Simulation of the mirror descent algorithm on distributions with different variances

D N Shiyan and A V Kolnogorov
Yaroslav-the-Wise Novgorod State University, 41, ul. B. St. Petersburgskaya, Veliky Novgorod, Russian Federation
Email: dsqq.tm@yandex.ru

Abstract. In this article, we consider the operation of the mirror descent algorithm in a random environment in which the distributions of one-step incomes have close expectations and different variances. Particular attention is paid to the version of the algorithm for batch processing which allows one to achieve better processing time without increasing losses. The results of numerical modeling of the operation of the algorithms are presented.

1. Introduction
Consider the task of processing an incoming data stream, for example, the arrival of computational tasks on a server. Having received the task, the server has to decide which executive machine to send it to. There can be several such machines and each of them works better on tasks of a certain type. We consider the flow of tasks to be uniform but the type of tasks is not known to the server in advance. Choosing a server is considered successful if the executor managed to complete the task at the scheduled time. Thus, the main task of the server is to maximize the number of successful executions when processing a stream of a fixed size.

In the literature, this problem is described as «n-armed bandit» [1–2], where n is the number of available actions. In case of a server, n is the number of different executive machines. Server operation is considered as control in a random environment with n available actions on a fixed horizon.

In this study, we restrict the number of available actions to two and consider a two-alternative random environment. Let’s consider a random process $\xi_n$, $n = 1, \ldots, N$, which values are interpreted as random incomes and depend only on the choice of the current action $x_n$, $x_n \in \{1,2\}$.

We introduce the loss function over the control horizon N as follows

$$L_N = \sum_{n=1}^{N} \left( \max(\rho_n^{(1)}, \rho_n^{(2)}) - \xi_n \right),$$

(1)

where $\mathbf{\rho}$ is the vector of mathematical expectations of incomes when choosing both actions.

In previous papers [3], [5] we considered the case of binary incomes, known as the Bernoulli distribution. Upon successful processing, we get 1, otherwise 0, i.e.,

$$\Pr(\xi_n = 1 | x_n = l) = \rho_l, \quad \Pr(\xi_n = 0 | x_n = l) = 1 - \rho_l, \quad l = 1,2$$

(2)

It turned out in [3], [5] that one has to study operation of the algorithms only at close values of the mathematical expectations of both actions. Otherwise, determining the best action is not difficult. Hence,
when modeling, we considered the case when probabilities of success in the choice of actions differ by a value \( d/\sqrt{N} \) with \( d > 0 \). Namely, the vector \( \overline{\rho} \) was presented in the following form

\[
\overline{\rho} = (\rho_1, \rho_2) = (p + d\sqrt{D/N}, p - d\sqrt{D/N}), \quad 0 < p < 1,
\]

where \( D = 0.25 \) is the maximum possible variance of one-step income of Bernoulli two-armed bandit.

In this case, with close mathematical expectations, we had almost equal variances of distributions of incomes for both actions. However, in real-world problems, variances can vary. In this study, we consider the operation of algorithms on specially selected distributions of incomes with close mathematical expectations but with different variances. A description of these distributions can be seen in section 4.

Section 2 describes the mirror descent algorithm for one-by-one data processing (Algorithm 1) for the task of minimizing the loss function. Section 3 provides two versions of the algorithm for batch processing (Algorithm 2 and Algorithm 3). Section 5 contains the results of numerical modeling of the operation of algorithms on various distributions of incomes.

2. Algorithm for one-by-one data processing

This algorithm is based on the algorithm from [4] but has some differences. In particular, the problem statement of the algorithm differs.

Let \( \overline{\rho}_n \) be the vector of probabilities, \( \overline{\xi}_n \) be the dual vector. The two-component vector \( \overline{\rho}_n \) contains an estimate of the probability of choosing the first or the second action, obtained by the algorithm at step \( n \). The vector \( \overline{\xi}_n \) is used to recalculate the values of \( \overline{\rho}_n \) at the next step. Set the initial values \( \overline{\rho}_0 = (0.5; 0.5) \) and \( \overline{\xi}_0 = (0; 0) \).

**Algorithm 1**

For all \( n = 1, \ldots, N \) perform the following actions:

1) Choose the action \( y_n \) by vector \( \overline{\rho}_{n-1} \):

\[
P(y_n = l) = p^{(l)}_{n-1}, \quad l = 1, 2;
\]

2) Get the amount of income \( \xi_n \) for the use of action \( y_n \)

\[
P(\xi_n = 1 | y_n = l) = \rho_l, \quad P(\xi_n = 0 | y_n = l) = 1 - \rho_l; 
\]

3) Calculate the stochastic gradient \( \overline{u}_n(\overline{\rho}_{n-1}) \) according to the following formula

\[
\overline{u}_n(\overline{\rho}_{n-1}) = \begin{cases} 
1 - \xi_n, & y_n = 1, \\
\frac{1}{\rho_n^{(1)}}, & y_n = 2;
\end{cases}
\]

4) Update vectors \( \overline{\xi}_n \) and \( \overline{\rho}_n \)

\[
\overline{\xi}_n = \overline{\xi}_{n-1} + \overline{u}_n(\overline{\rho}_{n-1}),
\]

\[
\overline{\rho}_n = \overline{G}_\beta(\overline{\xi}_n),
\]

where \( \beta_n = \beta_0\sqrt{D(n+1)}, \quad \beta_0 > 0 \) is a custom parameter,

\[
\overline{G}_\beta(\overline{\xi}_n) = \frac{1}{S_\beta(\overline{\xi}_n)} \left( e^{-\xi_n^{(1)}/\beta} + e^{-\xi_n^{(2)}/\beta} \right), \quad S_\beta(\overline{\xi}_n) = e^{-\xi_n^{(1)}/\beta} + e^{-\xi_n^{(2)}/\beta}.
\]

For the resulting loss function, normalization is performed as follows:
3. Algorithms for batch processing

If one sends tasks to executors in groups, he (or she) can get a significant gain in processing speed. In batch processing, the overall control horizon is divided into \( T \) stages, \( N = TM \), where \( M \) is the size of the group. Thus, the total processing time no longer depends on the control horizon but depends only on the number of stages \( T \). The effectiveness of the method also increases when using the same type of executors with the possibility of parallel execution.

Consider the first version of the algorithm for batch processing. The probability of applying a specific action to each task in the group is selected by the value of the vector \( p_{n-1} \). Only after processing the whole group the value of \( p_n \) is updated and is then used at the next stage.

**Algorithm 2**

For all \( t = 1, \ldots, T \) perform the following actions:

1) We introduce the vector \( \chi_t = (\chi_t^{(1)}, \chi_t^{(2)}) = (0,0) \).

2) Processing group \( t \). For all \( n = (t-1)M + 1, \ldots, tM \):
   a) Choose the action of \( y_n \) by vector \( \bar{p}_{n-1} \): \( P(y_n = l) = p_{n-1}^{(l)} \), \( l = 1, 2 \);
   b) Get the amount of income \( \xi_n \) for the use of action \( y_n \) using (4);
   c) Update vector \( \chi_t = \chi_t^{(n)} = \chi_t^{(n)} + (1 - \xi_n) \).

3) Calculate the stochastic gradient \( \bar{u}_t(\bar{p}_{t-1}) \) according to the following formula

\[
\bar{u}_t(\bar{p}_{t-1}) = \left( \frac{\chi_t^{(1)}}{p_{t-1}}, \frac{\chi_t^{(2)}}{p_{t-1}} \right)
\]

4) Update vectors \( \bar{\chi}_t \) and \( \bar{p}_t \):

\[
\begin{align*}
\bar{\chi}_t &= \bar{\chi}_{t-1} + \bar{u}_t(\bar{p}_{t-1}), \\
\bar{p}_t &= G_{\beta_t}(\bar{\chi}_t),
\end{align*}
\]

where \( \beta_t = \beta_0 \sqrt{DM(t + 0.5)} \).

Loss normalization function is the following

\[
L^{(2)}_N = \frac{L_N}{\sqrt{DN}}
\]

Let’s consider one more variant of the algorithm for group processing. It involves dividing the group into two parts, in proportion to the current value of \( p_n \), and then performs the corresponding action for each of the parts.

Let \( \bar{M}_t = (M_t^{(1)}; M_t^{(2)}) \), where \( M_t^{(1)} > 0, M_t^{(2)} > 0 \) and \( M_t^{(1)} + M_t^{(2)} = M \). Denote by \( [M_t^{(1)}], [M_t^{(2)}] \) the nearest integers to \( M_t^{(1)}, M_t^{(2)} \).

**Algorithm 3**

For all \( t = 1, \ldots, T \) perform the following actions:

1) Calculate for which part of the batch the first action should be used and for which the second one \( M_t^{(l)} = p_{t-1}^{(l)} \times M, l = 1, 2 \);
2) Apply the \( l \)-th action \([M^{(l)}_t]\) times and calculate the total vector of income \( \vec{\eta}_t \), where \( \xi_n \) is calculated by the expression (4)

\[
\eta^{(l)}_t = \sum_{n=0}^{dM+1} (1 - \xi_n) y_n = l
\]

3) Calculate the stochastic gradient \( \vec{u}_t (\vec{p}_{t-1}) \) according to the following formula

\[
\vec{u}_t (\vec{p}_{t-1}) = \begin{pmatrix} \eta^{(1)}_t \\ \eta^{(2)}_t \end{pmatrix} / \begin{pmatrix} p^{(1)}_{t-1} \\ p^{(2)}_{t-1} \end{pmatrix}
\]

4) Update vectors \( \vec{\xi}_t \) and \( \vec{p}_t \)

\[
\vec{\xi}_t = \vec{\xi}_{t-1} + \vec{u}_t (\vec{p}_{t-1}),
\vec{p}_t = \vec{P}_\varepsilon (\vec{p}_t),
\]

where \( \beta_t = \beta_0 \sqrt{DM(t + 0.5)} \) and \( \vec{P}_\varepsilon (\vec{p}_t) \) is a projection operator with some \( \varepsilon > 0 \)

\[
\vec{p}_t = \begin{cases} p'_t, & \text{if } p'^{(1)} \geq \varepsilon, p'^{(2)} \geq \varepsilon, \\ (1, \varepsilon), & \text{if } p'^{(1)} < \varepsilon, \\ (1 - \varepsilon, \varepsilon), & \text{if } p'^{(2)} < \varepsilon. 
\end{cases}
\]

The projection operator is used because \( M^{(l)}_t < 1 \) if corresponding \( p^{(l)}_{t-1} \) is small enough.

The loss normalization function in this case differs from previous algorithms:

\[
L^{(3)}_N = \frac{L_N}{\sqrt{DN}},
\]

where \( D \) is the maximum variance, for the Bernoulli distribution \( \hat{D} = pq, q = 1 - p \).

4. Description of distributions for numerical modeling

For modeling, we use discrete random variables distributed over three points selected in such a way that they have equal mathematical expectation of incomes when both actions are chosen at \( d = 0 \). With increasing \( d \), mathematical expectations gradually change, so that one of the actions is more advantageous but the difference in mathematical expectations is small. Let us denote the introduced random variables as \( W_1 \) and \( W_2 \) (hereinafter, these notations will be used to indicate the types of distributions used). Distributions are presented in tables 1–2.

**Table 1.** Three-point distribution for the first action.

| \( x_i \) | 0 | 0.5 | 1 |
|---|---|---|---|
| \( p_i \) | \( 1 - (p_2 + p_3) \) | \( p_2 = 0.3 + 0.25 \cdot d \sqrt{T} \), \( p_3 = 0.2 + 0.25 \cdot d \sqrt{T} \), | \( p_2 = [0.3...0.55] \), \( p_3 = [0.2...0.45] \) |
Table 2. Three-point distribution for the second action.

| $x_i$ | 0     | 0.5   | 1     |
|-------|-------|-------|-------|
| $p_i$ | $1-(p_2+p_3)$ | $p_2=0.5-0.5\cdot d\sqrt{T}$, | $p_3=0.1$ |
|       | $p_2=[0...0.5]$ |       |       |

Consider the characteristics of the obtained distributions. Given $d=0$, mathematical expectations of incomes are equal to $E(W_1)=E(W_2)=0.35$, variance and mean squared deviation are equal to $\text{Var}(W_1)=0.1525, \sigma(W_1) \approx 0.39$ for the first action and $\text{Var}(W_2)=0.1025, \sigma(W_2) \approx 0.32$ for the second action. It is easy to show that the maximum values of variances for the distributions $W_1$ and $W_2$ for $d>0$ are $\text{Var}_{\max}(W_1) \approx 0.1569$ and $\text{Var}_{\max}(W_2) \approx 0.1125$ respectively for each action. So $\hat{D}=\text{max}(\text{Var}_{\max}(W_1), \text{Var}_{\max}(W_2))=0.1569$. We also introduce a group of distributions $V_1$ and $V_2$ which values $x_i$ are exactly two times less, therefore, $\frac{E(W_1)}{E(V_1)}=\frac{E(W_2)}{E(V_2)}=2$ and $\frac{\sigma(W_1)}{\sigma(V_1)}=\frac{\sigma(W_2)}{\sigma(V_2)}=2$.

5. Results of numerical simulation
This section presents the results of numerical modeling of the operation of algorithms in various cases. Consider the operation of the Algorithm 3 on distributions with different variances. To this end, we use three types of distributions: Bernoulli ($\sigma_{\max}^B=0.5$), type $W$ ($\sigma_{\max}^W=0.4$) and type $V$ ($\sigma_{\max}^V=0.2$). We use simulation parameters: control horizon $N=10000$, group size $M=50$ and parameter $\beta_0=1.55$.

![Figure 1](image_url) Normalized loss for Algorithm 3 on various types of distributions.

On figure 1 one can see that Algorithm 3 works in a similar way regardless of the variance of the distribution as it was theoretically shown in [6].
Figure 2. Normalized losses for Algorithm 3 with a growing control horizon.

Now we consider the operation of the Algorithm 3 on a growing control horizon with a fixed packet size $M = 100$ and with Bernoulli ($B$) and $V$ distributions.

On figure 2 one can see that Algorithm 3 shows the same losses with a growing control horizon with distributions of different types, and does not depend on their variances, which differ significantly.

\[
\frac{\text{Var}_B}{\text{Var}_V} = 6.25, \quad \frac{\sigma^B_{\text{max}}}{\sigma^V_{\text{max}}} = 2.5.
\]

Figure 3. Normalized losses for Algorithm 2 (A2) for various $M$ and Algorithm 1 (A1).

However, Algorithms 1 and 2 have no longer the property of independence of the resulting losses from the variances for distributions of type $W$ and $V$, as well as for the Bernoulli distribution, Algorithm 2 converges to Algorithm 1 with a decrease in the size of group $M$. The simulation results for this case on distributions of type $W$ with $N = 10000$ and $\beta_0 = 2.25$ are shown on figure 3. The
small difference between Algorithm 2 at $M = 1$ and Algorithm 1 is explained by the difference in the calculation of $\beta_i$.

6. Conclusion
This article examined the operation of the mirror descent algorithm in a random environment in which the distributions of incomes have close mathematical expectations but different variances. Algorithms were presented for one–by–one data processing (Algorithm 1) and for group processing (Algorithm 2 and Algorithm 3).

It was shown that group processing using Algorithm 3 not only allows to optimize the total control time due to the parallel execution of tasks in each group, but also ensures the independence of the resulting losses from the variance of the distribution of income.

Algorithm 2 does not have the independence property of the resulting losses from the variance but it demonstrates convergence to Algorithm 1 with a decrease in the size of the processed group on distributions with different variances in the same way as on Bernoulli distributions.

Acknowledgments
The reported study was funded by RFBR, project no. 20-01-00062.

References
[1] Berry D A and Fristedt B 1985 Bandit Problems: Sequential Allocation of Experiments (Chapman & Hall: London)
[2] Cesa-Bianchi N and Lugosi G 2006 Prediction, Learning, and Games (Cambridge: Cambridge Univ. Press)
[3] Kolnogorov A V and Shiyan D N 2016 Parallel Version of the Mirror Descent Algorithm for the Two-Armed Bandit Problem Proc. of the 3rd International Conference on Mathematics and Computers in Sciences and in Industry (Chania: MCSI) 241–245
[4] Juditsky A, Nazin A V, Tsybakov A B and Vayatis N 2008 Gap-free Bounds for Stochastic Multi-Armed Bandit Proc. of the 17th World Congr. of the International Federation of Automatic Control (Seoul:IFAC) 17 11560–11563
[5] Shiyan D N and Kolnogorov A V 2019 Control in a weakly inhomogeneous two-alternative random environment using the mirror descent algorithm J. Phys.: Conf. Ser. 1352 012048
[6] Kolnogorov A V 2018 Gaussian two-armed bandit and optimization of batch data processing Problems of Information Transmission 54 84–100