GreedyAgent: Crafting Efficient Agents for Meta-learning from Learning Curves via Greedy Algorithm Selection

Jinyu He[1], Xiaowei Song[2], Xiaohan Yan[2], Nan Wang[2], Yuqi Miao[2], Zijian Jiang[2], Fei Chao[1], Yan Zhang[1], Shengchuan Zhang[1], and Rongrong Ji[1]

[1] School of Informatics, Xiamen University, Xiamen 361005, China
[2] College of Electronic and Information Engineering, Tongji University, Shanghai 201804, China
rrji@xmu.edu.cn

Abstract. With the rapid development of artificial intelligence and machine learning technologies, Automated Machine Learning (AutoML) has become a hot research area in recent years. Meta-learning plays an increasingly important role in AutoML. A key sub-problem—meta-learning from learning curves is an immature but gradually gaining attention area within the field of meta-learning. For instance, in MetaLC, the organizers designed a competition on meta-learning from learning curves, whose goal is to achieve the highest all-time performance, based on the algorithm's selection strategy learned by past learning curves. This competition aims to simulate the meta-learning process in an environment like reinforcement learning, where the strategies provided by the participants act as agents interacting with the environment. We innovatively propose a training strategy for agents called GreedyAgent, based on an approximate greedy algorithm. After reducing the problem of meta-learning from learning curves to a 0-1 knapsack problem, we proved that our greedy method could replace the dynamic programming method to find an approximate optimal solution. In both the development phase and the final evaluation phase, our method achieved state-of-the-art in test results. We compared five baseline methods and selected representative cases for detailed analysis. The competition's second round has been accepted by the first AutoML Conference. Besides, our solution ranked 1st among the 14 methods participating in the competition. The code of our method has been released: https://github.com/dragonbra/MetaLC-2nd-Round.

Keywords: AutoML, Meta-learning, Learning curves, Greedy Algorithm

1 Introduction

As the field of artificial intelligence and machine learning rapidly advances, Automated Machine Learning (AutoML) [1] has emerged as a focal area of research in recent years. Particularly, meta-learning [2], playing an increasingly vital role within Au-

toML, is defined as the process of 'learning to learn,' enabling machine learning algorithms to optimize future learning tasks using past experiences. This capability, commonplace in

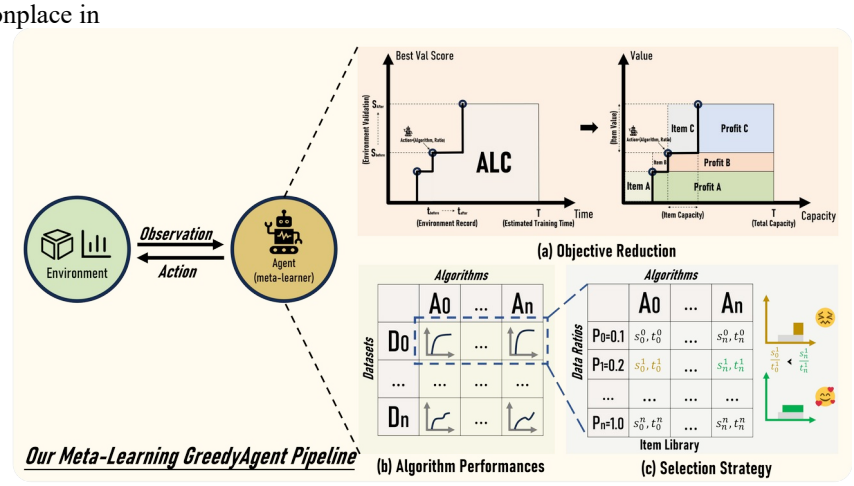

Fig. 1. The pipeline of the proposed GreedyAgent method in Meta-learning from learning-curves. Our method follows the greedy rule in selection strategies.

biological entities in nature who adapt quickly to changing environments by transferring knowledge across tasks, is still relatively nascent in artificial learning systems, which are primarily limited to transferring knowledge among very similar tasks.

In the current technological landscape, there is growing societal concern about the high computational resource consumption by artificial intelligence. Hence, developing learning systems that can effectively reuse their knowledge [3] is of paramount im-portance. Such systems could not only conserve resources but also potentially increase learning efficiency, thereby accelerating the learning process and improving performance on new tasks. To achieve this goal, researchers have extensively explored various domains, including few-shot learning, transfer learning, representational learning, continual learning, lifelong learning, and meta-learning.

Despite these efforts, research on meta-learning from learning curves [4] remains underdeveloped. Learning curves, critical tools for measuring incremental performance improvements of algorithms, can evaluate algorithms as functions of training time, iterations, or the number of samples. However, past challenges and research have not fully leveraged the potential of learning curves, often treating meta-learning merely as a recommendation issue, or focusing solely on specific aspects like few-shot learning.

Addressing this gap, organizers have introduced a new challenge [5], inspiring deeper exploration into meta-learning from learning curves within academic and industrial circles. The design of this challenge draws inspiration from previous studies, focusing on the simultaneous issues of algorithm selection and budget allocation. In this setting, the meta-learner needs to utilize partial information of training algorithms to reduce the cost of training them to convergence. Furthermore, the challenge fo-

cuses on the all-time performance of training curves on new datasets, evaluating agent training strategies using ALC(Area under Learning Curve) metrics.

In this paper, we innovatively introduce GreedyAgent, a greedy algorithm-based approach that learns sufficient greedy metric information from past algorithm training curves and can make strong, correct action recommendations based on this information. The pipeline of GreedyAgent method is as shown in Fig. 1.Our research presents a clear and complete mathematical justification, reducing the problem of meta-learning from learning curves to a 0-1 knapsack problem. Using a greedy algorithm that approximates the optimal solution, we achieve as optimal a performance as possible under constrained environmental settings, substituting for dynamic programming algorithms. Our method's advanced nature is validated through experimental and competitive evaluation, notably securing first place among 14 competing methods in the second round of the MetaLC competition, which was accepted at the 1st AutoML conference.

The main contributions of this paper are:

1. Introducing a simple yet effective meta-learning strategy, GreedyAgent, which won the MetaLC competition, outperforming others significantly (ALC scores in final phase: GreedyAgent gets 0.39, ranked in 1st place, which is 0.04 (~10%) higher than the 2nd place's method).

2. Deriving and demonstrating the problem of meta-learning from learning curves to a 0-1 knapsack problem, proving the correctness and effectiveness of our method.

3. Presenting a clear, white-box strategy with readily usable code, showcasing the efficiency and simplicity of our approach in meta-learning and transfer learning from learning curves without the need for high computational power.

2 Related Work

2.1 Automated Machine Learning

The advancement of Automated Machine Learning (AutoML) towards meta-learning marks a critical transition in artificial intelligence research. Meta-learning, though still in its infancy in the AutoML context, mainly involves knowledge transfer across similar tasks. With increasing emphasis on computational efficiency in AI systems, the significance of meta-learning is on the rise. Major contributions in this area include the study of few-shot learning [7], demonstrating meta-learning's flourishing in minimal data scenarios; [8]'s exploration of transfer learning, vital for knowledge sharing in meta-learning systems; [9]'s emphasis on the crucial role of representational learning in developing advanced learning mechanisms; [10]'s highlighting of continual learning's importance in adaptive meta-learning systems. Notably, [11] outlined the relatively less explored but fundamental area of meta-learning from learning curves, shedding light on understanding and optimizing the learning process itself. These diverse contributions point to a holistic evolution of meta-learning, aiming not just at task-specific performances but at a deeper understanding of the learning process in AI systems.

2.2 Meta-Learning from Learning Curves

Focusing on the use of incrementally gathered algorithm performance data over time, meta-learning from learning curves represents an emerging research field that encompasses a variety of approaches. Studies have shown that meta-learning has been applied in developing advanced recommendation systems, where algorithm selection is optimized based on historical performance data. The REVEAL [12] method introduces meta-learning as a novel challenge in reinforcement learning, thus broadening the scope and applications of meta-learning techniques. Additionally, various methods utilize learning curves for performance prediction, and despite relying on stringent policies, they contribute to more scalable meta-learning frameworks. Innovations in challenges, as demonstrated in works like [13, 14], play a key role in enhancing the efficiency of meta-learning using learning curves, showcasing the dynamic and continually evolving nature of this field.

MetaLC 1st Round [5] was accepted as part of the official selection of WCCI 2022 competitions, and MetaLC 2nd Round [6] was recognized as an integral component of the 1st AutoML Conference competitions. Our method won the championship in MetaLC 2nd Round.

3 Methodology

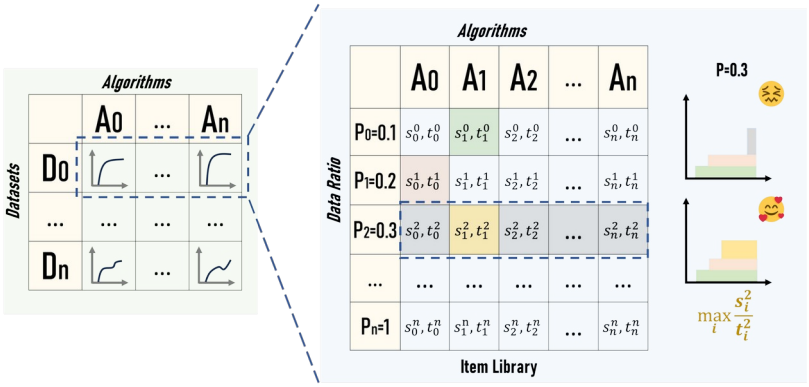

Fig. 2. Selection strategy. Our greedy algorithm tends to choose the algorithm with the highest performance-time ratio and growing partial p.

3.1 Problem Definition

Meta-learning from learning curves.
The problem we are investigating is as follows: Suppose we now have many different algorithms (as well as different hyperparameter settings for the same algorithms) and their learning curve results from training on numerous datasets in the past. We wish to perform meta-learning on this information to select the most suitable algorithm to apply on new datasets.

At the same time, since we typically hope to achieve better performance through transfer learning within a relatively short period, and it is often applied in scenarios with limited budgets, we are more concerned with the ALL-TIME Performance of the model. Regardless of the actual training budget in real-world scenarios, we hope our model can maintain commendable performance. To this end, we introduce the ALC (Area under Learning Curve) as the evaluation criterion.

Formal definition of MetaLC.

In the competition, the overall meta-learning process is defined as an environment akin to reinforcement learning.

Within this environment, our algorithms take on the role of an agent. Based on the algorithm learning curves from existing datasets and the information observable by the agent in the current environment (observation), a suggestion is made for the next training algorithm, corresponding to the training strategy that will be implemented, known as an action.

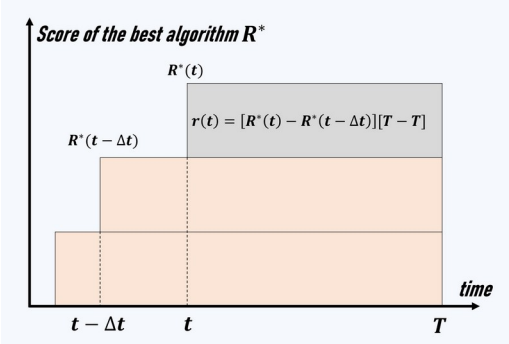

Fig. 3. The schematic diagram of ALC and the reward function r(t).

Upon receiving the proposed action from the agent, the environment trains using the selected algorithm and dataset proportion, updating the current scores, time spent, and other information, which is then incorporated into the observation and returned to the agent to facilitate the decision-making for the next training step. The meta-learning process iterates through this cycle repeatedly. The definitions of action and observation are as follows:

$$action = agent.suggest(observation)$$
$$observation = environment.reveal(action)$$

$action$: a tuple of (A, p), with:

A: index of the chosen algorithm,

p: decimal fraction of training data used, with the value of p in $[0.1, 0.2, 0.3, ..., 1.0]$.

$observation$: a tuple of $A, p, t, R\_train\_A\_p, R\_validation\_A\_p$¿, with:

A: index of the algorithm provided in the previous action,

t: the amount of time it took to train A with training data size of p, and make predictions on the training/validation/test datasets,
$R\_train\_A\_p$: performance score on the training dataset,
$R\_validation\_A\_p$: performance score on the validation dataset.

Due to the focus of this issue on the ALL-TIME performance of the final training strategy, as shown in Fig. , the ultimate evaluation metric is the ALC (Area under the Learning Curve).

$$ALC(Area\ \ under\ \ Learning\ \ Curve)=\int_{t=0}^{\tau} R_{test_{¿}}(t) \tag{1}$$

This implies that it is not just the final model accuracy score that is desirable, but rather maintaining a high accuracy score throughout the process that contributes to the accumulation of the ALC area (which can demonstrate good accuracy performance at any given moment). This approach will result in a higher ALC score.

3.2 Problem Reduction

Reduction to 0-1 knapsack problem.
From the perspective of reinforcement learning, the reward function that our agent can obtain each time in the environment can be defined as follows:

$$r(t)=\left(R^{¿}(t)-R^{¿}(t-\Delta t)\right)\times(\tau-t) \tag{1}$$

of which,

$$\begin{cases} R^{¿}(t) & ¿ \quad \max\limits_{k\le t} R_{test\_A\_p}(k), \\ R^{¿}(0) & ¿ \quad 0, \end{cases} \tag{1}$$

To model MetaLC using the 0-1 knapsack problem, we see the combination of (A,p) as different items:

$$item_i=(w_i,v_i) \tag{1}$$

$$w_i(weight)=\Delta t \tag{1}$$

$$v_i(value)=r(t) \tag{1}$$

Also, we make the following hypotheses based on the physical meaning of the problem:
Hypotheses.
1. For $action(A,p)$,
   1. When A are same, different p means $[\Delta t \propto p, R(t) \propto p]$,
   2. When p are same, different A means [Δt are approximately same].

2. v_i is related to the update amount of each $R(t)$, and $R^{\prime}(t)=\max_{k\le t} R(k)$, thus it is necessary to ensure that each $R(t)$ is increasing when p is increasing. That is, we can estimate the change in score each time with $R(t)=R^{\prime}(t)$,

3. τ is unknown, and in practice, it is a relatively limited time budget.

Thus, the problem is reduced to the 0-1 knapsack problem. However, there are several issues:

The τ in this context is unknown, meaning the total time for calculating the ALC metric is unknown. Since τ is unknown and, in practice, is a relatively limited time budget, it does not allow for the "item" with the highest $r(t)$ (the algorithm with the strongest performance, usually with the largest w_i) to obtain the highest v_i all at once. Therefore, ALC can more objectively reflect the all-time performance.

The correct solution to the 0-1 knapsack problem is dynamic programming; however, in the current agent-environment interaction steps, the specific v_i is not known before each observation is returned, making it impossible to attempt all items through dynamic programming. This point will be discussed in detail in 3.3.

3.3 Algorithm

Dynamic Programming.

Given the known τ and t on a new dataset, we can directly rank, and estimate based on past algorithm performances to obtain a definitive optimal solution for the agent to provide action recommendations, as well as the maximum value of the ALC metric here. Selecting algorithms based on known past performance is, in fact, the most intuitive approach to choosing which algorithm model to apply in real life. The dynamic programming solution is as follows:

$$dp(t)=max(dp(t),dp(t-\Delta t)+r(t)) \tag{1}$$

$$ans=dp(\tau) \tag{1}$$

However, in this scenario, τ and t are completely unknown. Although the $R(t)$ of an algorithm can be considered the best estimate, without knowing τ, and with each t only known after an action recommendation is given, we cannot determine the specific values of the interval distance Δt and $r(t)$ during each state transition update, which poses a significant challenge for implementing correct state transitions.

One approach would be to attempt each $action(A,p)$on the target dataset once to obtain the actual effects, allowing us to plan an action recommendation strategy that maximizes all-time performance. However, this clearly does not align with the real situation that the agent-environment interaction aims to simulate, thus, we cannot use dynamic programming to obtain the optimal solution.

Approximate Greedy Algorithm.

In the 0-1 knapsack problem, there are multiple intuitive greedy methods that can obtain approximate optimal solutions most of the time [15],

Greedy based on w_i.
Items are sorted by w_i and taken from smallest to largest. This conflicts with *Hypothesis 2* because, according to *Hypothesis 1* [$\Delta t \propto p$, $R(t) \propto p$], if we always choose the smallest w_i (which usually means that for the given $action(A, p)$, $p=0.1$, the $R(t)$ we obtain may not increase over time, and thus the $r(t)$, which is v_i, will be very small or even equal to 0.

Greedy based on v_i.
Items are sorted by v_i and taken from largest to smallest. This conflicts with *Hypothesis 3* because choosing the largest v_i may result in not obtaining a result within the τ time frame.

Greedy based on "cost-performance ratio" v_i / w_i.
Items are taken from largest to smallest according to the cost-performance ratio. The v_i used for sorting here has a different meaning from the actual v_i because, in the past training curve information, we also do not have a good grasp of the actual size of τ. Therefore, we approximate by using the training score $R(t_i)$ in place of v_i for calculations.

In addition to satisfying the condition of taking items from largest to smallest based on cost-performance ratio, additional design can be incorporated to control the increase of its $R(t)$ and the actual v_i as much as possible. This leads to better all-time performance and a larger ALC.

GreedyAgent. The pseudo-code of our method is shown in Algorithm 1 and the whole pipeline is shown in Fig. 1

Algorithm 1: GreedyAgent.suggest(Observation)

Input: Observation($\mathbf{A}, \mathbf{p}, \mathbf{t}, \mathbf{R}_{\text{train}_{\mathbf{A}_p}}, \mathbf{R}_{\text{valid}_{\mathbf{A}_p}}$),
Meta-train Datasets $\mathbf{D}$, Meta-train Algorithms $\mathbf{M}$
Output: Action($\mathbf{A}, \mathbf{p}$)

$\mathbf{p} \leftarrow \mathbf{p} + 0.1$; // we assume v_i growing by adding p_i
$max_{ratio}, max_{index} \leftarrow 0, 0$; // initialize maximum ratio, index
$N \leftarrow \mathbf{M}.size()$; // get algorithms number
for $i \leftarrow 1$ **to** N **do**
  if $time_{p_i} < \mathbf{p}$ **then**
    continue
  // choose algorithm with max v_i per w_i;
  if $score_{p_i}/time_{p_i} > max_{ratio}$ **then**
    $max_{ratio} = score_{p_i}/time_{p_i}$;
    $max_{index} = i$;
  $\mathbf{A} \leftarrow max_{index}$;
return $(\mathbf{A}, \mathbf{p})$

4 Experiments

4.1 Experiment Settings

Competition organizers developed a meta-dataset with pre-trained learning curves for 40 hyperparameter settings across 30 datasets, enriched with meta-features. The dataset included three sets of learning curves per task from training, validation, and test phases, with evaluation metrics specified in meta-features. In development, 15 datasets were used; the full 30 were assessed in the final phase. Meta-learning strategies were ranked by average ALC scores.

During meta-training, agents received datasets' meta-features, algorithms' hyperparameters, and learning curves for training, validation, and testing. Organizers provided 12,000 training curves for 40 algorithms based on four foundational algorithms with varying hyperparameters.

In meta-testing, agents engaged in reinforcement learning, selecting algorithms and training data amounts for new tasks, using training and validation curves to inform decisions. Phases ended upon time budget exhaustion, challenging agents to make strategic, optimized choices with limited information.

4.2 Baselines Comparison and Case Study

Baselines.

***Double Deep Q Network (DDQN)*:** In the meta-testing phase of the competition, a Double Deep Q-Network (DDQN) [16] was used as a baseline due to its relevance in Reinforcement Learning (RL).

Best on Samples (BOS): BOS strategy is an approach where algorithms are initially tested with a set of samples, and the one performing best is chosen to run on the entire dataset.

Freeze-Thaw Bayesian Optimization (FT): FT [14] is employed in the meta-learning environment, particularly when an available hyperparameter space exists, to efficiently and rapidly determine the optimal hyperparameters using the standard Bayesian optimization method.

Average Rank (AR): This approach is based on selecting algorithms for new datasets based on their rankings in past algorithms, which inspired us to construct an average ranking baseline.

Random Search (RS): RS [17] is a simple baseline method that performs a random search over the algorithm space. It randomly selects an algorithm and allocates arbitrary amounts of time for training and testing.

Case Study.

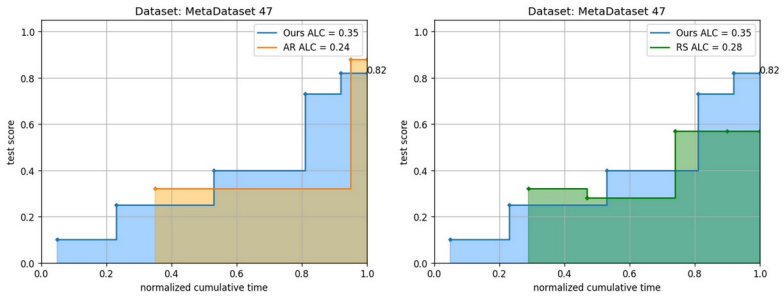

Fig. 4. Comparison of GreedyAgent and Average Rank(AR) algorithm (left), GreedyAgent and Random Search(RS) algorithm (right).

As show in Fig. above, it displays a case on the meta-dataset 47 from the sample test dataset, comparing our GreedyAgent method with the baseline methods AR (Average Rank) and RS (Random Search). This sample is highly representative, allowing us to conduct a case study on it.

On the left side of the figure, it can be observed that compared to the AR agent strategy executed on this dataset, our method can achieve an initial score sooner because the GreedyAgent tends to give suggestions with a small p at the beginning. This gives us a significant advantage when calculating the global all-time performance.

On the right side of the figure, compared to the RS agent strategy executed on this dataset, our GreedyAgent exhibits stronger stability. A noticeable score drop by the RS agent can be seen near the normalized cumulative time of 0.5. This occurs because the evaluation dataset is divided into training, validation, and test sets. During the interaction with the agent, the environment sorts based on the scores from the validation dataset resulting from the action (A, p) proposed by the agent. If a higher score than the existing validation set score is achieved, it will be updated as the new score.

However, the score on the test dataset before the update is unknown, which may introduce a decrease in the test dataset score.

4.3 Results and Analysis

Table 1. Comparison of ALC score with different baselines in development phase. The ALC score is evaluated on 15 real-world datasets. The **bold** and the underline represent the best and second-best results (all the same results are marked in the same style). DDQN, BOS, FT, AR and RS stands for different methods used as baselines. Our GreedyAgent performs the best on most of the datasets. And our method achieves the best performance on average.

Method	DDQN	BOS	FT	AR	RS	GreedyAgent (Ours)
0-adult	0.74	0.75	0.75	0.72	0.16	**0.76**
1-cadata	0.64	0.62	0.64	0.72	0.52	**0.73**
2-digits	0.64	0.73	0.73	0.74	0.25	**0.80**
3-dorothea	0.62	0.65	0.63	0.60	**0.67**	0.66
4-newsgroups	0.06	0.11	0.11	**0.15**	0.13	**0.15**
5-christine	0.39	0.37	0.36	0.32	0.30	**0.45**
6-jasmine	0.56	0.56	0.56	0.54	0.56	**0.57**
7-madeline	0.50	0.43	0.37	0.40	0.40	**0.53**
8-philippine	**0.39**	0.36	0.36	0.25	0.33	0.38
9-sylvine	**0.86**	0.85	0.84	0.79	0.84	0.85
10-albert	**0.26**	0.21	0.21	0.17	0.21	0.20
11-dilbert	**0.66**	0.47	0.47	0.37	0.28	0.64
12-fabert	**0.25**	0.24	0.20	0.17	0.22	**0.25**
13-robert	**0.22**	0.16	0.16	0.12	0.00	**0.22**
14-volkert	**0.13**	0.11	0.11	0.00	0.12	**0.13**
Average	0.46	0.44	0.43	0.40	0.33	**0.49**

During the development phase, we obtained detailed ALC experimental results for 40 algorithms with different hyperparameter configurations on 15 real datasets, shown in Table 1. It can be observed that in most cases, that is, on 10 out of 15 datasets, our method, GreedyAgent, achieved first place. On the remaining 4 out of 5 datasets, we also secured second place with a slight margin from the first. This demonstrates the excellence of our method's performance and its versatility across various datasets.

On the 10-albert dataset, our method showed a significant gap compared to the baseline provided by the organizers. This may be because the algorithm selected for its highest cost-performance ratio incurs more time consumption during the actual training process or fails to meet the score expectations derived from the training information of past datasets. This indicates that in actual transfer learning, the expected performance of an algorithm may vary significantly from its real performance.

Table 1. Comparison **of ALC score with different baselines in final phase**. The final score is obtained by averaging results over all the 30 datasets. Our method outperforms all the other methods. AT01, AT02 stands for other participants' agent strategy.

Method	DDQN	AT01	AT02	BOS	FT	GreedyAgent (Ours)
Final Score	0.35	0.32	0.31	0.30	0.30	**0.39**

In the final evaluation phase, we can obtain the overall ALC results for 40 algorithms with different hyperparameter configurations on 30 real datasets, shown in Table 1. Our GreedyAgent significantly outperforms other methods under comparison, which fully demonstrates the superiority of our approach.

5 Conclusion

This paper explores meta-learning using learning curves, a significant topic for integrating AutoML methods and guiding algorithm selection and training costs based on past data. To tackle high AI resource consumption, we introduce GreedyAgent, a new approach with a mathematical proof. It reformulates the problem as a 0-1 knapsack issue, highlighting the limitations of dynamic programming and suggesting an approximate greedy algorithm for efficiency. Our method outperforms in all phases, validating its effectiveness and reliability.

Acknowledgment. This work was supported by National Science and Technology Major Project (No. 2022ZD0118202), the National Science Fund for Distinguished Young Scholars (No.62025603), the National Natural Science Foundation of China (No. U21B2037, No. U22B2051, No. 62176222, No. 62176223, No. 62176226, No. 62072386, No. 62072387, No. 62072389, No. 62002305 and No. 62272401), and the Natural Science Foundation of Fujian Province of China (No.2021J01002, No.2022J06001).

References

1. He X, Zhao K, Chu X. AutoML: A survey of the state-of-the-art[J]. Knowledge-based systems, 2021, 212: 106622.
2. Vanschoren J. Meta-learning: A survey[J]. arXiv preprint arXiv:1810.03548, 2018.
3. Markus L M. Toward a theory of knowledge reuse: Types of knowledge reuse situations and factors in reuse success[J]. Journal of management information systems, 2001, 18(1): 57-93.
4. Mohr F, van Rijn J N. Learning Curves for Decision Making in Supervised Machine Learning--A Survey[J]. arXiv preprint arXiv:2201.12150, 2022.
5. Nguyen M H, Sun-Hosoya L, Grinsztajn N, et al. Meta-learning from Learning Curves: Challenge Design and Baseline Results[C]//2022 International Joint Conference on Neural Networks (IJCNN). IEEE, 2022: 1-8.

6. Nguyen M H, Sun L, Grinsztajn N, et al. Meta-learning from Learning Curves Challenge: Lessons learned from the First Round and Design of the Second Round[J]. arXiv preprint arXiv:2208.02821, 2022.
7. Wang Y, Yao Q, Kwok J T, et al. Generalizing from a few examples: A survey on few-shot learning[J]. ACM computing surveys (csur), 2020, 53(3): 1-34.
8. Zhuang F, Qi Z, Duan K, et al. A comprehensive survey on transfer learning[J]. Proceedings of the IEEE, 2020, 109(1): 43-76.
9. Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives[J]. IEEE transactions on pattern analysis and machine intelligence, 2013, 35(8): 1798-1828.
10. De Lange M, Aljundi R, Masana M, et al. A continual learning survey: Defying forgetting in classification tasks[J]. IEEE transactions on pattern analysis and machine intelligence, 2021, 44(7): 3366-3385.
11. Mohr F, van Rijn J N. Fast and informative model selection using learning curve cross-validation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023.
12. L. Sun-Hosoya, “Meta-Learning as a Markov Decision Process,” Theses, Universit´e Paris Saclay (COmUE), Dec. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/tel-02422144
13. Sun-Hosoya L, Guyon I, Sebag M. Activmetal: Algorithm recommendation with active meta learning[C]//IAL 2018 workshop, ECML PKDD. 2018.
14. Swersky K, Snoek J, Adams R P. Freeze-thaw Bayesian optimization[J]. arXiv preprint arXiv:1406.3896, 2014.
15. Sahni S. Approximate algorithms for the 0/1 knapsack problem[J]. Journal of the ACM (JACM), 1975, 22(1): 115-124.
16. Lv P, Wang X, Cheng Y, et al. Stochastic double deep Q-network[J]. IEEE Access, 2019, 7: 79446-79454.
17. Zabinsky Z B. Random search algorithms[J]. Department of Industrial and Systems Engineering, University of Washington, USA, 2009.