Asymptotic Analysis of Meta-learning as a Recommendation Problem

Zhengying Liu, Isabelle Guyon

Asymptotic Analysis of Meta-learning as a Recommendation Problem

Zhengying Liu[1] **and Isabelle Guyon**[1, 2]
[1]UFR Sciences & INRIA, Université Paris-Saclay, France, [2]ChaLearn, USA
{zhengying.liu, isabelle.guyon}@inria.fr

Abstract

Meta-learning tackles various means of learning from past tasks to perform new tasks better. In this paper, we focus on one particular statement of meta-learning: learning to recommend algorithms. We focus on a finite number of algorithms, which can be executed on tasks drawn *i.i.d.* according to a "meta-distribution". We are interested in generalization performance of meta-predict strategies, *i.e.,* the expected algorithm performances on new tasks drawn from the same meta-distribution. Assuming the perfect knowledge of the meta-distribution (*i.e.,* in the limit of a very large number of training tasks), we ask ourselves under which conditions algorithm recommendation can benefit from meta-learning, and thus, in some sense, "defeat" the No-Free-Lunch theorem. We analyze four meta-predict strategies: *Random*, *Mean*, *Greedy* and *Optimal*. We identify optimality conditions for such strategies. We also define a notion of meta-learning complexity as the cardinal of the minimal clique of complementary algorithms. We illustrate our findings on experiments conducted on artificial and real data.

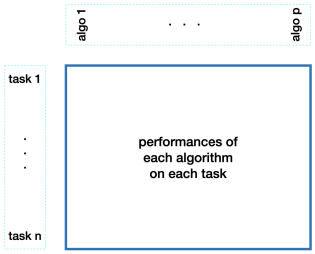
Introduction

Meta-learning [1, 17] is an active research field that aims at leveraging past learning experiences to improve or accelerate the learning process for future tasks. Inspired by [14], we study a class of meta-learning problems called *zero-level meta-learning* problems, as a special case of the *algorithm selection* problem introduced in [12]. We contrast it with other settings by introducing a novel meta-learning taxonomy. We analyze the problem of "meta-learnability" for *zero-level meta-learning* problems from a theoretical standpoint, highlighting cases that are not hampered by the No Free Lunch (NFL) theorems [20, 21, 19]. Roughly speaking, the NFL theorems state that all learning algorithms have the same average performance over all possible learning tasks, where the average is taken with a uniform distribution of the tasks. This sheds doubt on the efficacy of algorithm selection. However, as stated in [5], meta-learning offers a viable path to extract commonalities between tasks encountered in real life, which are not usually uniformly distributed. In this work, we consider assume that tasks are drawn *i.i.d.* form a given *meta-distribution*, but not necessarily uniform. We illustrate this class of problems in practical cases.

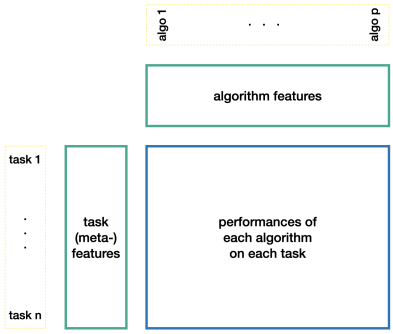
A taxonomy of meta-learning

In a meta-learning problem, each training example is a dataset or task (a task usually combines some data and performance metric for supervised learning, but may consist of a simulator and some rewards for reinforcement learning). A meta-dataset is therefore a set of tasks (or simply datasets), characterized by information on how well they can be resolved by a variety of algorithms. Based on which level of information is accessible, we categorize meta-learning algorithms into 3 families (Figure 1): **Zero-level meta-learning** uses only past *performances* of algorithms on tasks, as shown in Figure 1a. We call the performance matrix, with tasks or **D**atasets in lines and **A**lgorithms in columns, a **DA matrix**. It may contain missing values. This scenario is that of algorithm recommendation [12] in the "collaborative filtering" setting (*e.g.,* [10, 15, 14]). **First-level meta-learning** uses past performances plus meta-features of tasks and/or algorithms, as shown in Figure 1b (*e.g.,* [2, 11]). **Second-level meta-learning** uses full information of past tasks and algorithms, including all examples in all datasets and all code of all algorithms, as shown in Figure 1c (*e.g.,* [3], [4, 8], [13]).

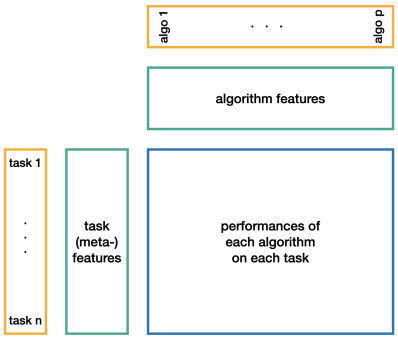
In this paper, we focus on *zero-level meta-learning*, *i.e.,* the collaborative filtering algorithm recommendation setting: the sole information available for meta-learning is a DA matrix of dimension (m tasks/datasets $\times$ n algorithms). At test time, the goal is to predict the performance coefficients of a new DA matrix, having the same algorithms, but different tasks/datasets. Both training and test matrices have lines drawn *i.i.d.* according to the same *meta-distribution* (also sometimes referred to as *mother distribution*, following the nomenclature introduced in [9]). In experiments, we use a finite meta-training sample (finite number of lines m). As in regular machine learning problems, one can estimate the meta-distribution of datasets/tasks with a certain precision, depending on the size of the available sample. However, our theoretical analyses assume the complete knowledge of the meta-distribution. In our setting the meta-distribution is the joint distribution of columns of the DA matrix, considered as random variables, *i.e.,* assuming that the DA matrix can be extended infinitely. We are thus in an asymptotic case and we will show that, under some assumption on the meta-distribution, certain meta-predict strategies perform better than others, such as random search.



(a) **Zero-level meta-learning** algorithms use only previously known performances of algorithms on tasks.



(b) **First-level meta-learning** algorithms use previously known performances plus meta-features of tasks and/or algorithms.



(c) **Second-level meta-learning** algorithms use full information of past tasks and algorithms (including all examples in all datasets and all code of all algorithms).

Figure 1: **Categorization of meta-learning in terms of the information used in the meta-dataset.**

Notations and Problem Setting

The DA Matrix

We consider the setting in which a DA matrix is available for meta-learning, that is a matrix of performances of algorithms on datasets/tasks. To simplify the analysis, we assume that scores are **binary**: 0 means algorithm failure and 1 algorithm success. We further assume that tasks/datasets (lines of the DA matrix) are drawn *i.i.d.* according to an unknown but fixed meta-distribution. Performances of algorithms on tasks/datasets can be thought of as random variables X_j, $j = 1 \dots n$, where n is the total number of algorithms considered. We call $\mathbf{X}$ the random vector $[X_1, X_2, \dots, X_n]$.

The goal of meta-learning is to learn from m samples of $\mathbf{X}$ constituting a training DA matrix, to devise a **meta-predict strategy**. After learning, this strategy is applied to find a successful algorithm as fast as possible, given a new task/dataset not seen before, *i.e.,* . by querying the performance of as few algorithms as possible.

Here we assume that we have an infinite number of training examples, such that the joint distribution $P(X_1, X_2, \dots, X_n)$ is known perfectly. Hence the **meta-training** procedures considered search for an optimal order of algorithms, knowing the meta-distribution perfectly. We ask ourselves the following questions: (i) Does the perfect knowledge of $P(X_1, X_2, \dots, X_n)$ allow us to outperform random search. (ii) Under what conditions (if any).

Criterion of evaluation

To evaluate the performance of any given meta-predict strategy, we consider as metric the area under the (meta-)learning curve. A learning curve $lc(t_i)$, $i = 1 \dots n$, is defined as the performance of the best algorithm queried so far, as a function of the number of algorithms queried. That is:

$$lc(i) = \max_{t_j, j=1\dots i} \{X_{t_1}, X_{t_2}, \dots, X_{t_i}\} , \qquad (1)$$

where we denote by X_{t_i} the score of the i^{th} algorithm queried. Next, we adopt a probabilistic notion of learning curve, considering a meta-test example (new task or dataset) as a random variable, *i.e.,* the expectation of $lc(t_i)$ over possible meta-test examples (a form of meta-generalisation):

$$LC(i) = \mathbb{E}\left[\max_{t_j, j=1\dots i} \{X_{t_1}, X_{t_2}, \dots, X_{t_i}\}\right] . \qquad (2)$$

Given that scores are either 0 or 1, the expected value of the maximum score seen so far is the probability that at least one algorithm was successful (score 1). This is also one minus the probability that all algorithms seen so far failed:

$$LC(i) = 1 - P(X_{t_1} = 0, X_{t_1} = 0, \dots, X_{t_i} = 0) . \qquad (3)$$

To evaluate the performance of meta-predict strategies having a stochastic component, we define

$$LC(i) = \mathbb{E}\left[1 - P(X_{t_1} = 0, X_{t_1} = 0, \dots, X_{t_i} = 0)\right] \qquad (4)$$

where the expectation runs over possible algorithm orderings. For strategies with a fixed pre-determined order of algorithms to be queried, taking this second expectation is not necessary. It is necessary for the Random strategy and when ties are broken at random.

Finally, we define the area under the learning curve, which we wish to maximize it over meta-predict strategies:

$$ALC(n) = \sum_{i=1}^{n} LC(i) . \qquad (5)$$

Meta-predict strategies

We consider four meta-predict strategies:

- **Random:** Query algorithms in uniformly random order.
- **Mean:** Query algorithms in order of their mean score value $\mathbb{E}[X_j] = P(X_j = 1)$.
- **Greedy:** Query first the algorithm with largest $P(X_{t_1} = 1)$. Then, query iteratively the next algorithm having the largest $P(X_{t_i} = 1 | X_{t_1} = 0, X_{t_2} = 0, \dots, X_{tn} = 0)$, until we find one sucessful algorithm; then the order of the remaining algorithms does not matter.
- **Optimal:** Query algorithms in the optimal order, maximizing $ALC(n)$.

For the Mean, Greedy, and Optimal strategies, we assume that ties are broken at random.

Meta-distributions

We consider four types of meta-distributions:

- **NFL:** No Free Lunch distribution: $P(X_1, X_2, \ldots, X_n) = P(X_1)P(X_2)\ldots P(X_n)$ and $P(X_j) = 0.5, \forall j = 1 : n$.
- **Indep:** $P(X_1, X_2, \ldots, X_n) = P(X_1)P(X_2)\ldots P(X_n)$ but for some j $P(X_j) \neq 0.5$. To make is more comparable to NFL, we assume that $(1/n)\sum_{j=1}^{n} P(X_j) = 0.5$.
- **Dep:** $P(X_1, X_2, \ldots, X_n) \neq P(X_1)P(X_2)\ldots P(X_n)$. We keep assuming $(1/n)\sum_{j=1}^{n} P(X_j) = 0.5$.
- **DepU: Dep** with uniform marginals $P(X_j) = 0.5, \forall j$.

Theoretical Results

In what follows, we prove the following propositions:

1. For the **NFL** distribution, the **Random** meta-predict strategy is as good as anything else.
2. For **Indep**, the **Mean** strategy is optimum.
3. For the **Dep** distribution, we might expect that the performance order should be **Random** $\leq$ **Mean** $\leq$ **Greedy** $\leq$ **Optimal**, however a variety of cases can arise:
   (a) **"Worst case"**: All strategies perform at chance level.
   (b) **"Best case"**: Greedy is optimal.
   (c) **Greedy** makes best "local" decisions, *i.e.,* increasing most the learning curve at any given point.
   (d) **Greedy** can be worse than **Optimal**
   (e) **Greedy** can be worse than **Random**
   (f) **Greedy** can be worse than **Mean**
   (g) **Mean** can be worse than **Random**
4. For the **DepU** distribution, the **Mean** meta-predict strategy performs no better than **Random**. We define a notion of meta-learning complexity C as the cardinal of the minimal clique of complementary columns (*i.e.,* columns having at least one successful algorithm in each line). There exists a meta-predict strategy such that $LC(C) = 1$, *i.e.,* the learning curve asymptote is reached in C steps.

Proofs

We simplify notations using: $P(X_{t_i} = 1) = p_i = 1 - q_i$. For **NFL** and **Indep** meta-distributions, learning curves are:

$$\begin{aligned} LC(i) &= 1 - P(X_{t_1} = 0)P(X_{t_2} = 0)\ldots P(X_{t_i} = 0) \\ &= 1 - q_1 q_2 \ldots q_i \end{aligned} \tag{6}$$

1. For the NFL distribution, the Random meta-predict strategy is as good as anything else.

For the **NFL** meta-distributions, $p_i = 1 - q_i = 0.5$. Thus,

$$LC(i) = 1 - (0.5)^i \,, \tag{7}$$

for all strategies. Thus **Random** is as good as anything else.

2. For Indep, Mean is optimum.

For the **Indep** distribution, the **Mean** strategy provides a ranking such that $q_1 \leq q_2 \leq \cdots \leq q_i$. Therefore the product $q_1 q_2 \ldots q_i$ for the **Mean** strategy will be smaller (or equal) to that obtained for any other order of algorithms.

$$\begin{aligned} LC(i) &= 1 - q_1 q_2 \ldots q_i \\ &\leq 1 - (0.5)^i \end{aligned} \tag{8}$$

3.a. Worse case scenario: All strategies perform at chance level

This case arises simply when all algorithms are identical: $X_1 = X_2 = \cdots = X_n$. In that case, by our previous assumption that on average algorithms perform at chance level, we have $P(X_j = 1) = 0.5$. For individual tasks, the (unique) algorithm we have will either be successful or fail, hence $lc(i) = 0$ of all i or $lc(i) = 1$ of all i. Thus $LC(i) = 0.5$ for all i, regardless of the strategy chosen.

3.b. Best case scenario: Greedy is optimal

This case arises, for example, when we have only 2 types of algorithms: an algorithm with scores X_1, and another one with exactly complementary scores: $X_2 = 1 - X_1$. All n algorithms are either of the first or the second type. We recall that by hypothesis, on average, algorithms perform at chance level. When we query a first algorithm, regardless of the strategy, predictions are therefore at chance level $LC(1) = 0.5$. But, as soon as we query a second algorithm with the greedy strategy, we get $LC(i > 1) = 1$ because $P(X_2 = 1|X_1 = 0) = 1$ and $P(X_1 = 1|X_2 = 0) = 1$, hence the **Greedy** strategy will pick up one of the versions of the complementary algorithm. The **Optimal** algorithm cannot beat it because $LC(1) = 0.5$ regardless of strategy.

3.c. Greedy makes the best local decision

The best local decision is the decision that increases most the learning curve at any given point. By definition,

$$\begin{aligned} LC(i) &= 1 - P(X_{t_1} = 0, X_{t_2} = 0, \ldots, X_{t_i} = 0) = 1 - \mathcal{P} \\ &= 1 - P(X_{t_i} = 0|X_{t_1} = 0, \ldots, X_{t_{i-1}} = 0) \\ &\qquad \cdot P(X_{t_1} = 0, \ldots, X_{t_{i-1}} = 0)) \,. \end{aligned} \tag{9}$$

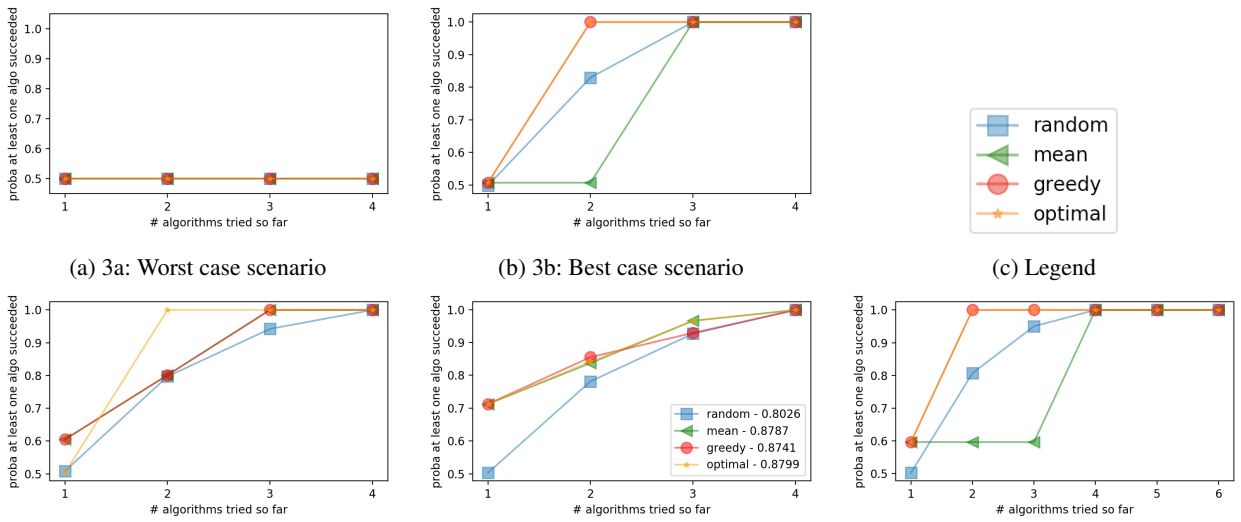
Hence the variation of learning curve is:

$$\begin{aligned} \Delta LC(i) &= LC(i+1) - LC(i) \\ &= \mathcal{P} \cdot (1 - P(X_{ti+1} = 0|X_{t_1} = 0, \ldots, X_{t_i} = 0) \\ &= \mathcal{P} \cdot P(X_{t_i} = 1|X_{t_1} = 0, \ldots, X_{t_i} = 0) \end{aligned} \tag{10}$$

By definition of the greedy strategy, unless the learning curve has already reached its maximum value, Greedy chooses at step i the algorithm with the largest $P(X_{t_i} = 1|X_{t_1} = 0, X_{t_2} = 0, \ldots, X_{t_i} = 0)$. Hence greedy makes the choice that increases most the learning curve, given past decisions, and ignoring what will happen in the future.

3.d. Greedy can be worse than Optimal

In this example we exploit the fact that the first step of the **Greedy** strategy is to choose the algorithm with best mean



(a) 3a: Worst case scenario (b) 3b: Best case scenario (c) Legend

(d) 3d: Greedy worse than Optimal (e) 3f: Greedy worse than Mean. (f) 3g: Mean worse than Random

Figure 2: **Learning curves on constructed examples.** (a) **"Worst-case" scenario:** All algorithms are identical. Random search is as good as anything else. (b) **"Best-case" scenario:** Two algorithms are exactly complementary (one succeeds when the other one fails). All other algorithms are independent of the two first ones and of one another. Greedy is optimal. (d) **Greedy worse than Optimal:** Both mean and greedy do not choose optimally the first point: it is the best performing algorithm by itself, but does not belong to the best performing pair. (f) **Greedy worse than Mean:** Mean provides by coincidence the optimal order, which Greedy does not select. We use $\epsilon = 0.1$ (see text). ALCs are shown in legend. (g) **Mean worse than Random:** Redundant versions of the best performing algorithm are included. Mean ranks them all first rather than selecting complementary algorithm.

score (the same as the **Mean** strategy). But this algorithm is not necessarily the most informative about what second algorithm should be chosen.

Assume that we have four types of algorithms: X_1, X_2, X_3, and $X_4 = 1 - X_3$, with $P(X_1 = 1) = 0.5 + \epsilon$ and $P(X_2 = 1) = 0.5 - \epsilon$, $0 < \epsilon \ll 1$, $P(X_3 = 1) = P(X_4 = 1) = 0.5$, $X_1 \perp\!\!\!\perp X_2 \perp\!\!\!\perp (X_3|X_4)$, $X_3 \not\perp\!\!\!\perp X_4$.

Greedy will choose algorithm 1 first, then X_3 or X_4 without distinction (assume it chooses X_3), then finally it will choose X_4 (as perfectly complementing X_3) and X_2 last. Therefore the learning curve of **Greedy** will be:

$$LC(1) \simeq 0.5, \quad LC(2) \simeq 1 - (0.5)^2, \quad LC(i \geq 3) = 1 \, .$$

In contrast the optimal strategy is to choose X_3 or X_4 first, without distinction. Assume it chooses X_3 first, then it will choose X_4 and reaches perfect prediction in only two steps. Therefore the learning curve of **Optimal** will be:

$$LC(1) \simeq 0.5, \quad LC(i \geq 2) = 1 \, , \tag{11}$$

and therefore **Optimal** is better than **Greedy**.

3.e. Greedy can be worse than Random

True since Greedy can be worse that Optimal and Random can reach Optimal by chance. However, it is more interesting to find out whether Greedy can perform well in "most cases" than Random does on average, since Greedy is a deterministic algorithm, while Random has a lot of variance. See Sections Empirical Results and Computational Considerations.

3.f. Greedy can be worse than Mean

In this example, we exploit the fact that "by chance", **Mean**, which is ordering its algorithms with the marginal probabilities, would order them in an optimal order, while **Greedy** would choose a sub-optimal order. This construction is possible with at least 4 algorithms that we call A, B, C, and D. Without loss of generality, we assume that $P(A = 0) < P(B = 0) < P(C = 0) < P(D = 0)$ (no ties). In this example we chose that:

- $P(A = 0) = 0.5 - 2\epsilon$, $P(B = 0) = 0.5 - \epsilon$, $P(C = 0) = 0.5 + \epsilon$, $P(D = 0) = 0.5 + 2\epsilon$, where $0 < \epsilon \ll 1$.
- $P(A = 0, B = 0, C = 0) = 0$. Note that this also means that $P(A = 0, B = 0, C = 0, D = 0) = 0$.

The example leads **Mean** to choose the optimal order A, B, C, D that reaches the asymptote of the learning curve in 3 steps, whereas **Greedy** does not: **Greedy** chooses A, D, C, B. This can happen if we have the following conditional probabilities:

Step	**Mean**	**Greedy**
1	$P(A = 0) = 0.5 - 2\epsilon$	$P(A = 0) = 0.5 - 2\epsilon$
2	$P(B = 0 \mid A = 0) = 0.5 + 2\epsilon$	$P(D = 0 \mid A = 0) = 0.5$
		$P(C = 0 \mid A = 0) = 0.5 + 2\epsilon$
3	$P(C = 0 \mid A = 0, B = 0) = 0$	$P(C = 0 \mid A = 0, D = 0) = 0.5$
		$P(B = 0 \mid A = 0, D = 0) = 0.5$

With these values the learning curve cross each other. We can compute the learning curves values and the ALC:

$$LC(i) = 1 - P(X_{t_1} = 0, X_{t_1} = 0, \ldots, X_{t_i} = 0) \, .$$

$$ALC = \sum_{i=1}^{n} LC(i)$$

	Mean	Greedy
$1 - LC(1)$	$0.5 - 2\epsilon$	$0.5 - 2\epsilon$
$1 - LC(2)$	$(0.5 - 2\epsilon)(0.5 + 2\epsilon)$	$(0.5 - 2\epsilon) \cdot 0.5$
$1 - LC(3)$	0	$(0.5 - 2\epsilon) \cdot 0.5 \cdot 0.5$
$1 - LC(4)$	0	0

Let us verify that ALC(**Mean**) $>$ ALC(**Greedy**). ALC(**Mean**) $>$ ALC(**Greedy**), if and only if $LC(2) + LC(3)$ is larger for **Mean** than for **Greedy** since $LC(1)$ and $LC(4)$ are identical. Equivalently, $2 - LC(2) - LC(3)$ is smaller for **Mean** than for **Greedy**. We have

$$\begin{aligned} &ALC(\textbf{Mean}) - ALC(\textbf{Greedy}) \\ &= (0.5 - 2\epsilon) \cdot 0.5^2 + (0.5 - 2\epsilon) \cdot 0.5 - (0.5 - 2\epsilon)(0.5 + 2\epsilon) \\ &= 1/8 - \epsilon/2 + 1/4 - \epsilon - 1/4 + 4\epsilon^2 \qquad (12) \\ &= 1/8 - \frac{3}{2}\epsilon + 4\epsilon^2 \end{aligned}$$

which is positive for small enough ϵ. For *e.g.,* $\epsilon = 0.1$, **Mean** has a better ALC than **Greedy** (we get $0.015 > 0$).

3.g. Mean can be worse than Random

This case arises in a similar example as example 3.b. We still have $X_2 = 1 - X_1$. But we assume that there is a small difference in the average score of two types of algorithms considered: $P(X_1) = 0.5 + \epsilon$ and $P(X_2) = 0.5 - \epsilon$, with $0 < \epsilon \ll 1$. In this case, the **Mean** strategy will rank first all the algorithms of type 1. Hence for the **Mean** strategy:

$$LC(i \leq n/2) \simeq 0.5,\ LC(i > n/2) = 1\ . \qquad (13)$$

For the random strategy, we also have $LC(i > n/2) = 1$ because half of the algorithms are of a complementary type, so even in case of extreme bad luck where we draw first all the algorithms that fail on a particular task, at $n/2 + 1$ we get one that succeeds (its complement). For the **Random** strategy, between at each step, we increase our probability of getting a good algorithm, yielding the learning curve:

$$LC(i \leq n/2) \simeq 1 - 0.5^i,\ LC(i > n/2) = 1\ . \qquad (14)$$

4. For the DepU distribution, the Optimal algorithm attains $LC = 1$ in C steps.

First, note that, for the DepU distribution, since all marginal distributions are identical (average performance of algorithms identical), the Mean strategy performs like the Random strategy (since ties are broken at random).

A set of algorithms $\{X_j\}, j = 1 \cdots p$ will be called **complementary** if and only if $P(X_1 = 0, X_2 = 0, \cdots, X_p = 0) = 0$. Hence, for each task, there exists at least one successful algorithm in that set. Further, we call a **clique** a minimal set of complementary algorithms, *i.e.,* such that removing any of its members breaks complementarity. Finally, we define a notion of **complexity of a meta-learning problem** C as the **cardinal of the smallest clique** (when there is one), and $C = n$ otherwise.

For a meta-learning problem of complexity C, there exists an meta-predict strategy such that $LC(C) = 1$. Indeed, it suffices to rank first the algorithms of the smallest clique. Note however that neither the Greedy not the Optimal strategies attain necessarily $LC(C) = 1$. Nonetheless, we will see in the experimental section (Section Empirical Resutls) that the smaller C, the larger the ALC of Greedy and Optimal.

Empirical Results

In this section, we compare the four meta-predict strategies considered on various meta-distributions. The theoretical results offer no guarantee of optimality of the **Greedy** method. But we offer empirical evidence of its effectiveness.

First we report results on synthetic data constructed such that the meta-distribution includes a single clique, and we vary the complexity C (size of the clique). Specifically, a DA matrix is constructed as follows: All values are initialized with -1 (missing); for C of its columns, a 1 is randomly placed in each line; the remaining -1 values are replaced by 0 or 1 randomly, such that the average values of each column is 0.5. In these experiments we use $n = 5$ and $m = 10000$ both for training and testing.

Figure 3 shows the learning curves for the various meta-predict strategies when C varies. We see that ALC performance does not change with C for the Random, Mean, and vanilla Greedy algorithms. It does improve for smaller C for the optimal strategy. We introduce a variant of the Greedy strategy called Greedy+ in which, at meta-training time, all algorithms are tried for the first position, then greedy search is performed. This algorithm is more computationally costly at meta-training time, but it results in a single algorithm ordering, hence is not more costly at meta-predict time. Greedy+ performs nearly as well as Optimal.

Figure 4 shows the relationship between the ALC (Area under Learning Curve) and complexity C.

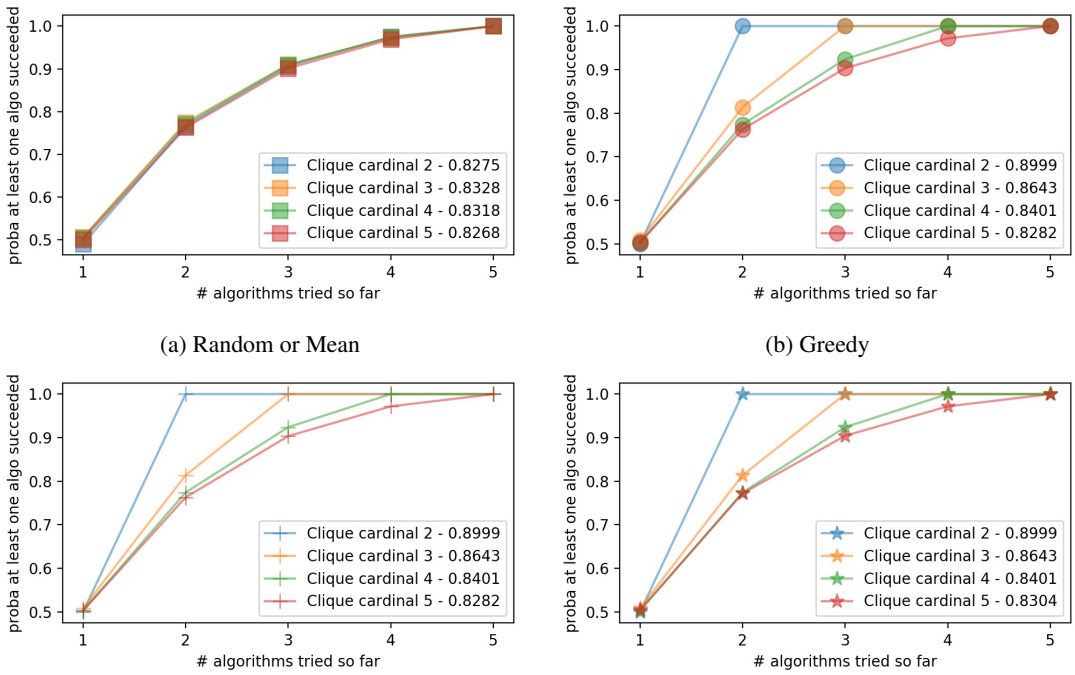
To illustrate the behavior of the meta-predict strategies considered, we also report results on benchmark meta-datasets used in [15]. These meta-datasets are Statlog (Australian Credit Approval) Data Set (21 tasks 24 algorithms), AutoML challenge dataset [6, 7] (30 tasks 17 algorithms), a subset of OpenML [16, 18] (76 datasets 292 algorithms) and an artificially generated dataset by [15] (50 tasks 20 algorithms). Since our setting considers only binary algorithm scores (failure or success), we binarized the meta-datasets (using the median value as threshold). We omit the Optimal strategy due to its prohibitive computational requirement, with a time complexity of $O(n!)$ (where n is the number of algorithms). Meta-predict performances are evaluated using the leave-one-dataset out estimator, as in the original paper.

The learning curves, shown in Figure 5, are qualitatively similar to those of the original paper [15], which we include in supplementary material, for convenience (Figure 6). Remarkably, for the first dataset (Figure 5a), the performance of Mean is worse than others both in the original and the binarized data. Notably, the Greedy algorithm generally performs best (or closely as well as the best), and always better than the Random strategy does on average.

Computational Considerations

Although this paper focuses on asymptotic analyses (infinite sample limit), for all practical purposes, we must evaluate the conditional probabilities from data, *i.e.,* a finite set of m training tasks. We provide a brief comparison of meta-predict strategies in that respect.

Unlike all the other strategies, which are deterministic and advocate one given ranking of algorithms, the **Random**



(a) Random or Mean

(b) Greedy

(c) Greedy+

(d) Optimal

Figure 3: **Learning curves for toy data, for varying complexity (clique cardinal).** All marginals (ave. algo. perf.) are identical to 0.5. Hence Mean has not advantage over Random. More subtly, neither does Greedy on average. Greedy+ however selects first the algorithms of the clique and performs as well as Optimal. ALC shown next to the clique cardinal in legend.

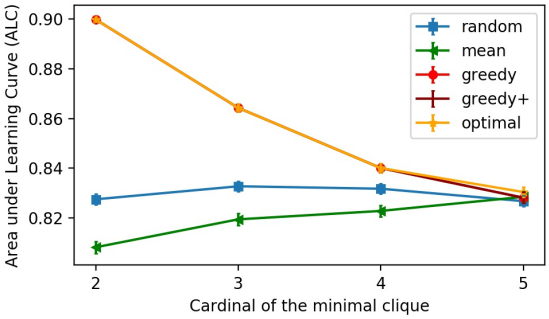


Figure 4: **ALC vs Clique cardinal C.** For Random, Mean, and Greeedy, the ALC does not vary (within the experimental error bar $\simeq VAL$. For Greedy+ and Optimal, the ALC decreases with C.

strategy has a large variance. At the first step, for instance, the variance of $LC(1)$ is $p_1 q_1$. To beat this variance, one would need to repeat random search many times, which defeats the purpose because after trying all n algorithms one is certain of finding the best one. Hence, although it may perform well on average, it is only useful as a theoretical baseline.

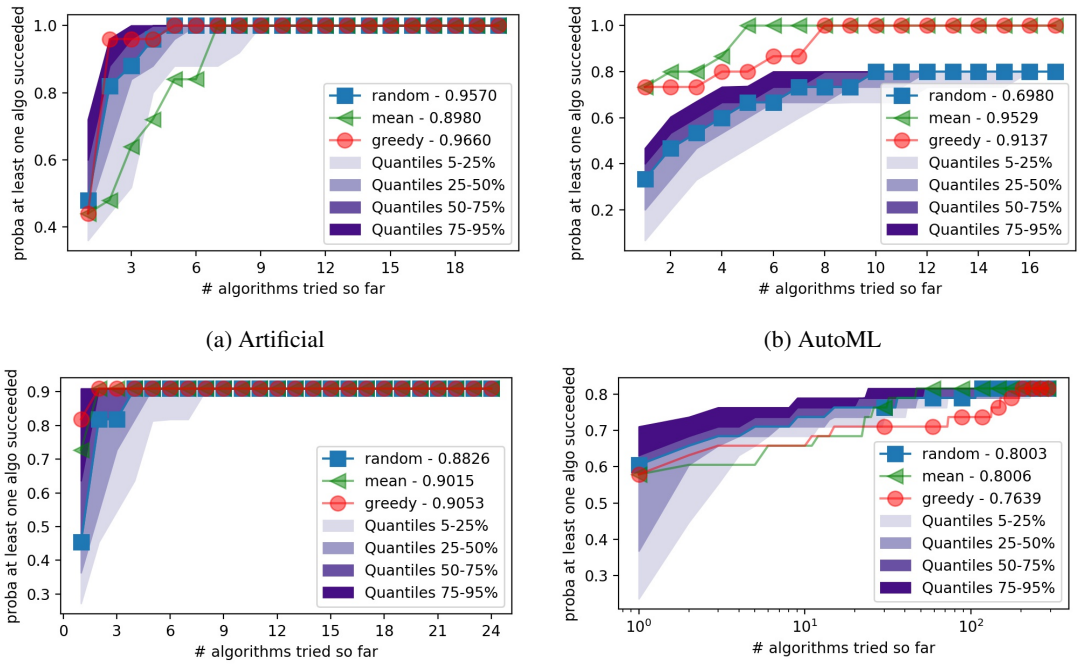
The three other strategies differ greatly in computational complexity at meta-training time:

- **Optimal:** To determine the optimal order of algorithms of the Optimal strategy, we need to conduct a search in a search space of $n!$ permutations, to find the permutation that maximize $ALC(n)$. For each permutation, we need to evaluate n conditional probabilities.
- **Greedy:** In contrast, the Greedy strategy evaluates only $n(n+1)/2$ conditional probabilities. The Greedy+ strategy requires repeating the search n times.
- **Mean:** The Mean strategy is much faster, since it requires only evaluation n marginal probabilities.

Conditional probabilities are estimated with fewer samples than marginal probabilities. This can yield to uncertainty in algorithm ranking for the Random and Optimal strategies. While Mean has n examples to evaluate all $P(X_j = 1)$. The number of examples decays exponentially with the number of conditions to evaluate $P(X_{t_i} = 1 | X_{t_1} = 0, X_{t_2} = 0, \ldots, X_{tn} = 0)$.

Discussion and Conclusion

Meta-learning as an algorithm recommendation problem is, to some extent, what every overview paper is doing: Analyzing results on past tasks, the authors generally attempt to rank algorithms in order of preference, such that readers would save time by trying the smallest possible number of algorithms before obtaining satisfactory results. This paper puts a formal framework around this problem and shows that, when algorithms are not independent of one another, ranking with the Mean strategy generally does not perform as well as the Greedy strategy, which in turns is often nearly



(a) Artificial

(b) AutoML

(c) Statlog

(d) OpenML

Figure 5: **Learning curves on benchmark datasets.** The shaded areas represent quantiles for the Random strategy.

optimum. We prove theoretically that if algorithms are independent of one another and have same average performance, all strategies perform at chance level. This situation is analogous the that of the NLF theorem. If they are independent but have different average performance, then the Mean strategy is optimal. If they have the same average performance, but are not mutually independent, the Mean algorithm performs at chance level, but the Greedy algorithm can potentially do better. However, seeding the Greedy algorithm properly is important. At the expense of a slightly slower meta-training time, the Greedy+ algorithm performs a lot better. Greedy algorithms make decisions that are only "locally" optimal, hence can be outperformed by the Optimal algorithm. However, they are much faster and are therefore good candidates for use in algorithm recommendation.

Further work includes moving from binary performance scores of algorithms to continuous scores, re-defining in this context the complexity of meta-learning, and studying the final sample case. Other extensions could be done to address the problem of missing data, and the problem of "warm starting" recommendation using algorithm meta-features (a *first-level* meta-learning problem). However, at least qualitatively, our binary *zero-level* meta-learning problem setting captures the essence of meta-learning as a recommendation problem, allowing us to sort out when and how meta-learning is possible.

Acknowledgements:

We are grateful to Lisheng Sun for sharing her benchmark datasets and her code.

References

[1] Brazdil, P.; Carrier, C. G.; Soares, C.; and Vilalta, R. 2008. *Metalearning: Applications to data mining*. Springer Science & Business Media.

[2] Feurer, M.; Klein, A.; Eggensperger, K.; Springenberg, J.; Blum, M.; and Hutter, F. 2015. Efficient and Robust Automated Machine Learning. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*, 2962–2970. Curran Associates, Inc. URL http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf.

[3] Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 1126–1135. International Convention Centre, Sydney, Australia: PMLR. URL http://proceedings.mlr.press/v70/finn17a.html.

[4] Franceschi, L.; Frasconi, P.; Salzo, S.; Grazzi, R.; and Pontil, M. 2018. Bilevel Programming for Hyperparameter Optimization and Meta-Learning. In *International Conference on Machine Learning*, 1568–1577. URL http://proceedings.mlr.press/v80/franceschi18a.html.

[5] Giraud-Carrier, C. G. 2005. Toward a Justification of Meta-learning : Is the No Free Lunch Theorem a Showstopper ?

[6] Guyon, I.; Bennett, K.; Cawley, G.; Escalante, H. J.; Escalera, S.; Tin Kam Ho; Macia, N.; Ray, B.; Saeed, M.; Statnikov, A.; and Viegas, E. 2015. Design of the 2015 ChaLearn AutoML challenge. In *2015 International Joint Conference on Neural Networks (IJCNN)*, 1–8. Killarney, Ireland: IEEE. ISBN 978-1-4799-1960-4. doi:10.1109/IJCNN.2015.7280767. URL http://ieeexplore.ieee.org/document/7280767/.

[7] Guyon, I.; Sun-Hosoya, L.; Boullé, M.; Escalante, H. J.; Escalera, S.; Liu, Z.; Jajetic, D.; Ray, B.; Saeed, M.; Sebag, M.; Statnikov, A.; Tu, W.-W.; and Viegas, E. 2018. Analysis of the AutoML Challenge series 2015-2018. In Hutter, F.; Kotthoff, L.; and Vanschoren, J., eds., *AutoML: Methods, Systems, Challenges*, The Springer Series on Challenges in Machine Learning. Springer Verlag. URL https://hal.archives-ouvertes.fr/hal-01906197.

[8] Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[9] Lopez-Paz, D.; Muandet, K.; Schölkopf, B.; and Tolstikhin, I. 2015. Towards a learning theory of cause-effect inference. In *International Conference on Machine Learning*, 1452–1461.

[10] Mısır, M.; and Sebag, M. 2017. Alors: An algorithm recommender system. *Artificial Intelligence* 244: 291–314.

[11] Muñoz, M. A.; Villanova, L.; Baatar, D.; and Smith-Miles, K. 2018. Instance spaces for machine learning classification. *Machine Learning* 107(1): 109–147. ISSN 1573-0565. doi:10.1007/s10994-017-5629-5. URL https://doi.org/10.1007/s10994-017-5629-5.

[12] Rice, J. R. 1976. The algorithm selection problem. In *Advances in computers*, volume 15, 65–118. Elsevier.

[13] Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, 4077–4087.

[14] Sun-Hosoya, L. 2019. *Meta-Learning as a Markov Decision Process*. phdthesis, Université Paris Saclay (COmUE). URL https://hal.archives-ouvertes.fr/tel-02422144.

[15] Sun-Hosoya, L.; Guyon, I.; and Sebag, M. 2018. Activmetal: Algorithm recommendation with active meta learning .

[16] Van Rijn, J. N.; Bischl, B.; Torgo, L.; Gao, B.; Umaashankar, V.; Fischer, S.; Winter, P.; Wiswedel, B.; Berthold, M. R.; and Vanschoren, J. 2013. OpenML: A collaborative science platform. In *Joint european conference on machine learning and knowledge discovery in databases*, 645–649. Springer.

[17] Vanschoren, J. 2018. Meta-Learning: A Survey. *arXiv:1810.03548 [cs, stat]* URL http://arxiv.org/abs/1810.03548. ArXiv: 1810.03548.

[18] Vanschoren, J.; van Rijn, J. N.; Bischl, B.; and Torgo, L. 2014. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter* 15(2): 49–60. ISSN 19310145. doi:10.1145/2641190.2641198. URL http://arxiv.org/abs/1407.7722. ArXiv: 1407.7722.

[19] Wolpert, D. 2001. The Supervised Learning No-Free-Lunch Theorems. In *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*. doi:10.1007/978-1-4471-0123-9_3.

[20] Wolpert, D. H. 1996. The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation* 8(7): 1341–1390. ISSN 0899-7667. doi:10.1162/neco.1996.8.7.1341. URL https://doi.org/10.1162/neco.1996.8.7.1341.

[21] Wolpert, D. H.; and Macready, W. G. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1(1): 67–82. ISSN 1089-778X. doi:10.1109/4235.585893. URL https://ti.arc.nasa.gov/m/profile/dhw/papers/78.pdf.

Supplementary material

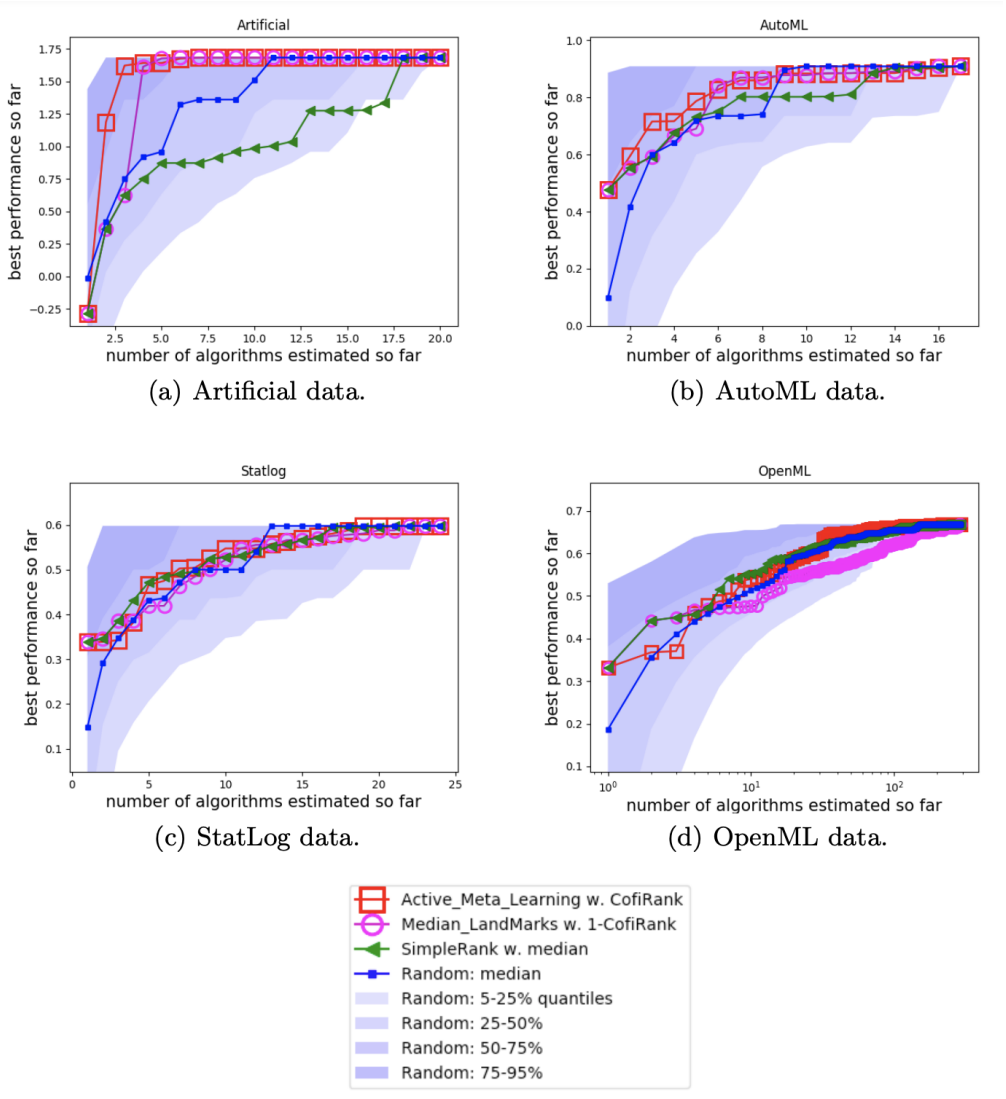
Asymptotic Analysis of Meta-learning as a Recommendation Problem

Zhengying Liu and **Isabelle Guyon**

To facilitate comparing our Figure 5 with the figure of the paper of Sun et al. [15], which uses the same benchmark meta-datasets, we reproduce her figure (Figure 6). She uses four methods:

1. Random search (similar to our Random strategy)
2. SimpleRank w. median (similar to our Mean strategy)
3. Active_Meta_Learning w. CofiRank (a Greedy method)
4. Median_LandMarks w. CofiRank (another Greedy method)

She represents the quantiles of Random search by shaded areas.



(a) Artificial data. (b) AutoML data.

(c) StatLog data. (d) OpenML data.

Figure 6: Learning curves in Lisheng Sun et al. [15]