Meta-algorithm to Choose a Good On-Line Prediction (Short Paper)

Alexandre Dambreville[1(✉)], Joanna Tomasik[2], and Johanne Cohen[3]

[1] LRI, CentraleSupélec, Université Paris-Sud,
Université Paris-Saclay, Orsay, France
Alexandre.Dambreville@lri.fr
[2] LRI, CentraleSupélec, Université Paris-Saclay, Orsay, France
Joanna.Tomasik@lri.fr
[3] LRI, CNRS, Université Paris-Saclay, Orsay, France
Johanne.Cohen@lri.fr

Abstract. Numerous problems require an on-line treatment. The variation of the problem instance makes it harder to solve: an algorithm used may be very efficient for a long period but suddenly its performance deteriorates due to a change in the environment. It could be judicious to switch to another algorithm in order to adapt to the environment changes.

In this paper, we focus on the prediction on-the-fly. We have several on-line prediction algorithms at our disposal, each of them may have a different behaviour than the others depending on the situation. First, we address a meta-algorithm named *SEA* developed for experts algorithms. Next, we propose a modified version of it to improve its performance in the context of the on-line prediction.

We confirm the efficiency gain we obtained through this modification in experimental manner.

1 Introduction

Let us assume that we have several algorithms at our disposal to solve a given problem. One of them may perform very well for a situation but badly for another situation whereas for another algorithm it is the opposite. If we were in an off-line scenario, we could determine in which situation we are and select the best algorithm once for all. In this paper, we address an on-line scenario, i.e. the environment may change with time and evolve from one situation to another. Our goal is to use a meta-algorithm that dynamically switches among the available algorithms.

First, we analyse a meta-algorithm named *Strategic Expert meta-Algorithm* (*SEA*) [3] and discuss its advantages and drawbacks in Sect. 2. Next, we modifie it (Sect. 3) to make it fit the environment quicker. We evaluate the performance of our meta-algorithm through numerical experiments in Sect. 4.

B. Bonakdarpour and F. Petit (Eds.): SSS 2016, LNCS 10083, pp. 126–130, 2016.
DOI: 10.1007/978-3-319-49259-9_10

2 Existing Meta-algorithm, *SEA*

Let us assume that we have n algorithms at our disposal. We denote M_i the average payoff of algorithm i since we used it, and N_i the number of steps on which we use algorithm i when it is selected. *SEA* (Strategic Expert meta-Algorithm [3]) alternates the exploration and exploitation phases as described in Algorithm 1:

Algorithm 1. SEA

For each $i \in [\![1; n]\!]$, $M_i \leftarrow 0$, $N_i \leftarrow 0$, iter $\leftarrow 1$
procedure SEA
  loop
    $U \leftarrow \text{Random}(0, 1)$
    if $U < 1/\text{iter}$ **then** $i \leftarrow \text{Random}([\![1; n]\!])$ ▷ *Exploration phase*;
    else $i \leftarrow \underset{i \in [\![1;n]\!]}{\text{argmax}}\ M_i$ ▷ *Exploitation phase*;
    $N_i \leftarrow N_i + 1$.
    Execute algorithm i for N_i steps;
    $R \leftarrow$ average payoff of i during these N_i steps;
    $M_i \leftarrow M_i + \frac{2}{N_i+1}(R - M_i)$;
    iter $\leftarrow$ iter $+ 1$;
  end loop
end procedure

The analysis of the *SEA* algorithm leads us to formulate a list of its advantages and a list of its drawbacks. Its strengths are:

1. If the environment does not change, *SEA* is able to find the best algorithm which fits the situation.
2. If the environment does change, the average reward of *SEA* is at least as good as the average reward of the best algorithm when it was played in infinite time (see Theorem 3.1 of [3]).

Its weaknesses are:

1. It is proved that, in the long run, all of the algorithms will be used countless times by *SEA*. However, if there are many algorithms available, some of them might not be tried before a long time. Indeed, the more the time passes, the smaller the probability of an exploration is (Lemma 3.1 of [3]).
2. *SEA* computes the mean M_i since the first iteration that is why M_i suffers from inertia when the number of iterations increases. Even a drastic change for the average payoff R may be impossible to be detected what slows down the switching between algorithms. In certain situations, it would have been advantageous to switch to a very efficient algorithm, but *SEA* is not reactive enough to do it (see Figs. 2a and b).

3 Our *Dynamic SEA*

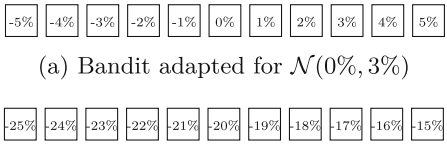
We modify *SEA*, trying to overcome its weaknesses mentioned above. For the second point, to make the mean be more reactive, instead of a long run mean, for M_i, we use the average payoff during the last N_i steps, i.e. at line 10 of Algorithm 1, we put $M_i \leftarrow R$. It allows *SEA* to have a good overview of the recent performance of an algorithm. Now, to switch to another algorithm, *SEA* just has to wait for a new exploration. This brings us to the first point of the drawbacks: an exploration may take a long time to come and it will take even much more time to try each algorithm.

In order to ensure more frequent explorations, we reset our meta-algorithm occasionally. During the exploitation of an algorithm i (line 6 of Algorithm 1), if the payoff is smaller compared to the previous iteration, we set $\forall i' \neq i, M_{i'} = \infty$ (after line 9). With this mechanism, the next exploitations will try each algorithm (different than i) at least once and then determine the best of them for the actual situation. Likewise, we use this mechanism to overcome the first weakness listed and we make our version of *SEA* try each algorithm at least once. Thereby we avoid having an untested algorithm for too long time.

4 Experiments

We start this section by explaining the experimental setup used. We evaluate our meta-algorithm for the following prediction problem. Let (D_i) be a positive integer sequence. This sequence is disturbed by a noise (N_i), which give us a jammed sequence $(J_i) = (D_i + N_i)$. At time i we receive the real data D_i and the jammed data for the next step J_{i+1}. Our goal at each time i is to recover D_{i+1} from J_{i+1}. We denote (R_{i+1}) the result of our recovering. To measure the performance of the result at time i, we propose to use a reward $\delta_i = \exp\left(-\left|\frac{R_i - D_i}{D_i}\right|\right)$, whose value always is in $(0; 1]$. If we obtain $R_i = D_i$ (the optimal result), the reward reaches the maximal value and $\delta_i = 1$. Moreover, the farther from D_i our result R_i is, the closer to 0 the reward δ_i is.

Our proposition consists in using multi-armed bandit algorithms [1]. A bandit is a method that offers us several strategies, represented by its arms, to play. Each arm has a certain reward attributed. At each time, a player choses a bandit arm and expects to win, i.e. to maximize the mean of the rewards obtained. In our problem, each arm corresponds to a modification of J_i, expressed in terms of a percentage $(x\%)$ of J_i. We denote $(Arm)(J_i) = J_i + x\%(J_i) = R_i$ the effect of an arm on the jammed value J_i.



(a) Bandit adapted for $\mathcal{N}(0\%, 3\%)$

(b) Bandit adapted for $\mathcal{N}(-20\%, 3\%)$

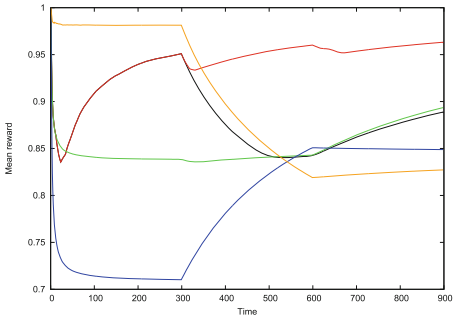
(c) Bandit adapted for $\mathcal{N}(+20\%, 3\%)$

Fig. 1. Three bandits

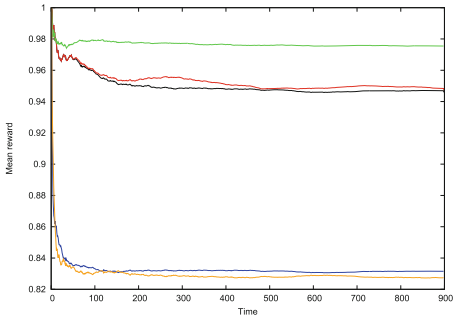
In our experiments, we use the trace of the 1998 World Cup Web site [2], which gives the number of requests by hour on the site as (D_i) (this trace is commonly used in the context of evaluation of scheduling algorithms). We generate different kind of noise on this trace in order to pinpoint the effect of our modifications and validate the dynamic version of *SEA*. We use a Gaussian noise for (N_i): at each time i, we set the mean and the variance as percentages of D_i: $N_i = \mathcal{N}(D_i\mu\%, D_i\sigma\%)$. More precisely, we divide (D_i) into three equal parts and we add a different noise on each of them. We denote $n_1 \to n_2 \to n_3$ the sequence of noise used. The four types of noise we use are: $n_{+20,\pm3} = \mathcal{N}(+20\,\%, 3\,\%)$, $n_{-20,\pm3} = \mathcal{N}(-20\,\%, 3\,\%)$, $n_{0,\pm3} = \mathcal{N}(0\,\%, 3\,\%)$ and $n_{0,\pm30} = \mathcal{N}(0\,\%, 30\,\%)$.

For the first three noise variants, we have three bandit algorithms, one specialized for each environment as illustrated in Fig. 1. The last noise variant has a great variability that makes it unpredictable for any of our bandit algorithms. We consider three scenarii: $n_{-20,\pm3} \to n_{20,\pm3} \to n_{0,\pm3}$, $n_{0,\pm3} \to n_{-20,\pm3} \to n_{20,\pm3}$ and $n_{0,\pm30} \to n_{0,\pm30} \to n_{0,\pm30}$.

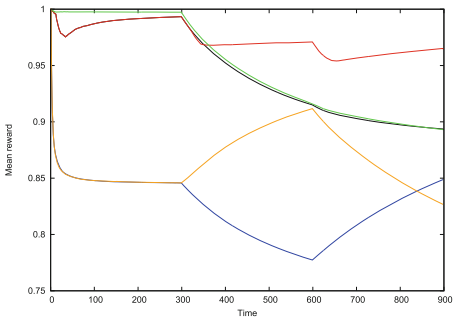
We show our results in Fig. 2 which represent the evolution of the average reward of our algorithms. Each curve is the mean of one hundred different runs of the algorithm.



(a) $n_{-20,\pm3} \to n_{20,\pm3} \to n_{0,\pm3}$



(c) $n_{0,\pm30} \to n_{0,\pm30} \to n_{0,\pm30}$



(b) $n_{0,\pm3} \to n_{-20,\pm3} \to n_{20,\pm3}$

Dynamic SEA
SEA
Bandit 1 (Fig. 1a)
Bandit 2 (Fig. 1b)
Bandit 3 (Fig. 1c)

(d) Legend

Fig. 2. Mean rewards of our algorithms

The half-width of confidential intervals computed at confidence level $\alpha = 0.05$ never exceeds 1.5 % of the corresponding mean. We do not thus incorporate them in the figures.

We discuss the results of our experiments for $n_{-20,\pm3} \rightarrow n_{20,\pm3} \rightarrow n_{0,\pm3}$ and for $n_{0,\pm3} \rightarrow n_{-20,\pm3} \rightarrow n_{20,\pm3}$ depicted in Figs. 2a and b respectively.

We build N_i in such a way that each bandit algorithm outperforms the others for a third of the time, and indeed, it is what we note in Figs. 2a and b. We observe that the *SEA* algorithm follows the best algorithm in average as time grows. Nevertheless, due to the inertia of the mean, *SEA* is very slow to switch from an algorithm to another. At the opposite, the *Dynamic SEA* can fit the environment very quickly.

For the last experiment in which the prediction is characterized by an excessive variability of the noise (Fig. 2c), both *SEA* and *Dynamic SEA* follow the first bandit (Fig. 1a) which has the best reward in average. Whatever the situation, *Dynamic SEA* is at least as good as *SEA*.

5 Conclusion

At first, we tested the *SEA* algorithm to dynamically choose an algorithm among those available. We observed the deterioration of *SEA* performance with time. The modification we brought to *SEA* improved its reactivity and its overall performance.

Acknowledgment. The PhD thesis of Alexandre Dambreville is financed by Labex Digicosme within the project E-CloViS (Energy-aware resource allocation for Cloud Virtual Services).

References

1. Bubeck, S., Cesa-Bianchi, N.: Regret analysis of stochastic and nonstochastic multi-armed bandit problems. Found. Trends Mach. Learn. **5**, 1–122 (2012)
2. http://ita.ee.lbl.gov/html/contrib/WorldCup.html
3. Farias, D.P.D., Megiddo, N.: Combining Expert Advice in Reactive Environments. J. ACM **53**, 762–799 (2006)