Automated Machine Learning with Monte-Carlo Tree Search

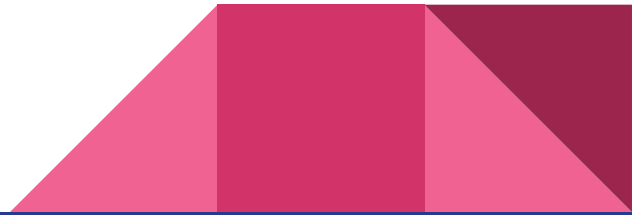
IJCAI-2019

Authors: Rakotoarison, Schoenauer, and Sebag
Presenter: James Quintero

University of California, Irvine

Outline

1. Hyperparameters
2. AutoML
3. Introduction to Auto-sklearn and Mosaic
4. Mosaic notation
5. Mosaic’s MCTS
6. Experiments
7. Results

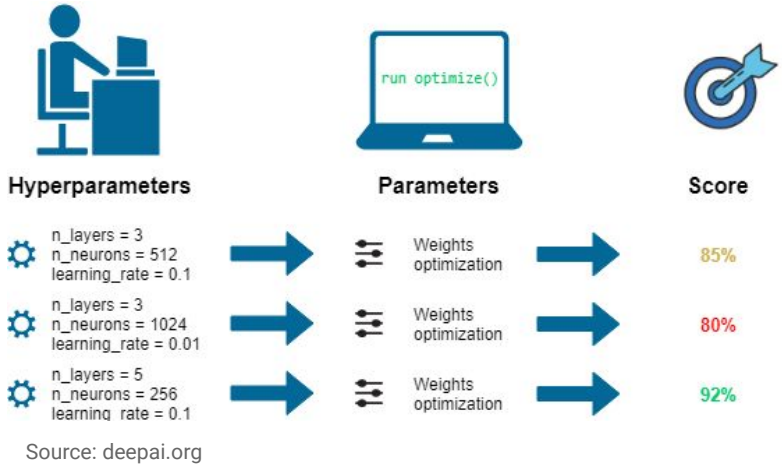


ML Hyperparameter Optimization

Two types of hyperparameters:

- Model
  - Ex: Neural network architecture type, feature layer size, number of hidden layers, and more
- Algorithm
  - Ex: Loss function for back-propagation, batch size, dropout amount, activation functions

Trivial approach - Grid search: Testing a range of combinations of algorithm hyperparameters, and returning the best performing combination.



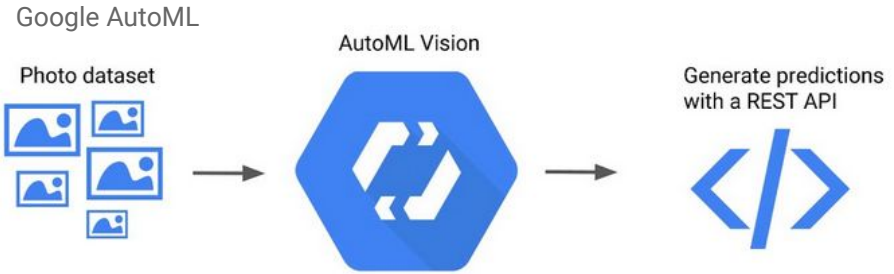
Source: deepai.org

AutoML

- Task that chooses the best performing model hyperparameters and algorithm hyperparameters to deliver the best performing overall system. Performance can be defined as accuracy, sensitivity, or a custom metric.
- Helps non-experts deploy an ML system with only requiring a dataset.
- **Auto-sklearn** is the current leader in AutoML competitions.

Notable companies who offer AutoML services:

- Google
- Microsoft



Introduction

- Desire of AutoML is to maximize F(x)

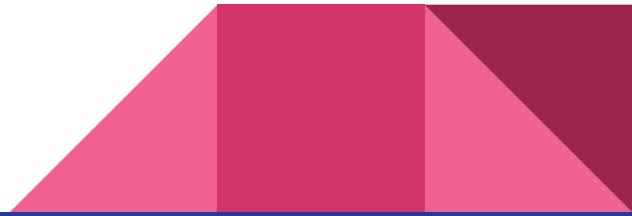
$$\text{Find } \mathbf{x}^* \in \arg\max_{x \in X} \mathcal{F}(\mathbf{x})$$

- Where X is the scope of ML configurations, and F(x) is the performance of configuration x.
- Pipeline: An ML configuration, like the choice of model type.
- Bayesian Optimization: A system of finding a global max for a black fox function by using a surrogate model.
- Surrogate model: A model of the desired outcome; In this case a model of the optimization objective F.

Auto-sklearn

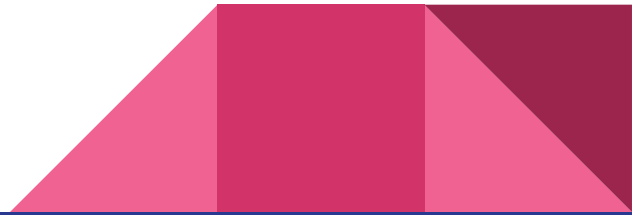
Reference: Efficient and Robust Automated Machine Learning

- Current state of the art, introduced in 2015
- Based on scikit-learn library
- Uses Bayesian optimization for optimizing model hyperparameters
- Uses Bayesian optimization for optimizing algorithm hyperparameters
  - Single surrogate model is used for determining performance.



Mosaic

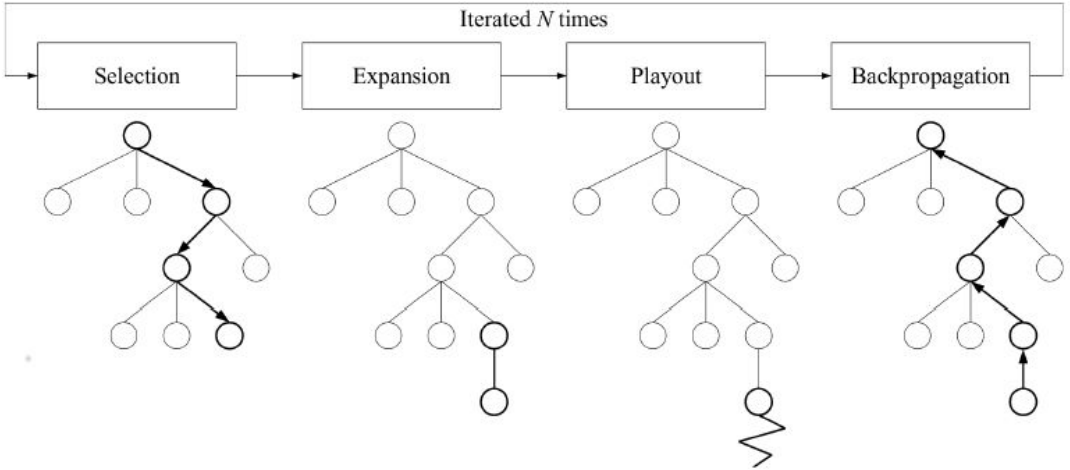
- Uses MCTS for optimizing model hyperparameters
- Uses Bayesian optimization for optimizing algorithm hyperparameters
- Different than Auto-sklearn in that the surrogate model(s) is used to couple the model hyperparameter optimization and algorithm hyperparameter optimization.



Monte-Carlo Tree Search

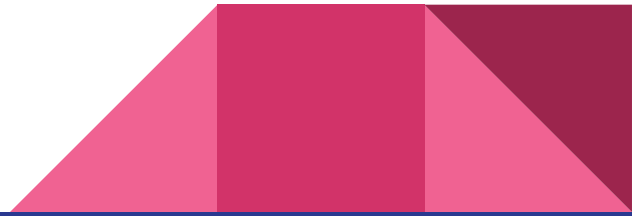
Four phases:

1. Selection
2. Expansion
3. Playout
4. Backpropagation



Search Initialization

- The initialization and selection of the first candidates is important in that its quality determines the surrogate model's quality.
- Auto-sklearn can use MetaLearning heuristics, by having an archive of (dataset_type, model) pairs. These models are known to work well with specified datasets.
- Auto-sklearn ensembling and Mosaic ensembling are made up of the weighted sum of the models learned along the search, where the weights are optimized on a validation set



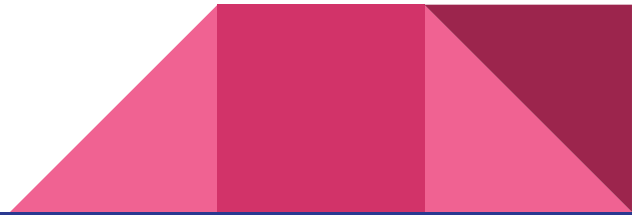
Mosaic Introduction

Let's introduce some notation

- An ML pipeline *x* involves a fixed ordered sequence of ℓ decisions.
- At the ith decision step, some algorithm $a_i \in A_i$ is selected (A_i is the set of possible algorithms at ith step)
  - EX: ith step might contain algorithms "Deep network, shallow network, recurrent network"
- $\Theta(a_i)$ = the space of hyper-parameters associated with a_i and $\theta_i \in \Theta(a_i)$
  - EX: θ_i can be hyperparameters "network depth, hidden layer size, etc" associated with an example a_i "Deep network"
- With $x = (a_1, \theta_1), \ldots, (a_\ell, \theta_\ell)$, a complete pipeline structure is an ℓ-uple of $a = (a_1,\ldots,a_\ell)$
  - its hyperparameters $\Theta(a) = (\Theta(a_1) \times \ldots \times \Theta(a_\ell))$
- X = the entire ℓ-uple space

Two Optimization Problems

- Remember, we have to optimize which model hyperparameters to use ($a \in A$), and its hyperparameters ($\theta(a) \in \Theta(a)$).
- At one extreme, one could optimize $\theta(a)$ for every considered a – a brute-force and very slow way.
- Auto-sklearn improves on this by using Bayesian Optimization for a_i and $\theta(a_i)$
- Instead, Mosaic estimates the performance of a_i from a few samples of $\theta(a)$, creating a surrogate model,



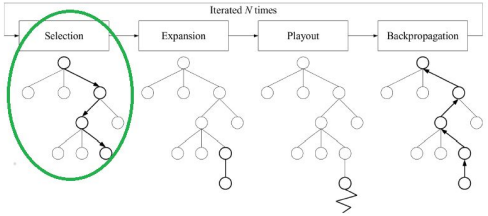
Creating the Surrogate Model

- With Mosaic and Auto-sklearn, a surrogate model is computed from all performances ($F(x_t = (a_t, \theta(a_t))$), we'll call F'.
- Authors average the first 100 F' to get $Q_{F'}$
- A probabilistic selection policy π is calculated from $Q_{F'}$

$$\pi(a|s) = \frac{\exp\left(Q_{\widehat{F}}(s,a)\right)}{\sum_{b \in \mathcal{A}_k} \exp\left(Q_{\widehat{F}}(s,b)\right)}$$

π is then used to enhance MCTS selection rule.

Mosaic's MCTS Selection Phase



Regular MCTS selection algorithm:

$$\text{select } \arg\max_a \left\{ \hat{\mu}_{s.a} + C_{ucb} \sqrt{\frac{\log n(s)}{n(s.a)}} \right\}$$

AlphaGoZero algorithm:

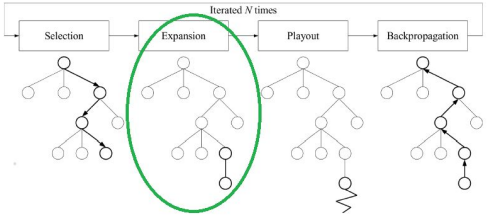
$$\operatorname*{argmax}_a \left(\overline{Q}(\mathbf{s}, a) + C_{ucb} * \pi(a|\mathbf{s}) * \frac{\sqrt{n(\mathbf{s})}}{1 + n(\mathbf{s}.a)} \right)$$

Q(s.a) = median of F(x) for all x in X(s.a)

n(s.a) = the number of visits to node s.a

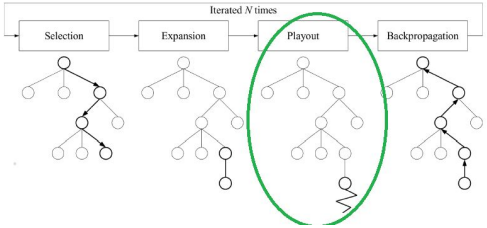
C_{ucb} = Constant that controls exploitation vs exploration trade-off.

Mosaic’s MCTS Expansion Phase



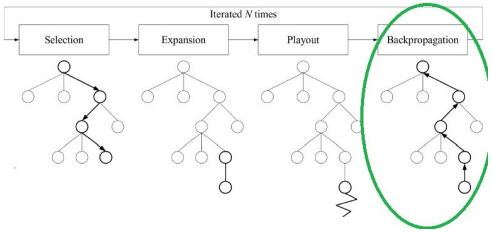
- More traditional MCTS Expansion uses RAVE and Progressive Widening to add new child nodes.
- Mosaic uses the surrogate performance $Q_{F'}(s.a)$ where the child that maximizes $Q_{F'}(s.a)$ is added to the MCTS tree

Mosaic’s MCTS Playout Phase



- A number of configurations are sampled locally around (a*, θ(a*))
- The sample that maximizes the Expected Improvement of F’ is retained. F(x) of the retained configuration is computed for use in Back propagation step.

Mosaic’s MCTS Back-propagation



- The calculated F(x) value is back-propagated up the tree, updating each current path’s Q(s.a) value (used in Selection phase).
- (x, F(x)) is added to the surrogate training set, and F’ is retrained.

- Mosaic stocks the MCTS process after the computational budget is exceeded. In the case of the experiments, it’s 1 hour per dataset).

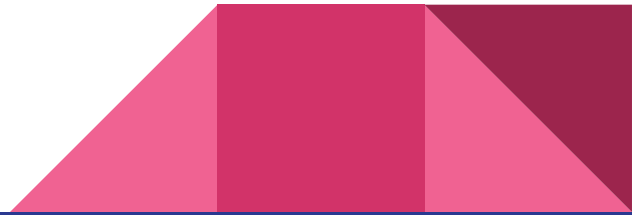
Different Initializations Tested

List of initializations for both Mosaic and Auto-sklearn that will be tested

- Vanilla
  - Regular Mosaic algorithm, no special initialization. Will return best pipeline
- Metalearning
  - Is initialized using Auto-Sklearn's metalearning initialization. First 25 neighboring configurations to the current dataset are selected, with Mosaic.Vanilla running the remainder.
- Ensemble
  - Mosaic.Vanilla finds the best pipeline, then an ensembled model is computed from the weighted sum of models computed along the AutoML search.

Search Space Specifics

- The search space involves
  - 16 ML algorithms
  - 13 pre-processing methods
  - 2 categorical encoding strategies
  - 4 missing values imputation strategies
  - 6 rescaling strategies and 2 balancing strategies.
- The size of the structural search subspace is 6,048. Less than 16*13*2*4*6 = 9984 because of parameter dependencies.
- For each configuration $x = (a, \theta(a))$, $\theta(a)$ involves a dozen scalar hyper-parameters on average.



Model training specifics

- Mosaic and Auto-sklearn were assessed on the OpenML repository and 100 extra classification datasets
- Overall computational budget is set to 1 hour for each dataset
- Trained on 70% of the training set with a cut-off time of 300 seconds, and performance F(x) is set to the model accuracy on the remaining 30%
    - F(x) is computed on the validation portion of the dataset
    - No hard limit for number of epochs
- Testing phase - At the end, the best configuration x* (on validation set) is trained on 100% of the training set and its **performance** on the (unseen) test set, is calculated.
- This performance is averaged over 10 independent runs, and the average is reported as the **system performance** on this dataset.
- Finally, the system performances from all systems, are **ranked**. The overall performance of a system is its average rank over all (the lower the better)
    - Each system is ranked 1^{st}, 2^{nd}, or 3^{rd}.

Results for all variants

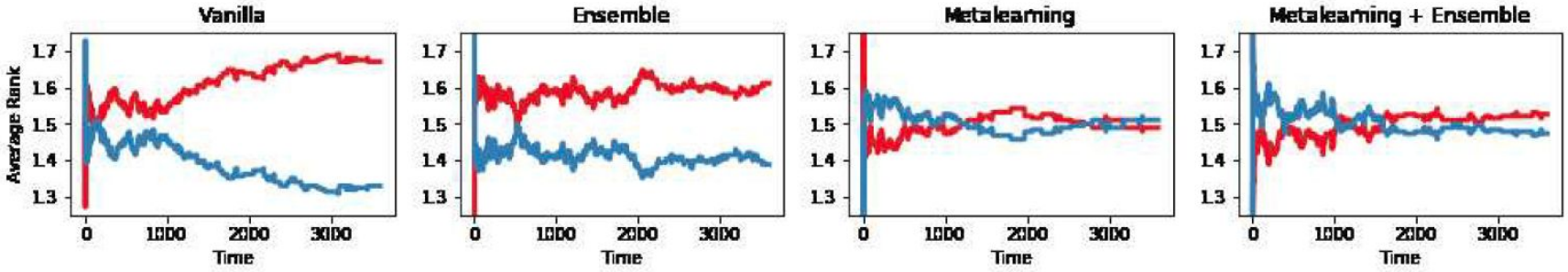
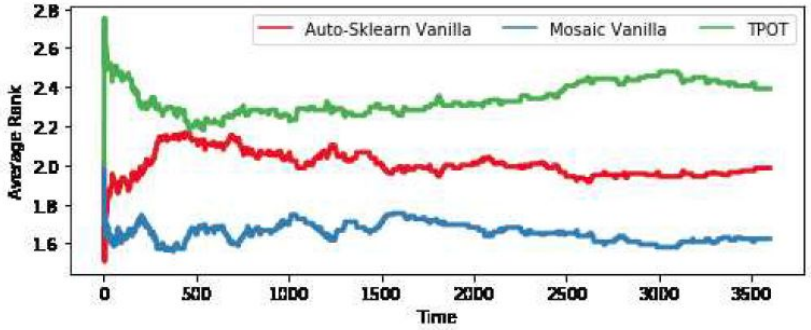


Figure 2: Comparative assessment of MOSAIC and AUTO-SKLEARN: Average performance rank (the lower the better) on OpenML-100 *vs* CPU time of the Vanilla, Ensemble, MetaLearning and Ensemble+MetaLearning variants (left to right). Better seen in color.

Results for Vanilla variant



Mosaic's MCTS optimization improves on Auto-sklearn's Bayesian optimization (and on TPOT, which uses evolutionary optimization only)

Results for Vanilla variant cont.

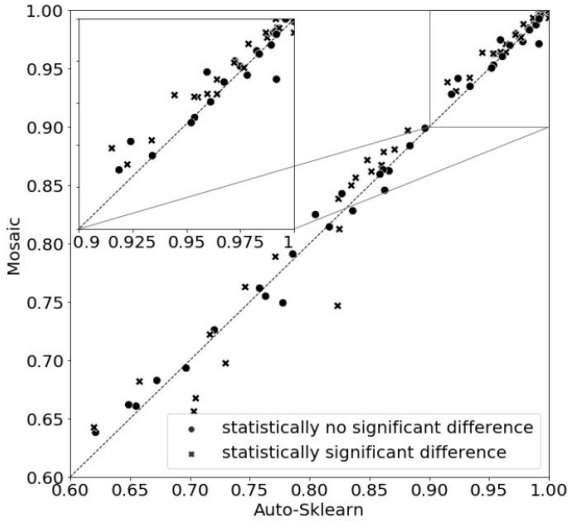


Figure 4: Performance of MOSAIC (y-axis) versus AUTO-SKLEARN (x-axis) on OpenML-100. Datasets for which the difference is statistically significant (resp. insignificant) after MWW test with confidence 5% are represented with a × (resp ●).

- Axis are Performance
- Significance determined by Mann-Whitney-Wilcoxon test
- Mosaic significantly outperforms Auto-sklearn on 21/100 datasets
- Auto-sklearn outperforms MOSAIC on 6/100 datasets
- Mosaic improves on Auto-sklearn on 35/100 datasets (though not in a statistically significant way).
- Auto-sklearn improves on Mosaic on 18/100 datasets (though not in a statistically significant way).
- Both are equal on 18/100 datasets
- Both systems crashed on 2/100 datasets

Results for Ensemble and MetaLearning variant

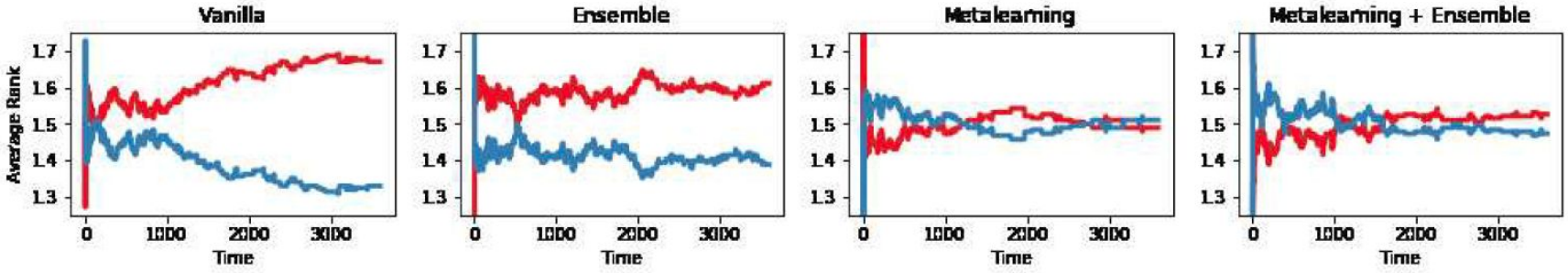


Figure 2: Comparative assessment of MOSAIC and AUTO-SKLEARN: Average performance rank (the lower the better) on OpenML-100 *vs* CPU time of the Vanilla, Ensemble, MetaLearning and Ensemble+MetaLearning variants (left to right). Better seen in color.

- The best Auto-sklearn configuration is mostly found during the initialization.
  - Therefore, Auto-sklearn.MetaLearning mostly explores the neighborhood of the initial configurations.
- Mosaic explores more at first, which could lead to overfitting by constantly improving the validation set performance at the expense of the test set performance

More Results for Mosaic Vanilla variant

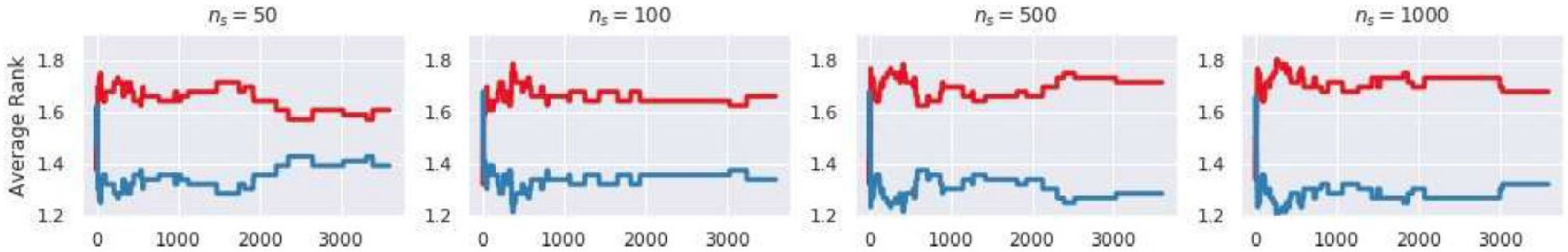


Figure 5: Sensitivity study w.r.t. hyper-parameters n_s for $C_{ucb} = 1.3$ and $PW = 0.6$: Average rank of MOSAIC.Vanilla against AUTO-SKLEARN.Vanilla. Better seen in color.

- The n_s hyperparameter has no significant effect on performance of Mosaic or Auto-sklearn.

Results of different Hyper-hyperparameters

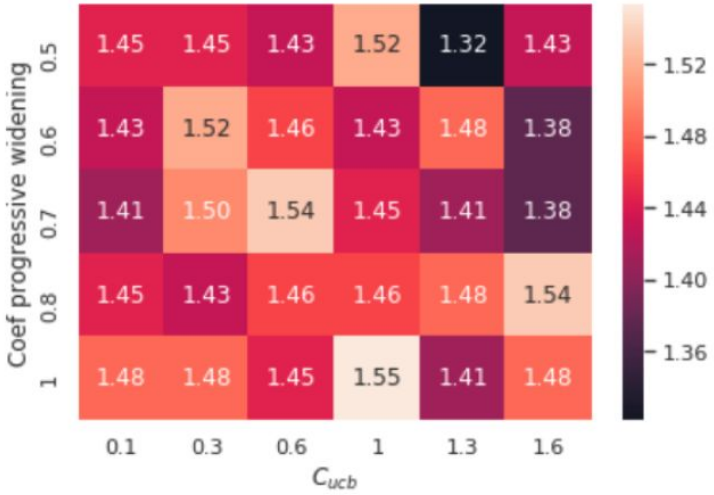
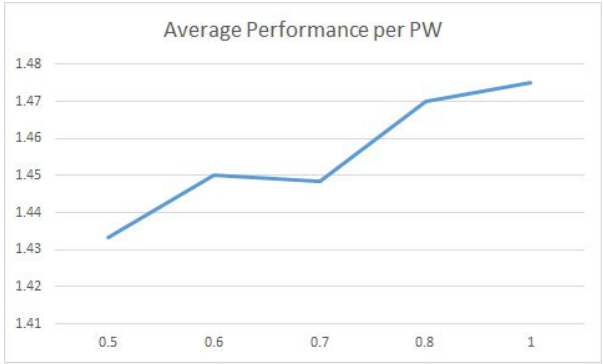
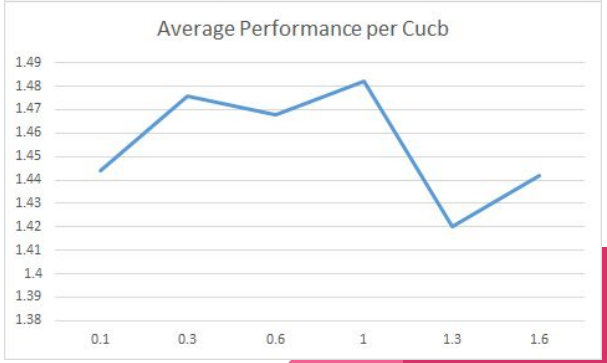


Figure 6: Sensitivity study w.r.t. hyper-parameters C_{ucb} and PW, for $n_r = 100$: Average rank of MOSAIC.Vanilla against AUTO-SKLEARN.Vanilla (the lower, the better).



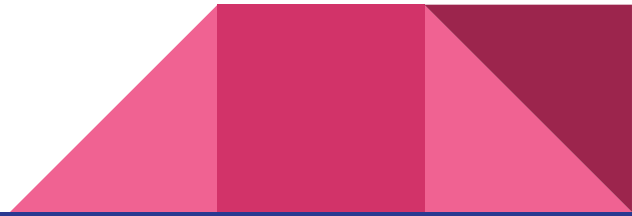


Discussion

- The main contribution of this paper is the new Mosaic algorithm that handles both the model hyperparameter optimizations and algorithm hyperparameter optimizations with surrogate models.
- For later stages of search, Mosaic differs from Auto-sklearn by switching to the exploitation of the most promising MCTS subtrees and avoiding poor performing regions, while Auto-sklearn continues to explore seemingly bad configurations.
- The results demonstrate that Mosaic's Vanilla and Ensemble variants significantly outperforms the challenge winner Auto-sklearn's Vanilla and Ensemble variants on the OpenML benchmark suite.
- For MetaLearning, the main contribution comes from the initialization, which Mosaic and Auto-sklearn both share.
- In the absence of a pre-existing archive, Mosaic offers itself as a robust AutoML algorithm, outperforming all others.

Question

How is Mosaic’s MCTS Selection algorithm different than the traditional MCTS algorithm?



References

- Automated Machine Learning with Monte-Carlo Tree Search, Herilalaina Rakotoarison, Marc Schoenauer, Michele Sebag, IJCAI-19. *28th International Joint Conference on Artificial Intelligence*, Macao, China, August 2019.
- Efficient and Robust Automated Machine Learning, Mathias Feurer, et al, NIPS-2015, Advanced in Neural Information Processing Systems 28, 2015.