Exploiting Multi-Step Sample Trajectories for Approximate Value Iteration

Robert Wright[1,2], Steven Loscalzo[1], Philip Dexter[2], and Lei Yu[2]

[1] AFRL Information Directorate, Rome, NY, USA
{robert.wright,steven.loscalzo}@rl.af.mil
[2] Binghamton University, Binghamton, NY, USA
{pdexter1,lyu}@binghamton.edu

Abstract. Approximate value iteration methods for reinforcement learning (RL) generalize experience from limited samples across large state-action spaces. The function approximators used in such methods typically introduce errors in value estimation which can harm the quality of the learned value functions. We present a new batch-mode, off-policy, approximate value iteration algorithm called Trajectory Fitted Q-Iteration (TFQI). This approach uses the sequential relationship between samples within a trajectory, a set of samples gathered sequentially from the problem domain, to lessen the adverse influence of approximation errors while deriving long-term value. We provide a detailed description of the TFQI approach and an empirical study that analyzes the impact of our method on two well-known RL benchmarks. Our experiments demonstrate this approach has significant benefits including: better learned policy performance, improved convergence, and some decreased sensitivity to the choice of function approximation.

1 Introduction

Temporal Difference (TD) based value iteration methods solve reinforcement learning problems by estimating the optimal value function over the problem space [14]. This function describes the maximal expected long-term value of taking actions in any given state of the domain and can be used to extract an optimal policy. Representing value functions exactly is infeasible in all but the most trivial domains giving rise to Approximate Value Iteration (AVI) algorithms [8]. Function approximation provides a mechanism for efficiently representing value functions, however, by their very nature they introduce generalization errors. Typical AVI methods are dependent on the approximation model to derive and propagate the long-term value of a state through the function space. The generalization error introduced by function approximation adversely impacts the derivation of long-term values and as a consequence the policy described by the learned function.

Reinforcement Learning (RL) problems are multi-step and as such the data used by value iteration methods typically comes in the form of sequential sets of experience samples known as trajectories. Trajectories are actual observed

paths through the problem space that describe how value is propagated through the state space without the error induced by function approximation. Despite the availability of this information, AVI based methods commonly ignore the sequential relationship between samples while deriving the value of a state and rely instead on the approximation model.

Figure 1 provides an illustration of the AVI generalization problem alluded to above. There are four sampled states, represented by the circles, that are part of two separate trajectories that were generated by arbitrary policies. In this example, assume that the immediate rewards for the transitions shown are zero and we are given the true optimal long-term values, V, for the transitions that go beyond states B and D, 4 and 2 respectively. The function approximation model here is a tiling abstraction represented by the rectangle, $\bar{S}$, that combines the value of states B and D to predict a generalized value, x where $2 < x < 4$, for any other state $\bar{S}$ contains. The value x may be a reasonable approximation of the value of unobserved states within $\bar{S}$, however it produces generalization errors for the values of B and D. With function approximation, standard AVI approaches will only use x to determine the long-term value for states A and C instead of the long-term values for the states that follow in their sampled trajectories, again 4 and 2 respectively. From the trajectories we can see that the value for state A in this example should be greater than that of state C's. But, due to the generalization error, value iteration will derive a value for those states based upon x that is an underestimate of the value of state A and, potentially, an overestimate of the value of state C.

Unfortunately this error will not just impact the derivation of the long-term value for the sampled states A and C. AVI methods calculate long-term value by back-propagating rewards along approximated transition paths. This error will therefore adversely affect the derivation of long-term value for all other sampled states that can reach states B and D. Errors such as this will propagate and reoccur through the function space and can reduce the quality of the learned policy. This form of generalization error is common among statistical function approximation, and is not limited to tiling abstractions.

The trajectory AB can be used to correct for the underestimation of the sampled value of A. Trajectory AB demonstrates that following some unknown policy from the sampled state B a value of at least 4 is attainable. Given this information, while updating the long-term value for the sampled state A we should use the long-term value implied by the trajectory, 4, rather than x because we know it is an underestimate. Unfortunately we cannot apply the same reasoning to correct for the potential overestimation in the sampled value of C. The trajectory CD does not provide definitive evidence that from state D a value greater than x is not achievable. Instead, the value of x should still be used while updating the long-term value estimate of C.

In this paper we present a new algorithm we call Trajectory Fitted Q-iteration (TFQI) that utilizes trajectory information in this way to reduce the impact of generalization error. It is a batch mode, model-free, off-policy RL algorithm based upon the Fitted Q-Iteration (FQI) framework [4]. FQI is a well-known

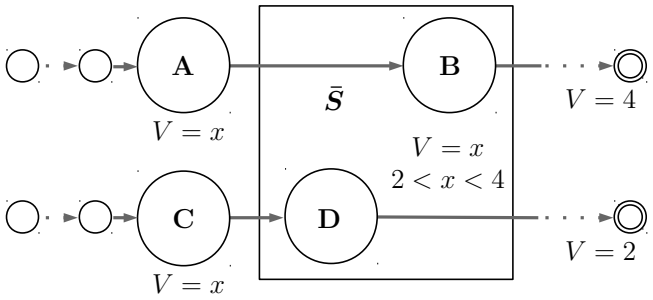


Fig. 1: A simple 4 state example scenario where AVI can produce misleading long-term value estimates. A, B, C, and D are states that have been sampled belonging to two sample trajectories AB and CD. $\bar{S}$ is a state abstraction that generalizes the value of the states it contains, B and D. The V's indicate the value derived for each state by standard AVI.

value iteration algorithm that has demonstrated great empirical performance on RL benchmarks [11]. In addition we provide an empirical analysis of this new approach that demonstrates TFQI learns policies that are superior to that of FQI using the same data and function approximation architecture with no significant additional computational costs beyond the original FQI formulation. TFQI not only demonstrates the ability to learn better policies, but also shows dramatically improved convergence and some decrease in the sensitivity to approximation model settings. The only additional assumption TFQI makes beyond FQI, and other off-policy TD methods, is the availability of trajectory based data which, as stated above, is the norm rather than the exception.

The remainder of the paper is structured as follows: Section 2 provides background and notation and a detailed description of FQI. Section 3 introduces TFQI and elaborates upon the ideas behind the new approach. In Section 4 we provide details for the setup of our empirical analysis. Section 5 presents the results as well as a discussion of their significance. Section 6 provides a brief overview of related work. Finally, in Section 7 we conclude and identify future directions for this research.

2 Background

Reinforcement learning is commonly discussed within the framework of Markov Decision Processes (MDP) [10]. An MDP, M, is a 5-tuple, $M = \langle S, A, P, R, \gamma \rangle$, where S is a set of states of the world, A is the set of actions, P is the state transition model such that $P(s, a, s') \in [0, 1]$ describes the probability of transitioning to state s' after taking action a in state s, R is the reward function such that $R(s, a) \in \mathbb{R}$ describes the immediate reward received for taking action a in state s, and γ is the discount factor on future rewards bound by $[0, 1)$.

The goal of RL is to derive an optimal policy, π^*, over M that maximizes the discounted long-term aggregate value that can be obtained starting from any

given state. A policy, π, is a mapping from the state space to the action space, $\pi = S \mapsto A$. π^* can be extracted from the optimal Q-function, $Q^*(s, a)$, defined by the solution to the Bellman equation:

$$Q^*(s,a) = R(s,a) + \gamma P(s,a,s') \max_{a' \in A} Q^*(s',a') \tag{1}$$

$$\pi^*(s) = \max_{a \in A} Q^*(s,a) \tag{2}$$

If the functions P and R are known then Q^* and π^* can be derived in a straightforward fashion using dynamic programming. However, in the model-free RL context P and R are unknown and Q^* must instead be estimated from samples. Samples describe single-step observations of transitions taken in the domain. They are represented by tuples, (s, a, s', r), consisting of a state s, and action a, the state s' transitioned to by taking a in s, and r, the immediate reward for that transition. Samples are collected through interactions with the domain by the agent itself, a third party, or by a generative model when available.

Value iteration combined with Temporal Difference (TD) [14] provides a model-free way to estimate $Q^*(s, a)$ directly from samples. Given a set of samples to learn from, value iteration can learn estimates of the Q-values for each sample by iteratively applying the Bellman update rule to each sample:

$$Q_n(s,a) \leftarrow r + \gamma \max_{a' \in A} Q_{n-1}(s',a') \tag{3}$$

Update rule (3) states that the Q-value for a state-action pair at iteration n is equal to the expected value of immediate reward plus the discounted long-term value. Long-term value here is estimated by the current maximum Q-value for the successor state. Each iteration of the approach refines the long-term value estimate for the state-action pair by back propagating value from subsequent transitions that receive rewards. Through successive iterations this approach is guaranteed to converge toward Q^* in small domains, where states can be represented exactly, and with infinite sampling guarantees.

Non-trivial problems, however, have state and action spaces that are too large to define the Q-function explicitly, so we must use function approximation to represent an approximate Q-function, $\hat{Q}$. Linear function approximation is commonly used in practice and is the type of approximation we utilize in this paper. In a linear function approximation scheme the value-function or Q-function is represented by a weighted, w, linear combination of k features, ϕ, defined over the state-action space:

$$\hat{Q}(s,a) = \sum_{i \leftarrow 1}^{k} w_i \phi_i(s,a) \tag{4}$$

The types of features and their number is domain dependent and crucial to the success of the approach. Typically those parameters are chosen a priori by a domain expert. The Q-function is approximated by deriving an appropriate weight vector through linear regression.

Algorithm 1 FQI(D, γ, N)

Require: D: set of samples, γ: discount factor, N: number of iterations to complete
1: $\hat{Q}_0 \leftarrow 0$ //Initialize Q-function to zero everywhere
2: **for** $n = 1$ to N **do**
3: **for all** $sample \in D$ **do**
4: $//sample = (s_t, a_t, r_t, s_{t+1})$
5: $Input_{sample} \leftarrow (s_t, a_t)$
6: $Target_{sample} \leftarrow r_t + \gamma \max_{a \in A} \hat{Q}_{n-1}(s_{t+1}, a)$
7: **end for**
8: $\hat{Q}_n \leftarrow Regression(Input, Target)$
9: **end for**
10: **Return** $\hat{Q}_N$

Representing the Q-function this way is much more efficient and the choice of representation (features) provides valuable generalization. Generalization allows for effective learning in complex domains even under limited sampling conditions. However, while generalization is a useful benefit of approximation, it is limited by errors that can produce misleading value estimates as was shown in the example given in Section 1.

2.1 Fitted *Q*-Iteration

Fitted Q-Iteration (FQI) is a general AVI framework for solving RL problems. It derives $\hat{Q}$ through a sequence of standard supervised learning regression problems that iteratively converge upon the fixed point solution of the Bellman equation, equation 1. **Algorithm 1** provides a detailed outline of the FQI framework. First, the algorithm begins with a Q-function initialized to zero, $\hat{Q}_0 = 0$, and a set of provided samples, D. At each iteration FQI solves a regression problem that produces a more accurate Q-function model to be used in subsequent iterations. The regression problem is defined by using the state and action pairs, (s, a), from the samples in D as the input patterns and the current approximated Q-value of the pair, $\hat{Q}_n(s, a)$, as the regression target. After performing regression, the updated Q-function model, $\hat{Q}_{n-1}$, combined with the Bellman update rule (3) is used to generate the next iterations regression targets (see line 6 of **Algorithm 1**). Given an appropriate choice in approximation model, through successive iterations this process converges and produces a model that approximates Q^*.

FQI has several desirable traits that have made it a widely used algorithm among the RL community. First, it is an off-policy algorithm which enables FQI to effectively utilize samples collected by any means. This is an important feature when samples are difficult to obtain or cannot be simulated. FQI is a batch-mode algorithm which gives it favorable sample efficiency when compared to single sample update approaches. Because of the generality of the approach, it can be paired with a variety of regression models, and allows for the approach to be adapted to any given problem domain. Additionally, it has demonstrated

competitive learning performance compared to that of other state-of-the-art RL algorithms [11].

However, FQI like other AVI methods suffers from the generalization problem discussed in Section 1. FQI's use of the Bellman update, shown on line 6 of **Algorithm** 1, explains why it is susceptible to this issue. The long-term value component of the update is based upon the maximal value that can be obtained from the successor state of the sample according to the function approximation model, $\hat{Q}_{n-1}$. It is not a strong assumption that the approximation model will contain generalization error, and as a result the derived long-term value of state-action pairs will be adversely impacted.

3 Trajectory Fitted Q-Iteration

AVI methods like FQI are sensitive to the generalization errors of an approximation because they assume that there is no other mechanism to propagate long-term reward through the function space than via the model. In most RL scenarios, however, this assumption is not true. Samples for RL are generally collected as part of a larger sequence known as a trajectory. These trajectories describe observed paths through the domain's state-action space that can also be used to propagate long-term reward without the generalization error of an approximation. In this section we present our approach, Trajectory Fitted Q-Iteration (TFQI) a new batch-mode, off-policy, AVI learning algorithm based on FQI that exploits trajectory data to improve the derivation of long-term value.

TFQI makes one additional assumption that other AVI approaches do not; the sample data to be learned over has been collected as sets of trajectories. A trajectory is a finite ordered sequence of samples that describe a series of successive transitions through the problem space. More formally, an n-step trajectory is comprised of the following ordered set of samples:

$$\{(s_{t_0}, a_{t_0}, r_{t_0}, s_{t_1})_1, (s_{t_1}, a_{t_1}, r_{t_1}, s_{t_2})_2, \ldots, (s_{t_{n-1}}, a_{t_{n-1}}, r_{t_{n-1}}, s_{t_n})_n\} \quad (5)$$

Trajectories are collected through episodic multi-step interactions with the problem domain performed either by the agent itself or by a third-party. TFQI is an off-policy approach in that it makes no assumptions on the quality of the policy used to generate the trajectories or that they were all produced by the same policy. Additionally, the provided set of trajectories can consist of trajectories of varied length and my not end with transitions to terminal states.

The key insight behind TFQI is that the long-term value for any sample, as part of a trajectory, can be estimated as either the discounted value predicted by the approximation model given the successor state or by the value of the successor sample from the trajectory. The later option is more resilient to generalization errors of the approximation scheme because it is derived from the real value relationship between successive samples and not the approximation. However, the value derived for the successor sample is based on the action taken by the successor, which is not assumed to be the optimal action for the successor state, and therefore can be an underestimate of the optimal value for the sample.

Algorithm 2 TFQI($Traj, \gamma, N$)

Require: $Traj$: a set of trajectories, γ: discount factor, N: number of iterations
1: $\hat{Q}_0 \leftarrow 0$
2: **for** $n = 1$ to N **do**
3: **for all** $trajectory \in Traj$ **do**
4: **for** $i = |trajectory|$ downto 1 **do**
5: $//sample_i = (s_t, a_t, r_t, s_{t+1})_i \in trajectory$
6: $Input_{sample_i} \leftarrow (s_t, a_t)$
7: $Target_{sample_i} \leftarrow r_t + \gamma \max(\max_{a \in A} \hat{Q}_{n-1}(s_{t+1}, a), Target_{sample_{i+1}})$
8: **end for**
9: **end for**
10: $\hat{Q}_n \leftarrow Regression(Input, Target)$
11: **end for**
12: **Return** $\hat{Q}_N$

As such, TFQI uses the maximal value of the two values as the estimate for the long-term reward while performing updates.

Algorithm 2 provides a detailed outline of the TFQI algorithm. Structurally TFQI is very similar to the FQI algorithm. The differences are: TFQI accepts a set of trajectories, $Traj$, as opposed to a general set of samples. TFQI iterates through the samples of each trajectory in reverse sequential order while computing the regression targets. This is necessary to obtain an updated estimate of the successor sample's value, $Target_{sample_{i+1}}$. Finally, the update rule (line 7) has been expanded to utilize the maximum of either the approximation model or successor sample value to estimate the long-term value of a sample. [3]

These modifications to the original FQI formulation enables TFQI to harness trajectory data more effectively. The enhanced update rule is the most meaningful of the changes. It provides a mechanism for long-term value to propagate backward through trajectories. This one function has two beneficial consequences. First, it reduces the impact of generalization errors by correcting underestimates of long-term value for sampled states, as shown in the example given in Section 1. This ability can lead to more accurate $\hat{Q}$-functions and hopefully improved policies. And second, it propagates long-term value through entire trajectories at each iteration. This effect can speed how long-term value is learned throughout the function space and improve convergence especially in reward sparse domains.

TFQI retains the desirable traits (batch-mode and off-policy) of FQI with negligible additional computational costs and no additional overhead. Per iteration, TFQI only adds $|D|$ more comparisons, where D is the set of samples,

[3] One or both of the long-term value estimates will be undefined for the last sample of each trajectory. If the sample transitions to an absorbing state, r_t is taken as the value for that sample. If the sample does not transition to an absorbing state, the value predicted by the approximation model is used.

than that of FQI. In the subsequent section we provide an empirical analysis comparing FQI and TFQI to explore the potential benefits of this approach.

4 Experimental Setup

We performed an empirical comparison of TFQI and FQI over several experiments to assess the impact of the enhanced use of trajectory information for AVI. Our analysis, provided in the subsequent Section 5, evaluates both approaches based on final learned policy performance, convergence speed, and sensitivity to the function approximation model. This section provides the details for the setup of this comparison.

4.1 Domains

Mountain Car [4] (MC) and the Acrobot (ACRO) Swing-Up [6] problems are the two well-known RL domains that we chose to use in our analysis. For our experiments we used implementations of both domains provided by RL-Glue [15]. In MC the learner is tasked with learning a policy for driving an underpowered car from the bottom of a deep valley to the top of the forward hill. The ACRO problem challenges the learner to derive a policy for an underpowered robot, simulating a gymnast on a high-bar, that starting from a still straight position below the bar rotates the leg of the robot, at the hip, in such a way that the robot gathers enough momentum to swing and raise the end of its leg above the high bar position.

The objective for both problems is to reach a goal state in as few steps as possible. As such, the rewards for all transitions in both problems is -1, except for transitions to the goal state which have 0 rewards. While evaluating the learned policies we limit the number of steps a policy has to achieve the objective to 300 steps for MC and 1000 steps for ACRO. Policies that take longer than those limits are considered unsuccessful policies for the purpose of our evaluation.

Both problems require function approximation because of their continuous state-spaces. MC has a 2-dimensional state-space and ACRO has a 4 dimensional state-space. The two domains each have 2 discrete actions: apply forward or reverse throttle for MC and apply forward or reverse torque at the robot's hip for ACRO.

4.2 Function Approximation Models

We use two forms of linear function approximation in our experiments for representing $\hat{Q}$ that differ in the types of features. The first is Radial Basis Functions (RBF) which overlays several Gaussian curves uniformly over the state space and uses their activation as features. It is a common choice for RL and has good localized generalization properties [6]. The other is based on Fourier basis functions. Fourier basis functions have greater global generalization properties and have only recently been explored in an RL context [6].

Ridge regression is used for both variants to train the weight vectors of the models. It applies l_2 regularization on the objective function to prevent overfitting. The amount of regularization is controlled by the shrinkage parameter, λ, that we manually set but can be tuned automatically using cross-validation. This form of function approximation paired with FQI is known to exhibit divergence behavior because errors in the approximation of value are potentially unbounded [4]. We are able to circumvent this issue for these specific problem domains by limiting the predicted value response of the model to be non-positive. Both domains have strictly non-positive rewards making it impossible for state-action pairs to have positive value.

4.3 Trajectory Generation

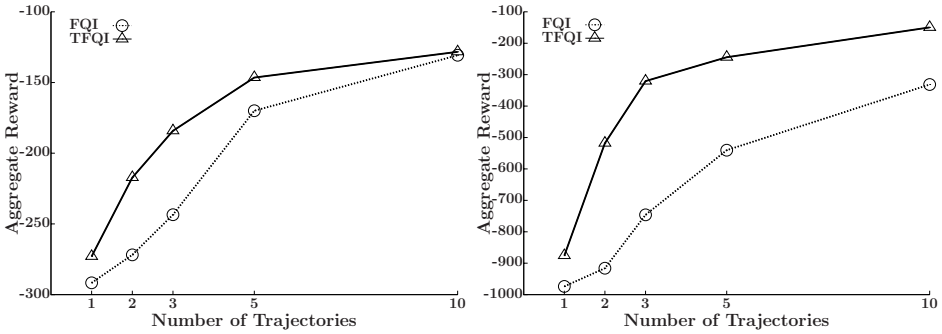
The NEAT [13] algorithm was used to generate large diverse sets of trajectories necessary for our analysis. NEAT solves RL problems via genetic algorithm that performs policy search. Each member of a NEAT population represents a potentially different policy. Trajectories were recorded by observing an evaluation of each policy on the problem domain. If more than one policy achieved the same aggregate value only one of the trajectories was recorded per NEAT run to maintain diversity. Several runs with varying random seeds for both problem domains generated the trajectory sets for our analysis. The recorded trajectories are all "complete" trajectories generated from successful policies. The trajectories are "complete" in that they all start from the same initial state and end with a transition to the goal state after some number of intermediate transitions.

There are roughly 1500 trajectories in the MC and 6000 trajectories in the ACRO data sets. The un-discounted aggregate values achieved by the trajectories in the sets ranges uniformly from -105 to -299 for MC and -70 to -999 for ACRO. In our experiments, for each run an identical set of trajectories is provided to both approaches. The trajectory sets vary from run to run and consist of randomly selected trajectories from the full generated sets.

4.4 Experiment Parameters

Unless stated otherwise, the learning parameters for all experiments are consistent for each domain. In our MC experiments RBF is used, and it is comprised of 25 features (k=25), $\lambda = 1.0$, and $\gamma = 0.9999$. For ACRO the Fourier basis functions are used, $k = 81$, $\lambda = 1.0$, and $\gamma = 0.9999$. These parameter settings were manually selected and were chosen because they produced the best observed overall learning performance for both methods across multiple experiments.

The reported results for all experiments are the average of 200 runs after 300 iterations (N=300). Additionally, we performed a paired t-test on the results from each approach to determine if differences in the results are statistically significant. We report the difference as being significant if the p-value < 0.05.



(a) Mountain Car (b) Acrobot

Fig. 2: Average converged policy performance for FQI and TFQI using increasing numbers of complete trajectories.

5 Results and Discussion

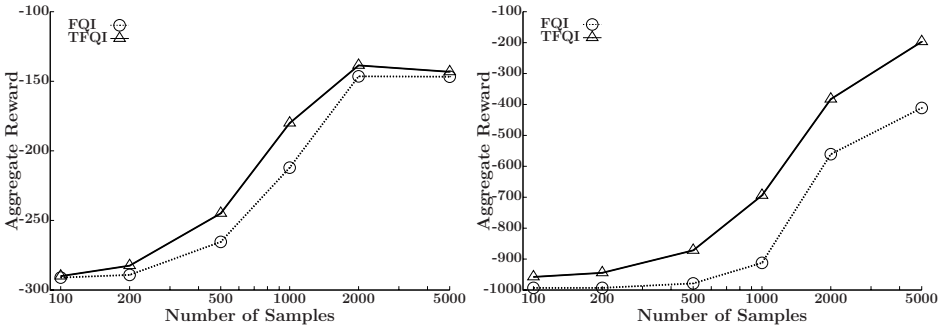
5.1 Learned Policy Performance

The primary goal of RL is to learn effective policies, therefore the most direct way of evaluating both FQI and TFQI is to compare them based upon their learned policy performance over the same sets of samples. Here we report the results from two sets of experiments that measure average un-discounted aggregate reward achieved by the learned policies after N iterations for both approaches. The experiment sets differ in the type of sample trajectories that are provided.

Our first set of experiments provides both approaches with increasing numbers of complete trajectories. This scenario is arguably the most realistic for an off-policy batch-mode algorithm. The learner is presented with some number of trajectories that demonstrate how to achieve a goal to varying degrees of success and the learner must derive the most effective policy from these samples. We show experiments increasing the number of provided trajectories from 1 to 10.

From the results given in Figure 2, it can be seen the enhanced use of trajectory data gives TFQI an advantage in learned policy performance. TFQI learns better policies on average than FQI in all of these experiments using the same data. The performance differences are significant in all cases except for the MC 10 trajectories experiment where both methods performed similarly. TFQI's advantage is most distinct in the ACRO problem where its average policy performance can be more than twice as good as that of FQI.

The second set of experiments evaluates policy performance when the methods are provided with fragments of complete trajectories. Although full trajectories may be more common, neither approach is dependent on the availability of such data. This set of experiments is designed to demonstrate whether or not TFQI can effectively exploit this more general form of trajectory data. The trajectory sets in these experiments were generated by randomly selecting trajectory fragments of random length from the repository. Fragments are added



(a) Mountain Car (b) Acrobot

Fig. 3: Average converged policy performance for FQI and TFQI using increasing numbers of samples from trajectory fragments.

until the aggregate number of samples reaches a desired threshold. Additionally, we guarantee that there are at least 3 fragments containing transitions to the goal state in each run. For this series of experiments we increased the aggregate size of the trajectory data incrementally from 100 to 5000 samples.

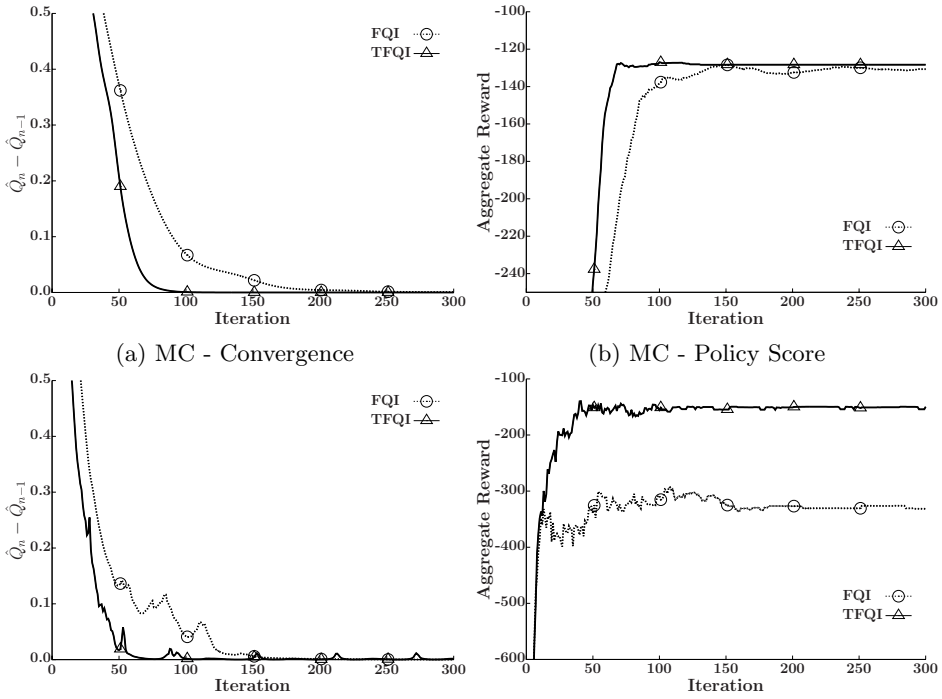
As shown in Figure 3 TFQI again demonstrates better learning performance than FQI on this more general form of trajectory data. All performance gaps shown in Figure 3b for the ACRO problem are significant as well as the 500, 1000, and 2000 sample runs for the MC problem. While the differences between the two approaches in these experiments are not quite as great as in the previous experiment there is still an improvement to be gained from utilizing this data and it can be significant.

Based on the results from these two experiments, when trajectory based data is available TFQI is clearly the preferable approach. The enhanced update allows TFQI to learn policies that are significantly better than FQI's on average. Improvements in performance are most pronounced when there are fewer samples, 2-5 trajectories or 500-1000 samples. Though we provide no theoretical bounds on sample efficiency of TFQI, these results suggest that our method can be more efficient than FQI in practice; a desirable trait when samples are costly to acquire.

5.2 Convergence

In addition to policy performance we compare FQI and TFQI on their convergence speed. Both methods iteratively refine $\hat{Q}$ by learning new sets of regression targets that estimate Q-values for the available samples. Through successive iterations, the difference in the regression targets from one iteration to the next, $\hat{Q}_n - \hat{Q}_{n-1}$, should approach zero and lead to a stable policy. Convergence speed is measured by the rate at which the difference approaches zero.

Figure 4 provides two sets of graphs for this analysis that show convergence of the $\hat{Q}$-function, Figures 4a and 4c, and average policy performance, Figure



(a) MC - Convergence

(b) MC - Policy Score

(c) ACRO - Convergence

(d) ACRO - Policy Score

Fig. 4: Convergence, (a) and (c), and policy, (b) and (d), performance graphs for two experiments comparing FQI and TFQI in both domains. For the convergence graphs, lower values approaching zero indicate improved performance. Alternatively, higher values in the policy score graphs are better.

4b and 4d. The graphs report the results of two experiments where TFQI and FQI were given 10 complete trajectories for each domain. For Figures 4a and 4c, $\hat{Q}_n - \hat{Q}_{n-1}$ is measured by the mean squared difference of the regression targets from one iteration to the next. In both experiments the convergence graphs show that TFQI appears to converge nearly 100 iterations earlier than FQI, demonstrating a significant speedup. This result can be attributed to the way TFQI is able to propagate long-term value through an entire trajectory within a single iteration. FQI, alternatively, must wait for value to propagate one transition step at a time per iteration. Figure 4b shows the learning performance of the methods from the same 10 trajectory experiment reported in Figure 2a. Interestingly, despite not deriving a significantly better policy on average, TFQI is able to derive a similar policy in far fewer iterations. For the ACRO domain this result is even more impressive. TFQI is able to learn a far better policy in fewer iterations using the same exact set of sample trajectories. In Figure 4c TFQI's convergence does exhibit some oscillations, however they do not have a significant impact on the quality of the policy as seen in Figure 4d.

5.3 Model Parameter Sensitivity

The success of any machine learning approach depends on the selection of an appropriate model for a given domain. RL is certainly no exception to this statement. Selecting an appropriate representation is critical and is the subject of intense research to find automatic methods [7]. Determining the most appropriate settings for any given model is difficult because there are often many different interrelated parameters that must be tuned. Our approach, while still very much dependent on model selection, utilizes trajectory data to calculate value in a way that is partially independent of the approximation model. It is plausible then that this approach may exhibit some degree of insensitivity to model choice.

In this section we provide a preliminary non-exhaustive set of experiments that compare the sensitivity of FQI and TFQI to the settings of various model parameters. The parameters that we change are the type of feature, ϕ, the number of features, k, and the shrinkage parameter for ridge regression, λ. For both domains we use the best performing settings on 10 complete trajectories as a baseline. We then deviate one of the parameters from the baseline settings in each experiment and measure the difference in the average policy performance, Δ, to determine sensitivity.

Table 1: Approximation model parameter sensitivity experiment results for Mountain Car. The first entry shows the baseline settings and performance for both methods. Subsequent entries show the policy performance and deviation from the baseline performance for varying model parameters for both methods. Significant results are **bold**.

ϕ	k	λ	FQI	FQI Δ	TFQI	TFQI Δ
RBF	25	1.0	-130.700	NA	-128.320	NA
RBF	25	0.1	-156.250	-25.550	**-146.460**	-18.140
RBF	25	10.0	-297.264	-166.564	**-238.044**	-109.724
RBF	16	1.0	-286.519	-155.819	**-200.724**	-72.404
RBF	36	1.0	-143.370	-12.670	**-134.870**	-6.550
Fourier	25	1.0	-194.260	-63.560	**-178.455**	-50.135

Table 1 provides the complete results for these experiments on the MC domain. In general TFQI exhibits less of a degradation in policy performance than FQI, supporting our hypothesis. The difference in degradation is most significant when the model parameters are changed to increase the generality of the model. Generalization is increased here when the number of features is decreased to 16 and when the λ parameter is increased to 10. This observation is somewhat expected given that the use of trajectory data by the TFQI approach reduces the impact of generalization error.

The results for the ACRO problem can be seen in Table 2. They are not as conclusive as the ones reported for MC. FQI actually shows a performance im-

provement when λ is reduced to 0.1 while learning over 10 complete trajectories. $\lambda = 1.0$ was found to be a good for learning over other amounts of trajectories and this result re-emphasises the difficulty in finding the best parameter settings. Still, even at this setting FQI does not match the performance of TFQI. Again, we can see that increasing the generalization of the model has significantly less of an impact on TFQI. When the number of features is increased to 256 both perform comparably. Given the relative number of features and samples (about 5000) this experiment might be suffering from model over-fitting. Both approaches struggle to find a successful policy when RBFs are used. Recall we limit the number of steps a policy has to achieve the goal to 1000. FQI is completely incapable of deriving a successful policy at these settings over 200 runs, whereas TFQI occasionally learned a competent policy. This last result provides further evidence of our hypothesis and shows that TFQI can learn in some scenarios where FQI cannot.

6 Related Work

Our approach is similar to learning from demonstration [2, 12] methods in that they also exploit trajectories. Learning from demonstration is a form of RL that takes examples, trajectories, demonstrated by known optimal or "good" behavior as input and attempts to use this data to derive policies that duplicate that behavior. While our approach could certainly take advantage of such data, it differentiates itself from most learning from demonstration work because no assumptions are made upon the quality of the policies used to generate the trajectories or that only a single policy was used to generate the trajectories.

The central idea behind our approach of using trajectory data to improve long-term value estimation is most similar to the idea of *augmented Bellman backups* described in [9]. In that work the authors define an update rule that uses either model induced value or value derived by a demonstration while performing a Bellman update of the value function. Our approach is a generalization of this

Table 2: Approximation model parameter sensitivity experiment results for Acrobot. The first entry shows the baseline settings and performance for both methods. Subsequent entries show the policy performance and deviation from the baseline performance for varying model parameters for both methods. Significant results are **bold**.

ϕ	k	λ	FQI	FQI Δ	TFQI	TFQI Δ
Fourier	81	1.0	-331.455	NA	**-149.690**	NA
Fourier	81	0.1	-274.495	56.960	-254.660	-104.970
Fourier	81	10.0	-951.565	-620.110	**-390.295**	-240.605
Fourier	256	1.0	-389.345	-57.890	-396.943	-247.253
RBF	81	1.0	-999.000	-667.545	-967.535	-817.845

idea and seamlessly intermixes the use of trajectory or value function model induced value in its update function. The intention of the *augmented Bellman backup* is to use demonstrations to accelerate the process of learning, whereas our intention is to improve the overall quality of learning. Like other learning from demonstration methods this approach assumes the availability of high quality demonstrations, unlike our approach.

Eligibility traces [14] are also similar in that they accelerate learning by applying TD updates to multiple samples along a trajectory. The idea was developed within an on-line on-policy context but it has been successfully ported to off-line off-policy algorithms. Nevertheless, the long-term values calculated by eligibility traces are derived exclusively from the approximated values of successor transitions, which makes them more susceptible to generalization error.

TFQI is not the first enhancement that has been made to the Fitted Q-Iteration framework since it was introduced in [4]. In [11] a multi-layer feed forward neural network architecture was substituted as the regression function, instead of the regression tree methods described in the original work, with improved empirical performance in standard RL benchmarks. More recently FQI was extended to work in continuous action space domains [1]. The extension proposed in this work is distinctly different and complimentary to these methods.

Finally, we must note that there are well known and explored theoretical issues when combing off-policy TD methods, such as FQI and TFQI, with function approximation. Off-policy TD methods are known to exhibit divergent behavior when paired with function approximators that do not provide bounds on approximation error [3]. Additionally, even when convergence can be ensured there are no known guaranteed bounds on the quality of the learned value function[5]. While the approach described in this paper does demonstrate empirical improvements, we do not claim to make a theoretical advancement regarding those issues.

7 Conclusion and Future Work

We have introduced TFQI and shown how its novel utilization of trajectory based data can reduce the impact of generalization error on the derivation of the value function. Our empirical analysis demonstrates how this generally available data can be further exploited to provide significant performance enhancements in the form of improved policies and quicker convergence. TFQI approach accomplishes all this while incurring no significant additional computational or memory costs over the standard FQI approach. One direction for future work is to explore the theoretical impact of this use of trajectory data.

Further, our empirical study found some evidence that trajectory based value updates provide some robustness to approximation model parameter settings. The value propagation information inherent in trajectories might then enable better modeling choices to be made a priori. We are additionally interested in investigating if trajectory data can similarly be harnessed to enhance automatic feature and model selection approaches.

Acknowledgements

This work is supported in part by grants from NSF (No. 0855204) and the AFRL Information Directorate's Visiting Faculty Research Program.

References

1. Antos, A., Munos, R., Szepesvári, C.: Fitted Q-iteration in continuous action-space mdps. In: NIPS (2007)
2. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robot. Auton. Syst. 57(5), 469–483 (May 2009)
3. Boyan, J.A., Moore, A.W.: Generalization in reinforcement learning: Safely approximating the value function. In: Advances in Neural Information Processing Systems 7. pp. 369–376. MIT Press (1995)
4. Ernst, D., Geurts, P., Wehenkel, L., Littman, L.: Tree-based batch mode reinforcement learning. Journal of Machine Learning Research 6, 503–556 (2005)
5. Kolter, J.Z.Z.: The fixed points of off-policy td. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 24, pp. 2169–2177 (2011)
6. Konidaris, G., Osentoski, S., Thomas, P.: Value function approximation in reinforcement learning using the Fourier basis. In: Proceedings of the Twenty-Fifth Conference on Artificial Intelligence. pp. 380–385 (August 2011)
7. Mahadevan, S.: Representation discovery in sequential decision making. In: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence. AAAI Press (2010)
8. Munos, R.: Error bounds for approximate value iteration. In: Proceedings of the 20th national conference on Artificial intelligence - Volume 2. pp. 1006–1011. AAAI'05, AAAI Press (2005)
9. Price, B., Boutilier, C.: Accelerating reinforcement learning through implicit imitation. Journal of Artificial Intelligence Research 19, 569–629 (2003)
10. Puterman, M.L.: Markov decision processes: discrete stochastic dynamic programming, vol. 414. Wiley-Interscience (2009)
11. Riedmiller, M.: Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In: Machine Learning: ECML 2005, pp. 317–328. Springer (2005)
12. Schaal, S.: Learning from demonstration. In: Advances in Neural Information Processing Systems 9. MIT Press (1997)
13. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
14. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning). The MIT Press (March 1998)
15. Tanner, B., White, A.: RL-Glue : Language-independent software for reinforcement-learning experiments. Journal of Machine Learning Research 10, 2133–2136 (September 2009)