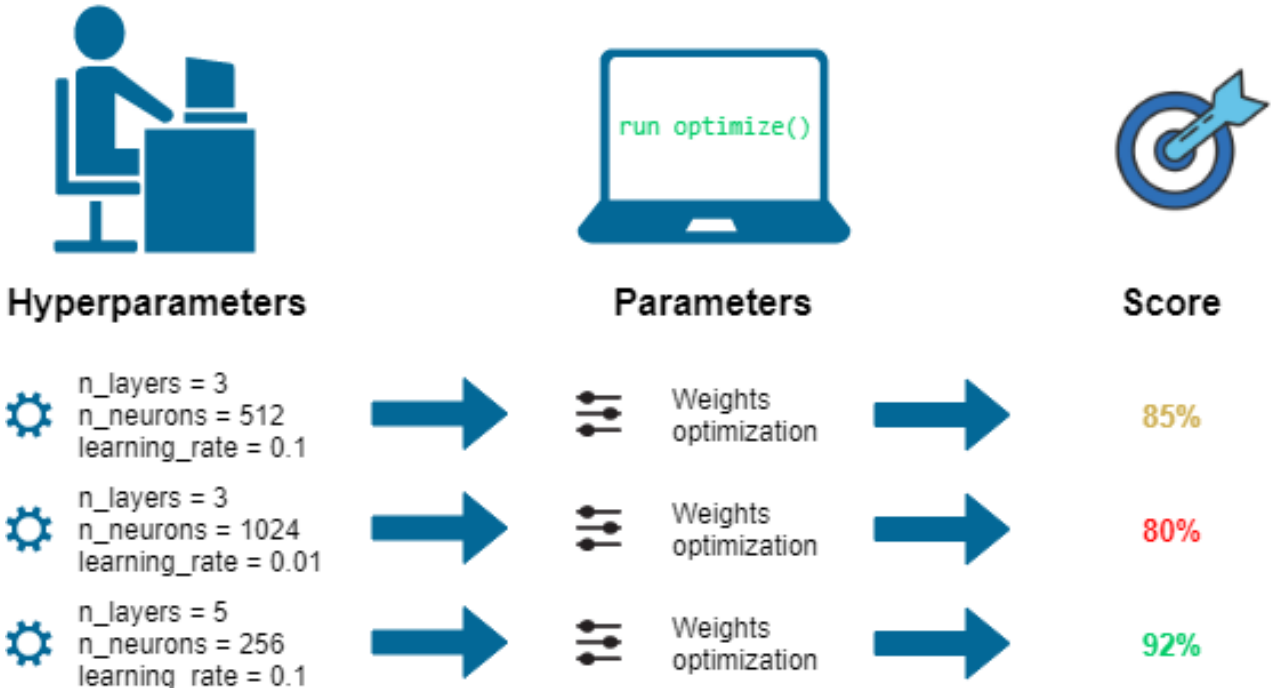
ML Parameters, Hyper-Parameters, HPO, CASH & AutoML

Basics

- Machine Learning models are composed of two different types of parameters:
  - **Hyperparameters** are all the parameters which can be arbitrarily set by the user before starting training (eg. number of estimators in Random Forest).
  - **Model parameters** are instead learned during the model training (eg. weights in Neural Networks, Linear Regression).
- Model parameters define how to use input data to get the desired output and are learned at training time.
- Hyperparameters determine how the model is structured in the first place.

Hyperparameter Tuning



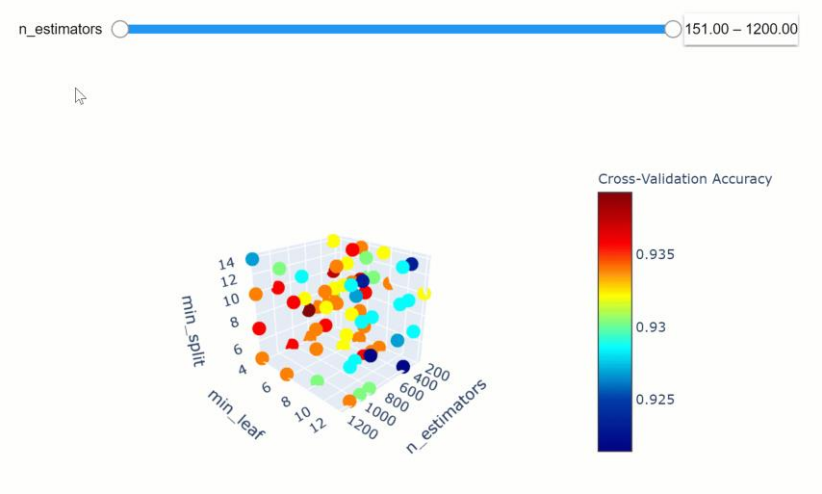
- Hyperparameters are (typically) set before the training process begins and are not (typically) learned from the data apriori.
- Examples of hyperparameters include the learning rate, regularization strength in a neural network, tree depth and number of trees in a decision tree.
- Hyperparameter tuning is a critical step in the machine-learning system learning & optimization process that can significantly improve the performance of your model.

$$w^{(t)} \leftarrow w^{(t-1)} - \eta \cdot \sum_{i \in B} \nabla_\theta \left(L_\alpha(f_\theta(x_i), y_i) \right) \Big|_{\theta^{(t)}}$$

α: denotes the hyperparameters during training (e.g., the learning rate η).

Hyperparameter Tuning and Optimization

- Machine Learning models hyperparameter tuning is a type of optimization problem.
- We have a set of hyperparameters and we aim to find the right combination of their values which can help us to find either the minimum (e.g., loss) or the maximum (e.g., accuracy) of a function (Figure below).
- This can be particularly important when comparing how different Machine Learning models performs on a dataset.
- In fact, it is wrong to compare an SVM model with the best Hyperparameters against a Random Forest model which has not been optimized.

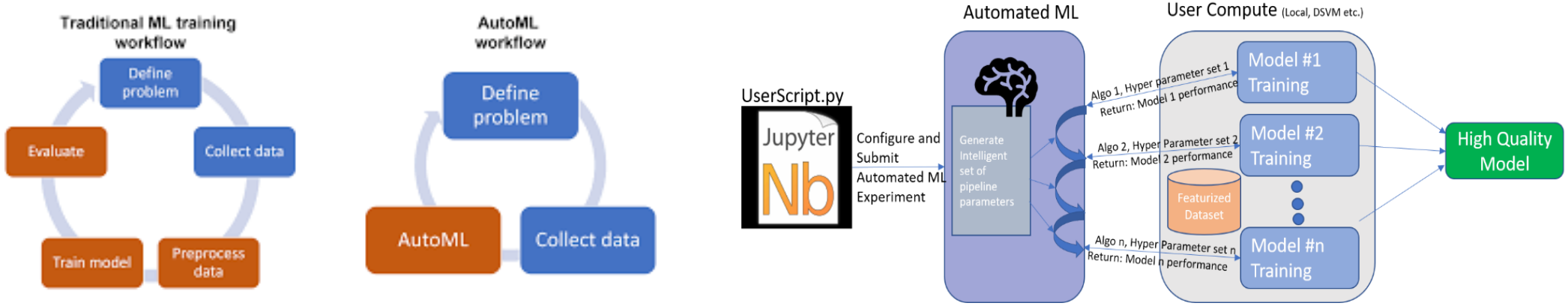


Combined Algorithm Selection and Hyperparameter Optimization (CASH)

- Selecting a data preparation pipeline, machine learning model, and model hyperparameters is a search problem.
- The possible choices at each step define a search space, and a single combination represents a point in that space that can be evaluated with a dataset.
- Navigating the search space efficiently is referred to as global optimization.
- This has been well understood for a long time in the field of machine learning, although perhaps tacitly, with focus typically on one element of the problem, such as hyperparameter optimization.
- The important insight is that there are dependencies between each step, which influences the size and structure of the search space.
- This requires that the data preparation and machine learning model, along with the model hyperparameters, must form the scope of the optimization problem and that the optimization algorithm must be aware of the dependencies between.
- This is a challenging global optimization problem, notably because of the dependencies, but also because estimating the performance of a machine learning model on a dataset is stochastic, resulting in a noisy distribution of performance scores (e.g. via repeated k-fold cross-validation).

AutoML

- Automated Machine Learning, or **AutoML** for short, is (in industry) viewed as critical for the optimization of most production grade ML systems due to complexities involved.
- More commercial products appear on the market, academic tools, and public, open-source AutoML libraries on a regular basis highlighting the need.
- As with every new technology that is new, unclear, and nebulously defined, AutoML is misunderstood.
- On one end, there are grandiose claims that it will send data analysts home
- On the other end, extreme statements say that it automates only the trivial part of the analysis.
- Different people provide different definitions of AutoML.
- Let us examine them



Why? Complexities and Successes of Well Tuned Deep Learning Algorithms



Speech recognition



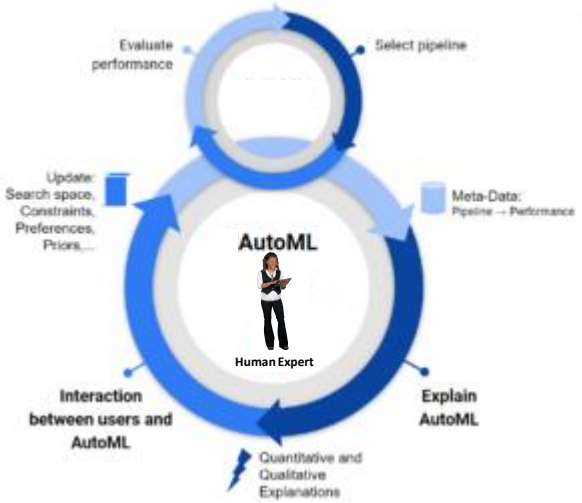
Computer vision in self-driving cars

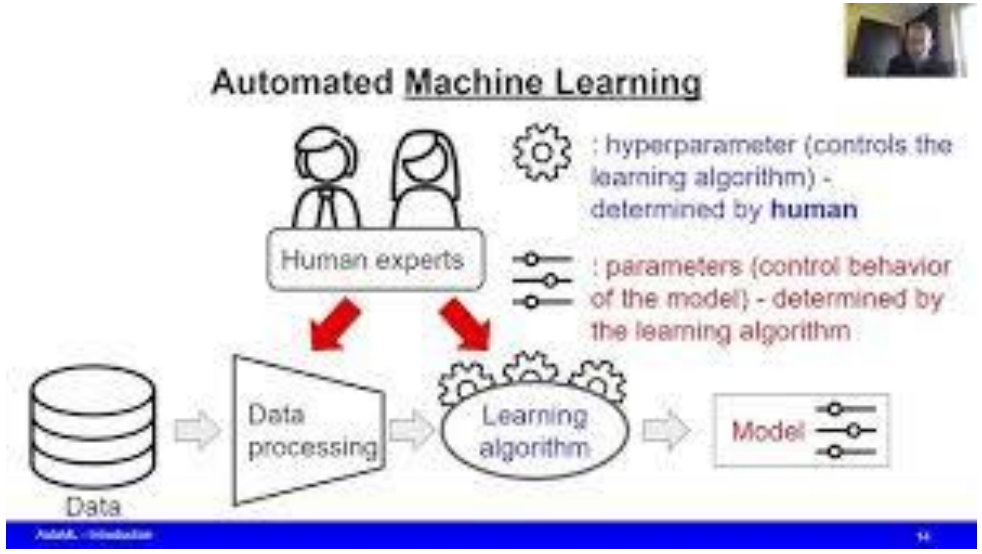


Reasoning in games

The Vision of (Predictive) AutoML

- What is AutoML?
  - Wikipedia defines it as *The automation of the machine learning process end-to-end.*
  - Isn't machine learning automated already? Given a dataset, we run a learning algorithm (e.g., Random Forest, SVM) with the default hyper-parameter values and get a model instance, automatically. So, what does automation really mean?
- The **goal** of AutoML is to gradually replace more of the rote **technical activities that a human expert does today** and produce results that the expert can comprehend and use to inform key decisions
- **These go way beyond just delivering a predictive model**.





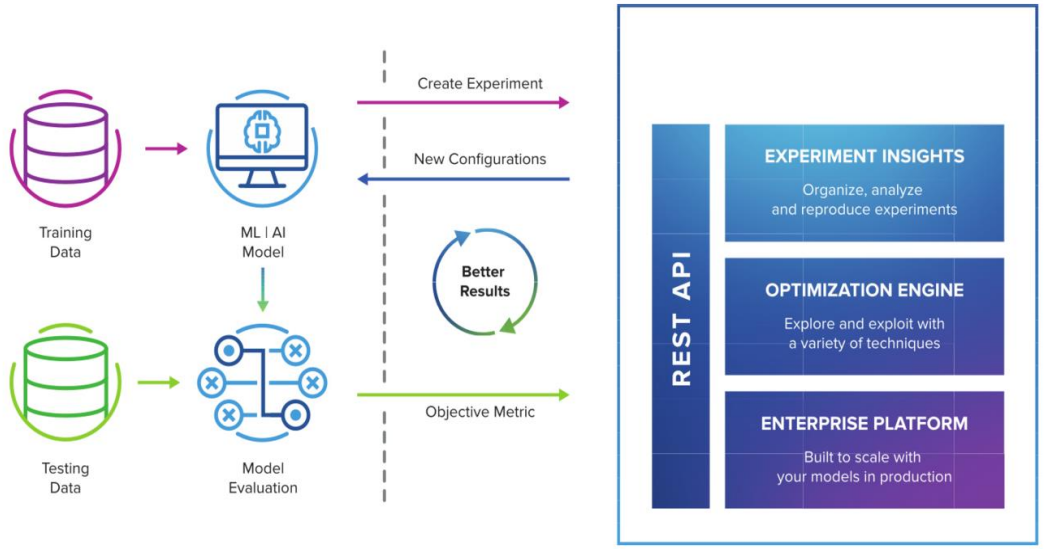
AutoML as a Good Data Scientist

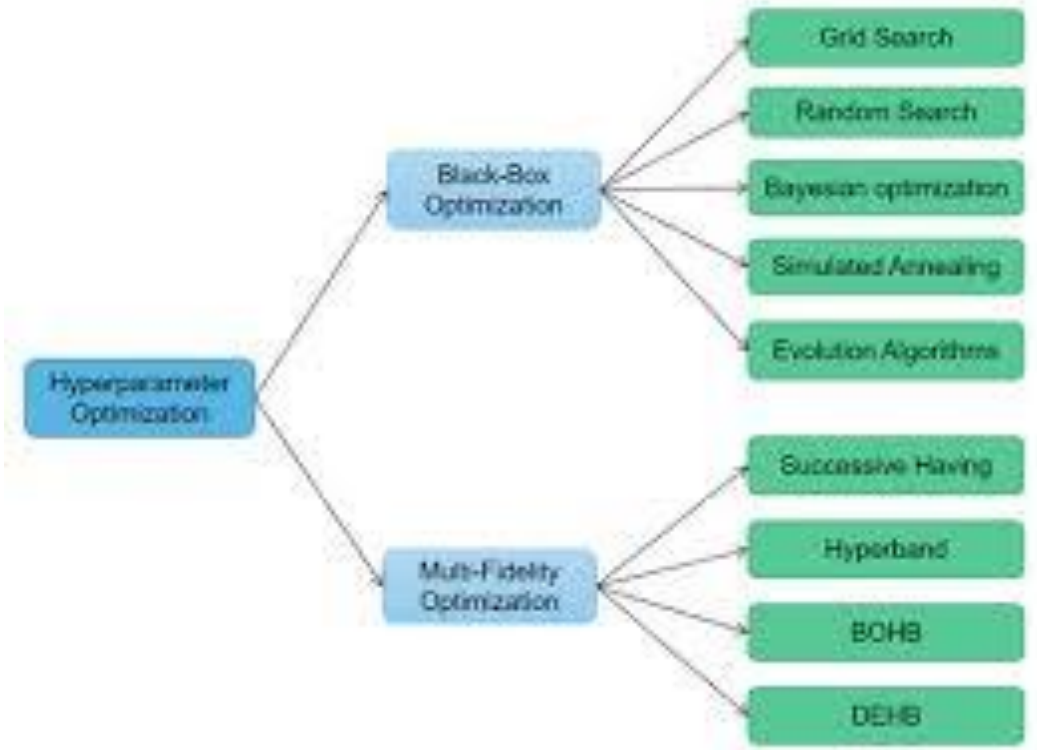
- A good data scientist would provide you with graphs, visuals, interpretations, insight, and suggestions.
- They would determine not only the appropriate classification algorithms to try, but also specialized preprocessing for your data types, try various representations for the data or the ML task, and apply feature extraction methods. And it does not stop there.
- After the model is in production, the data analyst needs to keep monitoring the sanity of the predictions in case of a data statistical distribution drift.
- If that is the case an alarm needs to be flagged and retraining of the model needs to take place.
- Most importantly, a good analyst will help you with making optimal decisions based on your model, like what is the optimal classification ***threshold*** when you apply the model, and which are the factors (features, variables) that may affect your subscribers' behavior.
- That is what AutoML should be all about. AutoML is the end goal, but we are not fully there just yet.



Hyper-Parameter Optimization (HPO)

- One of the sub-problems of an AutoML platform is to deliver the best possible model instance out of all available algorithms in its disposal.
- Each of these algorithms is “tunable” using hyper-parameters.
- Hyper-parameters are inputs to the algorithm that modify its behavior.
- **Running the same algorithm with different hyper-parameter values could lead to quite different predictive model instances**.
- So, the question that arises is how to tune their values and obtain the optimal model instance possible.
- Hyper-parameter Optimization (HPO) is the problem of automating the tuning of the hyper-parameters for a given algorithm.
- Several basic and advanced algorithms have appeared in the literature for HPO.





Examples

- For example, the lower the *K* in *K*-nearest neighbors or the lower the weight penalty lambda in ridge logistic regression, the more complex the model will be, the better the model will fit the training data, and the higher the probability of overfitting the model.
- Some algorithms take more than one hyper-parameter, e.g., the XGBoost algorithm accepts about a dozen hyper-parameters related to tuning learning performance!
- **Tuning the hyper-parameters of an algorithm can make a world of a difference in predictive performance!**
- From personal experience, **tuning "a few good algorithms" is, arguably, more valuable than using numerous algorithms with default settings**.
- The interpretation of the hyper-parameters is often related to their sensitivity to detecting patterns and fitting the data.
- HPO is challenging particularly for algorithms with many hyper-parameters to tune.

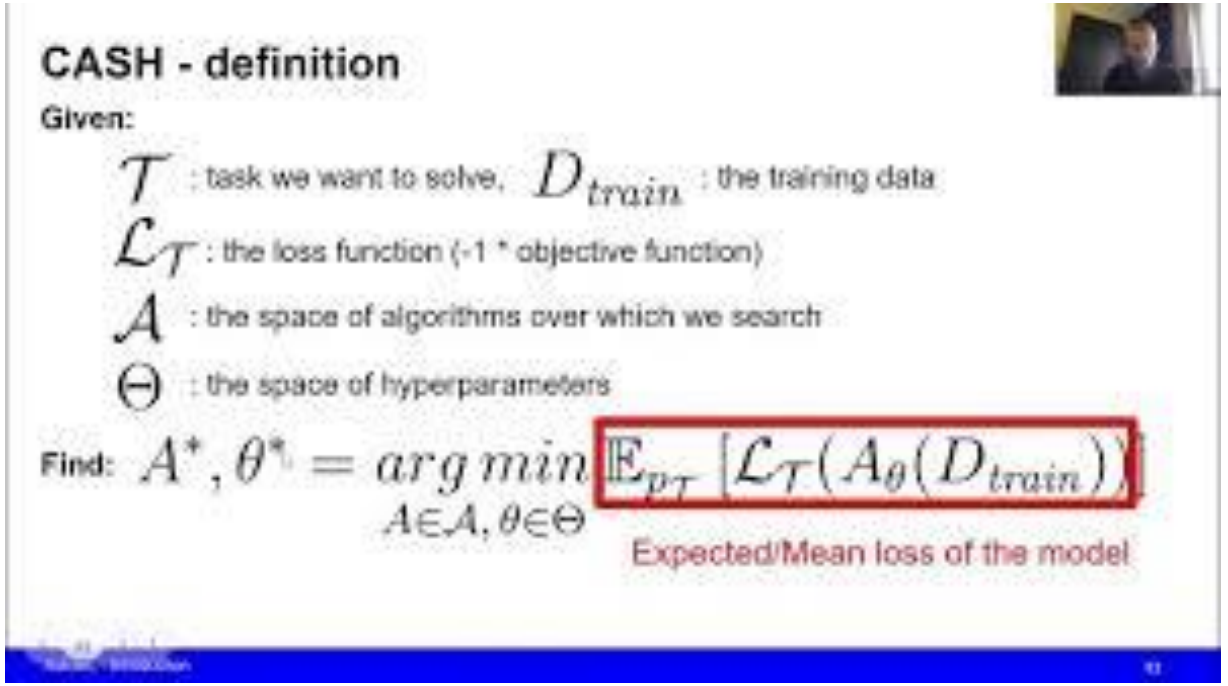
Re: https://www.researchgate.net/profile/Sayan-Putatunda/publication/331419832_A_Comparative_Analysis_of_Hyperopt_as_Against_Other_Approaches_for_Hyper-Parameter_Optimization_of_XGBoost/links/646836099533894cac7e9aa3/A-Comparative-Analysis-of-Hyperopt-as-Against-Other-Approaches-for-Hyper-Parameter-Optimization-of-XGBoost.pdf

Table 1. Algorithms along with their hyperparameters

Algorithm	Hyperparameters
Decision tree	min_samples_split
	max_depth
	min_samples_leaf
XGBoost	n_estimators
	learning_rate
	colsample_by_tree
	reg_lambda
	max_depth
	subsample
C-SVM	C (Soft margin constant)
	gamma
	kernel
Random forest	max_features
	max_depth
	min_samples_leaf
Neural network	number of hidden layers
	number of hidden nodes
	dropout
	activation function
	learning_rate
	momentum
	epochs
	batch size

Combined Algorithm and Selection of Hyper-parameters (CASH)

- HPO is often used in the context of optimizing the hyper-parameters of a ***single*** algorithm.
- However, the choice of the predictive modeling algorithm (e.g., SVM vs. Random Forests) can be encoded as a numerical hyper-parameter as well: 1 denotes SVMs, 2 denotes RFs, and so on.
- In addition, one may apply a series of analysis steps prior to modeling such as preprocessing, imputation of missing values, and feature selection.
- Accordingly, algorithm selection and hyper-parameter tuning needs to be considered for each step of the analysis.
- **When algorithm selection takes place in addition to hyper-parameter values,** the more specific term used is **Combined Algorithm and Hyper-parameter Selection** algorithms, or **CASH**.



Differences between AutoML, CASH, and HPO

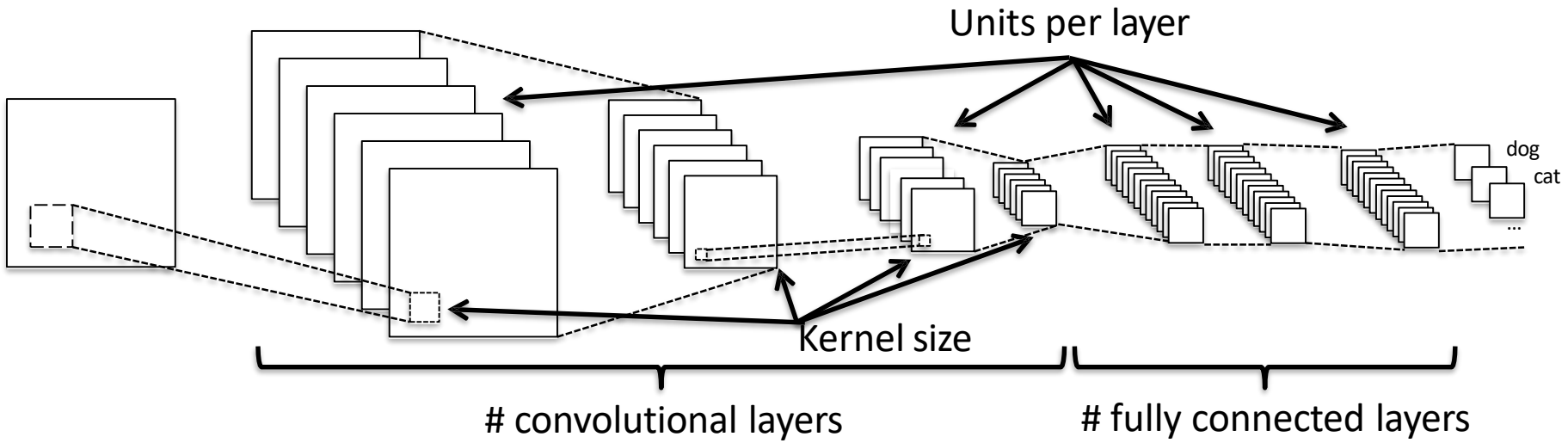
- Several exceedingly popular, particularly useful, and quite successful freeware libraries, like **auto sklearn**, **TPOT**, and **Gamma are, in my opinion, better described as CASH or HPO libraries and not AutoML**.
- So, here are the differences:
  - ***AutoML*** *aims to automate numerous functionalities of a human analyst including the production of a predictive model, estimation of its out-of-sample (i.e., on new, unseen samples) predictive performance, visualizations, interpretations, alerts about the possible problems of the analysis procedure, decision support with application of the model, monitoring of the operation of the model, automatic model retraining, and others.*
  - ***HPO*** *and* ***CASH*** *automate the search for an optimal predictive model.*

Going Deeper

One Problem with Deep Learning

Performance is very **sensitive** to **many hyperparameters**

- Architectural hyperparameters

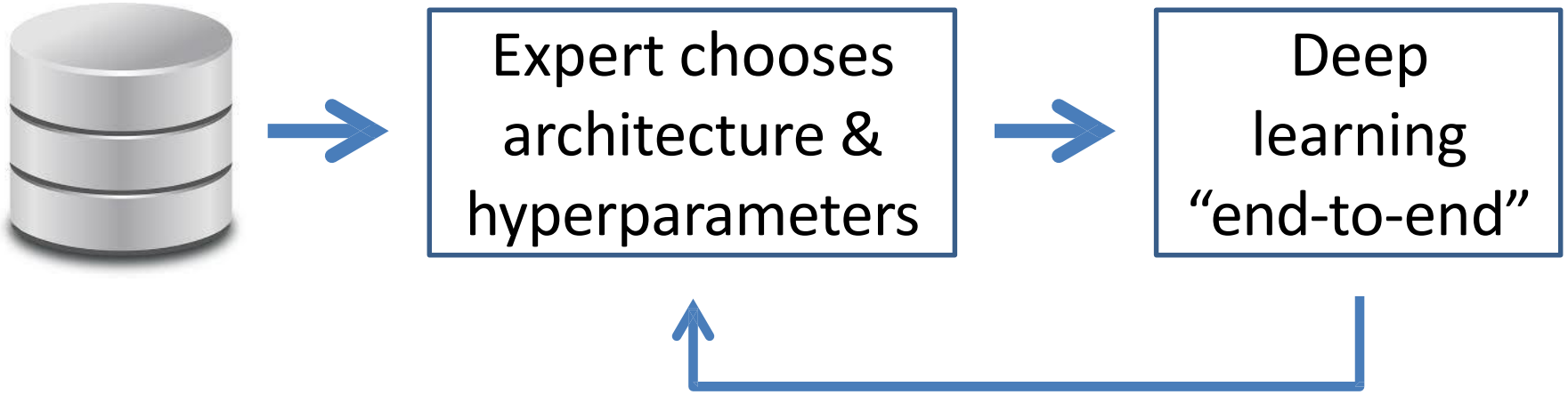


- Optimization algorithm, learning rates, momentum, batch normalization, batch sizes, dropout rates, weight decay, data augmentation, …

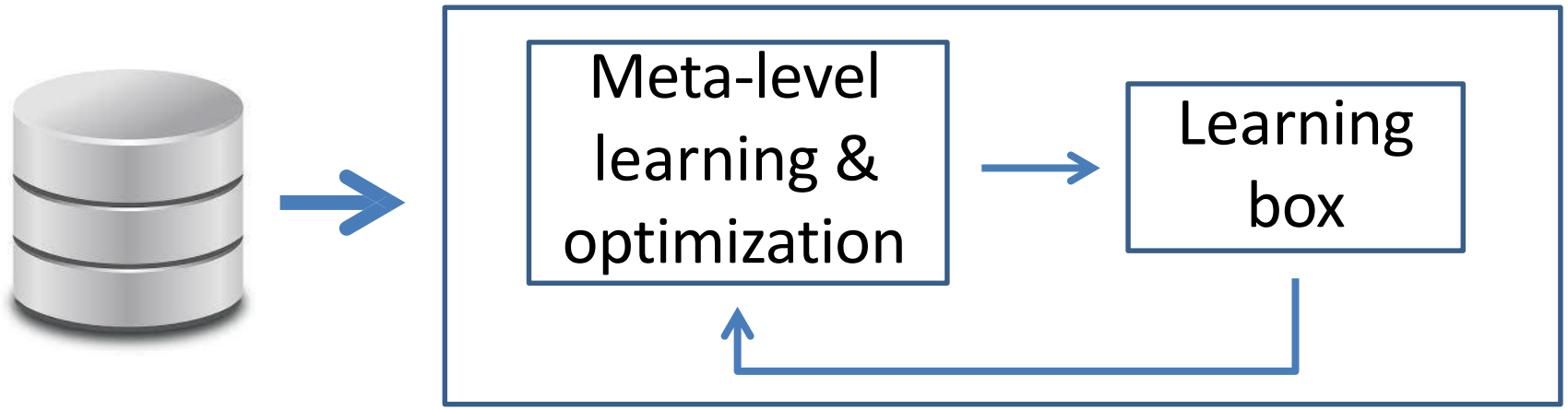
→ Easily 20-50 design decisions

Deep Learning and AutoML

Current deep learning practice



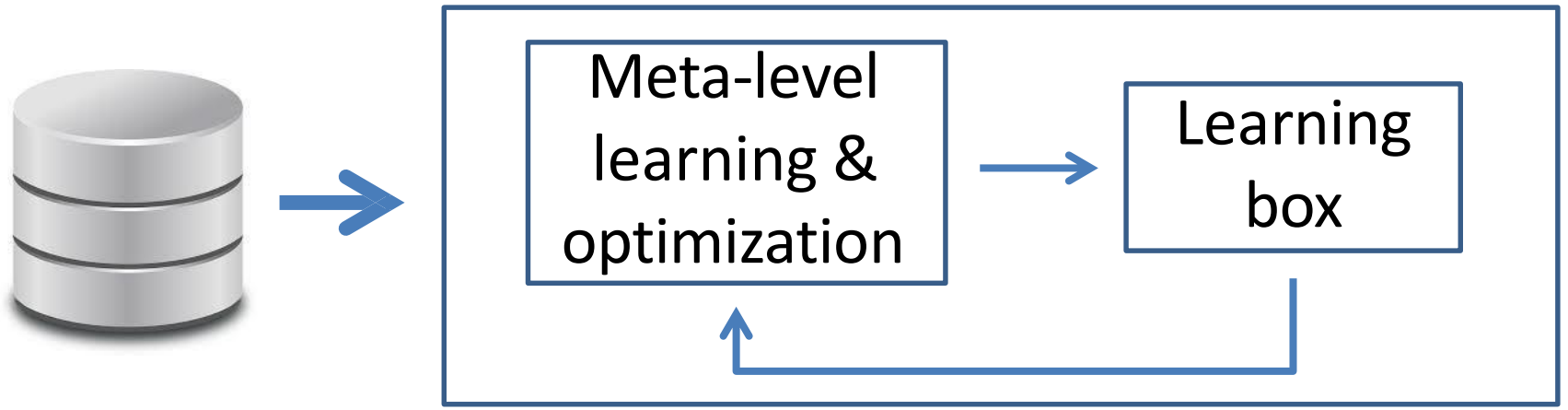
AutoML: true end-to-end learning



Learning box is not restricted to deep learning

- Traditional machine learning pipeline:
  - Clean & preprocess the data
  - Select / engineer better features
  - Select a model family
  - Set the hyperparameters
  - Construct ensembles of models
  - …

AutoML: true end-to-end learning



Topics Covered

AutoML

1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Examples
5. Conclusion

Outline

AutoML

1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Examples of AutoML
5. Wrap-up & Conclusion

Hyperparameter Optimization Definition

Definition: Hyperparameter Optimization (HPO)

Let

- $\boldsymbol{\lambda}$ be the hyperparameters of a ML algorithm A with domain $\boldsymbol{\Lambda}$,
- $\mathcal{L}(A_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$ denote the loss of A, using hyperparameters $\boldsymbol{\lambda}$ trained on D_{train} and evaluated on D_{valid}.

The hyperparameter optimization (HPO) problem is to find a hyperparameter configuration $\boldsymbol{\lambda}^*$ that minimizes this loss:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \mathcal{L}(A_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$$

Types of Hyperparameters

- Continuous

  Example: learning rate in NNs or GBMs

- Integer

  Example: #units, #trees in GBM

- Categorical
  - Finite domain, unordered
    - Example 1: algo ∈ {SVM, RF, NN}
    - Example 2: activation function ∈ {ReLU, Leaky ReLU, tanh}
    - Example 3: operator ∈ {conv3x3, separable conv3x3, max pool, …}
  - Special case: binary

Conditional hyperparameters

Conditional hyperparameters B are only active if other hyperparameters A are set a certain way

- Example 1:
  - A = choice of optimizer (Adam or SGD)
  - B = Adam‘s second momentum hyperparameter (only active if A=Adam)
- Example 2:
  - A = number of layers in a deep neural network
  - B = number of units in layer k (only active if A >= k)
- Example 3:
  - A = choice of classifier (RF or SVM)
  - B = SVM‘s kernel hyperparameter (only active if A = SVM)

AutoML as Hyperparameter Optimization

Definition: Combined Algorithm Selection and Hyperparameter Optimization (CASH)

Let

- $\mathcal{A} = \{A^{(1)}, \ldots, A^{(n)}\}$ be a set of algorithms
- $\mathbf{\Lambda}^{(i)}$ denote the hyperparameter space of $A^{(i)}$, for $i = 1, \ldots, n$
- $\mathcal{L}(A^{(i)}_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$ denote the loss of $A^{(i)}$, using $\lambda \in \mathbf{\Lambda}^{(i)}$ trained on D_{train} and evaluated on D_{valid}.

The Combined Algorithm Selection and Hyperparameter Optimization (CASH) problem is to find a combination of algorithm $A^* = A^{(i)}$ and hyperparameter configuration $\boldsymbol{\lambda}^* \in \mathbf{\Lambda}^{(i)}$ that minimizes this loss:

$$A^*_{\boldsymbol{\lambda}^*} \in \underset{A^{(i)} \in \mathcal{A}, \boldsymbol{\lambda} \in \mathbf{\Lambda}^{(i)}}{\arg\min} \mathcal{L}(A^{(i)}_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$$

AutoML as Hyperparameter Optimization

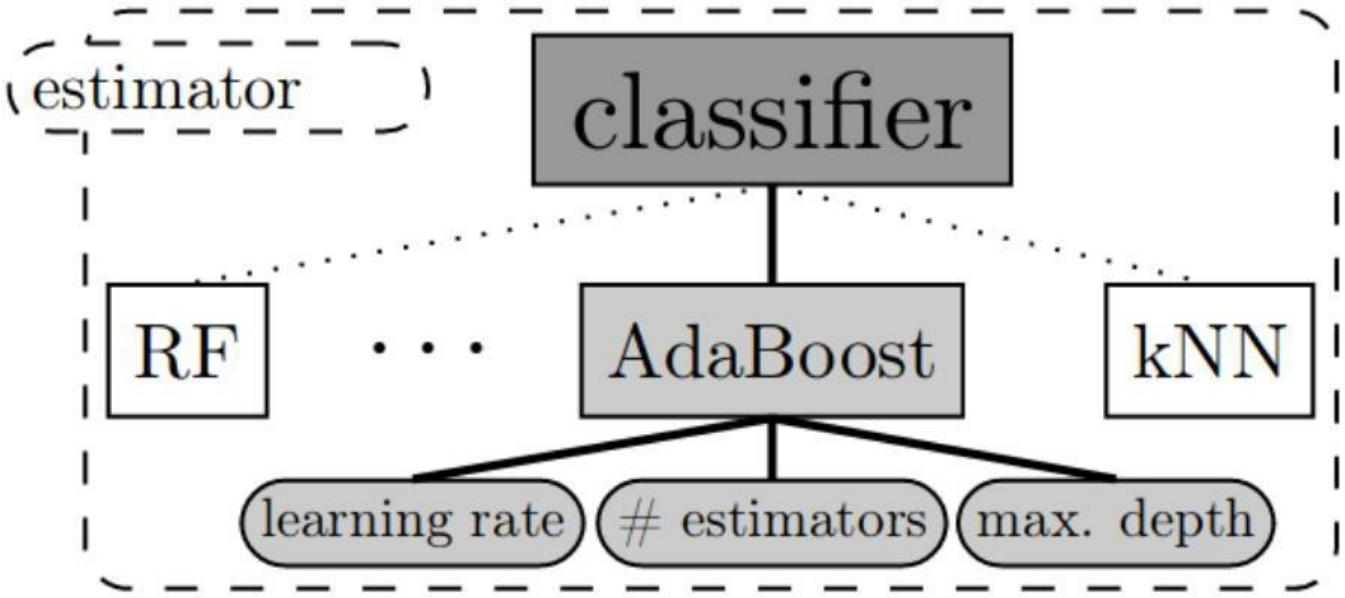
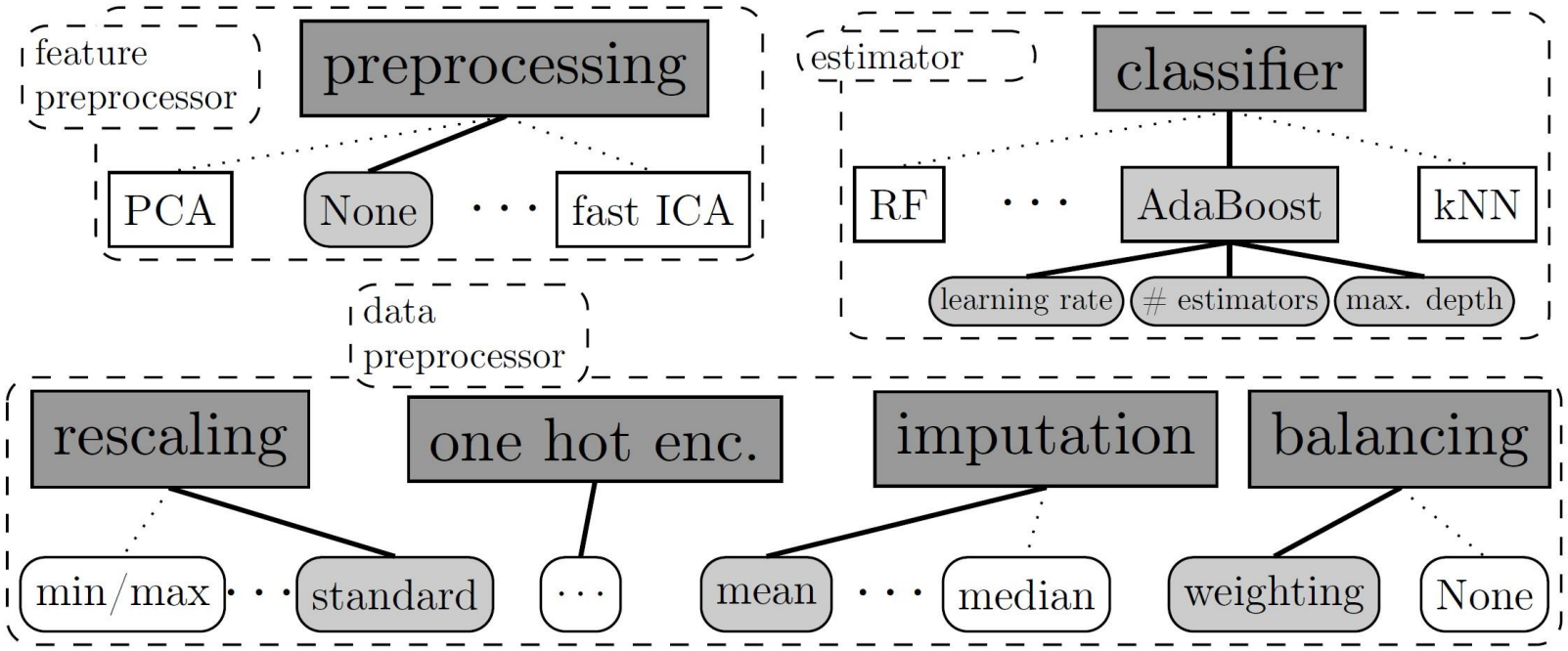


Illustration of the CASH problem in Auto-sklearn:

- 15 base classifiers
- Up to ten hyperparameters each
- Four levels of conditionality

AutoML as Hyperparameter optimization

Not limited to the classification algorithm:



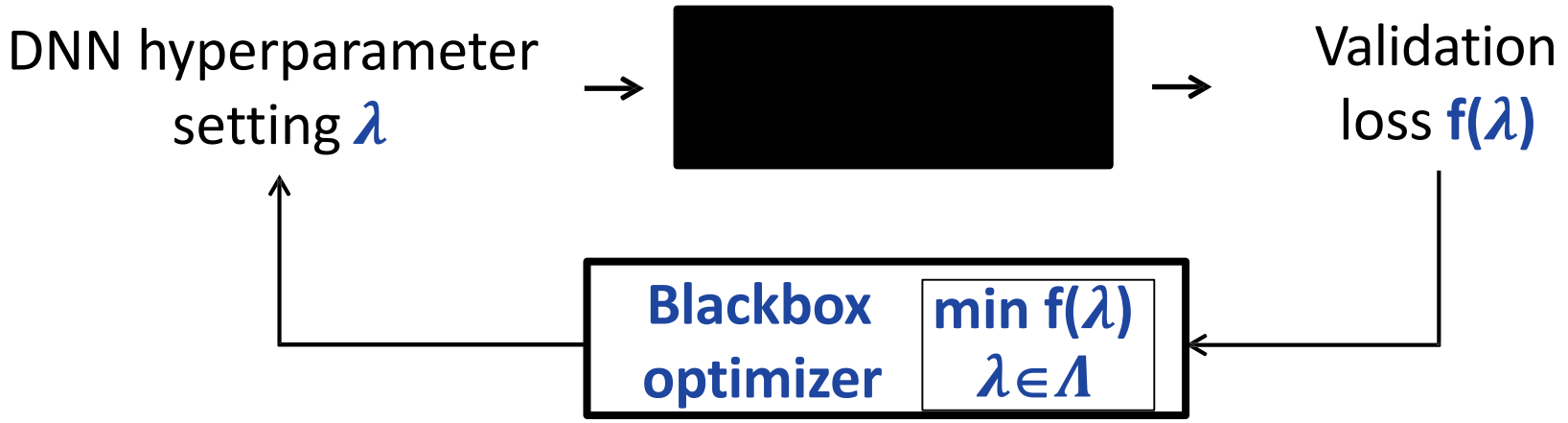
See also Thornton et al. (KDD 2013) which introduced the CASH problem.

Outline

General AutoML

1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Examples of AutoML
5. Wrap-up & Conclusion

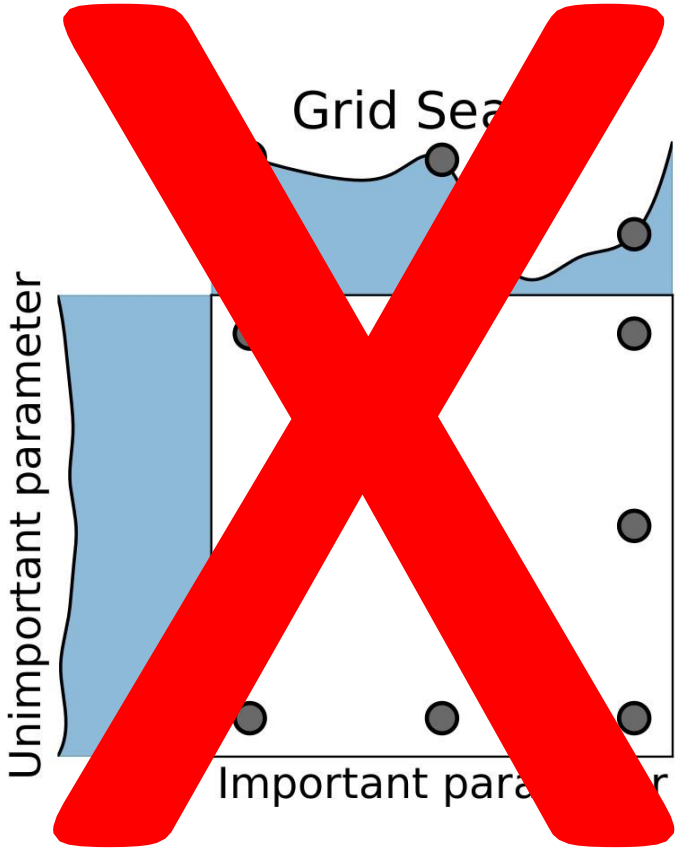
Blackbox Hyperparameter Optimization

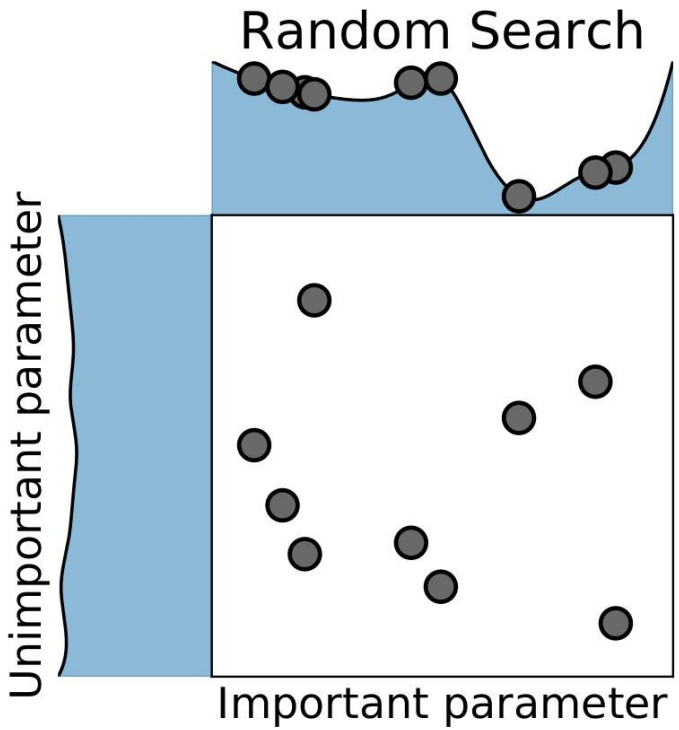


- The blackbox function is expensive to evaluate
  $\rightarrow$ sample efficiency is important

Grid Search and Random Search

- Both completely uninformed
- Grid search suffers from the curse of dimensionality
- Random search handles low intrinsic dimensionality better
- Example: an additive function ($y = f(x) + g(x)$)





Bergstra and Bengio, JMLR 2012; Image source: Feurer & Hutter, CC-BY 4.0

Bayesian Optimization

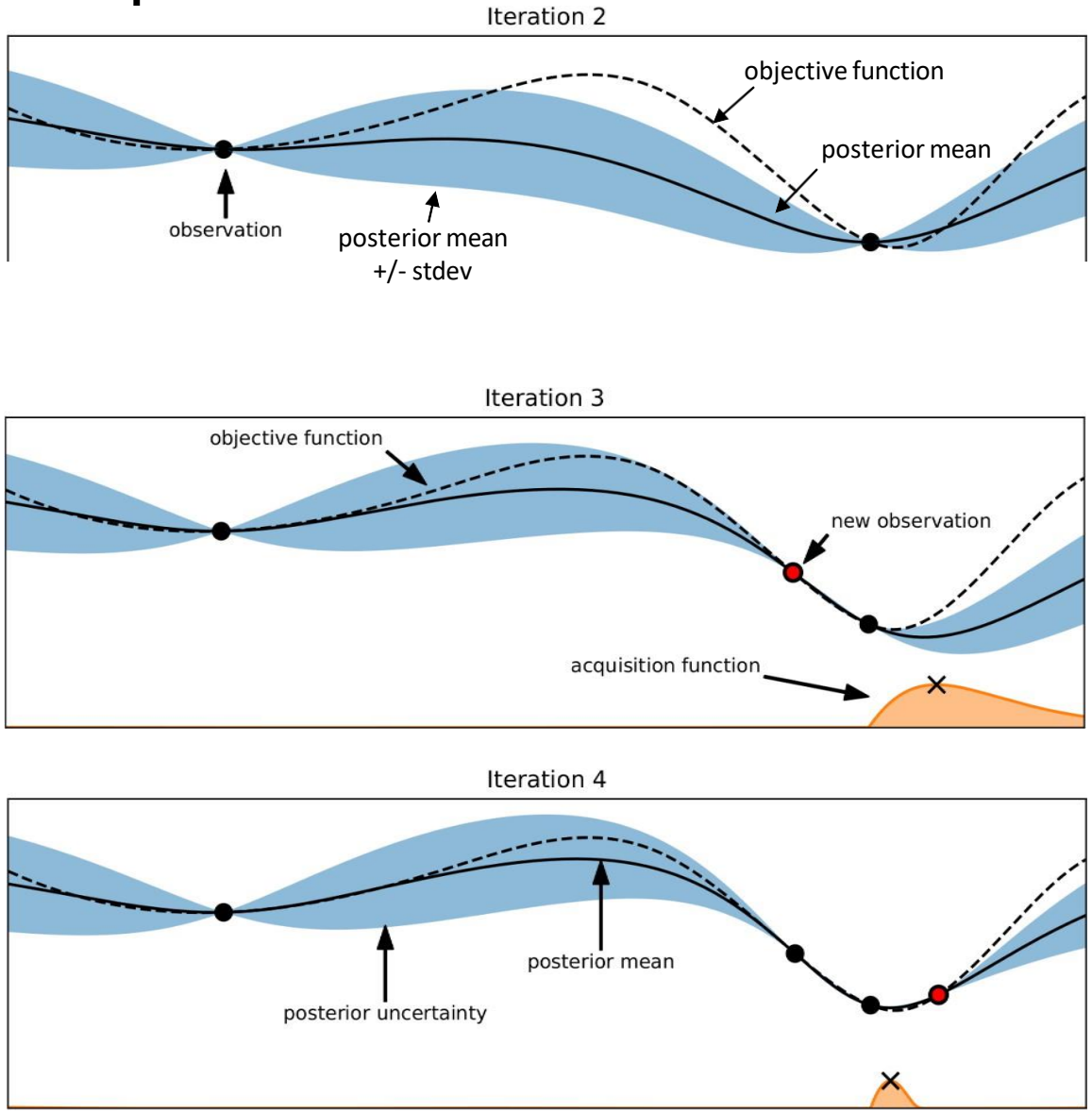


Image source: Feurer & Hutter, CC-BY 4.0

Acquisition Function: Expected Improvement

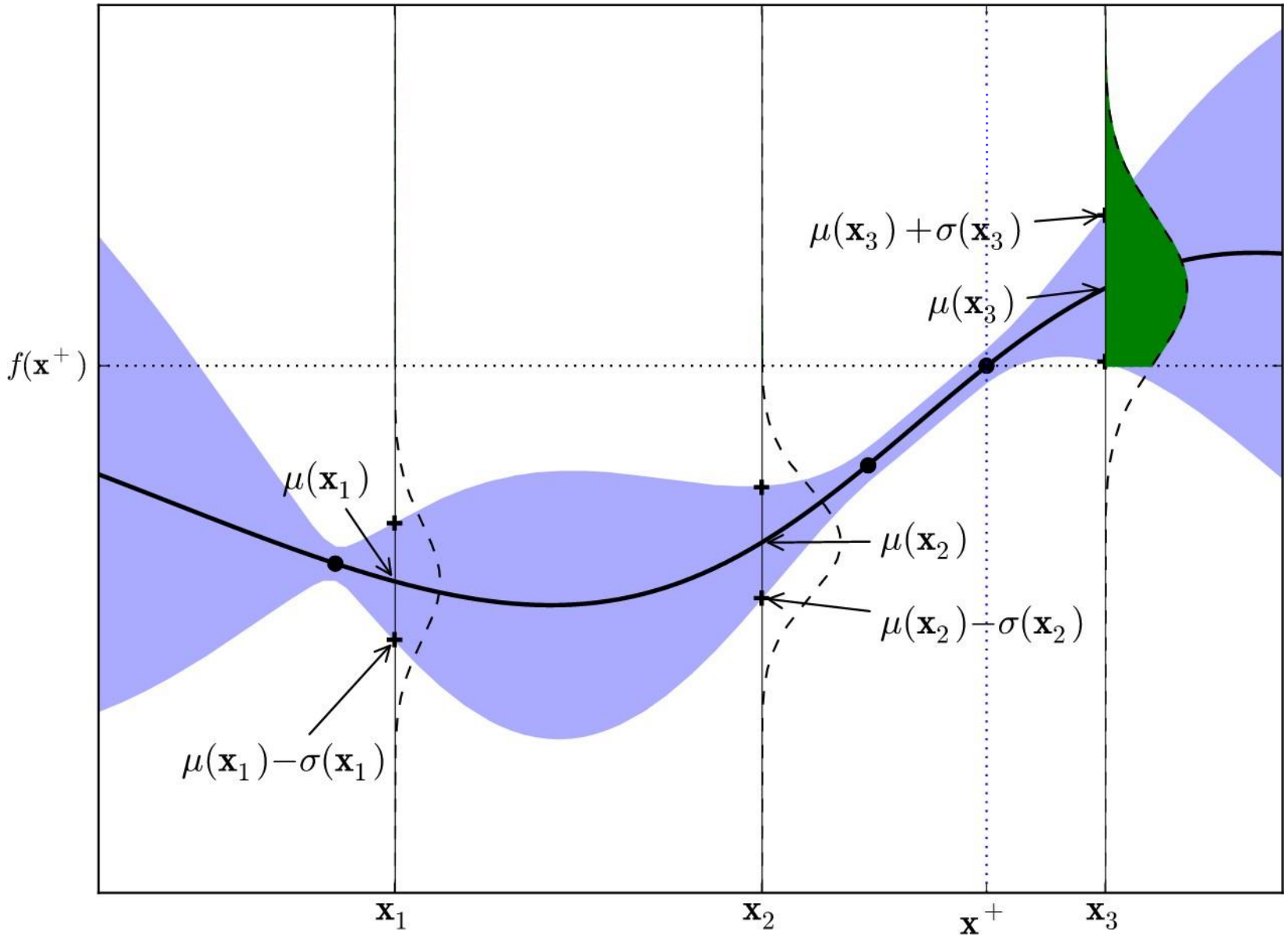


Image source: Brochu et al., arXiv:1012.2599

Bayesian Optimization

Approach

- Conduct an initial design
- Iteratively:
  - Fit a proabilistic model to the function evaluations $\langle \lambda, f(\lambda) \rangle$, most often a Gaussian process
  - Use that model to trade off Exploration vs. Exploitation in an acquisition function

Popular since Mockus [1974]

- Sample-efficient
- Works when objective is nonconvex, noisy, has unknown derivatives, etc
- Recent convergence results [Srinivas et al, 2010; Bull 2011; de Freitas et al, 2012; Kawaguchi et al, 2016; Nguyen et al., 2017; Berkenkamp et al., 2019]
- Excellent reviews by Shahriari et al. (IEEE, 2016) and Frazier (arXiv:1807.02811)

Example: Bayesian Optimization in AlphaGo

- During the development of AlphaGo, its many hyperparameters were tuned with Bayesian optimization multiple times.

- This automatic tuning process resulted in substantial improvements in playing strength. For example, prior to the match with Lee Sedol, we tuned the latest AlphaGo agent and this improved its win-rate from 50% to 66.5% in self-play games. This tuned version was deployed in the final match.

- Of course, since we tuned AlphaGo many times during its development cycle, the compounded contribution was even higher than this percentage.

[Chen et al., arXiv:1812.06855]

AutoML Challenges for Bayesian Optimization

- Problems for standard Gaussian Process (GP) approach :
  - Complex hyperparameter space
    - High-dimensional (low effective dimensionality) [e.g., Wang et al., 2013]
    - Mixed continuous/discrete hyperparameters [e.g., Hutter et al., 2011]
    - Conditional hyperparameters [e.g., Jenatton et al., 2017]
  - Noise: sometimes heteroscedastic, large, non-Gaussian
  - Model overhead (budget is runtime, not #function evaluations)

- Simple solution used in SMAC: random forests [Breiman, 2001]
  - Frequentist uncertainty estimate:
    variance across individual trees' predictions [Hutter et al, 2011]

Other methods

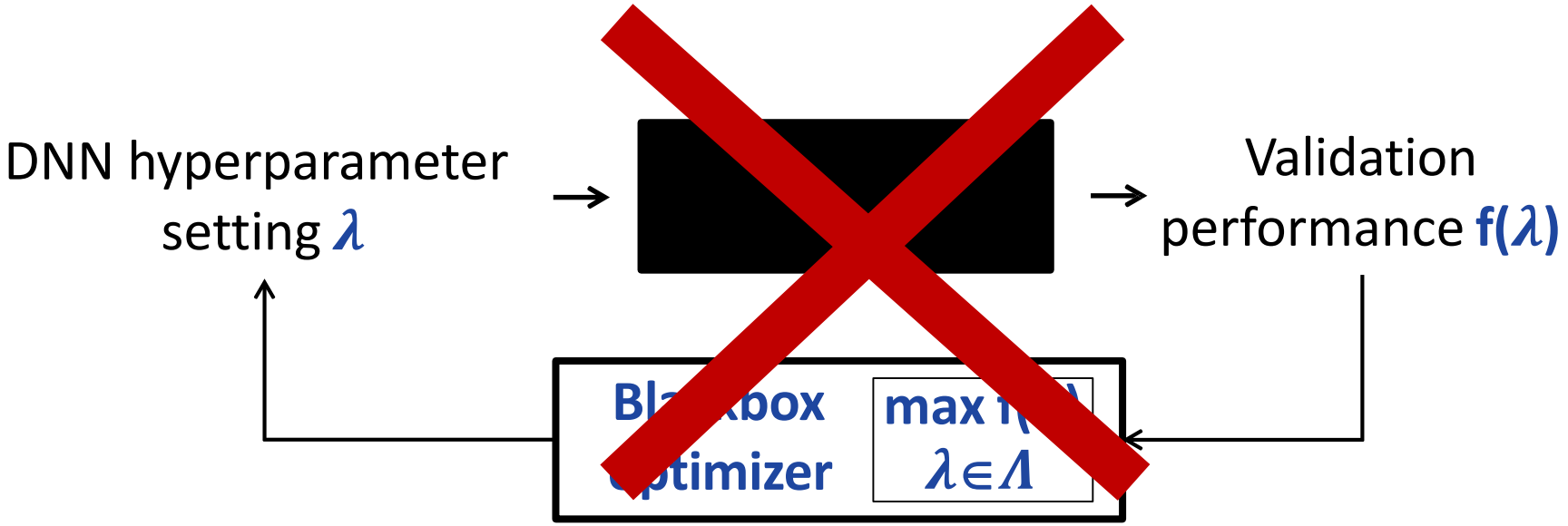
- Two recent promising models for Bayesian optimization
  - Neural networks with Bayesian linear regression using the features in the output layer [Snoek et al, ICML 2015]
  - Fully Bayesian neural networks, trained with stochastic gradient Hamiltonian Monte Carlo [Springenberg et al, NIPS 2016]
- Tree Parzen Estimator [Bergstra et al., 2011]
  - Non-parametric KDEs for $p(\lambda \text{ is good})$ and $p(\lambda \text{ is bad})$, rather than $p(y|\lambda)$
  - Ratio is proportional to Expected Improvement
- Population-based methods
  - Genetic algorithms, evolutionary algorithms, evolutionary strategies, particle swarm optimization
  - Embarassingly parallel, conceptually simple
- See Chapter 1 of the AutoML book for more information.

Outline

General AutoML

1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Examples of AutoML
5. Wrap-up & Conclusion

Beyond Blackbox Hyperparameter Optimization

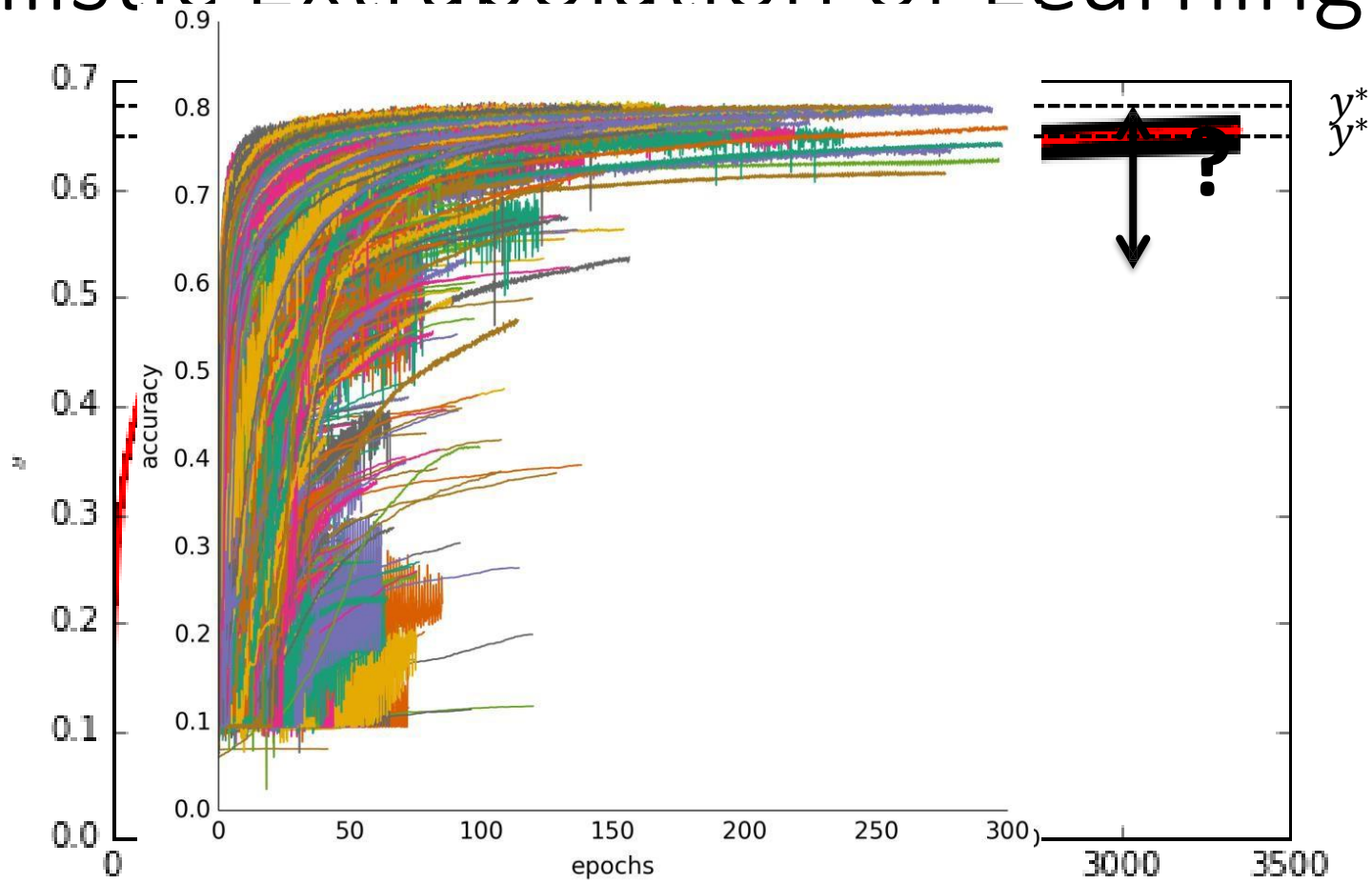


Too slow for DL / big data

Main Approaches Going Beyond Blackbox HPO

- Extrapolation of learning curves
- Multi-fidelity optimization
- Meta-learning [if there's time left]
- Hyperparameter gradient descent [see AutoML book]

Probabilistic Extrapolation of Learning Curves



- Parametric learning curve models [Domhan et al, IJCAI 2015]
- Bayesian neural networks [Klein et al, ICLR 2017]
- Linear combination of previous curves [Chandrashekaran and Lane, ECML2017]

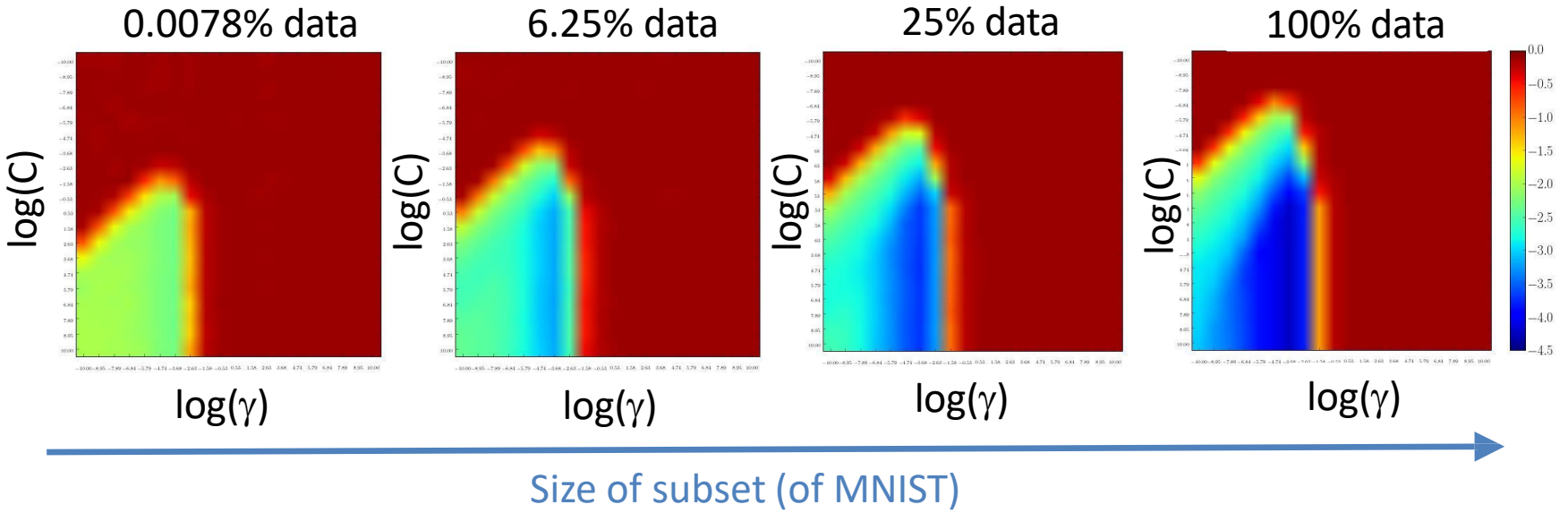
Multi-Fidelity Optimization

- Use cheap approximations of the blackbox, performance on which correlates with the blackbox, e.g.
  - Subsets of the data
  - Fewer epochs of iterative training algorithms (e.g., SGD)
  - Fewer trials in deep reinforcement learning
  - Downsampled images in object recognition

  - Also applicable in different domains, e.g., fluid simulations:
    - Less particles
    - Shorter simulations

Multi-fidelity Optimization

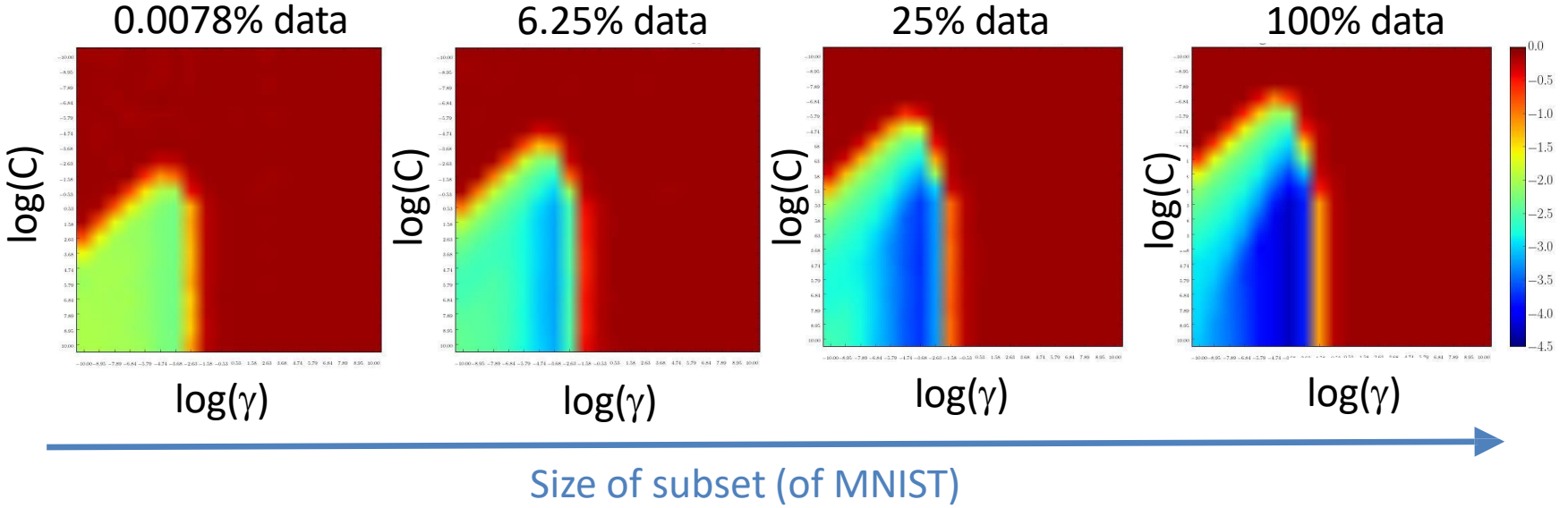
- **Make use of cheap low-fidelity evaluations**
  - E.g.: subsets of the data (here: SVM on MNIST)



  - Many cheap evaluations on small subsets
  - Few expensive evaluations on the full data
  - **Up to 1000x speedups** [Klein et al, AISTATS 2017]

Multi-fidelity Optimization

- **Make use of cheap low-fidelity evaluations**
  - E.g.: subsets of the data (here: SVM on MNIST)

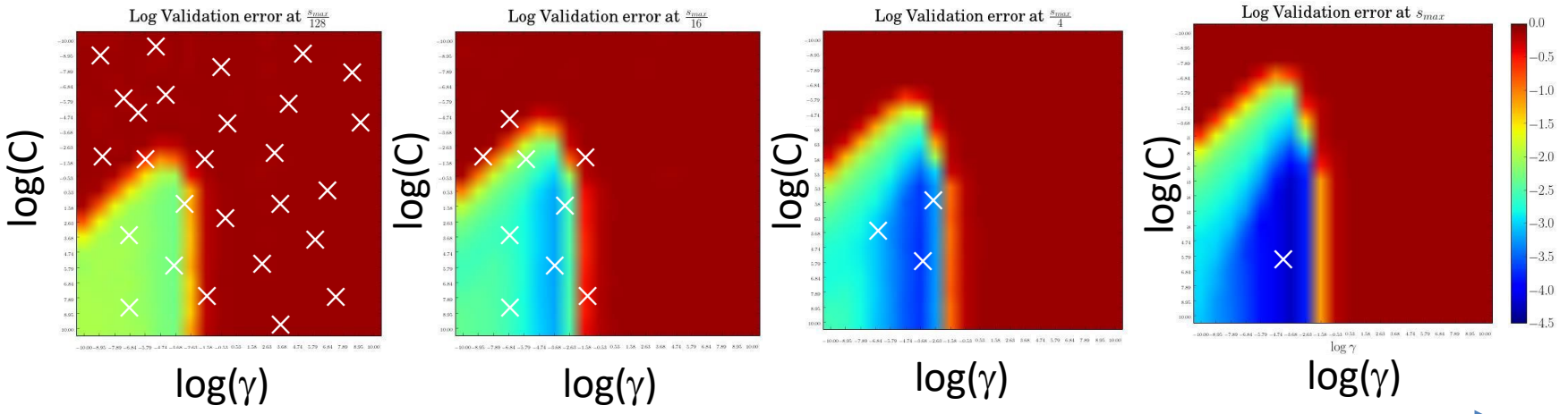


  - Fit a Gaussian process model f(λ,b) to predict performance as a function of hyperparameters λ and budget b
  - Choose both λ and budget b to maximize "bang for the buck"

[Swersky et al, NeurIPS 2013; Swersky et al, arXiv 2014; Klein et al, AISTATS 2017; Kandasamy et al, ICML 2017]

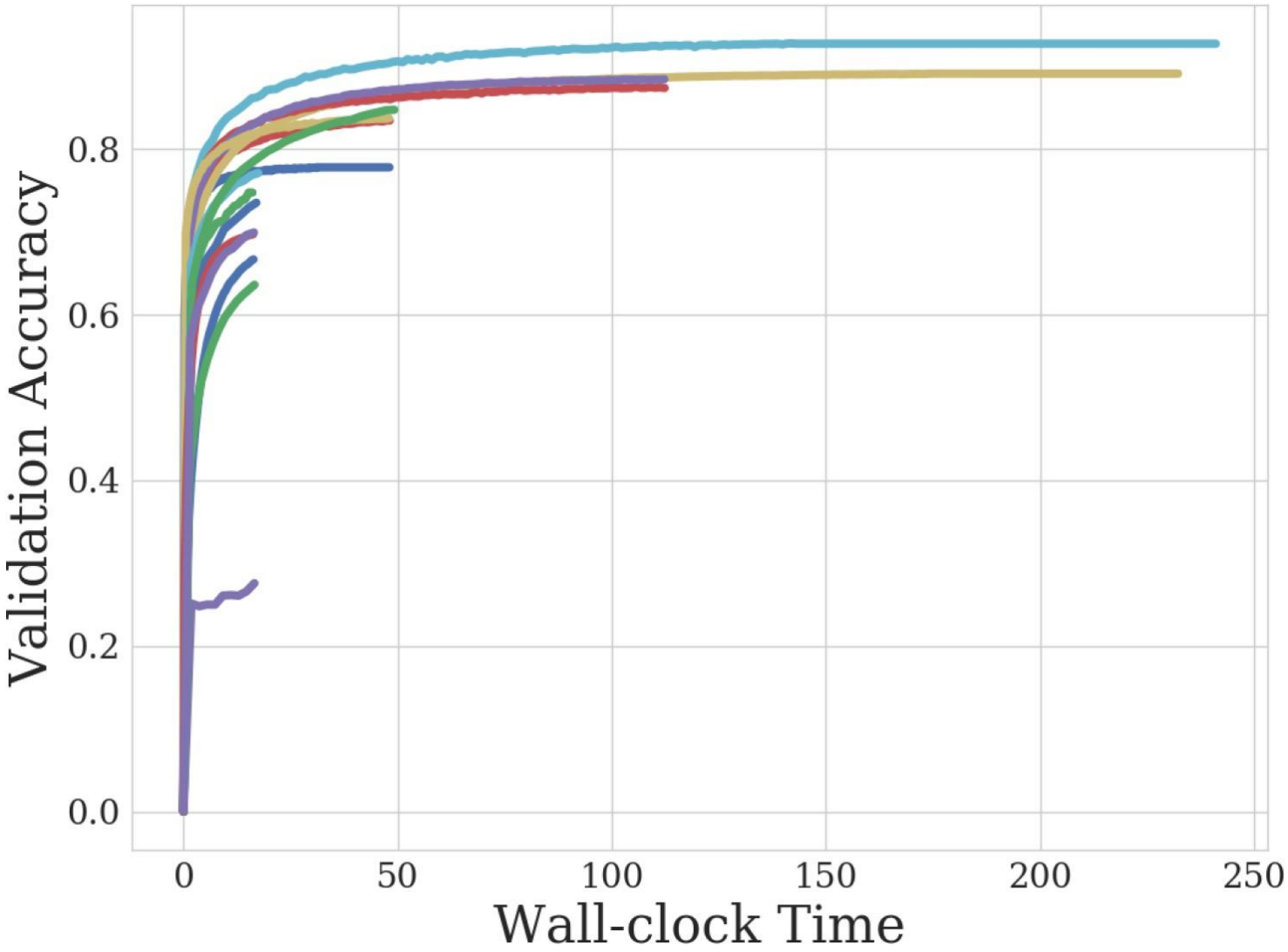
A Simpler Approach: Successive Halving (SH)

- **Idea:** Use a bandit to allocate more budget to promising configurations
- **Successive Halving** [Jamieson & Talwalkar, AISTATS 2016]
  - Randomly sample N configurations & evaluate on cheapest fidelity
  - Keep the top half, double its budget (or top third, triple budget)



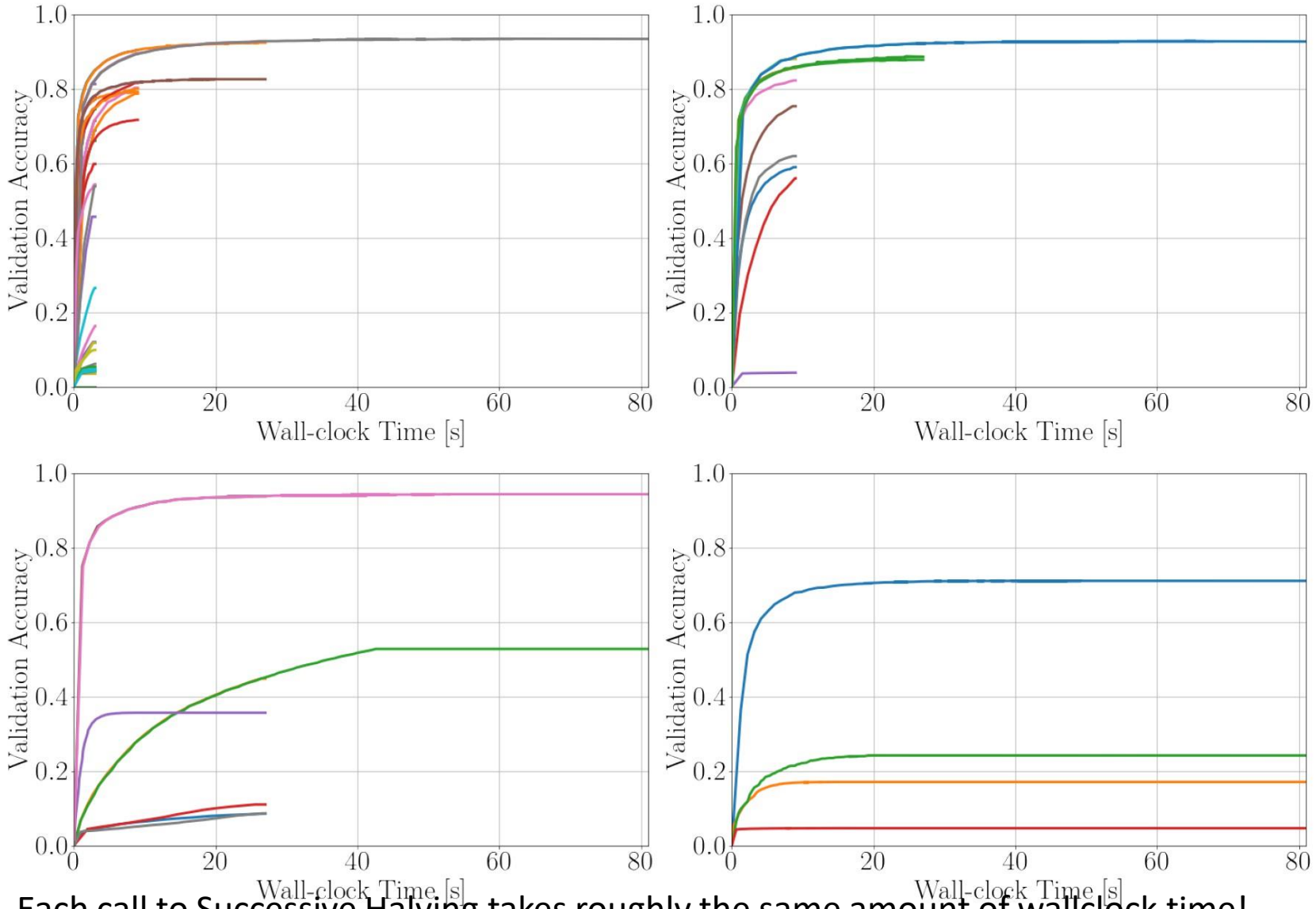
A Simpler Approach: Successive Halving (SH)

[Jamieson & Talwalkar, AISTATS 2016]



Hyperband (its first 4 calls to SH)

[Li et al, JMLR 2018]



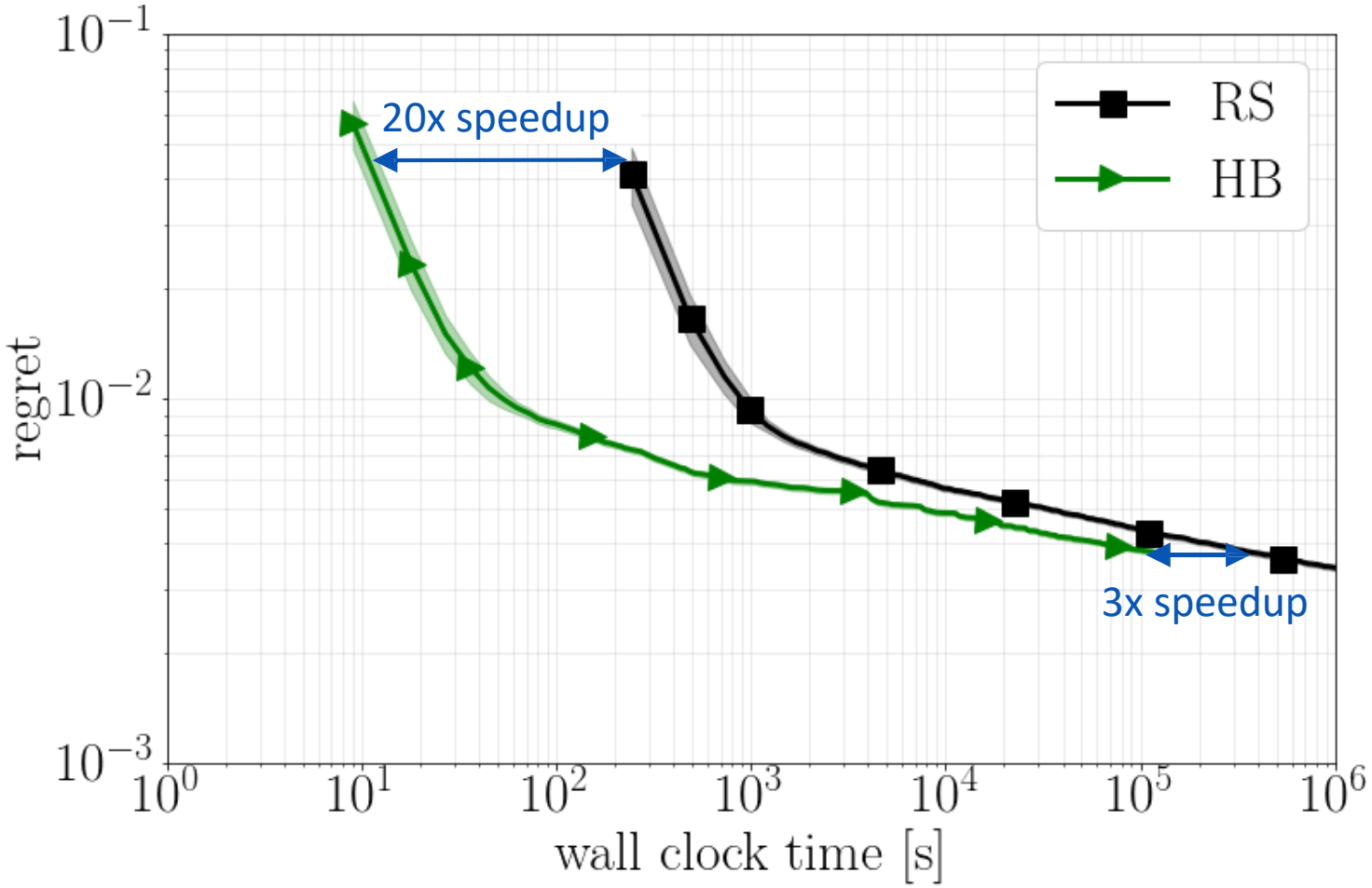
Each call to Successive Halving takes roughly the same amount of wallclock time!

BOHB: Bayesian Optimization & Hyperband

[Falkner, Klein & Hutter, ICML 2018]

- Advantages of Hyperband
  - Strong anytime performance
  - General-purpose
    - Low-dimensional continuous spaces
    - High-dimensional spaces with conditionality, categorical dimensions, etc
  - Easy to implement
  - Scalable
  - Easily parallelizable
- Advantage of Bayesian optimization: strong final performance
- Combining the best of both worlds in BOHB
  - Bayesian optimization
    - for choosing the configurations to evaluate (using a TPE variant)
  - Hyperband
    - for deciding how to allocate budgets

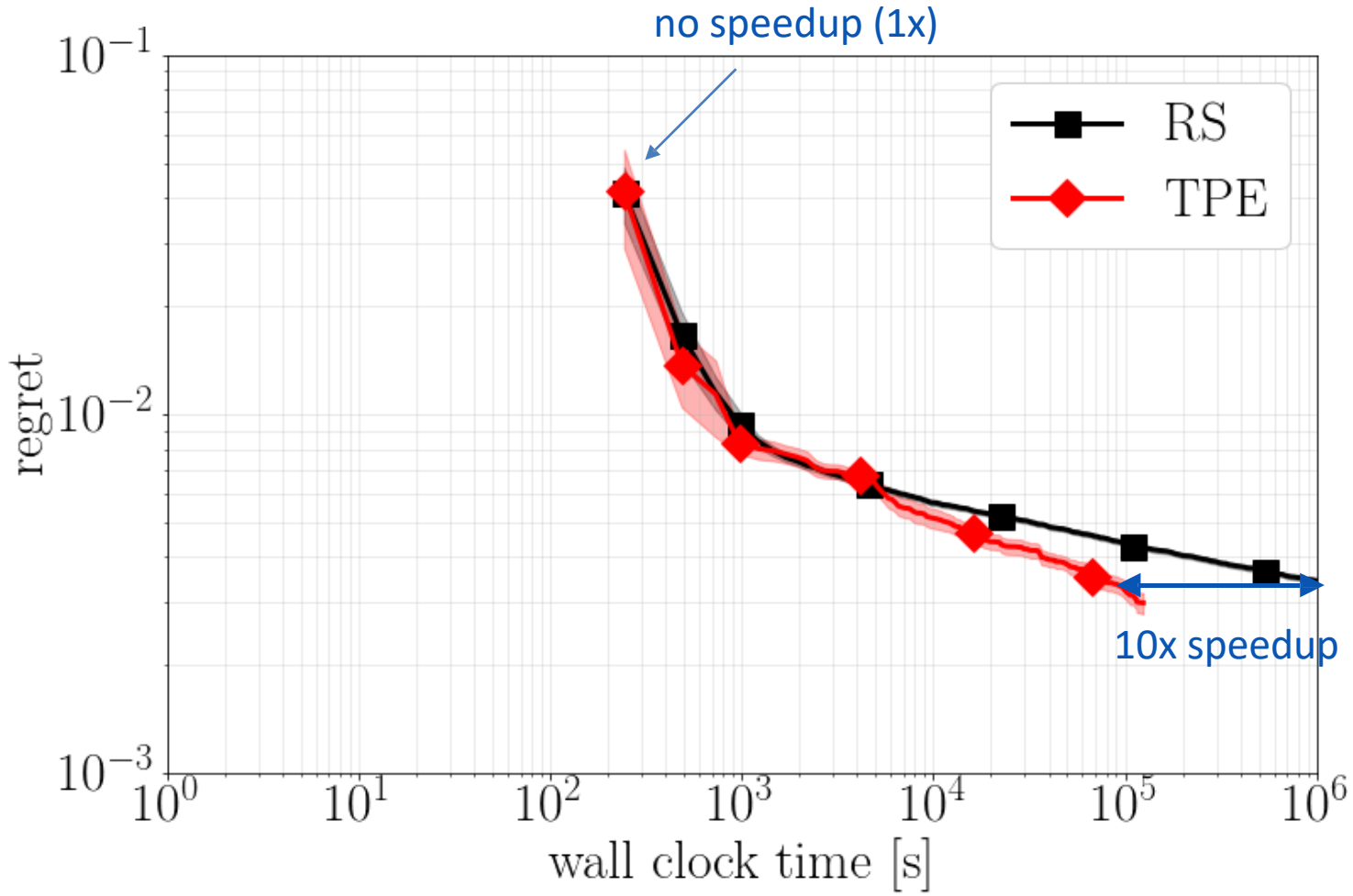
Hyperband vs. Random Search



Biggest advantage: much improved **anytime performance**

Auto-Net on dataset adult

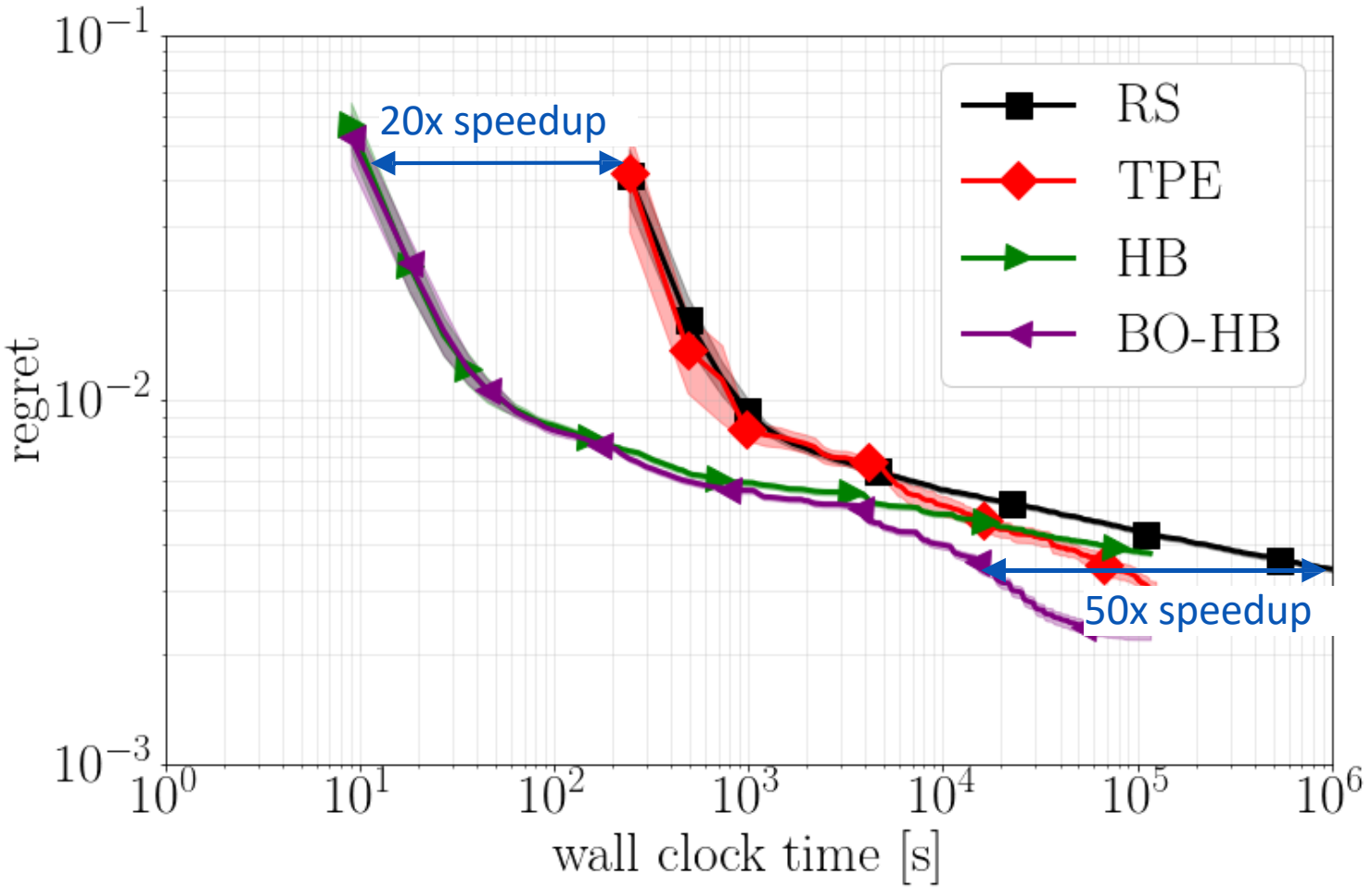
Bayesian Optimization vs Random Search



Biggest advantage: much improved **final performance**

Auto-Net on dataset adult

Combining Bayesian Optimization & Hyperband



Best of both worlds: strong **anytime and final performance**

Auto-Net on dataset adult

Outline

General AutoML

1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Examples of AutoML
5. Wrap-up & Conclusion

What can be automated?



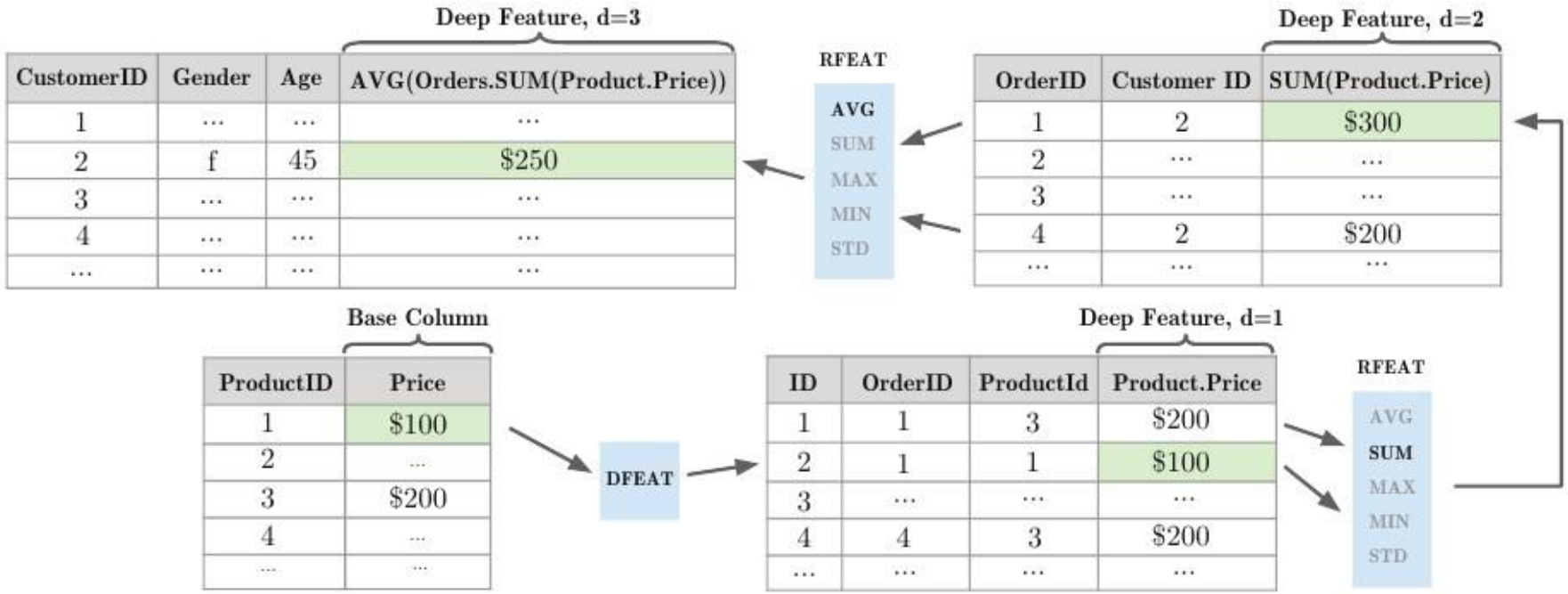
Image credit: Rich Caruana, AutoML 2015

Example I – Data cleaning and ingestion

- Automatically detect the dialect of CSV files
  [van den Burg et al., arXiv:1811.11242]
- Automatically classify data types
  [Valera and Ghahramani, ICML 2017]
- Automatically detect mistakes in the data gathering process
  [Sutton et al., KDD 2018]
- Check out the talk of Charles Sutton@AutoML Workshop 2019

Example II – Feature Engineering

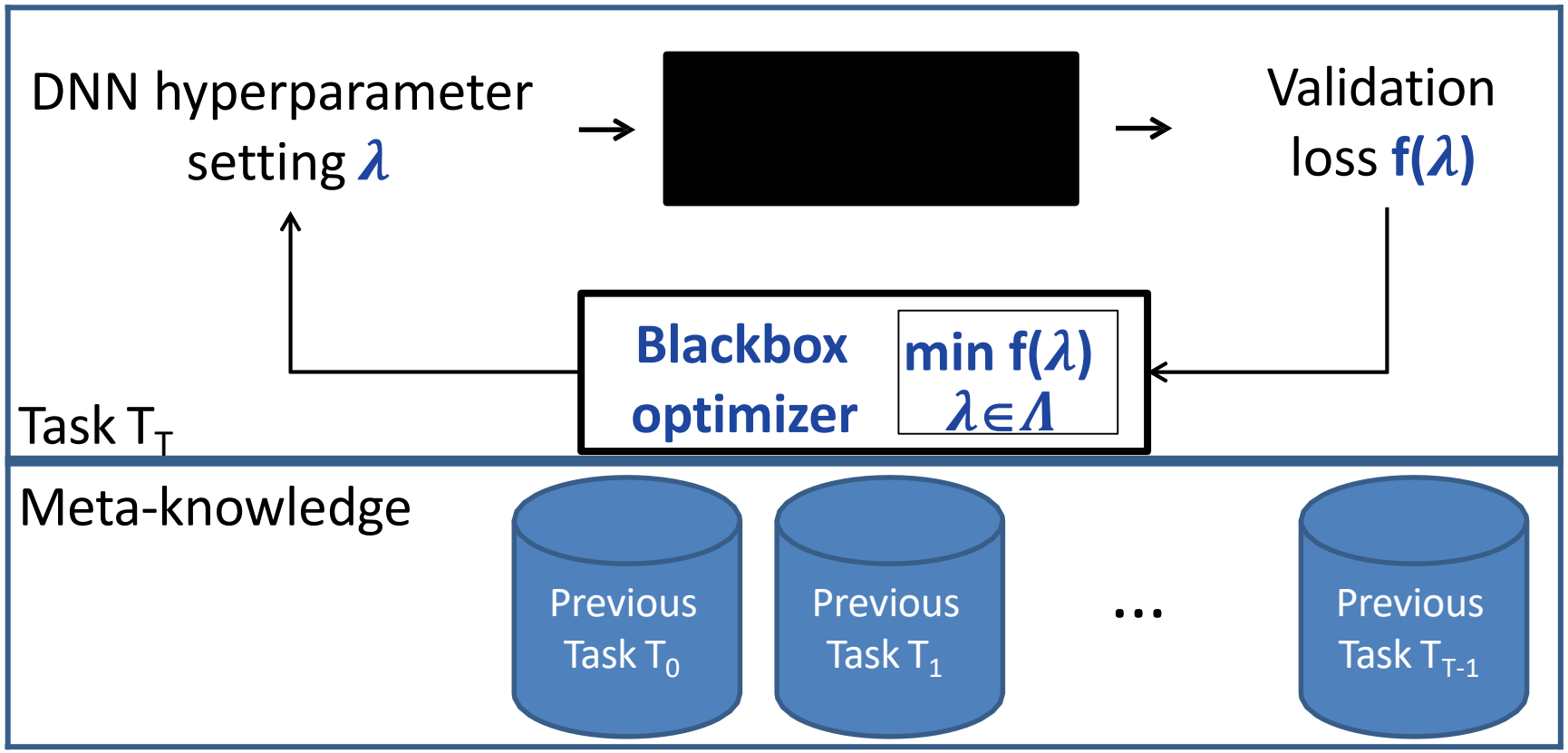
- From relational data bases:
  - Automatically aggregates information, can for example generate the *average sum of orders*
  - Requires post-hoc pruning of the features
  - [Kanter and Veeramachaneni, DSAA 2015]



Example III: Off-the-shelf Algorithms

- Reduce the amount of tuning:
  - Random Forests are excellent default classifiers
  - Learning rate adaption
    - rProp
    - RMSProp
    - …
    - Adam
    - …
    - Ranger (look ahead + rectified Adam)
  - Pre-trained Neural Networks
  - Better defaults
  - …

Example IV: Classical Meta-Learning

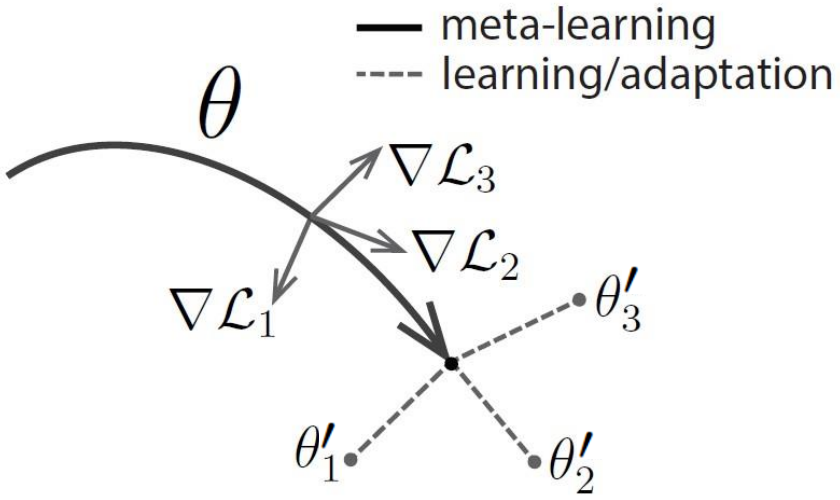


- Learning Defaults
- Joint Models for Bayesian Optimization
- Post-hoc analysis of hyperparameter importance

(see slides in the end for further info and links)

Example V: Learning Weight Inizializations

Model-Agnostic Meta-Learning (MAML) [Finn et al., ICML 2017]



while not done:

1. sample tasks T_i
2. update task weights

$$\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

3. update meta weights θ by solving

$$\min_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}) = \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)})$$

See also
REPTILE [Nichol et al., arXiv preprint 2018]
LEAP [Flennerhag et al., ICLR 2019]
iMAML [Rajeswaran et al., NeurIPS 2019]

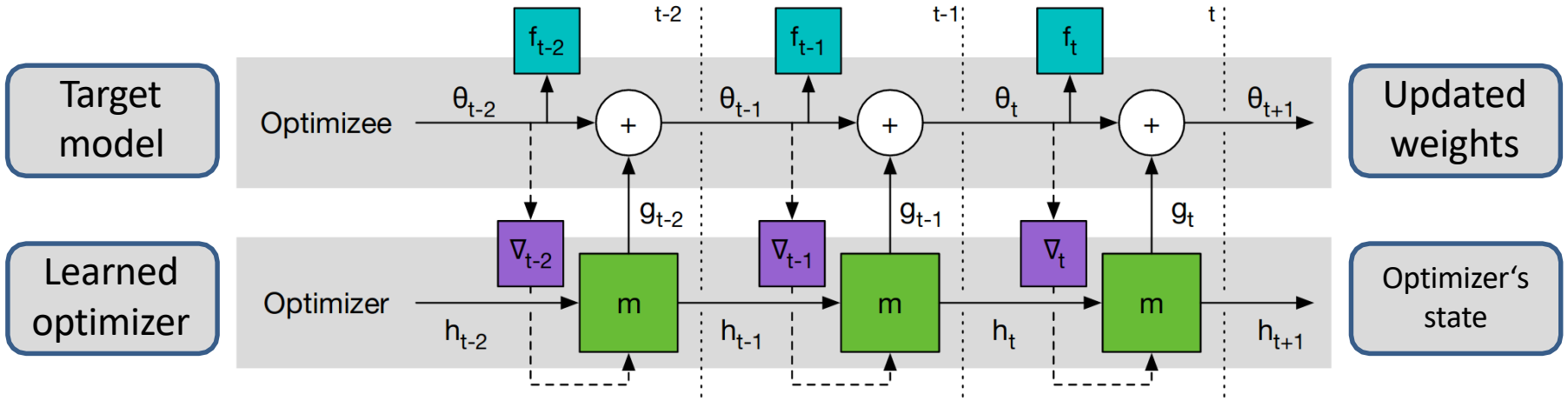
Example VI: Learning to Learn

Deep Learning

engineering features → learning features

Meta Learning

engineering to learn → learning to learn



[Andrychowicz et al., NeurIPS 2016]

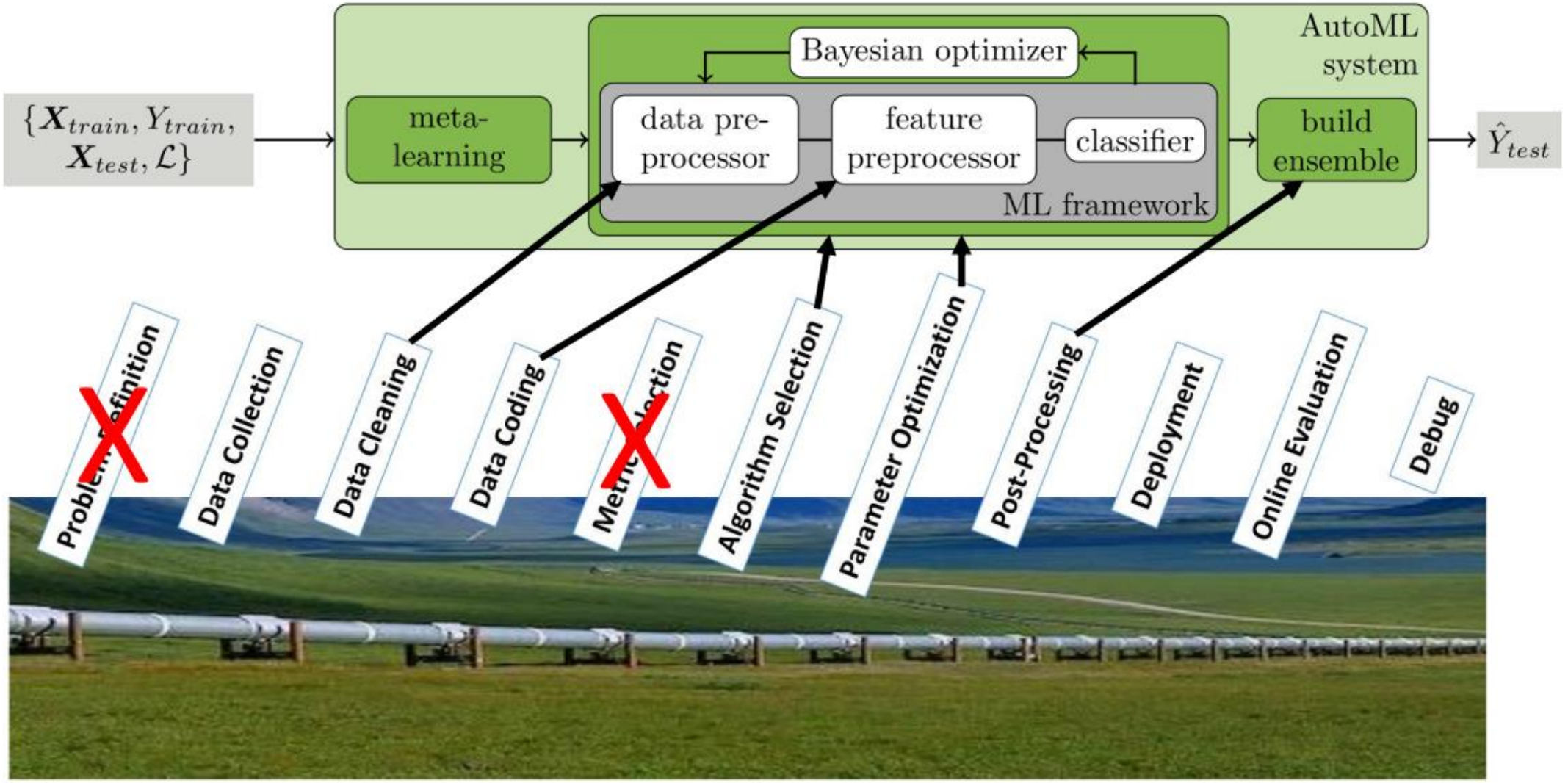


Image credit: Rich Caruana, AutoML 2015

Part 1: General AutoML

1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Examples of AutoML
5. Wrap-up & Conclusion

HPO for Practitioners: Which Tool to Use?

If you have access to multiple fidelities

- We recommend BOHB [Falkner et al, ICML 2018]
- https://github.com/automl/HpBandSter
- Combines the advantages of Bayesian optimization and Hyperband

If you do not have access to multiple fidelities

- Low-dim. continuous: GP-based BO
  (e.g., BoTorch, MLRMBO, Sigopt, GP version of SMACv3)
- High-dim, categorical, conditional: SMAC or Hyperopt
- Purely continuous, budget >10x dimensionality: CMA-ES

Open-source AutoML Tools based on HPO

- Auto-WEKA [Thornton et al, KDD 2013]
  - 768 hyperparameters, 4 levels of conditionality
  - Based on WEKA and SMAC
- Hyperopt-sklearn [Komer et al, SciPy 2014]
  - Based on scikit-learn & TPE
- Auto-sklearn [Feurer al, NeurIPS 2015]
  - Based on scikit-learn & SMAC
  - Uses meta-learning and posthoc ensembling
  - Won AutoML competitions 2015-2016 & 2017-2018
- H2O AutoML [no reference]
  - Uses implementations from H2O.ai
  - Based on random search and stacking
- TPOT [Olson et al, EvoApplications 2016]
  - Based on scikit-learn and evolutionary algorithms
- ML-PLAN [Mohr et al., Machine Learning 2018]
  - Based on WEKA and Hierarchical Task Networks

AutoML: Democratization of Machine Learning



Auto-sklearn also won the last two phases of the AutoML challenge human track (!)

- It performed better than up to 130 teams of human experts
- It is open-source (BSD) and trivial to use:

```
import autosklearn.classification as cls
automl = cls.AutoSklearnClassifier()
automl.fit(X_train, y_train)
y_hat = automl.predict(X_test)
```

automl.github.io/auto-sklearn



→ More in a hands-on session tomorrow

What have we learned?

1. AutoML by Hyperparameter Optimization

   AutoML can be phrased as an HPO problem

2. Black-box Hyperparameter Optimization

   We reviewed Bayesian optimization

3. Beyond black-box optimization

   Practically applicable by using domain knowledge

4. Meta-learning

   Increase practicality by using previous data

5. Examples

   AutoML can be used in almost every step of the ML pipeline

6. Open issues and future work

   Datasets, search space representation & overfitting

Further reading

- Automated Machine Learning: Methods, Systems, Challenges
  - Edited by Frank Hutter, Lars Kotthoff and Joaquin Vanschoren
  - Contains introductions to HPO, Meta-Learning and NAS
  - https://www.springer.com/de/book/9783030053178
- Various literature reviews on arXiv:
  - 1908.05557: Focus on open source software
  - 1810.13306: General and comprehensive
  - 1908.00709: Focuses mostly on NAS
  - 1905.01392: NAS survey
- AutoML workshop video recordings
  - icml2019.automl.org

