HAL
archives-ouvertes.fr

Meta online learning: experiments on a unit commitment problem

Jialin Liu, Olivier Teytaud

▶ To cite this version:

Jialin Liu, Olivier Teytaud. Meta online learning: experiments on a unit commitment problem. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Apr 2014, Bruges, Belgium. 2014. <hal-00973397>

HAL Id: hal-00973397
https://hal.inria.fr/hal-00973397

Submitted on 4 Apr 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Meta online learning: experiments on a unit commitment problem

Jialin Liu and Olivier Teytaud

TAO (Inria), LRI, UMR 8623 (CNRS - Univ. Paris-Sud),
bat 490 Univ. Paris-Sud 91405 Orsay, France, jialin.liu@lri.fr

Abstract. Online learning is machine learning, in real time from successive data samples. Meta online learning consists in combining several online learning algorithms from a given set (termed portfolio) of algorithms. The goal can be (i) mitigating the effect of a bad choice of online learning algorithms (ii) parallelization (iii) combining the strengths of different algorithms. Basically, meta online learning boils down to combining noisy optimization algorithms. Whereas many tools exist for combining combinatorial optimization tools, little is known about combining noisy optimization algorithms. Recently, a methodology termed *lag* has been proposed for that. We test experimentally the *lag* methodology for online learning, for a stock management problem and a cartpole problem.

1 Introduction

This paper is organized as follows. In this introduction, Section 1.1 is a brief overview of the online learning setting, Section 1.2 presents noisy optimization, Section 1.3 presents meta-learning. Then, Section 2 presents the algorithms used in this paper and Section 3 provides experimental results.

1.1 Online learning (OL)

Machine Learning (ML) consists in extracting, from data, a decision function. OL consists in doing this online. The input data themselves can be dynamically influenced by the current approximate decision function. Typically, a function measures the error made by the current approximate decision function, e.g. mean squared error for supervised ML, or reward in the case of control problems; this function is termed *objective function*. OL then boils down to the minimization, on stochastic samples, of the objective function. This work focuses on the black-box case - no internal property of the objective function is used, which is usual in reinforcement learning.We consider a stochastic objective function $R(\theta)$, with θ in a search domain. We assume that an algorithm has access to independent realizations of $R(\theta)$. The goal of the optimization algorithm is to approximate some θ^* such that $\forall\theta$, $\mathbb{E}R(\theta^*) \geq \mathbb{E}R(\theta)$. A black-box noisy optimization algorithm (NOA), for its n^{th} request to the black-box objective function, can only provide θ_n and receive a realization of $R(\theta_n)$.

1.2 Noisy optimization, a.k.a. stochastic approximation of minima

We present here the state of the art in stochastic approximation of minima. For short, we will assume without loss of generality that $\mathbb{E}R(\theta^*) = 0$. **Definitions.**

The black-box NOA is in charge of providing two sequences: (i) the θ_n, for $n \in \{1, 2, \dots, \}$, which is sent to the oracle black-box objective function, which will return a random independent realization of $R(\theta)$; (ii) the $\tilde{\theta}_n$, for $n \in \{1, 2, \dots, \}$, which is the current approximation of the optimum that the algorithm points out after having performed n function evaluations. Some algorithms and some published benchmark platforms do not distinguish θ_n and $\tilde{\theta}_n$. However, it is known that the best rates (in terms of convergence of $\tilde{\theta}_n$ to the optimum) can only be reached when $\tilde{\theta}_n$ (close to the optimum) and θ_n (a bit farther from the optimum) are distinguished. We will also use, in many cases, an additional sequence $\hat{\theta}_n$, when the algorithm's run is naturally splitted into iterations; $\hat{\theta}_n$ is then the approximation of the optimum obtained after iteration n. $\hat{\theta}_n$ is usually equal to $\tilde{\theta}_m$ for some $m \geq n$, where m depends on the number of function evaluations per iteration. We refer to [1] and references therein for the state of the art in noisy optimization.

1.3 Meta-learning

A recent trend in optimization is an online non-invasive combination of optimizers. Several solvers are available, and one of them must be selected online. This approach is particularly successful in combinatorial optimization[2, 3]. While some authors consider "invasive" combinations the simple selection, online, of the best algorithm is not trivial in stochastic contexts. We will here focus on non-invasive (except chaining) combinations of online NOA.

2 Algorithms

This section presents the algorithms used for the experiments in this paper. We present direct policy search (Section 2.1), which is a form of OL (Section 2.2), and then switch to the combination of several direct policy search algorithms (Section 2.3).

2.1 Direct Policy Search (DPS)

DPS is a form of reinforcement learning[4]. DPS consists in optimizing the parameters of a parametric policy.It boils down to stochastic optimization of parameters on simulations, as follows: (i) A state space S, an action space A and an initial state s_0 are defined. (ii) A transition function $(s, a) \mapsto t(s, a) \in S \times \mathbb{R}$ maps (state, action) pairs to (state, reward) pairs. This function is stochastic. (iii) A number T of time steps is given. (iv) A parametric control function $s \mapsto \pi_\theta(s)$ maps states to actions; it depends on a vector θ of parameters. (v) A simulation is a sequence $s_0, \dots, s_T$ with $(s_{t+1}, r_t) = t(s_t, a_t)$ and $a_t = \pi_\theta(s_t)$. The reward of the simulation is $R(\theta) = \sum_{t=0}^{T-1} r_t$. The reward $R(\theta) \in \mathbb{R}$ depends on θ and is stochastic. (vi) We use some NOA for finding $\theta^* \in \arg\min_\theta R(\theta)$. Black-box DPS is a control approach as follows, depending on (a) an initialization module (b) an update module (c) a recommendation module (d) a parametric controller π_θ: (1) **Initialization module:** An initial value θ_0

is proposed by the initialization module. (2) A budget (in time or in number of simulations) is given. (3) $n \leftarrow 0$. (4) While the budget is not exhausted: (4-a) Do a simulation by (v) above and get a reward r_n. (4-b) **Update module:** $\theta_{n+1} \leftarrow Update(\theta_0, r_0, \ldots, \theta_n, r_n)$. (4-c) **Recommendation module:** Define $\tilde{\theta}_n \leftarrow Recommend(\theta_0, r_0, \ldots, \theta_n, r_n)$. (4-d) $n \leftarrow n+1$. (5) Output $\tilde{\theta}_{n-1}$.

As a parametric controller, we will use either conformant planning (where the decision only depends on the time step) or fully connected Neural Networks (NN). We will, in this paper, do experiments on black-box DPS with NN.

2.2 Stochastic optimization algorithms

Let us note d the dimension of the search space. We will here apply 3 different NOAs with various parametrizations. **Fabian's algorithm** contains in each iteration: (i) Gradient estimation: the gradient g_n at iteration n is estimated by finite differences using estimates of $\mathbb{E}R(\hat{\theta}_{n-1} \pm \sigma_n e_i)$ where the e_i, $i \in \{1, \ldots, d\}$, are the standard orthonormal basis of the search space. $\sigma_n = 1/n^\gamma$. (ii) Update: an update is performed by $\hat{\theta}_n = \hat{\theta}_{n-1} + (a/n)g_n$. **Noisy Newton's algorithm**, as follows at each iteration: (i) Finite differences: the Hessian and gradients are estimated with approximate fitness values at points $\hat{\theta}_{n-1} \pm \sigma_n e_i \pm \sigma_n e_j$ where the e_i, $i \in \{1, \ldots, d\}$, are the standard orthonormal basis. $\sigma_n = A/n^\alpha$. (ii) Resampling: the objective function is resampled at each of these points $\lceil Bn^\beta \rceil$ times in order to decrease the variance by a factor $\lceil Bn^\beta \rceil$; (iii) Newton step: then the estimate $\hat{\theta}_n$ of the optimum is set at the minimum of the approximate quadratic model obtained with the Hessian and gradient above. **Resampling Self-Adaptive (μ, λ)-Evolution Strategy (RSAES):** (i) Classical offspring generation and selection;(ii) Resampling: Each of these offspring is resampled $\lceil Kn^\zeta \rceil$ times in order to reduce the variance.**We will define (in Section 2.3) additional solvers,** namely portfolios of the solvers above; the problem in this paper is precisely the definition of these portfolio methods.

2.3 Combining noisy optimization algorithms

Combining learning algorithms might, at first view, look like a simple task; just run each optimization algorithm separately and pick up the best performing one. However, testing which algorithm is best is simple in the case of deterministic optimization - just pick up the algorithm with best search point so far. But in stochastic optimization, evaluations are noisy, so evaluations must be repeated many times before we can know, by statistical tests, which one is best. The cost of such a comparison increases when points have close fitness values. When several optimization algorithms are to be compared, at least those who converge to the optimum will have similar fitness values. Therefore the comparison cost will increase. Let us consider a typical scenario as follows: (i) There are M stochastic optimization algorithms, termed S_1, S_2, ..., S_M. (ii) The noise is additive, with variance 1. (iii) Let us define $\tilde{\theta}_n^i$ the point in the state space recommended by optimization algorithm i after n fitness evaluations. (iv) Let us assume that all of them verify the classical $\log - \log$ convergence rate as

follows $\log(-\mathbb{E}R(\tilde{\theta}^i_n)) \simeq -\log n$. (v) Then $|\mathbb{E}R(\tilde{\theta}^i_n) - \mathbb{E}R(\tilde{\theta}^j_n)|$ is $\epsilon_{targetPrecision} = O(1/n)$. (vi) Comparing $\tilde{\theta}^i_n$ and $\tilde{\theta}^j_n$ correctly needs a comparison budget $b = \Omega(n^2)$ because the precision is $\epsilon_{statisticalPrecision} = \Theta(1/\sqrt{b})$. (vii) So we have in terms of numbers of evaluations: (a) a comparison budget $b \gg n$, for comparing just two recommendations; (b) a budget n for constructing $\tilde{\theta}^i_n$ (the budget for constructing θ^i_n is n by definition) (viii) This is very sad, because $b \gg n$, i.e. we spend far more budget on comparing recommendations $\{\tilde{\theta}^i_n; i \in \{1, \dots, M\}\}$ than in actually constructing them. This effect is the key in this paper. **A possible solution: NOPA (Noisy Optimization Portfolio Algorithm).** Typically, solvers have a smooth convergence rate, so that after some time, the solver which performs best is always the same. Therefore, instead of trying to compare $\tilde{\theta}^i_n$, we might compare $\tilde{\theta}^i_m$ for some $m \ll n$. Then, the difference between them is easier to detect. We therefore propose an iterative portfolio as follows, depending on sequences $n \mapsto k_n$, s_n, r_n with $k_n \leq n$ and r_n a non-decreasing sequence:

Iteration n of the portfolio $\{S_1, \dots, S_M\}$ containing M NOAs:

- **Initialization module:** If $n = 0$ initialize all S_i, $i \in \{1, \dots, M\}$.
- For $i \in \{1, \dots, M\}$:
  - **Update module:** Apply an iteration of solver S_i until it has received at least n data samples.
  - Let $\theta_{i,n}$ be the current recommendation by solver S_i.
- Comparison module: If $n = r_m$ for some m, then
  - For $i \in \{1, \dots, M\}$, perform s_m evaluations of the (stochastic) reward $R(\theta_{i,k_n})$ and define y_i the average reward.
  - Define $i^* \leftarrow \arg\min_{i \in \{1,\dots,M\}} y_i$.
- **Recommendation module:** $\tilde{\theta} = \theta_{i^*,n}$

The parameters are the 3 functions, as follows. The $(r_n)_{n \in \mathbb{N}}$ is the list of iterations at which algorithms are compared. s_n is the budget used for evaluating each solver at iteration r_n. k_n is the index of the recommendations used for comparisons at iteration n; i.e. it is the *comparison lag*[1]; at iteration n, we compare the solvers using what they recommended at iteration k_n and typically $k_n \ll n$.

The mathematical analysis of NOPA is beyond the scope of this experimental paper, but let us see some simple properties: (i) The number of data samples spent for solver i until iteration r_m of NOPA is $M \times r_m$ (within possibly the number of data samples used in the last iteration). Therefore, the number of data samples spent for applying the solvers is roughly $M \times r_m$. (ii) The number of data samples spent for comparing the solvers is $M \times \sum_{i=1}^m s_i$. (iii) **Comparisons use a small budget:** So if $\sum_{i=1}^m s_i = o(r_m)$, a proportion $(1 - o(1))$ of the data samples are used for the solvers themselves, and not for comparison purpose; therefore, the impact of the comparisons is negligible. (iv) **Comparisons are sufficiently well done:** Additionally, (iv-a) if the variance of $R(\theta)$ is roughly constant, then the standard deviation of y_i is $O(1/\sqrt{s_m})$, so we can find the optimum i^* within precision roughly $1/\sqrt{s_m}$ on $R(\theta_{i^*,k_n})$. (iv-b) if $\mathbb{E}R(\theta_{i,k_n}) = \Theta(1/k_n)$ (as is the case for Fabian's algorithm, for example, in the case of constant variance of $R(\theta)$, as shown in [5]), then $k_n = O(\sqrt{s_m})$ ensures that the precision loss (due to the comparison) is less than the optimization

precision. This preliminary analysis suggests two constraints for choosing the parametrization of NOPA, namely $\sum_{i=1}^{m} s_i = o(r_m)$ and $k_n = O(\sqrt{s_m})$.

3 Experiments

Several solvers are used in the experiments, as shown in Tab. 1. Due to length

Notation	Algorithm and parametrization
1 ($RSAES$)	RSAES with $\lambda = 10d$, $\mu = 5d$, $resampling_n = 10n^2$.
2 ($Fabian1$)	Fabian's solver with stepsize $\sigma_n = 10/n^{0.49}$, $a = 100$.
3 ($Fabian2$)	Fabian's solver with stepsize $\sigma_n = 10/n^{0.05}$, $a = 100$.
4 ($Newton1$)	Newton's solver with stepsize $\sigma_n = 10/n$, $resampling_n = n^2$.
5 ($Newton2$)	Newton's solver with stepsize $\sigma_n = 100/n^4$, $resampling_n = n^2$.
6 ($P.12345$)	NOPA of solvers 1 to 5 with $k_n = \lceil n^{0.1} \rceil$, $r_n = n^3$, $s_n = 15n$.
7 ($P.12345 + Sharing$)	Solver 6 ($P.12345$) with information sharing enabled.
8 ($P.22$)	NOPA of 2 (identical) solver 2, with $k_n = \lceil n^{0.1} \rceil$, $r_n = n^3$, $s_n = 15n$.
9 ($P.22 + Sharing$)	Solver 8 ($P.22$) with information sharing enabled.
10 ($P.222$)	NOPA of 3 (identical) solver 2, with $k_n = \lceil n^{0.1} \rceil$, $r_n = n^3$, $s_n = 15n$.
11 ($P.222 + Sharing$)	Solver 10 ($P.222$) with information sharing enabled.

Table 1: Solvers used in the experiments. n: iteration number. *Sharing* means that when NOAs are compared, the current iterate of the best is used, for all NOAs, as next iterate. In conclusions below, $P.22*$ means that the conclusion applies to both $P.22$ and $P.22 + Sharing$. All experiments below correspond to a budget of 160s and are repeated 30 times.

constraints, we do not present the artificial experiments performed on $R(\theta) = f(\theta) + Df(\theta)^{z/2} Gaussian$, with $f(\theta) = -\sum_{i=1}^{d} (\theta_i - \theta_i^*)^2 / i^p$, tested in dimension $d \in \{2, 5\}$, with $z \in \{0, 1, 2\}$ ($z = 2$ is the so-called multiplicative noise, $z = 0$ is the additive noise, $z = 1$ is intermediate), with $p = 0$ (well conditioned) or $p = 2$ (ill conditioned), $D = 0.1$ (low noise) or $D = 1$ (strong noise). Basically, NOPA performed similarly to the best of its NOAs, in all cases for $P.22*$ and $P.222*$ and in most (but not all) cases for $P.12345*$. **Realistic experiments**, on a Unit Commitment (minimization) problem, are presented in Tab. 2. $P.22*$ and $P.222*$ performed very well, often significantly better than each of their NOAs. On the other hand, $P.12345*$ failed, because $RSAES$ is parametrized for using a large number of evaluations per iteration, so that the budget was almost monopolized by $RSAES$; this leads to the conclusion that NOAs which have far more evaluations per iteration than other NOAs disrupts NOPA. **Experiments with NN**, on a Cart-Pole problem, are presented in Tab. 3.

4 Conclusions

We tested empirically NOPA, which combines several NOAs. NOPA is usually nearly as efficient as the best of the NOAs, except when the budget in terms of number of evaluations is not sufficient, compared to the number of evaluations in one of the NOAs (in the unit commitment problem, $RSAES$ has monopolized most of the budget).Importantly, NOPA could stabilize the uncertainty; $P.22*$ and $P.222*$ perform often better than their NOAs, by discarding unlucky runs, in spite of the additional costs of multiple NOAs runs.

St, T, d	$P.22$	$P.22 + S.$	$P.222$	$P.222 + S.$	Best NOA	Worst NOA
3, 21, 63	0.61 ± 0.07	0.63 ± 0.03	0.63 ± 0.05	0.63 ± 0.07	0.49 ± 0.08	0.81 ± 0.05
4, 21, 84	0.75 ± 0.02	0.75 ± 0.03	0.79 ± 0.05	0.76 ± 0.03	0.69 ± 0.06	1.27 ± 0.06
5, 21, 105	0.53 ± 0.04	0.58 ± 0.08	0.58 ± 0.03	0.52 ± 0.05	0.58 ± 0.04	1.44 ± 0.16
6, 15, 90	0.40 ± 0.05	0.39 ± 0.06	0.37 ± 0.06	0.39 ± 0.06	0.38 ± 0.06	0.96 ± 0.13
6, 21, 126	0.53 ± 0.08	0.54 ± 0.08	0.55 ± 0.07	0.54 ± 0.07	0.54 ± 0.07	1.78 ± 0.37
8, 15, 120	0.53 ± 0.03	0.50 ± 0.05	0.53 ± 0.02	0.51 ± 0.05	0.51 ± 0.04	1.70 ± 0.10
8, 21, 168	0.69 ± 0.04	0.77 ± 0.09	0.73 ± 0.06	0.71 ± 0.04	0.71 ± 0.06	2.68 ± 0.02
7, 21, 147	0.70 ± 0.07	0.70 ± 0.05	0.70 ± 0.07	0.70 ± 0.07	0.69 ± 0;06	2.28 ± 0.08

Table 2: Stochastic Unit Commitment problems, conformant planning. Given a same budget, a NOPA of identical solvers can outperform its NOAs. St: number of stocks. RSAES is usually the best NOA for small dimensions and variants of Fabian for large dimension.

Solver	2 neurons, $d = 9$	4 neurons, $d = 17$	8 neurons, $d = 33$
1 ($RSAES$)	-0.458033±0.045014	-0.421535±0.045643	-0.351726±0.051705
2 ($Fabian1$)	0.002226±5.29923e-05	0.002089±1.57766e-04	0.00221±8.14518e-05
3 ($Fabian2$)	0.002318±9.80792e-05	0.002238±1.14289e-04	0.00236±1.51244e-04
4 ($Newton1$)	0.002229±6.08973e-05	-0.030731±0.111294	0.002247±1.19829e-04
5 ($Newton2$)	0.00227±5.2989e-05	0.002217±7.80888e-05	0.002307±9.96404e-05
6 ($P.12345$)	-0.408705±0.068428	-0.3917±0.071791	-0.320399±0.050338
7 ($P.12345 + S.$)	-0.42743±0.05709	-0.403707±0.056173	-0.354043±0.069576

Table 3: Approximate convergence rates $\log(-R(\tilde{\theta}_n))/\log(n)$ for Cart-Pole, a multimodal problem, using NN. n: evaluation number. Fabian's algorithm and Newton's algorithm are not able to solve multimodal problem. The problem is easily solved by NOPA, because one solver is much better than the others - this makes comparisons easy.

References

[1] Marie-Liesse Cauwet, Jialin Liu, and Olivier Teytaud. Algorithm portfolios for noisy optimization: Compare solvers early. In *Proceedings of Lion8*, 2014.

[2] Lars Kotthoff. Algorithm selection for combinatorial search problems: A survey. *CoRR*, abs/1210.7959, 2012.

[3] Eugene Nudelman, Kevin Leyton-Brown, H. Hoos Hoos, Alex Devkar, and Yoav Shoham. Understanding random sat: beyond the clauses-to-variables ratio. In M. Wallace, editor, *Principles and Practice of Constraint Programming CP 2004, LLNCS 3258*, volume 3258 of Lecture Notes in Computer Science, pages 438–452. Springer Berlin / Heidelberg, 2004.

[4] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction.* MIT Press., Cambridge, MA, 1998.

[5] Vaclav Fabian. Stochastic Approximation of Minima with Improved Asymptotic Speed. *Annals of Mathematical statistics*, 38:191–200, 1967.