Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms

KDD-2013

Chris Thornton, Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown

Department of Computer Science, University of British Columbia

Contents

- Introduction
- Methods
- Experiments
- Improvements

Introduction

AutoML

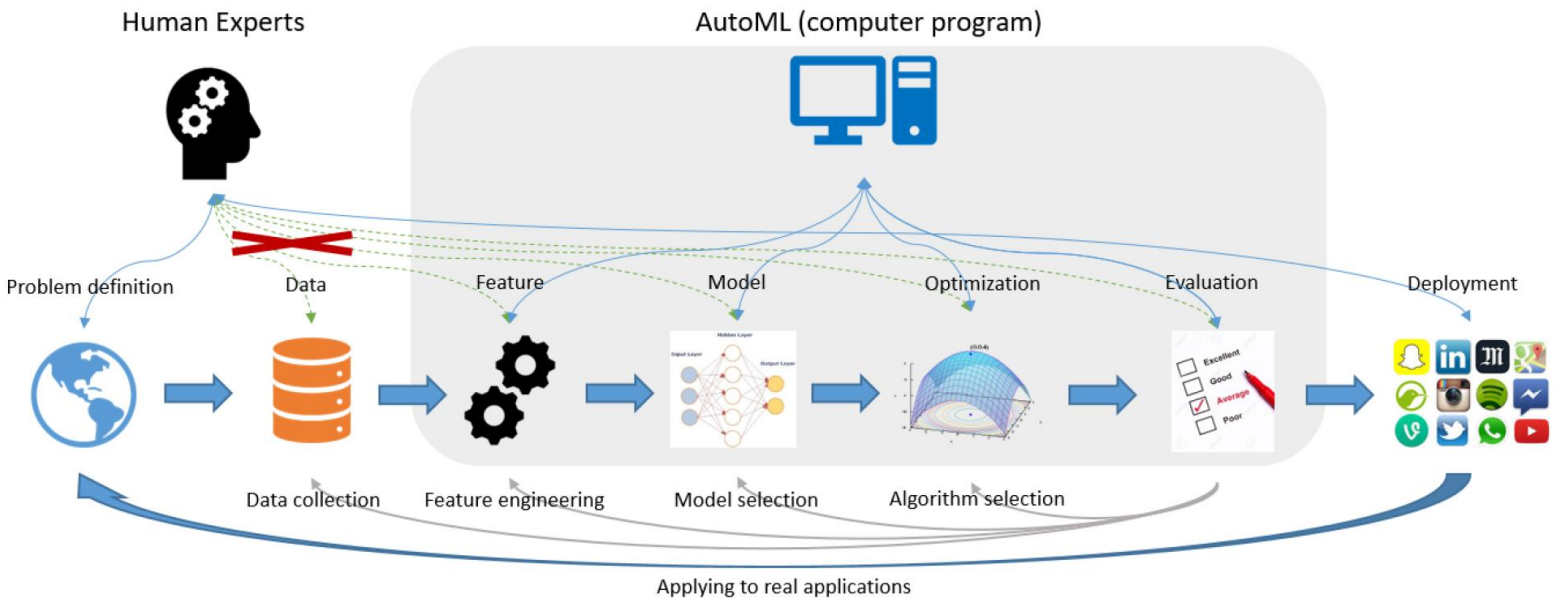


Fig. 1. To use machine learning techniques and obtain good performance, humans usually need to be involved in data collection, feature engineering, model and algorithm selection. This picture shows a typical pipeline of machine learning application, ands how can AutoML involve with the pipeline and minimize participations of humans.

Taking Human out of Learning Applications: A Survey on Automated Machine Learning [arXiv]

Introduction

Applications of AutoML

- Automatic model selection
  - Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms [KDD-2013]
  - Efficient and robust automated machine learning [NIPS-2015]
- Neural architecture search
  - Neural architecture search with reinforcement learning [ICLR-2017] 800 GPUs are used!
  - Progressive neural architecture search [ECCV-2018]
- Automatic feature engineering
  - Deep feature synthesis: Towards automating data science endeavors [ICDSAA-2015]

Taking Human out of Learning Applications: A Survey on Automated Machine Learning [arXiv]

Introduction

- Many different machine learning algorithms exist; taking into account each algorithm's hyperparameters, there is a staggeringly large number of possible alternatives overall.
- This suggests a natural challenge for machine learning: given a dataset, to automatically and simultaneously choose a learning algorithm and set its hyperparameters to optimize empirical performance.
- Combined algorithm selection and hyperparameter optimization problem (short:CASH)

Contents

- Introduction
- Methods
- Experiments
- Improvements

Methods

Model Selection

$$A^* \in \underset{A \in \mathcal{A}}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A, \mathcal{D}_{\text{train}}^{(i)}, \mathcal{D}_{\text{valid}}^{(i)})$$

- $\mathcal{L}(A, \mathcal{D}_{\text{train}}^{(i)}, \mathcal{D}_{\text{valid}}^{(i)})$ is the loss achieved by algorithm A when trained on $\mathcal{D}_{\text{train}}^{(i)}$ and evaluated on $\mathcal{D}_{\text{valid}}^{(i)}$. The k-fold cross-validation is used.

Hyperparameter Optimization

$$\lambda^* \in \underset{\lambda \in \Lambda}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A_\lambda, \mathcal{D}_{\text{train}}^{(i)}, \mathcal{D}_{\text{valid}}^{(i)})$$

- $\lambda \in \Lambda$ are hyperparameters of a given learning algorithm A.

Combined Algorithm Selection and Hyperparameter Optimization (CASH)

$$A^*{}_{\lambda^*} \in \underset{A^{(j)} \in \mathcal{A}, \lambda \in \Lambda^{(j)}}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A_\lambda^{(j)}, \mathcal{D}_{\text{train}}^{(i)}, \mathcal{D}_{\text{valid}}^{(i)})$$

Methods

Sequential Model-Based Optimization (SMBO)

- CASH: $$A^*_{\lambda^*} \in \underset{A^{(j)} \in \mathcal{A}, \lambda \in \mathbf{\Lambda}^{(j)}}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A^{(j)}_{\lambda}, \mathcal{D}^{(i)}_{\text{train}}, \mathcal{D}^{(i)}_{\text{valid}})$$
- This problem can be reformulated as a single combined hierarchical hyperparameter optimization problem with $\mathbf{\Lambda} = \mathbf{\Lambda}^{(1)} \cup \cdots \cup \mathbf{\Lambda}^{(k)} \cup \{\lambda_r\}$
- where λ_r is a new hyperparameter that selects between $A^{(1)}, \ldots, A^{(k)}$.
- Using Sequential Model-Based Optimization to tackle it.

Algorithm 1 SMBO

1: initialise model $\mathcal{M}_L$; $\mathcal{H} \leftarrow \emptyset$
2: **while** time budget for optimization has not been exhausted **do**
3: $\boldsymbol{\lambda} \leftarrow$ candidate configuration from $\mathcal{M}_L$
4: Compute $c = \mathcal{L}(A_{\boldsymbol{\lambda}}, \mathcal{D}^{(i)}_{\text{train}}, \mathcal{D}^{(i)}_{\text{valid}})$
5: $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\boldsymbol{\lambda}, c)\}$
6: Update $\mathcal{M}_L$ given $\mathcal{H}$
7: **end while**
8: **return** $\boldsymbol{\lambda}$ from $\mathcal{H}$ with minimal c

Methods

Selecting the Next Hyperparameter Configuration λ

- Using acquisition function $a_{\mathcal{M}_\mathcal{L}} : \Lambda \to \mathbb{R}$ to quantify (in closed form) how useful knowledge about λ

Expected Improvement (EI): a prominent acquisition function

$$I_{c_{min}}(\boldsymbol{\lambda}) := \max\{c_{min} - c(\boldsymbol{\lambda}), 0\}$$

- c_{min} is an existing given error rate
- We can compute the expectation of $c(\boldsymbol{\lambda})$ about current model $\mathcal{M}_\mathcal{L}$

$$\mathbb{E}_{\mathcal{M}_\mathcal{L}}[I_{c_{min}}(\boldsymbol{\lambda})] := \int_{-\infty}^{c_{min}} \max\{c_{min} - c, 0\} \cdot p_{\mathcal{M}_L}(c \mid \lambda)\, dc$$

Methods

- Main difference between existing SMBO algorithms: the model class they employ.

Sequential Model-based Algorithm Configuration (SMAC)

- SMAC uses random forest models (*which are not usually treated as probabilistic models*)
- SMAC obtains a predictive mean $\mu_{\boldsymbol{\lambda}}$ and variance ${\sigma_{\boldsymbol{\lambda}}}^2$ of $p(c \mid \boldsymbol{\lambda})$ over the predictions of its individual trees for $\boldsymbol{\lambda}$
- It then models $p_{\mathcal{M}_{\mathcal{L}}}(c \mid \boldsymbol{\lambda})$ as a Gaussian $\mathcal{N}(\mu_{\boldsymbol{\lambda}}, {\sigma_{\boldsymbol{\lambda}}}^2)$
- EI can be computed by the closed-form expression:

$$\mathbb{E}_{\mathcal{M}_{\mathcal{L}}}[I_{c_{min}}(\boldsymbol{\lambda})] = \sigma_{\boldsymbol{\lambda}} \cdot [u \cdot \Phi(u) + \varphi(u)]$$

  - where $u = \frac{c_{\min} - \mu_{\boldsymbol{\lambda}}}{\sigma_{\boldsymbol{\lambda}}}$ and φ and Φ denote the probability density function and cumulative distribution function of a standard normal distribution, respectively.

Sequential Model-Based Optimization for General Algorithm Configuration [LION-2011]

Contents

- Introduction
- Methods
- Experiments
- Improvements

Experiments

Baseline Methods

- **Ex-Def**: performing exhaustive 10-fold cross-validation on the training set and to return the classifier with the smallest average misclassification error across folds.
- **Random Grid**: considering a grid of hyperparameter settings for each of base classifiers, and performs a random search in the union of these grids.

Experiments

Table 4: Performance on both 10-fold cross-validation and test data. Random Grid Search was run once for an average of 400 hours per dataset; for SMAC and TPE, we performed 25 runs of 30 hours each. We report results as median percent error rate across 100 000 bootstrap samples simulating 4 parallel runs. Ex-Def is deterministic. Test error rates are determined by training the selected model/hyperparameters on the entire 70% training data and computing the accuracy on the previously unused 30% test data. Boldface indicates the lowest error within a block of comparable methods. SC denotes correlation coefficients (see Section 5.4).

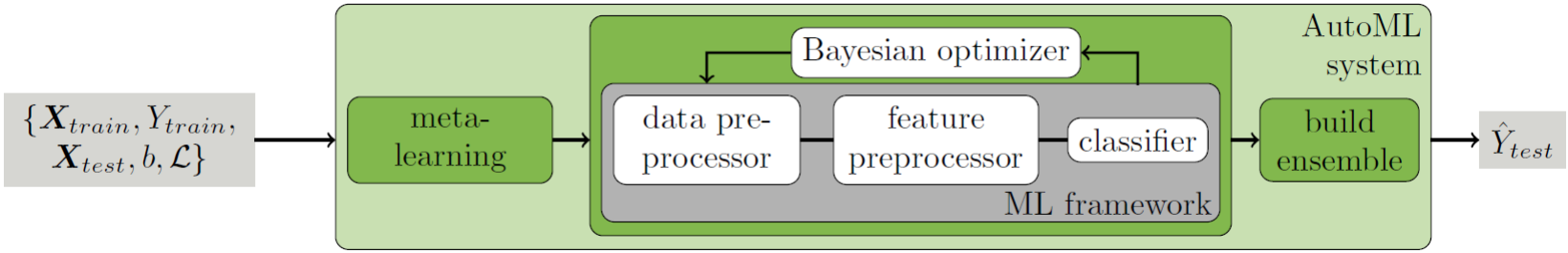
Dataset	Oracle Perf. (%)				10-Fold C.V. Performance (%)				Test Performance (%)				SC	
	Ex-Def		Rand. Grid		Ex-Def	Rand. Grid	Auto-WEKA		Ex-Def	Rand. Grid	Auto-WEKA		TPE	SMAC
	Best	Worst	Best	Worst			TPE	SMAC			TPE	SMAC		
Dexter	7.78	52.78	**5.00**	58.33	10.20	7.48	9.90	**5.48**	8.89	**5.00**	9.44	7.22	0.82	0.25
GermanCredit	26.00	38.00	**25.00**	63.67	22.45	22.45	21.43	**19.59**	**27.33**	**27.33**	27.67	28.33	0.31	0.20
Dorothea	**4.93**	99.24	**4.93**	99.24	6.03	6.03	6.93	**5.52**	6.96	6.96	6.96	**6.38**	0.95	0.40
Yeast	40.00	68.99	**37.53**	68.99	39.43	38.87	**35.03**	36.27	**40.45**	40.90	41.12	**40.45**	0.36	0.49
Amazon	**28.44**	99.33	**28.44**	99.33	**43.94**	**43.94**	48.43	48.30	**28.44**	**28.44**	37.56	37.56	0.92	0.97
Secom	7.87	14.26	**7.66**	40.64	6.25	6.12	6.25	**5.34**	8.09	8.30	**7.87**	**7.87**	-0.10	-0.56
Semeion	8.18	92.45	**6.08**	92.45	6.52	6.52	6.91	**4.86**	8.18	8.18	8.18	**5.03**	0.84	0.73
Car	0.77	29.15	**0.19**	31.66	2.71	1.54	0.94	**0.71**	0.77	0.19	**0.00**	0.58	0.12	0.75
Madelon	**17.05**	50.26	**17.05**	51.03	25.98	24.26	24.26	**20.87**	21.38	**20.77**	**20.77**	21.15	0.44	0.43
KR-vs-KP	0.31	48.96	**0.21**	51.04	0.89	0.70	0.45	**0.32**	**0.31**	0.52	0.52	**0.31**	0.22	0.32
Abalone	73.18	84.04	**72.55**	89.23	73.33	72.45	72.20	**71.76**	73.18	72.79	**72.71**	73.02	0.15	0.10
Wine Quality	36.35	60.99	**36.08**	81.62	38.94	37.28	35.94	**34.74**	37.51	36.08	**33.56**	33.70	0.73	0.85
Waveform	14.27	68.80	**14.20**	68.80	12.73	12.73	12.57	**11.71**	14.40	14.40	**14.20**	14.40	0.36	0.26
Gisette	2.52	50.91	**2.38**	50.91	3.62	3.27	3.70	**2.42**	2.81	2.38	2.57	**2.24**	0.69	0.79
Convex	**25.96**	50.00	**25.96**	50.57	28.68	28.50	29.04	**24.70**	25.96	26.76	25.45	**22.05**	0.98	0.84
CIFAR-10-Small	65.91	90.00	**64.54**	90.00	66.59	65.11	57.97	**57.76**	65.91	64.54	56.65	**55.93**	0.93	0.80
MNIST Basic	5.19	88.75	**3.79**	88.75	5.12	4.00	13.64	**3.64**	5.19	3.79	18.03	**3.56**	1.00	0.87
Rot. MNIST + BI	63.14	88.88	**57.28**	90.96	66.15	59.75	73.04	**59.61**	63.14	58.16	69.86	**55.84**	0.50	0.95
Shuttle	0.0138	20.8414	**0.0069**	20.8414	0.0328	0.0263	**0.0230**	**0.0230**	0.0138	0.0276	**0.0069**	**0.0069**	0.60	0.73
KDD09-Appentency	1.74	6.97	**1.64**	54.08	1.88	1.88	1.88	**1.75**	1.75	1.77	**1.74**	**1.74**	0.89	1.00
CIFAR-10	**64.27**	90.00	**64.27**	90.00	65.54	65.54	66.68	**63.21**	64.27	64.27	64.80	**62.39**	0.33	0.69

Contents

- Introduction
- Methods
- Experiments
- Improvements

Improvements

Methods



- Adding two components to Bayesian hyperparameter optimization of an ML framework:
  - Meta-learning for initializing the Bayesian optimizer
  - Automated ensemble construction from configurations evaluated during optimization

Efficient and Robust Automated Machine Learning [NIPS-2015]

Improvements

Experiments

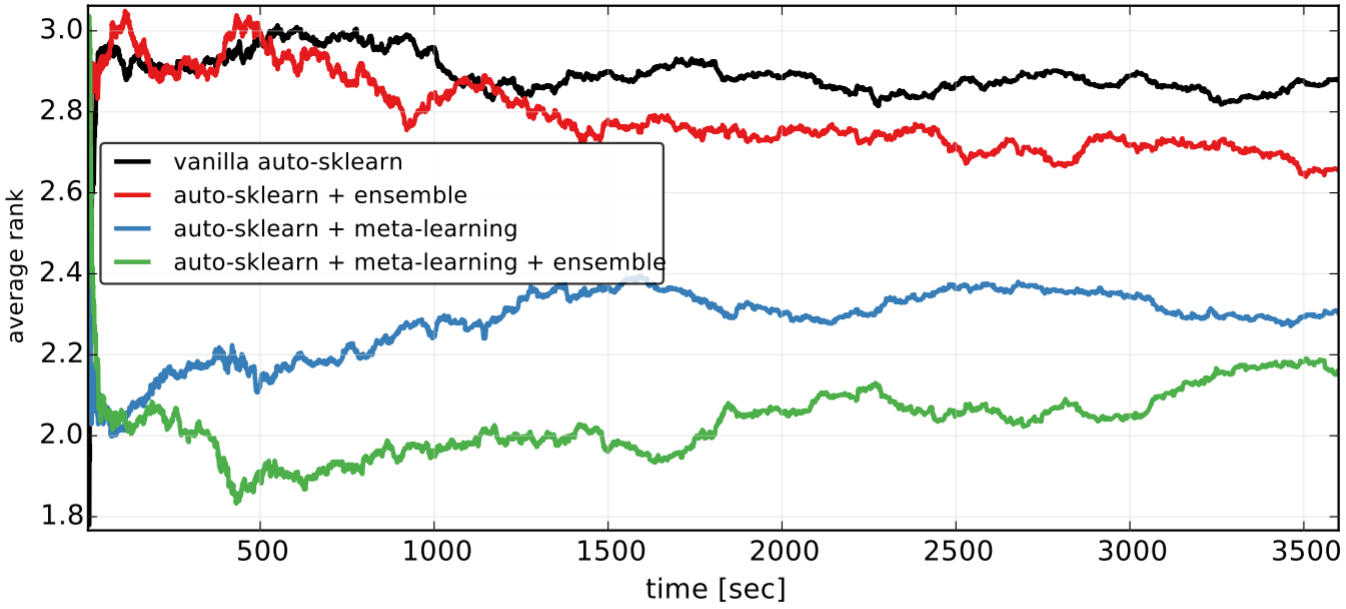


Figure 3: Average rank of all four AUTO-SKLEARN variants (ranked by balanced test error rate (BER)) across 140 datasets. Note that ranks are a relative measure of performance (here, the rank of all methods has to add up to 10), and hence an improvement in BER of one method can worsen the rank of another.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

Efficient and Robust Automated Machine Learning [NIPS-2015]