

Learning Optimal Parameter Values in Dynamic Environment: An Experiment with Softmax Reinforcement Learning Algorithm

Riyaz T. Sikora
University of Texas at Arlington
rsikora@uta.edu

1. Introduction

Many learning and heuristic search algorithms require tuning of parameters to achieve optimum performance. In stationary and deterministic problem domains this is usually achieved through off-line sensitivity analysis. However, this method breaks down in non-stationary and non-deterministic environments, where the optimal set of values for the parameters keep changing over time. What is needed in such scenarios is a meta-learning (ML) mechanism that can learn the optimal set of parameters on-line while the learning algorithm is trying to learn its target concept.

In this paper we present a simple meta-learning algorithm to learn the temperature parameter of the Softmax reinforcement-learning (RL) algorithm. We test the effectiveness of this meta-learning algorithm in two domains. The first is the classic reinforcement learning problem known as the k -armed bandit problem. The second domain is a stylized problem in e-procurement. It involves a context of strategic interaction consisting of homogeneous sellers of a single raw material or component vying for business from a single buyer. The sellers are modeled as artificial agents that learn increasingly effective bidding strategies.

We model non-stationarity in the first domain of k -armed bandit problem by periodically switching the reward distributions. The second domain is in effect a non-stationary and non-deterministic learning problem since it is a game of strategic interaction involving multiple agents. We model one of the sellers as using the Softmax RL algorithm. We consider the other seller to be using one of several different learning algorithms. We show that the best value of the temperature parameter for the Softmax agent varies depending on the learning algorithm used by the other seller. In both domains we show the improvement in performance brought on by the use of meta-learning.

2. Reinforcement Learning using Softmax Action-Selection Method

RL methods involve learning what to do so as to maximize (future) payoffs. RL agents use trial and error in formulating the best actions (i.e., learning strategies) to take based on the past rewards received for different actions. Sutton and Barto (1998) provides an excellent overview of RL. All RL methods are based on estimating action-values $Q(a)$ - i.e., the estimated reward for taking action a . In our experiments we use the time-weighted method where the action value estimates are the sample average of past observed rewards, with recent rewards getting a higher weight. If an action a has been taken k times in the past then the $(k+1)^{\text{th}}$ estimate of the action-value will be given by the following:

$$\begin{aligned} Q_{k+1}(a) &= \mathbf{a} \sum_{i=1}^{k+1} r_i(a) \\ &= Q_k(a) + \mathbf{a} [r_{k+1}(a) - Q_k(a)] \end{aligned} \quad (1)$$

where $r_i(a)$ is the actual reward received when taking action a for the i^{th} time, and \mathbf{a} is the learning rate. Note that the RL agent has to keep track of only Q_k for each action.

There are several action-selection methods that an RL agent can use in deciding the next action to take based on the estimated action-values. Each method assigns a probability distribution over the actions based on the estimated action-values. We consider the Softmax action-selection method (Sutton and Barto, 1998). Softmax uses a Gibbs or Boltzmann distribution given below.

$$\Pr(a) = \frac{e^{Q(a)/t}}{\sum_{i=1}^n e^{Q(i)/t}} \quad \text{where } t > 0 \quad (2)$$

All the RL methods are biased by their initial estimates of Q_0 . However, for the time-weighted average action-value method that we use, this bias decreases over time and eventually disappears. The initial action value can still be useful in encouraging exploration in the beginning if they are selected optimistically. For all our experiments, we therefore use an optimistic initial value of Q_0 . For the Softmax method the parameter t is known as temperature. At higher temperatures the method becomes a random search with all the actions selected equiprobably. At very low temperatures the method approaches a greedy search where it always selects the best action. Ideally in a static environment, the Softmax method should start out with a high temperature value so that it can explore all the actions, but over time it should slowly reduce the temperature so that it can exploit the best actions learned. However, as mentioned earlier, in dynamic and non-stationary environments the optimal temperature parameter might not remain static.

3. Meta-learning

Meta-learning concerns itself with improving the performance of a learning system for a particular task through experience with a family of related tasks (Vilalta and Drissi, 2002). In our case we want to improve the performance of the softmax RL method by dynamically learning the best value of the temperature parameter. The softmax learning algorithm, like all other RL methods, uses another parameter called the learning rate (α in eq. 1). In this paper our focus is only on learning the temperature parameter. However, rather than use a single value of learning rate we introduce the concept of using a variable learning rate that switches between a high and a low value. The main idea behind this principle is to learn quickly while losing and slowly while winning. This helps in convergence by giving more time for the other players to adapt to changes in the player's strategy that at first appear beneficial, while allowing the player to adapt more quickly to other players' strategy changes when they are harmful. The modified form of action-value estimation using the variable rate is given below.

$$\begin{aligned} Q_{k+1}(a) &= Q_k(a) + \alpha_{low}[r_{k+1}(a) - Q_k(a)] \text{ if } r_{k+1}(a) > Q_k(a) \\ &= Q_k(a) + \alpha_{high}[r_{k+1}(a) - Q_k(a)] \text{ else} \end{aligned} \quad (3)$$

For meta-learning of the temperature parameter τ , we apply a second layer of another RL method called ϵ -greedy. We consider a set of k intervals for the values of τ . An action in this instance then involves selecting the appropriate interval from which the value of τ is chosen. In ϵ -greedy, the best action is taken most of the time (i.e., with probability $1-\epsilon$), but with a small probability (ϵ) any of the remaining k actions are chosen randomly.

$$\begin{aligned} \Pr(a) &= (1 - \epsilon) \quad \text{if } a = \arg \max_i (Q(i)) \\ &= \frac{\epsilon}{k-1} \quad \text{else} \end{aligned} \quad (4)$$

Since each reward received after an episode of the game is dependent on the choice of τ as well as the action a taken by the learning system, we maintain a dual set of action value estimate vectors $Q(\tau, a)$ and $Q(\tau)$. Both are updated using the eq. (3) above. $Q(\tau, a)$ is used in selecting the action a , where as $Q(\tau)$ is used in selecting the temperature τ . The details of the meta-learning algorithm are given in figure 1.

4. Results & Discussion

We test the performance of the meta-learning algorithm in the following two problem domains.

***k*-Armed Bandit Problem**

The first domain in which we study the effectiveness of using meta-learning in setting the temperature parameter of the softmax RL algorithm is the k -armed bandit problem that has been used extensively in studying various RL algorithms. There are k actions available to an agent and each action returns a reward from a different distribution. The goal is to maximize the rewards through repeated plays. For our experiments, we use a 10-armed bandit problem. Action i ($i = 1 \dots 10$) returns a reward from a Normal distribution with a mean of $1000+100i$ and a standard deviation of 100 i.e., $N(1000+100i, 100)$.

We test the performance of the softmax algorithm using the temperature parameter values of 5, 50, and 500. For the meta-learning version of the softmax algorithm we use the following 4 intervals for learning the temperature parameter: [4.0, 6.0] [49.0, 51.0] [99.0, 101.0] and [499.0, 501.0]. To test the effectiveness of the meta-learning algorithm we introduce non-stationarity by rotating the reward distributions among the 10 actions. We control the dynamism of the changing environment by varying the frequency with which the environment changes. All experiments are repeated 10 times with different random seeds and are run for 3000 iterations.

The results containing the average reward and the standard deviation from the last 1000 iterations are presented in table 1. For stationary problems in general a low value τ does best since it exploits the best action a lot more. Since the best action remains static in stationary environments, the learning algorithm with low value of τ is quickly able to learn the best action and exploit it. As can be seen from the table 1, softmax with $\tau = 5$ performs the best. For the stationary environment the performance of $\tau = 5$ approaches that of the best action with an average reward of 2000 and a std. deviation of 100. As the environment becomes more and more dynamic these returns keep falling as it takes some time for all the algorithms to learn the new best action. In contrast, the performance of the meta-learning (ML) algorithm not only approaches that of $\tau = 5$ when the environment is stationary, but it returns better rewards as the environment becomes more dynamic. Since ML is not tied down to a single value of τ , it is able to respond faster to the changes in the environment.

Learning Bidding Strategies in an e-Procurement Problem

For the second problem domain we consider a modified version of the e-procurement problem presented in (Bandyopadhyay *et al.* 2005). The problem involves 2 homogeneous sellers x and y , each with production capacity of k and a variable cost of c , vying for business from a single buyer with a reserve price of r and a total demand of Q such that the following two conditions are satisfied: $Q > k$ and $2k > Q$. The seller bidding the lowest price sells to capacity with the residual demand going to the seller with the higher-bid. We modify the problem by splitting the effective price range $[c, r]$ into n equally sized price bands. The problem can now be analyzed as a discrete two-player symmetric game with n actions available to each player (i.e., an n -armed bandit problem). Each action involves selecting a bid price from the respective price band with a uniform distribution. Sikora & Sachdev (2005) show that the Nash equilibrium for such a problem is unstable and present results with various algorithms trying to learn the best bidding strategies.

We consider the following values for our experiments: $Q = 100$, $k = 65$, $c = 40$, $r = 80$, $n = 10$. We model one of the sellers as using the Softmax RL algorithm. We consider two stationary environments where the other player uses a fixed strategy (i.e., always bids from the same highest or lowest price band). To simulate non-stationary environment, we consider the other seller to be using one of several different learning algorithms. The learning algorithms we consider in this paper are ϵ -greedy, a genetic algorithm (Goldberg, 1989), and the same softmax algorithm used by the first seller. Note that a game of strategic interaction involving two or more agents is in effect a problem of learning a “moving target.” Since the other agent in our problem is also adapting, the “best response” or the optimal policy may be changing as the agent is learning.

The results containing the average reward and the standard deviation from the last 1000 iterations are presented in table 2. As expected, softmax with $\tau = 5$ does best in stationary environments. However, no single value of τ does best in the non-stationary environment. The meta-learning algorithm, on the other hand, tries to learn the best value of τ while it is learning the bidding strategy, and it approaches the

performance of the best τ in both stationary and non-stationary environments. Figure 2 presents the spread (i.e., high, average, and low values) of the rewards earned in each scenario. Only one figure is presented from the two stationary scenarios as the spreads for both were similar. It is clear that the performance of the meta-learning algorithm approaches that of the best value of τ in all the scenarios.

5. Conclusions

Fine tuning or optimal setting of parameters has been the Achilles heel of many heuristic search and learning algorithms. Many applications involving dynamic environments do not afford the luxury of using off-line sensitivity analysis to set the parameters values. Instead a meta-learning approach is required that can learn the best values of the parameters to use on-line while the search and learning algorithm is performing its task. In this paper we presented such a meta-learning approach for learning the temperature parameter of the softmax reinforcement learning algorithm. We presented results from two different domains to show the efficacy of this approach.

References

- Bandyopadhyay, S., Barron, J.M., Chaturvedi, A.R., "Competition Among Sellers in Online Exchanges," *Information Systems Research*, vol. 16 (1), 47 – 60, 2005.
- Goldberg, D. *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison Wesley, 1989
- Sikora, R. and Sachdev, V., "Learning Optimal Seller Strategies with Intelligent Agents: Application of Evolutionary and Reinforcement Learning," *Proc. of the 15th Workshop on Information Technologies and Systems*, pp. 75-80, Las Vegas, NV, December 10-11 2005
- Sutton, R.S. and Barto, A.G. *Reinforcement Learning: An Introduction*, MIT Press, 1998.
- Vilalta, R. and Drissi, Y., "A Perspective View and Survey of Meta-Learning," *Artificial Intelligence Review*, pp. 77-95, 18 (2002).

	Stationary environment	Non-stationary environment with changes every		
		100 trials	50 trials	10 trials
$\tau \neq 5$	1999.74 (100.93)	1933.90 (179.26)	1874.76 (233.72)	1666.54 (358.85)
$\tau = 50$	1979.67 (115.51)	1908.30 (230.06)	1840.62 (289.54)	1648.52 (374.61)
$\tau = 500$	1704.19 (279.44)	1648.91 (319.78)	1605.24 (333.36)	1552.41 (314.47)
ML	1997.97 (104.74)	1933.16 (188.17)	1887.17 (234.93)	1706.52 (329.71)

Table 1. Average reward realized for the 10-armed bandit problem during the last 1000 trials

	Stationary environment		Non-Stationary environment		
	FixedLow	FixedHigh	Self	0.1Greedy	GA
$\tau = 5$	1330.14 (12.86)	2209.77 (23.82)	1888.15 (171.03)	1337.34 (36.60)	1393.34 (100.67)
$\tau = 50$	1311.75 (77.41)	2205.57 (97.08)	1800.52 (224.85)	1640.41 (259.24)	1418.53 (34.40)
$\tau = 500$	1065.79 (85.85)	1854.84 (146.52)	1275.23 (143.56)	1637.02 (230.88)	1328.87 (35.09)
ML	1326.54 (17.95)	2206.36 (29.64)	1802.01 (230.08)	1600.75 (276.02)	1417.30 (37.56)

Table 2. Average reward realized for the e-procurement problem during the last 1000 trials

```

Let
 $N_t$  and  $N_a$  be the number of intervals for the parameters  $\tau$  and  $a$ .
 $Q(\tau, a)$  and  $Q(\tau)$  be the estimated profit vectors.
 $Pr(\tau)$  and  $Pr(a)$  be the probability distributions for  $\tau$  and  $a$  resp.

Initialize
 $Q(\tau, a)$  and  $Q(\tau)$  vectors to optimistic initial values.
 $Pr(\tau)$  and  $Pr(a)$  to  $1/N_t$  and  $1/N_a$  resp.
update the values of  $\tau$  and  $a$  using the above prob. distributions.

repeat {
  play the game using action  $a$  and obtain a reward  $r$ 
  update the profit vectors  $Q(\tau, a)$  and  $Q(\tau)$  using eq. (3)
  every 10 iterations do {
    update  $Pr(\tau)$  using eq. (4) with  $\epsilon = 0.1$ 
    update  $\tau$  using its updated prob. distribution
  }
  update  $Pr(a)$  using eq. (2)
  update  $a$  using its updated prob. distribution
}
until (the stopping criteria is met)

```

Figure 1. The meta-learning algorithm

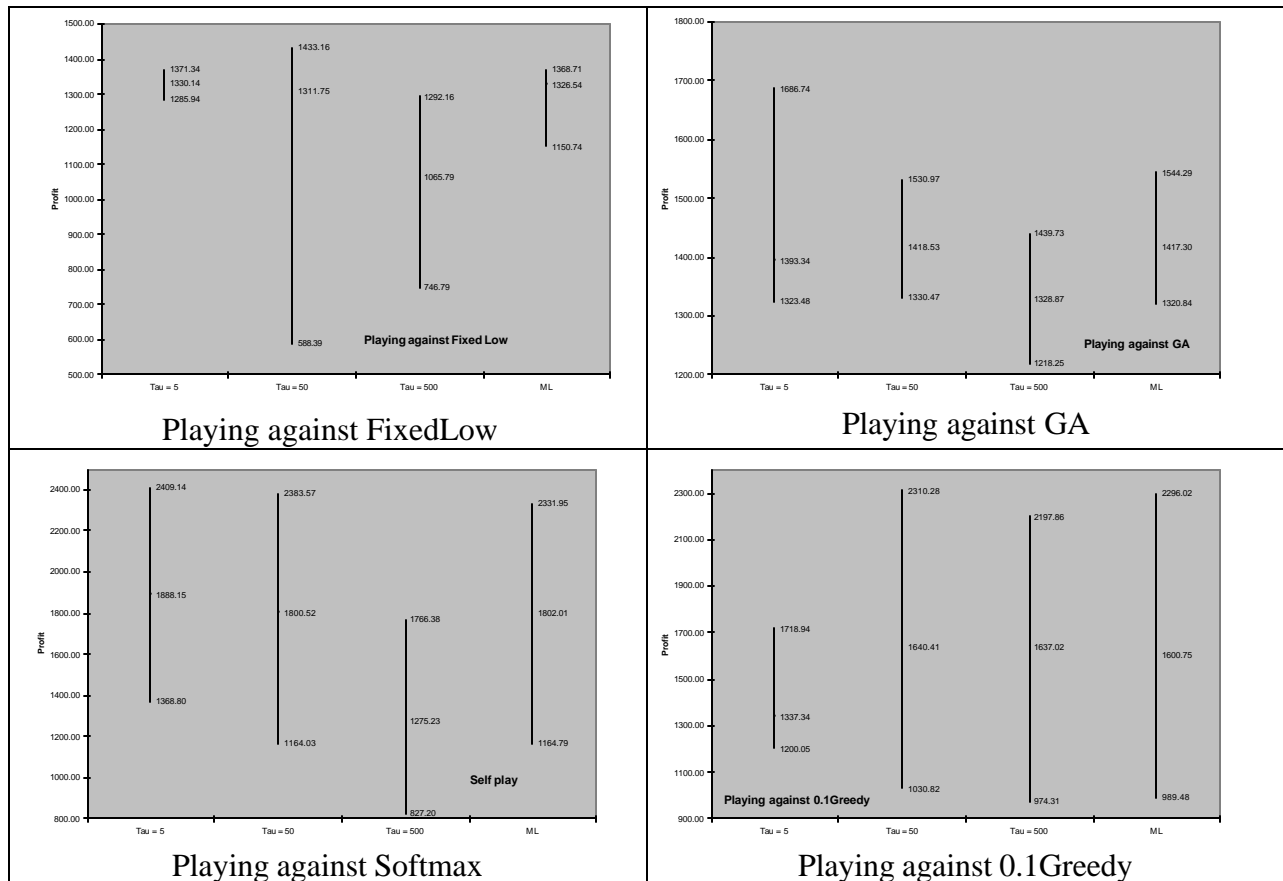


Figure 2. Spread of the rewards earned in each scenario