Monte Carlo Tree Search for Algorithm Configuration: MOSAIC

Herilalaina Rakotoarison and Michèle Sebag

TAU

CNRS — INRIA — LRI — Université Paris-Sud

NeurIPS MetaLearning Wshop — Dec. 8, 2018
Monte Carlo Tree Search for Algorithm Configuration: MOSAIC

Herilalaina Rakotoarison and Michèle Sebag
Tackling the Underspecified

CNRS — INRIA — LRI — Université Paris-Sud

NeurIPS MetaLearning Wshop — Dec. 8, 2018
AutoML: Algorithm Selection and Configuration

A mixed optimization problem

$$\text{Find } \lambda^* \in \arg \min_{\lambda \in \Lambda} L(\lambda, P)$$

with $\lambda$ a pipeline and $L$ the predictive loss on dataset $P$

Modes

- offline hyper-parameter setting
- online hyper-parameter setting

Approaches

- Bayesian optimization: SMAC, Auto-SkLearn, AutoWeka, BHOB
  Hutter et al., 11; Feurer et al. 15; Kotthoff et al. 17; Falkner et al. 18
- Evolutionary Computation
  Olson et al. 16; Choromanski et al. 18
- Bilevel optimization
  Franceschi et al. 17, 18
- Reinforcement learning
  Andrychowicz 16; Drori et al. 18
Monte Carlo Tree Search

Game playing when no good evaluation function and huge search space.

- Upper Confidence Tree (UCT)
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution
    - Path visited most often

Within learning
  - Feature selection
  - Active learning

Kocsis & Szepesvári 06, Gelly & Silver 07

Auer et al. 02

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search

**Game playing** when no good evaluation function and huge search space.

- **Upper Confidence Tree (UCT)**
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution
    - Path visited most often

**Within learning**
- Feature selection
- Active learning

---

Kocsis & Szepesvári 06, Gelly & Silver 07

---

Auer et al. 02

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search

Game playing when no good evaluation function and huge search space.

- Upper Confidence Tree (UCT)
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes

- Returned solution
  - Path visited most often

Within learning
  - Feature selection
  - Active learning

Kocsis & Szepesvári 06, Gelly & Silver 07

Auer et al. 02

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search

**Game playing** when no good evaluation function and huge search space.

- **Upper Confidence Tree (UCT)**
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution
    - Path visited most often

**Within learning**
- Feature selection
- Active learning

---

Kocsis & Szepesvári 06, Gelly & Silver 07

Auer et al. 02

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search

Game playing when no good evaluation function and huge search space.

- Upper Confidence Tree (UCT)
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution
    - Path visited most often

Within learning
- Feature selection
- Active learning

Kocsis & Szepesvári 06, Gelly & Silver 07

Auer et al. 02

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search

Game playing when no good evaluation function and huge search space.

- Upper Confidence Tree (UCT)
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution
    - Path visited most often

Within learning
  - Feature selection
  - Active learning

Kocsis & Szepesvári 06, Gelly & Silver 07

Auer et al. 02

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search

Kocsis & Szepesvári 06, Gelly & Silver 07

Game playing when no good evaluation function and huge search space.

- Upper Confidence Tree (UCT)
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution
    - Path visited most often

Within learning

Feature selection
Active learning

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search

Game playing when no good evaluation function and huge search space.

- Upper Confidence Tree (UCT)
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution
    - Path visited most often

Within learning
- Feature selection
- Active learning

Kocsis & Szepesvári 06, Gelly & Silver 07

Auer et al. 02

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search

Game playing when no good evaluation function and huge search space.

- Upper Confidence Tree (UCT)
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution
    - Path visited most often

Within learning
- Feature selection
- Active learning

Kocsis & Szepesvári 06, Gelly & Silver 07

Auer et al. 02

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search

Game playing when no good evaluation function and huge search space.

- Upper Confidence Tree (UCT)
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution
    - Path visited most often

Within learning
  Feature selection
  Active learning

Kocsis & Szepesvári 06, Gelly & Silver 07

Auer et al. 02

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search

Game playing when no good evaluation function and huge search space.

- Upper Confidence Tree (UCT)
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Select next action bis (random phase)
  - Compute instant reward
  - Update information in visited nodes
- Returned solution
  - Path visited most often

Within learning
- Feature selection
- Active learning

Kocsis & Szepesvári 06, Gelly & Silver 07

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search

Game playing when no good evaluation function and huge search space.

- Upper Confidence Tree (UCT)
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution
    - Path visited most often

Within learning
- Feature selection
- Active learning

Kocsis & Szepesvári 06, Gelly & Silver 07

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search

Kocsis & Szepesvári 06, Gelly & Silver 07

Game playing when no good evaluation function and huge search space.

- Upper Confidence Tree (UCT)
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
- Returned solution
  - Path visited most often

Within learning
  - Feature selection
  - Active learning

Auer et al. 02

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search

Game playing when no good evaluation function and huge search space.

- Upper Confidence Tree (UCT)
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution
    - Path visited most often

Within learning
  - Feature selection
  - Active learning

Kocsis & Szepesvári 06, Gelly & Silver 07

Auer et al. 02

Gaudel, Sebag, 10
Rolet, Teytaud, Sebag, 09
Monte Carlo Tree Search for AutoML

1. **Select** next action (alg/hyperparameter)

   \[
   \text{select } \arg \max_i \left\{ \mu_i + c \sqrt{\frac{\log N}{n_i}} \right\}
   \]

   with \( \mu_i \) average reward, \( n_i \) number visits, 
   \( N = \sum_i n_i \)

2. **Add a node**: new alg or hyper-parameter;

3. **Random phase**: complete pipeline with default/random choices.

4. **Compute reward** \( \nu \): predictive accuracy of pipeline

5. Use \( \nu \) to **update** \( \mu_i \), increment \( n_i \) in all visited nodes
Mosaic: MCTS for AutoML

Overview

▶ Search space: \{ Preprocessing algs \} \times \{ Algorithms \}
▶ Fixed sequence of choices:
  1. Preprocessing alg
  2. hyper-parameters of pre-processing alg
  3. Algorithm
  4. hyper-parameters of Alg.

Key features

▶ CPU management
  (every \( \Delta t \), kills unpromising pipelines; increases evaluation resources for others)
▶ Sample continuous hyperparameters: Progressive widening

Increase number of sampled values like \( \left\lfloor \sqrt{N} \right\rfloor \)
A MOSAIC Tree

root (11, 0.64)

preprocessing (11, 0.64)

SelectKBest (6, 0.66)

SelectKBest_k-20 (3, 0.74)

model (2, 0.89)

LogisticRegression (1, 0.88)

SelectKBest_k-12 (2, 0.45)

model (1, 0.10)

PCA_n_components-13 (3, 0.64)

PCA (4, 0.55)

model (2, 0.51)

LogisticRegression (1, 0.88)
Search space

<table>
<thead>
<tr>
<th>Preprocessing</th>
<th>PCA</th>
<th>n_components</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SelectKBest</td>
<td>k, score_func</td>
</tr>
<tr>
<td></td>
<td>Gaussian Random Projection</td>
<td>n_components, eps</td>
</tr>
<tr>
<td></td>
<td>No preprocessing</td>
<td>-</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Logistic regression</td>
<td>C, penalty, solver</td>
</tr>
<tr>
<td></td>
<td>SGD Classifier</td>
<td>learning rate, penalty, alpha, l1 ratio, loss</td>
</tr>
<tr>
<td></td>
<td>KNN classifier</td>
<td>K, metric, weights</td>
</tr>
<tr>
<td></td>
<td>XGBoost classifier</td>
<td>learning rate, max depth, gamma, subsample, regularization</td>
</tr>
<tr>
<td></td>
<td>LDA</td>
<td>n_components, learning decay</td>
</tr>
<tr>
<td></td>
<td>Random forest</td>
<td>criterion, max features, max depth, bootstrap, min sample split</td>
</tr>
</tbody>
</table>
Experiments and Results

AutoML Challenge PAKDD 2018

- Binary classification (Final phase)
- 10 or 20 minutes time budget for each dataset
- Metric: balanced accuracy

<table>
<thead>
<tr>
<th></th>
<th>Set 1</th>
<th>Set 2</th>
<th>Set 3</th>
<th>Set 4</th>
<th>Set 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>aad_freiburg</td>
<td>0.5533</td>
<td>0.2839</td>
<td>0.3932</td>
<td>0.2635</td>
<td>0.6766</td>
</tr>
<tr>
<td>mosaic</td>
<td>0.5382</td>
<td>0.3161</td>
<td>0.3376</td>
<td>0.3182</td>
<td>0.6317</td>
</tr>
<tr>
<td>narnars0</td>
<td>0.5418</td>
<td>0.2894</td>
<td>0.3665</td>
<td>0.2005</td>
<td>0.6922</td>
</tr>
<tr>
<td>W</td>
<td>wang</td>
<td></td>
<td><strong>0.5655</strong></td>
<td><strong>0.4851</strong></td>
<td>0.2829</td>
</tr>
</tbody>
</table>

Final test phase: mosaic ranked second w.r.t. average rank.
## Experiment 2: extended pre-processing search space

<table>
<thead>
<tr>
<th>Methods</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td><code>n_components</code>, <code>whiten</code>, <code>svd_solver</code>, <code>tol</code>, <code>iterated_power</code></td>
</tr>
<tr>
<td>KernelPCA</td>
<td><code>n_components</code>, <code>kernel</code>, <code>gamma</code>, <code>degree</code>, <code>coef0</code>, <code>alpha</code>, <code>eigen_solver</code>, <code>tol</code>, <code>max_iter</code></td>
</tr>
<tr>
<td>FastICA</td>
<td><code>n_components</code>, <code>algorithm</code>, <code>max_iter</code>, <code>tol</code>, <code>whiten</code>, <code>fun</code></td>
</tr>
<tr>
<td>Identity</td>
<td>-</td>
</tr>
<tr>
<td>IncrementalPCA</td>
<td><code>n_components</code>, <code>whiten</code>, <code>batch_size</code></td>
</tr>
<tr>
<td>SelectKBest</td>
<td><code>score_func</code>, <code>k</code></td>
</tr>
<tr>
<td>SelectPercentile</td>
<td><code>score_func</code>, <code>percentile</code></td>
</tr>
<tr>
<td>LinearSVC Pre-processing</td>
<td><code>C</code>, <code>class_weight</code>, <code>max_iter</code></td>
</tr>
<tr>
<td>ExtraTreesClassifier</td>
<td><code>n_estimators</code>, <code>criterion</code>, <code>max_depth</code>,</td>
</tr>
<tr>
<td>Pre-processing</td>
<td><code>min_samples_split</code>, <code>min_samples_leaf</code>,</td>
</tr>
<tr>
<td></td>
<td><code>min_weight_fraction_leaf</code>, <code>max_features</code>,</td>
</tr>
<tr>
<td></td>
<td><code>max_leaf_nodes</code>, <code>class_weight</code></td>
</tr>
<tr>
<td>FeatureAgglomeration</td>
<td><code>n_clusters</code>, <code>affinity</code>, <code>linkage</code></td>
</tr>
<tr>
<td>PolynomialFeatures</td>
<td>degree</td>
</tr>
<tr>
<td>RBFSampler</td>
<td><code>gamma</code>, <code>n_components</code></td>
</tr>
<tr>
<td>RandomTreesEmbedding</td>
<td><code>n_components</code>, <code>max_depth</code>, <code>min_samples_split</code>,</td>
</tr>
<tr>
<td></td>
<td><code>min_samples_leaf</code>, <code>min_weight_fraction_leaf</code>,</td>
</tr>
<tr>
<td></td>
<td><code>max_leaf_nodes</code>, <code>min_impurity_decrease</code></td>
</tr>
</tbody>
</table>
### Experiment 2: extended algorithm search space

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinearDiscriminantAnalysis</td>
<td>solver, shrinkage</td>
</tr>
<tr>
<td>QuadraticDiscriminantAnalysis</td>
<td>reg_param</td>
</tr>
<tr>
<td>DummyClassifier</td>
<td>-</td>
</tr>
<tr>
<td>AdaBoostClassifier</td>
<td>base_estimator, n_estimators, learning_rate, algorithm</td>
</tr>
<tr>
<td>ExtraTreesClassifier</td>
<td>n_estimators, criterion, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_features, max_leaf_nodes, class_weight</td>
</tr>
<tr>
<td>RandomForestClassifier</td>
<td>n_estimators, criterion, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_features, max_leaf_nodes, class_weight, bootstrap</td>
</tr>
<tr>
<td>GradientBoostingClassifier</td>
<td>loss, learning_rate, n_estimators, max_depth, criterion, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, subsample, max_features, max_leaf_nodes</td>
</tr>
<tr>
<td>SGD Classifier</td>
<td>learning_rate, penalty, alpha, l1 ratio, loss, epsilon, eta0, power_t, class_weight, max_iter</td>
</tr>
</tbody>
</table>
Experiment 2: extended algorithm search space, foll.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron</td>
<td>penalty, alpha, max_iter, tol, shuffle, eta0</td>
</tr>
<tr>
<td>RidgeClassifier</td>
<td>alpha, max_iter, class_weight, solver</td>
</tr>
<tr>
<td>PassiveAggressiveClassifier</td>
<td>C, max_iter, tol, loss, class_weight</td>
</tr>
<tr>
<td>KNeighborsClassifier</td>
<td>n_neighbors, weights, algorithm, leaf_size, p, metric</td>
</tr>
<tr>
<td>MLPClassifier</td>
<td>hidden_layer_sizes, activation, solver,</td>
</tr>
<tr>
<td></td>
<td>alpha, batch_size, learning_rate,</td>
</tr>
<tr>
<td></td>
<td>learning_rate_init, power_t, max_iter,</td>
</tr>
<tr>
<td></td>
<td>shuffle, warm_start, momentum, nesterovs_momentum, early_stopping, validation_fraction, beta_1, beta_2, epsilon</td>
</tr>
<tr>
<td>SVC</td>
<td>C, max_iter, tol, loss, class_weight, kernel, degree, gamma, coef0</td>
</tr>
<tr>
<td>DecisionTreeClassifier</td>
<td>criterion, splitter, max_depth, min_samples_split,</td>
</tr>
<tr>
<td></td>
<td>min_samples_leaf, min_weight_fraction_leaf,</td>
</tr>
<tr>
<td></td>
<td>max_features, max_leaf_nodes,</td>
</tr>
<tr>
<td></td>
<td>min_impurity_decrease, class_weight</td>
</tr>
<tr>
<td>ExtraTreeClassifier</td>
<td>criterion, splitter, max_depth, min_samples_split,</td>
</tr>
<tr>
<td></td>
<td>min_samples_leaf, min_weight_fraction_leaf,</td>
</tr>
<tr>
<td></td>
<td>max_features, max_leaf_nodes,</td>
</tr>
<tr>
<td></td>
<td>min_impurity_decrease, class_weight</td>
</tr>
</tbody>
</table>
Experiment 2

On 133 datasets from the OpenML repository
1 hour per (dataset, run); 10 runs per dataset.

Average rank (lower is better) of MOSAIC and Vanilla Auto-Sklearn across 102 datasets (Datasets on which the performance of both methods differs statistically according to Mann-Whitney rank test with $p = 0.05$).
Discussion

Monte Carlo Tree Search for Algorithm Configuration

- Proof of concept

Limitations

- Order of hyper-parameters
- Time allocation
Perspectives

Short term
  ▶ Refine initialization
  ▶ Extend to constrained satisfaction

Medium term
  ▶ Learn value of parameters across datasets
  ▶ Improving the sampling of continuous hyperparameter values

Long term
  ▶ Learning Meta-Features: a (the) key Meta-Learning task.