A suggestion for a Web page personalizing Algorithm

Pankayaraj.P
Undergraduate
Department of Computer Engineering
University of Peradeniya
Sri Lanka
Email: pankayaraj1995@gmail.com

Dr. C. K. Walgampaya
Senior Lecturer
Department of Engineering Mathematics
University of Peradeniya
Sri Lanka
Email: ckw@pdn.ac.lk

***Abstract*—When it comes to user customization it is essential to capture users preferences in an optimal manner so that the user can be served based on his past preferences. The concept behind this work is to formulate an a methodology for an online advertising shop to customize it's advertisement presentation using the existing algorithms in the literature.**

***Index Terms*—GPSARSA , Bayesian Reinforcement Learning , Advertisement Personalization**

I. Introduction

The task of the algorithm is to find the next shop to suggest for the user on his time line based on his past preferences. Users preferences will be captured by the ratings he give for a shop when it is shown in his time line and by the fact weather he marks some shop as visited. Most part of the final suggested algorithm follows the 2003 paper named Bayes Meets Bellman: The Gaussian Process Approach to Temporal Difference Learning [1]. The idea of Markov Decision Processes was used to design the algorithm. The idea is to represent the shops that are supposed to be suggested next as the possible actions leading to the next states and consider the currently displayed shop as the current state. The rating and the fact user marks some shop as visited is modified as a rewards. This now the problem can be modified into a Markov Decision Process for each user.

II. State Representation

The state representation of a shop was made to primarily capture it's geographical location and the property of the shop. Thus a trade off is made between these two vectors with more priority given towards the geographical location. With $i > j$

$$StateVector = i*LocationVector + j*CharactersiticVector$$

Even though the idea of using all the shops that are available in the database as the adjacent states sounds profound it attracts a load of computational resources and optimization which can be expensive given this is a product aimed for local users. Thus only having the adjacent shops within a radius as the adjacent state space for a user reduces the complexity of the problem while not compromising the accuracy. Thus the total state space is defined as the shops around a certain radius around the user.

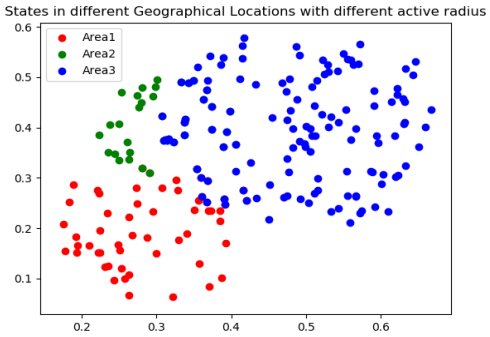

Fig. 1. States

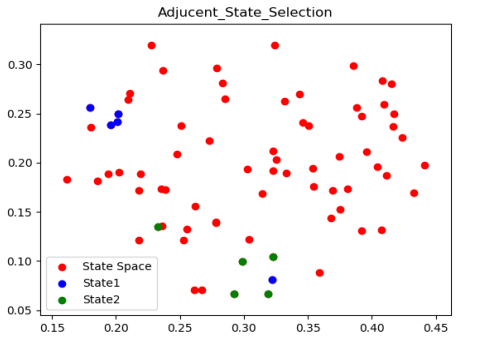

Fig. 2. Adjacent State Selection

Now in a given state space we select the states with the lowest euclidean distance as the adjacent state space. And to avoid the states getting stuck at a particular position a random state is selected from the state space as an adjacent state.

$$Distance = (CurrentState - State)^2$$

$$AdjucentState = argsort(Distance)[:n] + RandomState$$

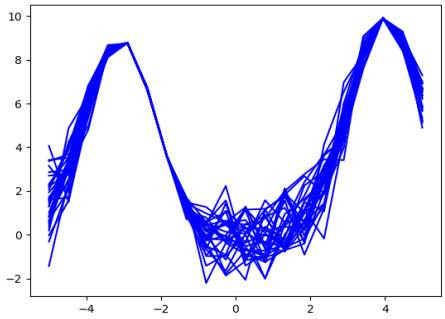

Fig. 3. A Gaussian Process

III. Algorithm Description

A Markov Reward Process is made of a tuple S, R, γ which is State, Reward, and decay parameter. A value function V for a state s is defined as

$$V(s) = E(\sum_{t=0}^{\infty} \gamma^t * R(s_t, s_{t+l}) \mid s_0 = s)$$

where the expectation is taken over the policy π based state transition $p^{\pi}(s_{t+1} \mid s_t)$. The action that leads to a state with the most V is selected in a solved Markov Decision Process. There are several ways to obtain the solution, optimal policy π_*. One such method is to use a value function estimator that is trained such that when given a particular state representation it spits out the corresponding value function. Gaussian Process is the value function estimator used in our case because of the confidence interval it gives apart from the value estimate.

The training data is provided to the estimator by TD(0) update as per the paper[1] which is

$$v(s_t) = v_t + \gamma * \delta_t$$

$$\delta_t = r_t + \gamma * v(s_{t+1}) - v(s_t)$$

$$r_t = R(s_{t+1}, s_t)$$

The TD(0) rule can be written as a matrix format as follows

$$H = \begin{bmatrix} 1 & -\gamma & 0 & \dots & 0 \\ 0 & 1 & -\gamma & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & -\gamma \end{bmatrix}$$

$$R_{t-1} = H_t V_t + N_{t-1}$$

$$N_{t-1} = noise$$

Then a Gaussian process can be fit as mentioned in the paper [1] for the equation as follows

$$\begin{bmatrix} [R_{t-1}] \\ V(s) \end{bmatrix} \sim N \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} H_t K_t H_t^T + \Sigma_{t-1} & H_t k_t(s) \\ (H_t k_t(s))^T & k(s,s) \end{bmatrix}$$

where $k(s,s), k_t(s), K_t$ are kernel dependent functions as defined in the paper [1].Due to the complexity of the Gaussian Process to be used as it is the paper [1] suggests an online algorithm which performs at $O(n^2)$ compared to the $O(n^3)$ complexity the original performs at. To reduce the no of states that are involved in the estimation process it uses a sparsification test where it takes the least square solution for the Hilbert space transformation of the new state from the available states (also as a Hilbert space transformation) and if the error in the estimation is lesser than than a threshold then it let them represent the new state. Or else it adds the new state to the existing states. The least square solution and error is as follows

$$argmin_a(a^T K_{t-1} a - 2a^T k_{t-1}(s_t) + k_{tt})$$

$$error = k_{tt} - k_{t-1}(s_t)^T a_t$$

Then as per the respective parameters mentioned in the paper for the case of error being less than the threshold

$$H_t = \begin{bmatrix} H_{t-1} \\ a_{t-1} - \gamma * a_t^T \end{bmatrix}$$

$$\alpha_t = \alpha_{t-1} + c_t / s_t(\Delta k_{t-1}^T \alpha_{t-1} - r_{t-1})$$

$$C_t = C_{t-1} + 1/s(c_t c_t^T)$$

For the case of error being less than the threshold

$$H_t = \begin{bmatrix} H_{t-1} & 0 \\ a_{t-1}^T & -\gamma \end{bmatrix}$$

$$\alpha_t = \begin{bmatrix} \alpha_{t-1} + c_t / s_t(\Delta k_{t-1}^T \alpha_{t-1} - r_{t-1}) \\ \gamma / s_t(\Delta k_{t-1}^T \alpha_{t-1} - r_{t-1}) \end{bmatrix}$$

$$C_t = \begin{bmatrix} C_{t-1} + 1/s(c_t c_t^T) & \gamma / s_t(c_t) \\ \gamma / s_t(c_t)^T & \gamma^2 / s_t \end{bmatrix}$$

Then the value function and the confidence interval can be found in $O(n^2)$ as follows

$$valuefuncton = k_t(s)^T \alpha_t$$

$$confidenceInterval = k_{ss} - k_t(s)^T C_t k_t(s)$$

IV. Test

In order to test the algorithm in the absence of real time data we have decided to test it on a simulation. We choose total adjacent states as 4. Then they were mapped into a 2D grid and then the tests were run on that grid. Now an example of zero indexed 10*10 grid is considered.As an extreme case it was assumed that the user kept on avoiding the ratings until he visits the shops at position (5, 8) and (7,2). Thus the reward system is initially modified as give a reward of 1 at those 2 states and 0 elsewhere.

And the value function result is obtained as in figure 5

The same system was now made to penalize itself with -1 every time it didn't give the user what he wanted and was given a 0 reward when it got to the user's wanted stage.

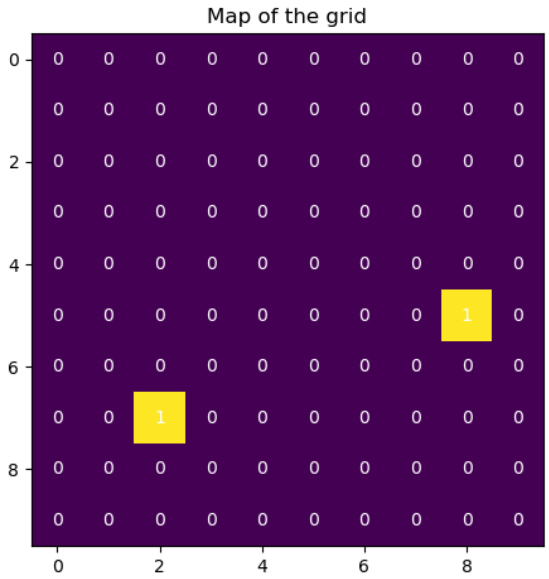

Fig. 4. A 1,0 reward grid

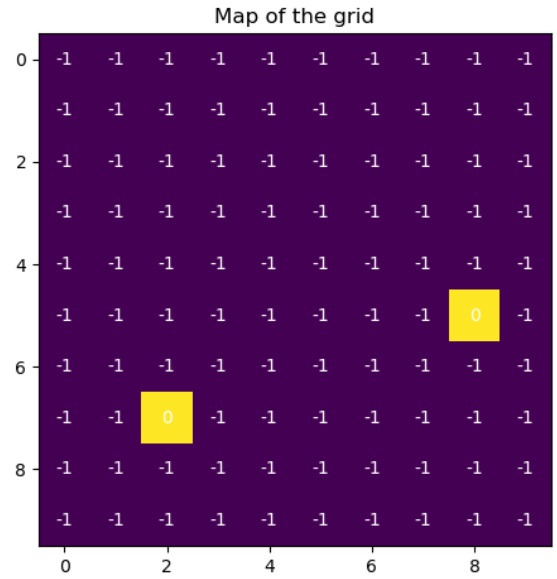

Fig. 6. A 0,-1 reward grid

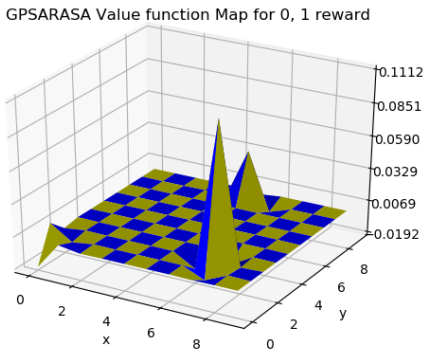

Fig. 5. GPSARSA action value function for a 0, 1 grid

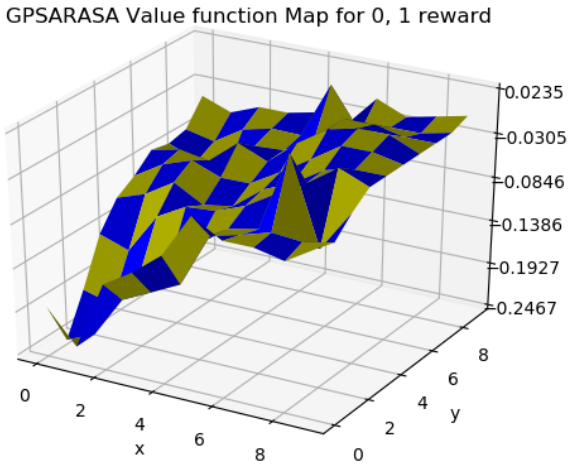

Fig. 7. GPSARSA action value function for a -1, 0 grid

And the results are as in figure 7.

Important fact to observe in are the peaks of the graph and the dynamics of the graph leading to those peaks. In both graphs the algorithm peaked at the customer's favoured places and near (0, 0) when customer doesn't have any favourite it fell down and gradually increased towards the favoured places. According to the results we can interpret the systems work as follows. That is when the user is in a random shop the algorithm will maximize the value function and then climb towards the visited shop. Thus the user can view advertisements from the similar shops and also the new offer advertisements from the already visited shop. The value function at the intermediate state of the crawling shows how it penalizes the undesired states that are visited. Thus on the next stage it would explore more.

Furthermore in all the diagrams the scale of value functions doesn't change by a greater margin due to our usage of rewards $< \|1\|$. Thus the estimator can adapt the change with ease when the user's preference changes thus enabling to add dynamics.Next if we assume that the user is giving ratings along a path and visiting the above mentioned states. To simulate that scenario we let our grid provide a reward of 5 for both visited states and then a reward of 0 for a certain path and -1 for the rest of the states and let the user start from (0, 0) state.

The value function we got was in figure 10

It has made smaller spikes for the path we defined and larger

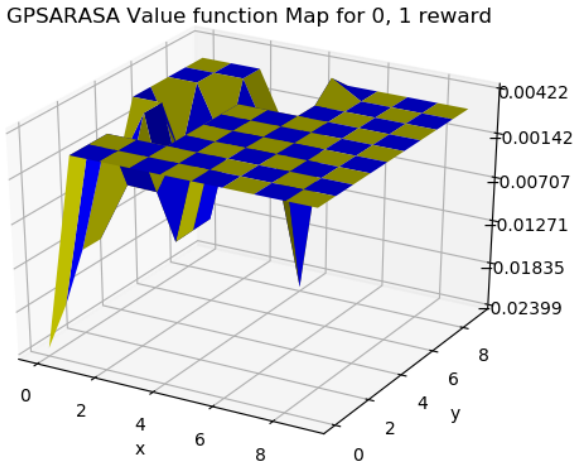

Fig. 8. GPSARSA action value function at the intermediate stage in -1, 0 reward grid

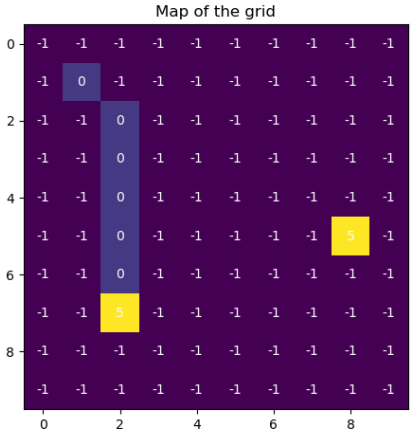

Fig. 9. A grid with a path

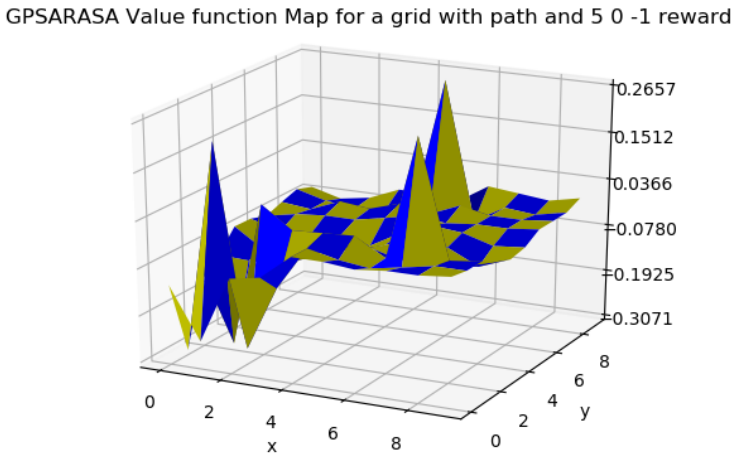

Fig. 10. GPSARSA action value function for the grid with a path

spikes for the visited shops. Thus capturing more information with ratings. In all these cases a higher reward was proposed for visited shops in the hindsight that a customer would want to know more of the new offers from the shops he has visited. These dynamics can be varied given what is the priority and the reward can also be treated as a hyper parameter of the algorithm. Furthermore in all the above processes the algorithm represented the states by using 30-40 states instead of the whole 100 states thus solving part of the third problem mentioned under the previous section

REFERENCES

[1] Yaakov Engel, Shie Mannor, Ron Meir (2003) *Bayes Meets Bellman: The Gaussian Process Approach to Temporal Difference Learning*