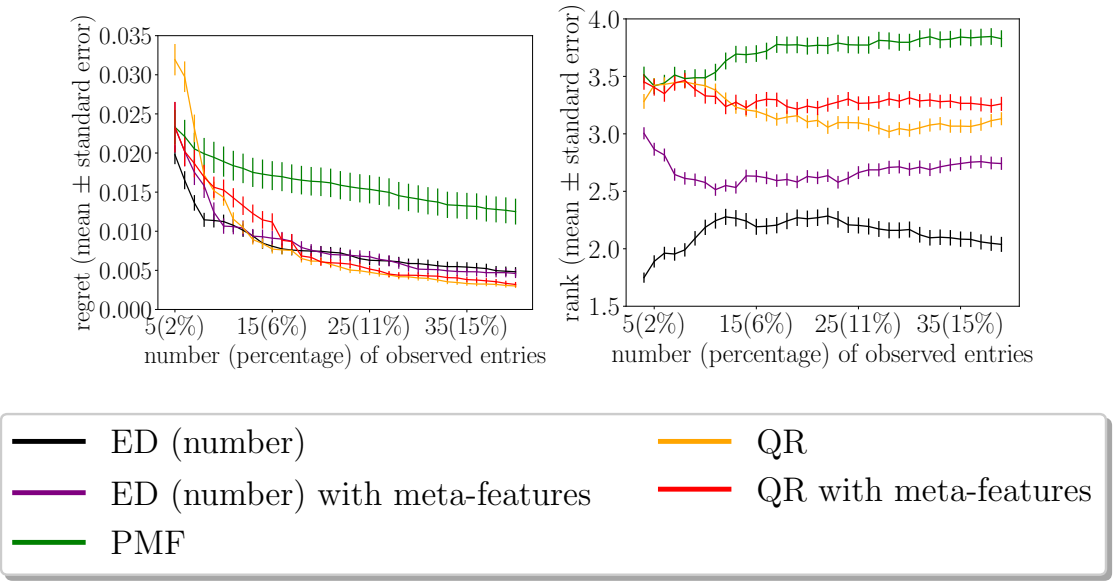
(d) UCI (meta-test)

- red: Oboe with experiment design
- blue: auto-sklearn
- green: Oboe with random initializations

source: OBOE [YAKU19]

Oboe experiments (continued)

Experiment design finds more informative pipelines than GP regression:



- ED (number):
  maximize $\log\det\left(\sum_{j\in S} y_j y_j^\top\right)$ subject to $|S| \leq \ell, S \subseteq [n]$
- "with metafeatures": pick the first 5 pipelines by closeness of meta-features

source: OBOE [YAKU19]

Discussion

How do trends in ML impact AutoML?

- cheaper compute $\Rightarrow$ less interested in model selection?
- deployment of deep learning $\Rightarrow$ more interested in neural architecture search?
- different data types, different methods
- performance gain from new methods vs. from better implementation
- ...

References I

James Bergstra and Yoshua Bengio.

Random search for hyper-parameter optimization.
Journal of machine learning research, 13(2), 2012.

Eric Brochu, Vlad M Cora, and Nando 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.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola.

Autogluon-tabular: Robust and accurate automl for structured data.
arXiv preprint arXiv:2003.06505, 2020.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter.

Efficient and robust automated machine learning.
Advances in neural information processing systems, 28, 2015.

Nicolo Fusi, Rishit Sheth, and Melih Elibol.

Probabilistic matrix factorization for automated machine learning.
Advances in neural information processing systems, 31, 2018.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar.

Hyperband: A novel bandit-based approach to hyperparameter optimization.
Journal of Machine Learning Research, 18(185):1–52, 2018.

References II

Randal S Olson, Ryan J Urbanowicz, Peter C Andrews, Nicole A Lavender, La Creis Kidd, and Jason H Moore.
Automating biomedical data science through tree-based pipeline optimization.
In *European conference on the applications of evolutionary computation*, pages 123–137. Springer, 2016.

Bernhard Pfahringer, Hilan Bensusan, and Christophe G Giraud-Carrier.
Meta-learning by landmarking various learning algorithms.
In *ICML*, pages 743–750, 2000.

Chengrun Yang, Yuji Akimoto, Dae Won Kim, and Madeleine 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, 2019.

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

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