Networks of Sensors

Onderwater, M.

2016

document version
Publisher's PDF, also known as Version of record

7

Discovery of Structured Optimal Policies in Markov Decision Processes

In this chapter we continue work on VFD, the novel method for discovery of relative value functions for Markov Decision Processes that we introduced in Chapter 6. VFD discovers algebraic descriptions of relative value functions using ideas from the Evolutionary Algorithm field and, in particular, these descriptions include the model parameters of the MDP. We extend that work and demonstrate how additional information about the structure of the MDP can be included in VFD. For this we use the same example MDP as in Chapter 6, and include prior knowledge that the optimal policy is of threshold type. We let VFD learn an expression for this threshold in terms of the model parameters, and numerically inspect its performance. We demonstrate that this alternative use of VFD also yields near-optimal policies, illustrating that VFD is not restricted to learning relative value functions and can be applied more generally.

This chapter is based on the results presented in [7] and [8].

7.1 Introduction

We use the example MDP from Section 6.5 to illustrate this alternative application of VFD. For convenience we repeat the description of the MDP here. Figure 7.1 shows a queue with Poisson arrivals (rate λ) and two servers with exponential service rates μ_1 and μ_2 ($\mu_1 > \mu_2$). Arriving jobs are put into the queue and, when they reach the head of the queue, they have to be assigned non-preemptively to either the fast server (S_1) or the slower server (S_2). This decision is taken after a job completion, as well as after a job arrival. The state space of the resulting MDP is $\mathcal{X} = \mathbb{N} \times \{0, 1\}$, where a state $(x, i) \in \mathcal{X}$ reflects that there are x jobs in the queue and at S_1, and i jobs at S_2. From [81] we have the optimality equation

$$g + V(x,i) = x + i + \lambda W(x+1,i) + \mu_1 W((x-1)^+,i) + \mu_2 W(x,0) \qquad (7.1)$$

with

$$\begin{aligned} W(x,0) &= \min\{V(x,0); V(x-1,1)\} \quad \text{if } x > 0, \\ W(0,i) &= V(0,i), \\ W(x,1) &= V(x,1). \end{aligned} \qquad (7.2)$$

The function $W(x,i)$ reflects the decision to be taken after the occurrence of an event. In particular, if S_2 is empty the decision is between leaving the job in the queue ($V(x,0)$) or moving one job from the queue to S_2 ($V(x-1,1)$), as shown in Eq. (7.2). If the queue and S_1 are empty then moving a job is not possible and the state of the system does not change ($W(0,i) = V(0,i)$). Also, if the second server is busy the state does not change ($W(x,1) = V(x,1)$). In Eq. (7.1), the second, third, and fourth term on the right-hand side correspond to the decision upon a job arrival, a job completion at S_1, and a job completion at S_2, respectively. Finally, the constant g is the time-average cost.

For the MDP in Eq. (7.1), we know that the optimal policy of Eq. (7.1) is of threshold type (see [81] for a proof). To be precise, for given model parameters λ, μ_1, μ_2 the optimal policy states that server S_2 should be used whenever $x > T$, i.e., when x exceeds a certain threshold T. By viewing this threshold as a function of the model parameters, we can apply VFD and discover an algebraic expression for T in terms of the model parameters (λ, μ_1, μ_2). Compared to using VFD for discovering relative value functions, this alternative application has the advantage that it is much faster, since the threshold T depends only on model parameters (λ, μ_1, μ_2) and not on state (x, i). Most importantly though, it demonstrates that VFD is flexible and not restricted to learning relative value functions.

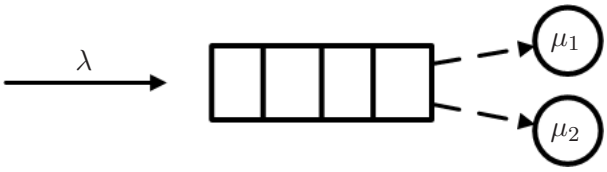

FIGURE 7.1: A queueing system with jobs arriving at rate λ, and with two servers S_1 (at the top) and S_2 (at the bottom). Both servers handle jobs from the queue, but server S_1 is faster than S_2 (reflected by service rates μ_1 and μ_2 such that $\mu_1 > \mu_2$). Upon completion of a job, and upon arrival of a new job, a controller decides whether to assign a job from the queue to either S_1 or S_2 (dashed line). See also Figure 6.4 and the description in Section 6.5.

The remainder of this chapter is structured as follows. First, in Section 7.2.1 we describe the setup of the experiments in this chapter. This includes the choice of sample points, and of the value of VFD's parameters. Section 7.3 demonstrates the results of applying VFD to discovery of thresholds, and Section 7.4 has concluding remarks.

7.2 Setup of experiments

In this section we describe the experiment where VFD learns an expression for the threshold T. Prior to running VFD, we make a minor change to VFD to facilitate the fact that a threshold has only integer values. Also, we determine the contents and location of the sample point sets, and VFD's parameters. As in Chapter 6, we make reasonable choices for the sample point sets and VFD's parameters, rather than seeking choices that lead to, e.g., the fastest run times. Our focus is on demonstrating that VFD can indeed be used to learn an algebraic description of a threshold policy that has near-optimal performance.

7.2.1 Learning thresholds with VFD

We denote the threshold discovered by VFD as $\widetilde{T}$, following the notation of Chapter 6. Before running experiments on VFD, we make one small adjustment to the error criterion used by VFD in Eq. (6.1). The threshold T is an integer number, whereas running VFD unchanged yields a $\widetilde{T}$ that potentially returns non-integer values. Additionally, the threshold is independent of the state (x, i), so we do not need to take the maximum over sample points anymore.

To this end, we modify the error on a sample point set $\mathcal{E}_q$ from Eq. (6.1) to

$$\mathcal{E}_q = \frac{\left|\lfloor \widetilde{T} \rfloor - T\right|}{T}, \tag{7.3}$$

where $\lfloor \widetilde{T} \rfloor$ denotes the largest integer smaller than $\widetilde{T}$. Also, we included notation T and $\widetilde{T}$ to emphasize that VFD discovers thresholds in this chapter. The expression for the total error E remains the same as in Eq. (6.2). This change to VFD is purely for convenience to avoid scenarios where VFD spends time improving a threshold $\widetilde{T} = 5.3$ to $\widetilde{T} = 5$, even though the improvement has no effect on the resulting policy. VFD also works well when the threshold is considered as a decimal number, as we demonstrated in [8].

7.2.2 Sample point sets

Recall that, when learning relative value functions with VFD, each sample point set contains several sample points (x, i) on the relative value function $V(x, i)$ for fixed parameters λ, μ_1, μ_2. For the current problem we can suffice with a single point in each sample point set, namely the threshold value T found by value iteration. Note that by making this change we only affect the input to VFD, and not the algorithm itself.

As in Chapter 6 we base the choice for the parameters λ, μ_1, μ_2 corresponding to each sample point set on the load $\rho = \lambda/\mu_1 \in$ of a $M/M/1$ system. In the region $0 \leq \rho < 0.7$ the load on the system is low, and possible wrong decisions in a policy have little impact. Therefore, we focus the experiment in this chapter on the region $0.7 \leq \rho \leq 1$. We want to let VFD discover a function that is close to optimality, so we place the sample point sets close together at intervals of 0.025. This results in $Q = 11$ sets at $\rho = 0.700, 0.725, \ldots, 0.925, 0.950$. Then we generate parameters μ_1 and μ_2 uniformly from $[0, 1]$, and set $\lambda_1 = \rho\mu_1$. In generating these values we also ensure that $\mu_1 > \mu_2$ and that $\lambda + \mu_1 + \mu_2 = 1$. The resulting parameters are in Table 7.1. Recall from Chapter 6 that when applying VFD to MDPs with a large dimensionality, having many sample point sets is typically not computationally feasible. Each (large) sample point set is used with each tree in the population to determine how well that tree fits, so having many sample point sets increases VFD's run time severely. In this chapter, however, each sample point set contains just one point, so we can use many of them without affecting run time too much.

Set	ρ	λ	μ_1	μ_2
0	0.700	0.369	0.527	0.104
1	0.725	0.373	0.515	0.112
2	0.750	0.379	0.506	0.115
3	0.775	0.396	0.512	0.092
4	0.800	0.409	0.511	0.080
5	0.825	0.450	0.546	0.003
6	0.850	0.447	0.525	0.028
7	0.875	0.424	0.484	0.092
8	0.900	0.466	0.518	0.015
9	0.925	0.445	0.481	0.074
10	0.950	0.485	0.510	0.005

TABLE 7.1: Model parameters per sample point set.

Name	In example	Allowed values
SEED	3,151,492	[0,MAXINT]
MU	1,000	[1,MAXINT]
LAMBDA	500	[1,MAXINT]
MIN_ERROR	0	[0,1]
MAXELEMENTSINTREE	100	[1,MAXINT]
APPLYMUTATIONPROB	0.05	[0,1]
DIVERSITY_THRESHOLD	0.01	[0,MAXDOUBLE]

TABLE 7.2: The parameters available to VFD, the values assigned to them for the example MDP in Section 7.1, and the values allowed by VFD.

7.2.3 Setting parameters of VFD

Running VFD requires the specification of several parameters, listed in Table 7.2. Following the reasoning in Section 6.5.3 we keep parameters SEED, MU, LAMBDA, and DIVERSITY_THRESHOLD at the same value as in Table 6.1. Parameter MIN_ERROR we set to 0 because we want VFD to discover a perfectly fitting function on the sample point sets. For parameter APPLYMUTATIONPROB we rely on experience from the GP community and set it to 0.05. The remaining parameter, MAXELEMENTSINTREE, is determined with short experimental runs, suggesting that VFD is capable of finding well-fitting relative value functions with MAXELEMENTSINTREE= 100. The resulting values for the VFD parameters are shown in the second column of Table 7.2.

Set	ρ	T	$\widetilde{T}$
0	0.700	2	2
1	0.725	2	2
2	0.750	2	2
3	0.775	3	3
4	0.800	3	3
5	0.825	32	32
6	0.850	6	6
7	0.875	2	2
8	0.900	8	8
9	0.925	3	3
10	0.950	13	13

TABLE 7.3: Threshold $\widetilde{T}$ of the policy discovered by VFD, compared to threshold T of the optimal policy.

7.3 Numerical results

We run VFD with the sample point sets from Section 7.2.2 and the parameters from Section 7.2.3. The threshold $\widetilde{T}$ discovered by VFD is

$$\begin{aligned}\widetilde{T} = 3\lambda - 2\mu_1 - 0.47 - \frac{\lambda - 0.12}{\lambda + \mu_1 - \lambda\mu_1 - 0.82} \\ + \frac{\lambda}{\mu_2}(\mu_1 - 0.40)(\lambda - \mu_1\mu_2) \\ - 0.47(\mu_1 - \lambda)\Big[\lambda + \mu_1 - \lambda(2\mu_1 - \mu_2) + 1 - \frac{\lambda + 2\mu_1}{\mu_2}\Big].\end{aligned} \tag{7.4}$$

In Table 7.3 we list the threshold $\widetilde{T}$ from VFD, and the threshold T of the optimal policy. The table demonstrates that the discovered expression for the threshold does indeed perfectly match the optimal values.

Next, we inspect the performance of the expression in Eq. (7.4) on model parameters that it was not trained on. To this end, we select 10 values for ρ, different from those in Table 7.3, and generate new parameters λ, μ_1, μ_2 as before. The new model parameters are shown in Table 7.4 in the first four columns. Then we calculate the threshold $\widetilde{T}$ (sixth column), and the optimal threshold T (fifth column). Table 7.4 demonstrates that the expression discovered by VFD yields thresholds that closely resemble the optimal thresholds,

ρ	λ	μ_1	μ_2	T	$\widetilde{T}$
0.710	0.411	0.578	0.011	17	17
0.735	0.421	0.573	0.007	25	25
0.760	0.422	0.555	0.023	8	8
0.785	0.429	0.546	0.025	7	7
0.810	0.443	0.548	0.009	15	15
0.835	0.416	0.498	0.086	2	3
0.860	0.429	0.499	0.072	3	3
0.885	0.458	0.517	0.025	6	6
0.910	0.474	0.521	0.004	18	19
0.935	0.465	0.497	0.038	4	4

TABLE 7.4: Threshold $\widetilde{T}$ of the policy discovered by VFD, compared to threshold T of the optimal policy. The model parameters are different from the ones VFD was given as input.

even for the new model parameters. We repeated this experiment several times, and the function discovered by VFD consistently returned thresholds close to optimal.

In order to investigate the run time of the alternative use of VFD described in this chapter, we repeat the process from Section 6.6.5. There, we recorded a median run time of 2 minutes and 21 seconds over 25 runs. In the current setup, running VFD 25 times gives a median run time of 47 seconds. This is faster than before, even with a low value for MIN_ERROR and many sample point sets ($Q = 11$) to cover the parameter space.

7.4 Conclusion

In this chapter we continued work on VFD, a novel algorithm for discovering relative value functions for Markov Decision Processes. We applied VFD to an example MDP, for which we know that the optimal policy is of threshold type. Instead of discovering the relative value function with VFD, we learned an algebraic expression for the threshold in terms of the model parameters. We numerically inspected the resulting threshold and compared the corresponding policy to the optimal policy. The results demonstrate that his alternative use of VFD also yields near-optimal policies and thresholds that closely resemble the optimal value.