Two Approaches on Accelerating Bayesian Two Action Learning Automata

Hao Ge[1], Haiyu Huang[2], Yulin Li[3], Shenghong Li[1(✉)], and Jianhua Li[1]

[1] Shanghai Jiao Tong University, Shanghai, China
{sjtu_gehao, shli, lijh888}@sjtu.edu.cn
[2] The 28th Research Institute of China Electronic Technology Group Corporation, Nanjing, China
abel89@126.com
[3] Dalian No. 24 High School, Dalian, China
yulinlin981@hotmail.com

Abstract. Bayesian Learning Automata (BLA) are demonstrated to be as efficient as the state-of-the-art automaton in two action environments, and it has parameter-free property. However, BLA need the explicit computation of a beta inequality, which is time-consuming, to judge its convergence.

In this paper, the running time of BLA is concerned and two approaches are proposed to accelerate the computation of the beta inequality. One takes advantage of recurrence relation of the beta inequality, the other uses a normal distributions to approximate the beta distributions. Numeric simulation are performed to verify the effectiveness and efficiency of those two approaches. The results shows these two approaches reduce the running time substantially.

Keywords: Bayesian Learning Automata · Recurrence relation · Moment matching · Normal approximation

1 Introduction

Learning Automata (LA) are simple self-adaptive decision units that were firstly investigated to mimic the learning behavior of natural organism [1]. The pioneer work can be traced back to 1960s by the Soviet scholar Tsetlin [2, 3]. Since then, LA has been extensively explored and it is still under investigation as well in methodological aspects [4–9] as in concrete applications [10–17]. One intriguing property that popularize the learning automata based approaches in engineering is that LA can learn the stochastic characteristics of the external environment it interacts with, and maximizing the long term reward it obtains through interacting with the environment.

On the development of learning automata, accuracy and convergence rate becomes two major measurement to evaluate the goodness of an algorithm. The former is defined as the probability of a correct convergence and the latter is defined as the average iterations for a learning automaton to get converged.

Most of the reported schemes in the field of LA has two or more tunable parameters, making themselves capable of adapting to different environments. The performance of

D.-S. Huang et al. (Eds.): ICIC 2016, Part III, LNAI 9773, pp. 239–247, 2016.
DOI: 10.1007/978-3-319-42297-8_23

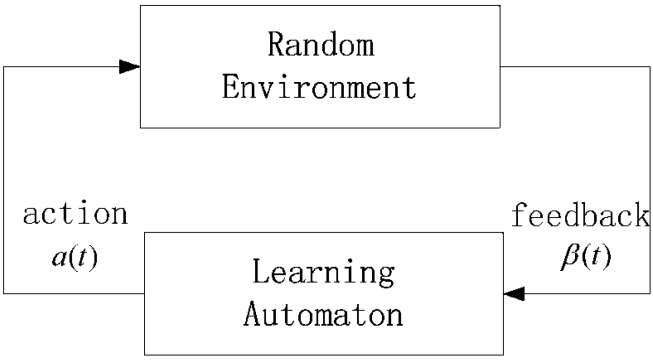

Fig. 1. Feedback connection of automaton and environment.

an automaton is highly dependent on the selection of those parameters. Generally, according to the ε-optimality property of LA, the probability of converging to the optimal action can be arbitrarily close to one. More specifically, as long as the learning resolution is large enough, we can obtain the correct result with an arbitrary accuracy.

Parameter tuning can be a complicated and time consuming procedure, and learning in two action stationary environments is the most fundamental scenario. In order to explore new mechanism to reduce the difficulty in parameter tuning, we took the first step by proposing a parameter-free Bayesian learning automata in two action environment [18]. The proposed BLA achieves the same accuracy and convergence rate as the sate-of-art automaton, besides, it holds the parameter-free property.

The parameter-free property is defined as a property that the performance of a automaton are independent of the selection of parameters, or a set of selected parameters can be universally applied to all environments. This implies the procedure can be omitted without scarifying accuracy and convergence rate. However, BLA also suffers from a heavy computational burden due to the explicit computation of a beta inequality.

In this paper, we propose two approaches to reduce the computational burden. One takes advantage of recurrence relation of the beta inequality, the other uses a normal distributions to approximate the beta distributions. Then we will verify the two approaches via simulations and analyze the simulation results.

The rest of this paper organized as follows. Section 2 introduces the BLA briefly, points out its drawback and advocates two possible ways to overcome it. Extensive simulations are given in Sects. 3 and 4 concludes the paper.

2 Parameter-Free Two Action Learning Automaton

First, we shall introduce the algorithm we proposed in [18] briefly.

Algorithm Bayesian Learning Automaton

1: Initial $\alpha_1 = 1, \beta_1 = 1, \alpha_2 = 1, \beta_2 = 1$
2: **Repeat**
3: Compute the probability $P_1 = Pr(e_1 > e_2)$, where e_1 and e_2 are random variables that follow distribution $Beta(\alpha_1, \beta_1)$ and $Beta(\alpha_2, \beta_2)$ respectively, and $P_2 = 1 - P_1$.
4: Choose an action α_i according to equation (1).
5: Receive a feedback from the environment and update the parameter of Beta distributions: $\alpha_i = \alpha_i + 1$ if a reward is received and $\beta_i = \beta_i + 1$ if a penalty is received.
6: **Until** $max(P_1, P_2) > 0.99$

$$a_i = \begin{cases} argmin_i(\alpha_i + \beta_i) & \text{when } \mathrm{P}_1 \neq \mathrm{P}_2 \\ \text{randomly chosen} & \text{when } \mathrm{P}_1 = \mathrm{P}_2 \end{cases} \tag{1}$$

We may notice the exact value of probability $P_1 = Pr(e_1 > e_2)$ is evaluated in step 3 and utilized in step 6 to judge if the convergence criteria is satisfied. The computation of probability $P_1 = Pr(e_1 > e_2)$ is implemented by the following equations.

If we regard $P_1 = Pr(e_1 > e_2)$ as a function of $\alpha_1, \beta_1, \alpha_2, \beta_2$ and denote it as $g(\alpha_1, \beta_1, \alpha_2, \beta_2)$, we have:

$$g(\alpha_1, \beta_1, \alpha_2, \beta_2)$$

$$= \sum_{i=0}^{\alpha_1 - 1} \frac{B(\alpha_2 + i, \beta_1 + \beta_2)}{(\beta_1 + i)B(1 + i, \beta_1)B(\alpha_2, \beta_2)} \tag{2}$$

$$= \sum_{i=0}^{\beta_2 - 1} \frac{B(\beta_1 + i, \alpha_1 + \alpha_2)}{(\alpha_2 + i)B(1 + i, \alpha_2)B(\alpha_1, \beta_1)} \tag{3}$$

$$= 1 - \sum_{i=0}^{\alpha_2 - 1} \frac{B(\alpha_1 + i, \beta_1 + \beta_2)}{(\beta_2 + i)B(1 + i, \beta_2)B(\alpha_1, \beta_1)} \tag{4}$$

$$= 1 - \sum_{i=0}^{\beta_1 - 1} \frac{B(\beta_2 + i, \alpha_1 + \alpha_2)}{(\alpha_1 + i)B(1 + i, \alpha_1)B(\alpha_2, \beta_2)} \tag{5}$$

The above four equivalent equations indicate that the $g(\alpha_1, \beta_1, \alpha_2, \beta_2)$ can be evaluated with $\mathcal{O}(\min(\alpha_1, \alpha_2, \beta_1, \beta_2))$ by any programming language where the log-beta function is well defined. However, it's still can be improved through the following two approaches.

2.1 Recurrence Relationship

The recurrence relations that given in [19] are:

$$g(\alpha_1+1,\beta_1,\alpha_2,\beta_2)=g(\alpha_1,\beta_1,\alpha_2,\beta_2)+\frac{h(\alpha_1,\beta_1,\alpha_2,\beta_2)}{\alpha_1} \tag{6}$$

$$g(\alpha_1,\beta_1+1,\alpha_2,\beta_2)=g(\alpha_1,\beta_1,\alpha_2,\beta_2)-\frac{h(\alpha_1,\beta_1,\alpha_2,\beta_2)}{\beta_1} \tag{7}$$

$$g(\alpha_1,\beta_1,\alpha_2+1,\beta_2)=g(\alpha_1,\beta_1,\alpha_2,\beta_2)-\frac{h(\alpha_1,\beta_1,\alpha_2,\beta_2)}{\alpha_2} \tag{8}$$

$$g(\alpha_1,\beta_1,\alpha_2,\beta_2+1)=g(\alpha_1,\beta_1,\alpha_2,\beta_2)+\frac{h(\alpha_1,\beta_1,\alpha_2,\beta_2)}{\beta_2} \tag{9}$$

where $h(\alpha_1,\beta_1,\alpha_2,\beta_2)=\frac{B(\alpha_1+\alpha_2,\beta_1+\beta_2)}{B(\alpha_1,\beta_1)B(\alpha_2,\beta_2)}$.

Since one of the four equation can be used to update $g(\alpha_1,\beta_1,\alpha_2,\beta_2)$ after each iteration, and $\alpha_1,\ \beta_1,\ \alpha_2,\ \beta_2$ are all initialized as one according to the algorithm. So we can simply compute any $g(\alpha_1,\beta_1,\alpha_2,\beta_2)$ iteratively starting from $g(1,1,1,1)=0.5$. A demo of learning process is illustrated in Fig. 2. In Fig. 2, the red-colored number means it changes at the corresponding time instance. Since the terms with green background are known at previous time instance, so only the term with yellow background should be evaluate to get a new value that may be used in the next time instance. By such way, it is obvious that $g(\alpha_1,\beta_1,\alpha_2,\beta_2)$ could be computed with $\mathcal{O}(1)$ by any programming language where the log-beta function is well defined. That is doubtlessly a big improvement and can be expected to reduce the computational burden significantly.

2.2 Normal Approximation

Normal distribution, also known as Gaussian distribution, is a very common continuous probability distribution in probability theory. As advocated in [20], beta distribution may sometimes be approximated by a normal distribution and this approximation becomes exact asymptotically as the beta distribution parameters increase.

The technique used here is called moment matching, which maps a general probability distribution $\mathcal{G}_1$ to another general probability distribution $\mathcal{G}_2$. The two probability distributions share the same moment (mean, variance, etc.).

So the key idea is $\Pr(X_B > Y_B) \approx \Pr(X_N > Y_N)$, where X_B and Y_B are independent beta random variables X_N and Y_N their corresponding normal approximations.

We denote the shared mean of X_B and X_N as μ_X, then

$$\mu_X=\frac{\alpha_X}{\alpha_X+\beta_X} \tag{10}$$

And we denote the shared variance of X_B and X_N as σ_X^2, then

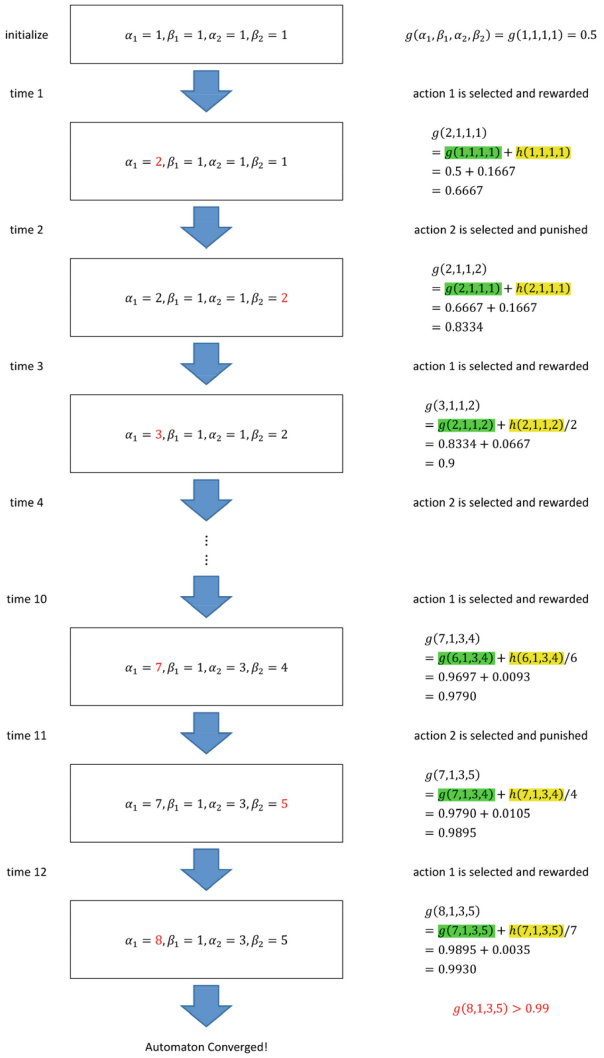

Fig. 2. Demonstration of a learning process that utilizing recurrence relationship (Color figure online)

$$\sigma_X^2 = \frac{\alpha_X \beta_X}{(\alpha_X + \beta_X)^2(\alpha_X + \beta_X + 1)} \tag{11}$$

Then

$$\Pr(X_N > Y_N) = \Phi\left(\frac{\mu_X - \mu_Y}{(\sigma_X^2 + \sigma_Y^2)^{1/2}}\right) \tag{12}$$

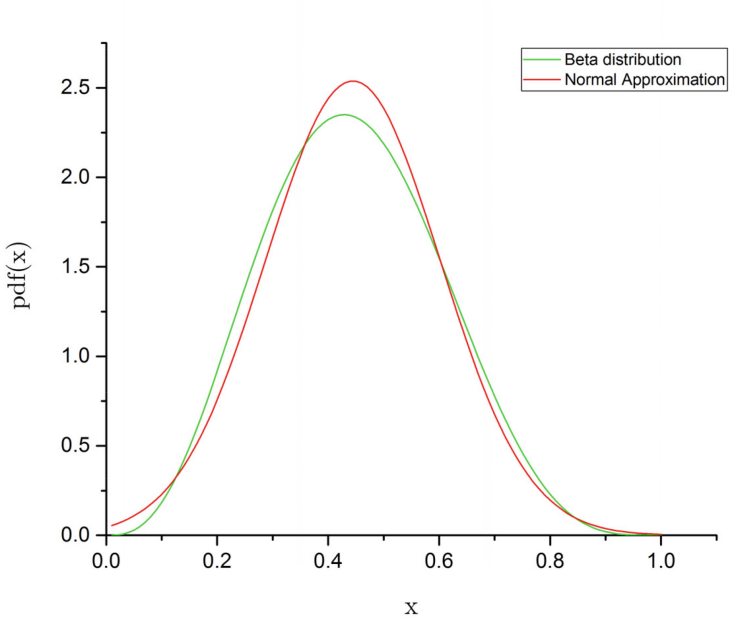

Fig. 3. The probability density function (pdf) of Beta distribution with $\alpha = 4,\ \beta = 5$ and the pdf its corresponding normal approximation. (Color figure online)

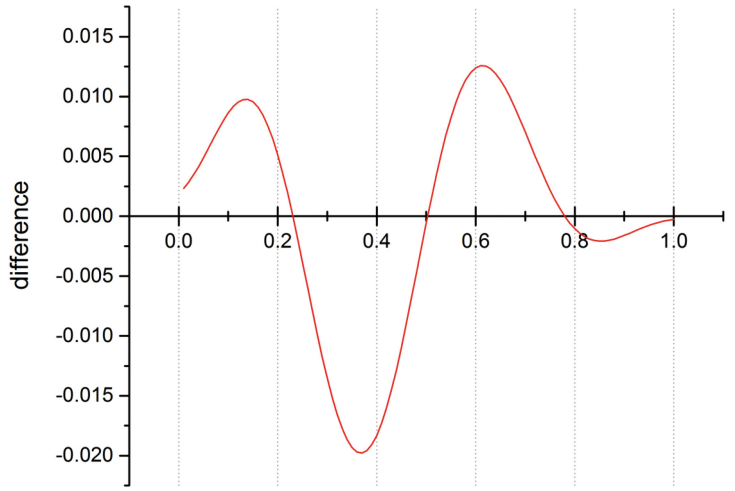

Fig. 4. The difference between the cumulative probability functions of beta distribution with $\alpha = 4,\ \beta = 5$ and its normal approximation

Where Φ denote the cumulative distribution function of the standard normal distribution. As $\Phi(x) = \frac{1}{2}\left[1 + erf\left(\frac{x}{\sqrt{2}}\right)\right]$, so the target formula could be computed by using the relative error function.

It was reported by Cook [20] that if the distribution parameters of X_B and Y_B take on integer values between 1 and 10 inclusive, the maximum absolute error is 0.05069 and the average absolute error over the parameter values is 0.006676. Figure 3 illustrates the probability density functions of a beta distribution and its normal approximation and Fig. 4 shows the differences between the cumulative probability function of the two distributions. It's worth mentioning that the difference will approaches zero asymptotically as the beta distribution parameters increase. That's the foundation of an accurate convergence.

3 Experiments

In this section, simulations are performed to verify the effectiveness of the two approaches. To make a comparison, the environments are the same as environments used in [18], which are:

E1:{0.90, 0.60}.
E2:{0.80, 0.50}.
E3:{0.80, 0.60}.
E4:{0.20, 0.50}.

The iterations and accuracy (number of correctly converged/number of experiments) of each algorithm are summarized in Table 1, where 250000 independent simulations are performed to get averaged iterations and accuracy.

The simulation codes are programmed by C++ with Boost library, which provides the beta random number generator, relative error function, beta function and log-beta function. Then the codes are performed on our workstation, which has Dual-CPU 2.6 GHz Intel Xeon E5-2670 with 4 Gbytes of RAM and a 500 Gbytes hard disk.

The running time, accuracy and averaged iterations of the three different approaches in the four environments are summarized in Tables 1, 2 and 3, respectively.

Table 1. The running time of 250000 independent simulations in E1 to E4

Environment	Original approach	Recurrence relation approach	Normal approximation approach
E_1	151.508 s	0.945 s	0.377 s
E_2	342.406 s	1.123 s	0.399 s
E_3	3004.4 s	2.19 s	0.725 s
E_4	493.156 s	1.113 s	0.395 s

From the tables, we can see Normal Approximation approach needs the least interactions with the environments, but at a cost of a small decline in accuracy. One distinct advantage is that it's about 401 to 4144 times faster than original approach with respect to running time.

Table 2. The accuracy of 250000 independent simulations in E1 to E4

Environment	Original approach	Recurrence relation approach	Normal approximation approach
E_1	99.9 %	99.9 %	99.9 %
E_2	99.9 %	99.9 %	99.8 %
E_3	99.7 %	99.7 %	99.5 %
E_4	99.9 %	99.9 %	99.7 %

Table 3. The averaged iterations to get converged in E1 to E4

Environment	Original approach	Recurrence relation approach	Normal approximation approach
E_1	42	42	40
E_2	49	49	45
E_3	99	99	93
E_4	49	49	45

Recurrence Relation approach keeps the same accuracy and required interactions with original approach, but still be about 160 to 1371 times faster than original approach.

4 Conclusion

In this paper, two approaches for reducing the computation time is proposed and experimentally verified. The simulation results shows both the two approaches are faster than the original approach, one with a slight loss in accuracy but another without any loss. These two approaches are expected to enhance the applicability of BLA in engineering.

However, the problem still remains open, problems such as how can the scheme be extended to be applicable in multi-action environments are supposed to be our future work.

Acknowledgement. This research work is funded by the National Science Foundation of China (61271316), 973 Program of China (2013CB329605), Key Laboratory for Shanghai Integrated Information Security Management Technology Research, and Chinese National Engineering Laboratory for Information Content Analysis Technology.

References

1. Narendra, K.S., Thathachar, M.: Learning automata-a survey. IEEE Trans. Syst. Man Cybern. **4**, 323–334 (1974)
2. Tsetlin, M.L.: On the behavior of finite automata in random media. Avtomatika i Telemekhanika **22**, 1345–1354 (1961)

3. Tsetlin, M.: Automaton Theory and Modeling of Biological Systems. Academic Press, New York (1973)
4. Agache, M., Oommen, B.J.: Generalized pursuit learning schemes: new families of continuous and discretized learning automata. IEEE Trans. Syst. Man Cybern. Part B Cybern. **32**(6), 738–749 (2002)
5. Papadimitriou, G.I., Sklira, M., Pomportsis, A.S.: A new class of ε-optimal learning automata. IEEE Trans. Syst. Man Cybern. Part B Cybern. **34**(1), 246–254 (2004)
6. Zhang, X., Granmo, O.-C., Oommen, B.J.: On incorporating the paradigms of discretization and bayesian estimation to create a new family of pursuit learning automata. Appl. Intell. **39** (4), 782–792 (2013)
7. Zhang, J., Wang, C., Zhou, M.: Last-position elimination-based learning automata. IEEE Trans. Cybern. **44**(12), 2484–2492 (2014)
8. Ge, H., Jiang, W., Li, S., Li, J., Wang, Y., Jing, Y.: A novel estimator based learning automata algorithm. Appl. Intell. **42**(2), 262–275 (2015)
9. Jiang, W., Li, B., Tang, Y., Philip Chen, C.L.: A new prospective for learning automata: a machine learning approach. Neurocomputing (2015)
10. Song, Y., Fang, Y., Zhang, Y.: Stochastic channel selection in cognitive radio networks. In: IEEE Global Telecommunications Conference, GLOBECOM 2007, pp. 4878–4882, November 2007
11. Oommen, B., Hashem, M.: Modeling a student-classroom interaction in a tutorial-like system using learning automata. IEEE Trans. Syst. Man Cybern. Part B Cybern. **40**(1), 29–42 (2010)
12. Horn, G., Oommen, B.: Solving multiconstraint assignment problems using learning automata. IEEE Trans. Syst. Man Cybern. Part B Cybern. **40**(1), 6–18 (2010)
13. Cuevas, E., Wario, F., Zaldivar, D., Perez-Cisneros, M.: Circle detection on images using learning automata. IET Comput. Vision **6**(2), 121–132 (2012)
14. Yazidi, A., Granmo, O.-C., Oommen, B.: Learning automaton based online discovery and tracking of spatiotemporal event patterns. IEEE Trans. Cybern. **43**(3), 1118–1130 (2013)
15. Misra, S., Krishna, P., Kalaiselvan, K., Saritha, V., Obaidat, M.: Learning automata based QoS framework for cloud IaaS. IEEE Trans. Netw. Serv. Manage. **11**(1), 15–24 (2014)
16. Kumar, N., Misra, S., Obaidat, M.: Collaborative learning automata based routing for rescue operations in dense urban regions using vehicular sensor networks. IEEE Syst. J. **9**(3), 1081–1090 (2015)
17. Vahidipour, S., Meybodi, M., Esnaashari, M.: Learning automata based adaptive Petri net and its application to priority assignment in queuing systems with unknown parameters. IEEE Trans. Syst. Man Cybern. Syst. **45**(10), 1373–1384 (2015)
18. Ge, H., Yan, Y., Li, J., Guo, Y., Li, S.: A parameter-free gradient Bayesian two-action learning automaton scheme. In: Proceedings of International Conference on Communications Signal Processing and Systems (2015)
19. Cook, J.D.: Exact calculation of beta inequalities. Technical report, UT MD Anderson Cancer Center Department of Biostatistics, Technical report (2005)
20. Cook, J.D.: Fast approximation of Beta inequalities. Technical report, UT MD Anderson Cancer Center Department of Biostatistics, Technical report (2012)