Auto–WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA

Author: Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown

Problem:
Increasing number of non-expert users trying to use machine learning tools. Wrong algorithm and parameter leads to results that are not ideal.

Existing Packages

- WEKA
- mlr

Input of these packages

- Learning algorithm
- Hyperparameters

-Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)

-Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casalicchio, G., Jones, Z.M.: mlr: Machine Learning in R. Journal of Machine Learning Research 17(170), 1–5 (2016), http://jmlr.org/papers/v17/15-066.html

CASH Problem:

Combined Algorithm Selection and Hyperparameter Optimization

-Given a dataset, automatically and simultaneously choosing a learning algorithm and setting its hyperparameters to optimize empirical performance

Challenge:
-Response function noisy

-Space is high dimensional

CASH: single hierarchical hyperparameter optimization problem

Preliminaries

$f : \mathcal{X} \mapsto \mathcal{Y}$ Y is either finite (classification) or continuous (regression)

Training data set: $\{d_1, \ldots, d_n\}$ $d_i = (\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$

A learning algorithm *A* tries to map d_i to such a function

Most learning algorithms A further expose to hyperparameter $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$

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

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

CASH

Given a set of algorithms $\mathcal{A} = \{A^{(1)}, \ldots, A^{(k)}\}$ with associated hyperparameter spaces $\mathbf{\Lambda}^{(1)}, \ldots, \mathbf{\Lambda}^{(k)}$, we define the combined algorithm selection and hyperparameter optimization problem (CASH) as computing

$$A^{*}{}_{\boldsymbol{\lambda}^{*}} \in \operatorname*{argmin}_{A^{(j)} \in \mathcal{A}, \boldsymbol{\lambda} \in \mathbf{\Lambda}^{(j)}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A_{\boldsymbol{\lambda}}^{(j)}, \mathcal{D}_{\text{train}}^{(i)}, \mathcal{D}_{\text{valid}}^{(i)}). \tag{4.1}$$

- Single combined hierarchical hyperparameter optimization question

$$\mathbf{\Lambda} = \mathbf{\Lambda}^{(1)} \cup \cdots \cup \mathbf{\Lambda}^{(k)} \cup \{\lambda_r\}$$

Sequential Model-Based Optimization (SMBO)

Algorithm 1 SMBO

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

Acquisition function: positive expected improvement $I_{c_{min}}(\boldsymbol{\lambda}) := \max\{c_{min} - c(\boldsymbol{\lambda}), 0\}.$

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

Sequential model-based algorithm configuration (SMAC)

- Random Forest
- Obtain mean and variance

$$\mathbb{E}_{\mathcal{M}_{\mathcal{L}}}[I_{c_{min}}(\boldsymbol{\lambda})] = \sigma_{\boldsymbol{\lambda}} \cdot [u \cdot \Phi(u) + \varphi(u)], \quad \text{where } u = \frac{c_{\min} - \mu_{\boldsymbol{\lambda}}}{\sigma_{\boldsymbol{\lambda}}}$$

- Make progressively better estimates of this mean by evaluating these terms one at a time, thus trading off accuracy and computational cost
- In order for a new configuration to become a new incumbent, it must outperform the previous incumbent in every comparison made
- Implements a diversification mechanism to achieve robust performance even when its model is misled: every second configuration is selected at random

Auto-WEKA

- Focused on classification algorithms in WEKA
- Meta-Methods: 1 single base classifier and its parameters as input
- Ensemble methods: up to 5 learners
- Associated either a uniform or log uniform prior with each numerical parameter, depending on its semantics
- Auto-WEKA uses SMAC to solve CASH

Source code:
https://github.com/automl/autoweka

Base Learners			
BayesNet	2	NaiveBayes	2
DecisionStump*	0	NaiveBayesMultinomial	0
DecisionTable*	4	OneR	1
GaussianProcesses*	10	PART	4
IBk*	5	RandomForest	7
J48	9	RandomTree*	11
JRip	4	REPTree*	6
KStar*	3	SGD*	5
LinearRegression*	3	SimpleLinearRegression*	0
LMT	9	SimpleLogistic	5
Logistic	1	SMO	11
M5P	4	SMOreg*	13
M5Rules	4	VotedPerceptron	3
MultilayerPerceptron*	8	ZeroR*	0
Ensemble Methods			
Stacking	2	Vote	2
Meta-Methods			
LWL	5	Bagging	4
AdaBoostM1	6	RandomCommittee	2
AdditiveRegression	4		
AttributeSelectedClassifier	2	RandomSubSpace	3
Feature Selection Methods			
BestFirst	2	GreedyStepwise	4

Fig. 4.1 Learners and methods supported by Auto-WEKA, along with number of hyperparameters $|\Lambda|$. Every learner supports classification; starred learners also support regression

Datasets

- 15 sets from the UCI repository [13]
- the ‘convex’, ‘MNIST basic’ and ‘rotated MNIST with background images’ tasks used in [5];
- The appentency task from the KDD Cup ’09
- two versions of the CIFAR-10 image classification task [21] (CIFAR-10-Small is a subset of CIFAR-10, where only the first 10,000 training data points are used rather than the full 50,000.)
- For datasets with a predefined training/test split, used that split. Otherwise, randomly split the dataset into 70% training and 30% test data.
- Each dataset: time budget of 30h
- Each method: 25 runs with different random seeds, then bootstrap sampling to repeatedly select 4 random runs and report the one with best cross-validation performance

Table 4.1 Datasets used; *Num. Discr.*. and *Num. Cont*. refer to the number of discrete and continuous attributes of elements in the dataset, respectively

Name	Num Discr.	Num Cont.	Num classes	Num training	Num test
Dexter	20,000	0	2	420	180
GermanCredit	13	7	2	700	300
Dorothea	100,000	0	2	805	345
Yeast	0	8	10	1,038	446
Amazon	10,000	0	49	1,050	450
Secom	0	591	2	1,096	471
Semeion	256	0	10	1,115	478
Car	6	0	4	1,209	519
Madelon	500	0	2	1,820	780
KR-vs-KP	37	0	2	2,237	959
Abalone	1	7	28	2,923	1,254
Wine Quality	0	11	11	3,425	1,469
Waveform	0	40	3	3,500	1,500
Gisette	5,000	0	2	4,900	2,100
Convex	0	784	2	8,000	50,000
CIFAR-10-Small	3,072	0	10	10,000	10,000
MNIST Basic	0	784	10	12,000	50,000
Rot. MNIST + BI	0	784	10	12,000	50,000
Shuttle	9	0	7	43,500	14,500
KDD09-Appentency	190	40	2	35,000	15,000
CIFAR-10	3,072	0	10	50,000	10,000

Baseline Method

- **Ex-Def**: perform 10-fold cross validation on the training set for each technique with unmodified hyperparameters, and select the classifier with the smallest average misclassification error across folds
- **Grid Search**: performs an exhaustive search over a grid of hyperparameter settings for each of the base learners, discretizing numeric parameters into three points (Expensive, some datasets require more than 10,000 CPU hours)
- **Random Search**: picking algorithms and hyperparameters sampled at random, and computes their performance on the 10 cross-validation folds until it exhausts its time budget (750 CPU hours per dataset for cross-validation, then 120 CPU hours for sampling without replacement)

Table 4.2 Performance on both 10-fold cross-validation and test data. Ex-Def and Grid Search are deterministic. Random search had a time budget of 120 CPU hours. For Auto-WEKA, we performed 25 runs of 30 h each. We report results as mean loss across 100,000 bootstrap samples simulating 4 parallel runs. We determined test loss (misclassification rate) by training the selected model/hyperparameters on the entire 70% training data and computing accuracy on the previously unused 30% test data. Bold face indicates the lowest error within a block of comparable methods that was statistically significant

	Oracle Perf. (%)				10-Fold C.V. performance (%)				Test performance (%)			
	Ex-Def		Grid search									
Dataset	Best	Worst	Best	Worst	Ex-Def	Grid search	Rand. search	Auto-WEKA	Ex-Def	Grid search	Rand. search	Auto-WEKA
Dexter	7.78	52.78	**3.89**	63.33	10.20	**5.07**	10.60	5.66	8.89	**5.00**	9.18	7.49
GermanCredit	26.00	38.00	**25.00**	68.00	22.45	20.20	20.15	**17.87**	27.33	**26.67**	29.03	28.24
Dorothea	4.93	99.24	**4.64**	99.24	6.03	6.73	8.11	**5.62**	6.96	5.80	**5.22**	6.21
Yeast	40.00	68.99	**36.85**	69.89	39.43	39.71	38.74	35.51	40.45	42.47	43.15	40.67
Amazon	28.44	99.33	**17.56**	99.33	43.94	**36.88**	59.85	47.34	28.44	**20.00**	41.11	33.99
Secom	7.87	14.26	**7.66**	92.13	6.25	6.12	5.24	**5.24**	8.09	8.09	8.03	**8.01**
Semeion	8.18	92.45	**5.24**	92.45	6.52	4.86	6.06	**4.78**	8.18	6.29	6.10	**5.08**
Car	0.77	29.15	**0.00**	46.14	2.71	0.83	**0.53**	0.61	0.77	0.97	**0.01**	0.40
Madelon	**17.05**	50.26	**17.05**	62.69	25.98	26.46	27.95	**20.70**	21.38	21.15	24.29	**21.12**
KR-vs-KP	0.31	48.96	**0.21**	51.04	0.89	0.64	0.63	**0.30**	0.31	1.15	0.58	**0.31**
Abalone	73.18	84.04	**72.15**	92.90	73.33	72.15	72.03	**71.71**	73.18	73.42	74.88	73.51
Wine Quality	36.35	60.99	**32.88**	99.39	38.94	35.23	35.36	**34.65**	37.51	34.06	34.41	33.95
Waveform	14.27	68.80	**13.47**	68.80	12.73	12.45	12.43	**11.92**	14.40	14.66	14.27	14.42
Gisette	2.52	50.91	**1.81**	51.23	3.62	2.59	4.84	**2.43**	2.81	2.40	4.62	**2.24**
Convex	25.96	50.00	**19.94**	71.49	28.68	**22.36**	33.31	25.93	25.96	23.45	31.20	**23.17**
CIFAR-10-Small	65.91	90.00	**52.16**	90.36	66.59	**53.64**	67.33	58.84	65.91	56.94	66.12	**56.87**
MNIST Basic	5.19	88.75	**2.58**	88.75	5.12	**2.51**	5.05	3.75	5.19	**2.64**	5.05	3.64
Rot. MNIST + BI	63.14	88.88	**55.34**	93.01	66.15	**56.01**	68.62	57.86	63.14	57.59	66.40	**57.04**
Shuttle	0.0138	20.8414	**0.0069**	89.8207	0.0328	0.0361	0.0345	**0.0224**	0.0138	0.0414	0.0157	**0.0130**
KDD09-Appentency	1.7400	6.9733	**1.6332**	54.2400	1.8776	1.8735	1.7510	**1.7038**	1.7405	1.7400	1.7400	**1.7358**
CIFAR-10	64.27	90.00	**55.27**	90.00	65.54	**54.04**	69.46	62.36	64.27	63.13	69.72	**61.15**

Result

- Auto-WEKA outperforms baseline methods 15/21
- Grid search and random search better than Ex-Def
- Auto-WEKA good on large datasets 12/13
    - Risk of overfitting decreases with large datasets
- Substantial improvements, relative reduction of test misclassification rate exceeding 16% in 3/21 cases