Small-sample Reinforcement Learning - Improving Policies Using Synthetic Data - preprint.pdf

Stephen W. Carden
James Livsey, US Census Bureau

Available at: https://works.bepress.com/stephen_carden/13/
Small-sample Reinforcement Learning: Improving Policies Using Synthetic Data

Stephen W. Carden*

Department of Mathematical Sciences, Box 8093
Georgia Southern University
Statesboro, GA 30460
scarden@georgiasouthern.edu

James Livsey

Center for Statistical Research and Methodology
U.S. Census Bureau

Abstract

Reinforcement learning (RL) concerns algorithms tasked with learning optimal control policies by interacting with or observing a system. In computer science and other fields in which RL originated, large sample sizes are the norm, because data can be generated at will from a generative model. Recently, RL methods have been adapted for use in clinical trials, resulting in much smaller sample sizes. Nonparametric methods

*Corresponding author.
are common in RL, but are likely to over-generalize when limited data is available. This paper proposes a novel methodology for learning optimal policies by leveraging the researcher’s partial knowledge about the probability transition structure into an approximate generative model from which synthetic data can be produced. Our method is applied to a scenario where the researcher must create a medical prescription policy for managing a disease with sporadically appearing symptoms.

**Keywords:** decision theory, sample size, fitted Q-iteration, marginalized transition models, reinforcement learning, nonparametric
1 Introduction

A Markov Decision Process (MDP) is a stochastic process, usually considered in discrete time, in which a system can reside in one of a given set of states, $S$. The term *agent* is used to describe the decision-making entity that interacts with the environment. At each time step, the agent observes the current state and chooses among a set of actions, $A$. The current state and action determine the probabilities that govern transitions to the next state. After the action is chosen, a real-valued reward is received. For full generality, the reward is allowed to be a random variable with a distribution depending on the current state and action. After the system transitions to a new state, the agent again makes an observation and chooses an action, and this process repeats. A *policy* is a function mapping states to actions; simply put, a policy tells the agent “when in state $s$ choose action $a$”. The goal is to find a policy that maximizes reward.

It is useful to classify an MDP with regards to how much knowledge the researcher has about the system, which is closely tied to how observations can be sampled from the process [9]. The strongest assumption is that of *complete knowledge*, in which all transition probabilities and the expected values of all rewards are known exactly. For small problems, optimal policies can be calculated exactly [14]. For many practical applications however, transition probabilities and reward distributions are not known exactly, but data from the system can be observed or simulated. Reinforcement learning (RL) is the field concerned with using data to learn optimal or near-optimal policies [18]. The most traditional setting is the *online simulation model* in which observations are collected sequentially as the agent interacts with the system in an unbroken chain of experience [19]. This model is appropriate for situations in which the researcher can observe the agent interacting with the system, but does not know enough about
the system dynamics to create a simulation or have enough control to change the system’s state at will. On the other hand, the *generative model* is used when the researcher has enough knowledge about the system to create a simulation, or enough control to initiate the system from any arbitrary state [10]. With this model, the researcher has access to observations from any state and as many observations as desired. Therefore, the limitation in the search for an optimal policy is the computational load associated with extracting information from the observations.

Previous research in sample-size related RL techniques have dealt almost entirely with generative models. The work includes meta-algorithms that seek to reduce computational expense by identifying small samples from which a near-optimal policy can be learned [15] and algorithm modifications that seek to reduce the computational cost as more observations are included [2]. The generative model assumption is often reasonable in disciplines where reinforcement learning was developed (computer science, artificial intelligence, robotics, etc.); however, there has also been growing use of reinforcement learning in clinical trials.

The popular Sequential Multiple Assignment Randomized Trials (SMART) design [11] is essentially the exploration phase of RL applied to a clinical setting, where the actions are medical or behavioral interventions. In the context of a SMART design, the assumption of a generative model is too strong. For example, the state space for a study participant might consist of health or demographic variables. The researcher cannot manipulate these variables in order to make observations from arbitrary states. Furthermore, the possibility of creating a generative model by simulation is out of the question because the unknown transition probabilities, which describe how subjects will react to the interventions, are the point of doing the study in the first place. Because the
researcher must only use observations from the states which are represented by the study participants, the number of observations resulting from a SMART or similar clinical trial will be orders of magnitude less than what is typical in RL.

For example, consider the following sample sizes in experiments where RL was used on some standard benchmark problems [4]. The first experiment used 2,010 observations to solve a stochastic discrete MDP with 22 state-actions, the second used 58,090 observations to solve a deterministic MDP with two continuous state variables and a single binary action variable, the third used 193,237 observations to solve a deterministic MDP with 4 continuous state variables and a single binary action variable, and so on. For comparison, a SMART trial comparing antipsychotic interventions had an overall sample size of 450 [17]. It is in this type of situation our methods can help produce improved policies over existing RL techniques.

The literature on RL with limited data is under-developed. There is work on how to select the best abstraction of the state space when limited data is available [8] and methods for quantifying how uncertainty from limited data propagates through the learning algorithm [6], but there is a need for further research in extracting good policies from small data sets. Our research question is whether the performance of a policy learned from a small number of observations can be beat by a policy making use of an approximate generative model. We will use a new technique for using partial knowledge about an MDP with limited data to improve the policies obtained. We propose a methodology that can use the available observations to fit a model for the state transitions and use this model to generate “synthetic” observations for the underrepresented regions of the state-action space. We show that adding synthetic data to the original data will reduce the amount of smoothing required by the RL methods and ultimately lead to better policies.
1.1 Reinforcement Learning Background

A central component of many RL algorithms is the approximation of the state-action value-function. The state-action value-function \( Q : S \times A \rightarrow \mathbb{R} \) is implicitly defined by the state transition structure and reward random variables, but the structure is not a priori known. For this reason, approximating the \( Q \) function works best with highly flexible regression methods such as decision trees [4], neural networks [5], or nonparametric methods such as kernel smoothing [3]. Of key importance is the amount of smoothing or generalization produced by the regression method. Generally, the data set will not contain an instance of every possible state and action combination, so the regression method must generalize the information contained in the data set to unobserved situations. If sizable regions of the state-action space are not represented in the data, then the regression method will require an unhealthy amount of smoothing to approximate the \( Q \) function there, and the approximation will likely be poor.

The ideal solution would be to observe more data from the unrepresented state-actions, but this is often not possible. Consider a pharmacological study in which a new medication’s ability to prevent symptoms is being investigated. The conditions of the patients form a state space, and the treatments form an action space. It may not be possible to find patients such that the entire state space is adequately represented. Our method of including synthetic data will reduce the amount of smoothing required by the regression method and produce better policies.

In this paper, we will consider the case of MDPs where key state variables are binary and fit an appropriate model to estimate transition probabilities for the corresponding two-state Markov chain. Marginalized Transition Models (MTM) [7] [1] are well-suited to this purpose. This class of models was chosen for its flexibility; MTM is a combination of generalized linear marginal regression and
a model that characterizes serial dependence between observations.

In Section 2, Markov Decision Processes will be defined and an algorithm for finding an approximately optimal policy will be outlined. Section 3 introduces the concept of synthetic data and details how Marginalized Transition Models can be used to generate synthetic data. Section 4 gives a proof of concept example. Finally, Section 5 concludes.

2 Markov Decision Processes

The basics of Markov Decision Processes will now be described. For further introduction, we recommend the texts by Puterman [14] and Ross [16]. This paper will consider MDPs where states and actions are represented by realvalued scalars or vectors. Though algorithms do exist for continuous states and actions [3], we will restrict our attention to finite state and action spaces. For ease of exposition, we will assume the states and actions have been transformed to take integer values, though this is not necessary. Let \( S \), the state space, be a finite subset of \( \mathbb{Z}^{d_S} \) where the dimension \( d_S \) is the number of state variables. Let \( A \), the action space, be a finite subset of \( \mathbb{Z}^{d_A} \) where the dimension \( d_A \) is the number of action variables. Note that \( S \times A \) is a subset of \( \mathbb{Z}^d \) where \( d = d_S + d_A \). Elements of this space will be written \((s,a)\) to make the distinction between state and action clear, but for all computational purposes they are best regarded as integer-valued vectors. Transitions between states are governed by a function \( P : S \times S \times A \rightarrow [0,1] \) where \( P(s,s',a) \) denotes the probability of transitioning from state \( s \) to state \( s' \) when action \( a \) is used. For each state and action \((s,a) \in S \times A\), there is a bounded random variable reward \( R(s,a) \).

The Markov Decision Process proceeds as follows: the process begins in some state \( s \in S \) and the agent chooses an action \( a \in A \). A random reward \( R(s,a) \) is received and the system transitions to a new state in accordance with the
probability mass function $P(s, \cdot, a)$. The system progresses to the next time step, and this scenario repeats from the new state. The reward distributions and transition probabilities are assumed to have the Markov property: they are conditionally independent of past observations given the present observation.

A policy $\pi$ is a function $\pi : S \to A$. Intuitively, it tells the agent which action to use for any state the system may be in. We seek a policy that will maximize some long-term measure of reward. In some settings, policies are allowed to be stochastic or non-stationary, but we will restrict our attention to stationary, deterministic policies. Once a policy is chosen, the action that will be used in each state is determined and the MDP reduces to a Markov chain. Formally, define the transition probabilities under a policy as $P_\pi(s, s') := P(s, s', \pi(s))$. This simply says that under a policy, the actions used are no longer freely chosen but determined by that policy. Notice that $P_\pi$ represents transition probabilities for a Markov chain on the space $S$.

Let $\gamma, 0 < \gamma < 1$, be a discount factor. $\gamma$ is a parameter that determines the relative worth of rewards that will be received in the future as compared to rewards received in the present. Values of $\gamma$ that are close to one can be interpreted as representing patience in an agent who assigns nearly as much value to future rewards as it does to present rewards. Conversely, $\gamma$ near zero represents a greedy agent who is more interested in short-term reward. Let $(s_t, a_t)$ be the (random) state and action at time $t$. Under a policy $\pi$, the value of a state $s$ is defined under the expected total discounted reward criterion:

$$V^\pi(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) | s_0 = s \right].$$

A policy $\pi^*$ is optimal if it satisfies $V^{\pi^*}(s) = \sup_{\pi} V^\pi(s), \forall s \in S$.

There are multiple classes of solution methods, and the best method depends on how much information is available about the MDP. Under the com-
plete knowledge assumption an optimal policy can be recovered using dynamic programming in a straightforward manner if the number of states and actions is small [14]. Under the generative model assumption, then observations from any arbitrary state can be generated and passed through a reinforcement learning algorithm to find an approximately optimal policy.

In practice, a researcher may find that neither of these assumptions are met. Complete knowledge of all transition probabilities and reward means is a strong condition, but the researcher will often have partial knowledge about the system dynamics. For example, while the transition probabilities are typically not known exactly, the researcher may have reason to assume a parametric model. Furthermore, it is often the case that the researcher defines the reward structure, so it is common for reward means to be known. This paper is proposing a sort of hybrid: a solution method that uses partial knowledge about transition probabilities and reward means to create an approximate generative model. Using the approximate generative model to produce synthetic observations will improve the policy obtained by reinforcement learning. Figure 1 illustrates the situation for which our method is intended.

Our hybrid method will use a value-function based reinforcement learning algorithm. The state-action value function, often termed the \emph{Q-value} function [19], under a policy $\pi$ is defined as

$$Q^\pi(s,a) := E[R(s,a)] + \gamma \sum_{s' \in S} V^\pi(s') P(s,s',a).$$

Intuitively $Q^\pi(s,a)$ is the average value obtained if we start in state $s$, utilize action $a$, and then follow policy $\pi$ thereafter. Consider the Q-values associated with an optimal policy $\pi^*$,

$$Q^*(s,a) := Q^{\pi^*}(s,a).$$
If we can learn the values \( Q^*(s,a) \) for all \((s,a)\), then an optimal policy can easily be recovered by setting

\[
\pi^*(s) = \operatorname{argsup}_{a \in A} Q^*(s,a).
\]

### 2.1 Fitted Q-Iteration

Fitted Q-Iteration is a learning algorithm for estimating the optimal \( Q \)-values, \( Q^*(s,a) \), of an MDP. Fitted Q-Iteration learns from a set of 4-tuples, each representing an observation of the MDP, \( \mathcal{F} = \{(s_k, a_k, r_k, u_k)\} \). Here \( s_k \) is the state at observation \( k \), \( a_k \) is the action, \( r_k \) is the reward, and \( u_k \) is the state transitioned to. The order of the observations - whether they come from a single observation trajectory or multiple observations - is irrelevant. Fitted Q-Iteration also requires a discount factor \( \gamma \), a regression method for producing state-action value estimates \( \hat{Q}_n(s,a) \), and a stopping condition. Fitted Q-Iteration proceeds as follows.

1. Initialize state-action value estimates \( \hat{Q}_0(s,a) = 0 \) for all \((s,a)\) \( \in S \times A \).
2. For \( n > 0 \), repeat the following until the stopping condition is met.
   
   (a) Build a training set \( \mathcal{T}S = \{(i_k, o_k)\} \) of inputs and outputs where

   \[
i_k = (s_k, a_k) \\
o_k = r_k + \gamma \max_{a \in A} \hat{Q}_{n-1}(u_k, a).
   \]

   (b) Use the regression method to create \( \hat{Q}_n(s,a) \) from the training set.

Pseudocode for Fitted Q-Iteration is given by Algorithm 1. Notice that the input or explanatory variables do not change across iterations; only the output
or response variables do. Fitted Q-iteration effectively reduces reinforcement learning to iterated regression.

It will be rare for every \((s, a) \in S \times A\) to be represented in the training set. It is possible that large regions of \(S \times A\) are not represented. Also, because the shape of the function \(Q(\cdot)\) is unknown, the regression method should be highly flexible. This flexibility typically comes from a piece-wise approach, smoothing, or local methods using data that are “nearby” according to some metric. Local approximations will be poor if the data from which the approximation is built is too sparse or outside a smoothing window.

3 Synthetic Data Generation

In the clinical setting, budget and time constraints restrict the amount of data that can be collected. In situations involving rare diseases, hazardous collection conditions, expensive medical studies, or long-term longitudinal studies, data may be difficult or costly to obtain. In this situation - where there is not enough data to believe our state-action space is adequately represented for Fitted Q-iteration - we will use an approximate generative model to improve the learned policy. We estimate the transition probabilities to produce an approximate generative model so that additional data can be generated. We assume throughout that the transition probabilities can be modeled reasonably well through the values of present state and action variables. The type of model used to estimate transition probabilities depends on whether the state and action variables are binary, discrete, continuous, etc. In the following example we will restrict our attention to the case where the key state variable for which transition probabilities need to be modeled is binary, and there is a single binary action variable. It should be noted that the action could be multivariate or continuous while still using this transition probability estimation method, but other aspects of
the Fitted Q-iteration algorithm would change.

3.1 Marginalized Transition Models

In this section we review a stochastic model, Marginalized Transition Models (MTM) [7] [1], for estimating transition probabilities and creating an approximate generative model. This is but one approach for estimating these probabilities. For each application of our method, an appropriate model should be chosen which fits the structure of the MDP in question. In MTM, transition probabilities depend implicitly on explanatory variables via two relations: the marginal mean of the response variable, and the correlation with the most recent response variable. Both the marginal mean and strength of correlation depend on the values of the explanatory variables.

Suppose that observations are being made on $m$ individuals, and that individual $i$ is observed for $n_i$ time steps. Let $Y_{i,j}$ for $i = 1, \ldots m$ and $j = 1, \ldots n_i$ represent the binary responses. For each individual and time step, a vector of covariates $X_{i,j}$ is observed and recorded. The first assumption of the model is that the marginal mean of $Y_{i,j}$, denoted by $\mu_{i,j}^M = E[Y_{i,j}|X_{i,j}]$, is specified by

$$\text{logit}(\mu_{i,j}^M) = X_{i,j}^T \beta$$

(1)

where $\beta$ is a vector of fixed but unknown parameters measuring the influence of the explanatory variables on the marginal mean and the logit function is $\text{logit}(p) = p/(1 - p)$.

In addition to being influenced by the covariates, the second assumption is that there is a serial dependence between values of the response variable. Because the variable is binary, a two-state Markov chain can be used to model
the serial dependence. Define

\[ p_{i,j,0} = P(Y_{i,j} = 1|Y_{i,j-1} = 0) \]
\[ p_{i,j,1} = P(Y_{i,j} = 1|Y_{i,j-1} = 1) \]

to be the transition probabilities defining the Markov chain. Then the marginal means must satisfy

\[ \mu_{i,j}^M = p_{i,j,1} \mu_{i,j-1}^M + p_{i,j,0} (1 - \mu_{i,j-1}^M). \] (2)

The values \( p_{i,j,0} \) and \( p_{i,j,1} \) will depend on how strong the correlation between successive observations is and may be stronger or weaker depending on the values of the covariates. The strength of correlation is modeled through the odds ratio

\[ \Psi_{i,j} = \frac{p_{i,j,1}/(1 - p_{i,j,1})}{p_{i,j,0}/(1 - p_{i,j,0})}. \] (3)

The odds ratio is related to the conditional mean of the response,

\[ \mu^C = E[Y_{i,j}|X_{i,j},Y_{i,j-1}], \]

by

\[ \logit(\mu^C_{i,j}) = \logit(p_{i,j,1}) + \log(\Psi_{i,j})Y_{i,j-1} \]

and is related to the covariates by

\[ \log(\Psi_{i,j}) = Z_{i,j}^T \alpha. \] (4)

Here \( Z_{i,j} \) is, in general, a subset of \( X_{i,j} \). \( \alpha \) is a vector of fixed but unknown parameters relating the strength of correlation to the explanatory variables.

For the purposes of reinforcement learning, we are mainly interested in the
transition probabilities $p_{i,j,0}$ and $p_{i,j,1}$. These are implicitly defined by $\alpha$ and $\beta$, and can be solved for using equations (1) through (4).

4 Example

In this section, we use our proposed method combining reinforcement learning and MTM to find improved policies for an example in which the transition probabilities meet the assumptions of MTM exactly.

This problem is inspired by the modeling of schizophrenia symptoms using MTM by Heagerty [7]. Once one can estimate the probability that a patient will present symptoms, a natural extension is how that knowledge can be used by a physician in prescribing medication. Suppose that we are treating a chronic disease with symptoms that subside and reappear. At regular time intervals, the patient will be inspected and symptoms will be classified as present or not present. The physician can prescribe a drug that has some success in preventing symptoms but has serious side effects of its own, so it should not be over-prescribed.

We will assume the appearance of symptoms at the next time step is related to four variables: time since last appearance of symptoms, time since last receiving medication, whether the patient currently has an active prescription, and whether the patient currently has symptoms. What prescription policy should the physician follow in order to prevent symptoms while also limiting the side effects of the drug?

At times $t = 1, 2, \ldots$ the patients are observed. The first explanatory variable $X_{1,t} \in \{1, 2, \ldots, 9, 10\}$ represents how long it has been since symptoms were last observed. We assume that effects from time periods longer than ten units are no larger than the effect from ten units. The second explanatory variable $X_{2,t} \in \{1, 2, \ldots, 9, 10\}$ represents how long it has been since the patient received
a dose of medication. Again, we assume that after ten time units the drug has been fully flushed from the system and the additional effect (or lack thereof) beyond ten time units is insignificant. The third and last explanatory variable \( X_{3,t} \in \{0, 1\} \) indicates whether a prescription is made at time \( t \). Prescriptions last one time unit. The binary response variable \( Y_t \in \{0, 1\} \) represents whether symptoms develop as a result of the values of the explanatory variables at time \( t \). Notice that \( Y_t \) will not be observed by the physician until the next observation at time \( t + 1 \).

One may ask why the variable \( X_{3,t} \) is necessary when the variable \( X_{2,t} \) could be adjusted so that it can take on a zero value. \( X_{3,t} \) is present because the medication can be expected to have a significant effect when the patient is actively taking medication. This effect would diminish rapidly when the patient stops taking medication and then diminish slowly after that. Additionally, \( X_{3,t} \) represents the action chosen by the physician and is the action variable for the reinforcement learning model.

For the purposes of a reinforcement learning algorithm, variables \( X_1, X_2, \) and \( Y \) are regarded as state variables, and \( X_3 \) is the action variable. Thus the state at time \( t \) consists of the state variables known to the physician as influencing the symptoms observed at time \( t \). Also, the symptoms observed at time \( t \) are an effect of observations and actions made at time \( t - 1 \), hence the state at \( t \) is \( s_t = [X_{1,t-1}, X_{2,t-1}, X_{3,t-1}, Y_{t-1}] \). Upon observation of this state, the physician chooses whether to issue a prescription at time \( t \) or not, so the action \( a_t \) is the assignment of \( X_{3,t} \) to either zero or one. Table 1 summarizes each variable and lists its role with respect to MTM and RL.

For this example, we assume the appearance of symptoms follows an MTM model. The true (but unknown to the learning algorithm) MTM parameters are \( \beta = [.1 \ 3 \ -1] \) (corresponding to variables \( X_1, X_2, \) and \( X_3 \)) and \( \alpha = 1 \).
(corresponding to variable $X_1$). These values were chosen because they are reasonable and generate a problem with a non-trivial solution.

Table 2 defines the reward structure. The reward is a deterministic function of whether the patient experiences symptoms and whether medication is prescribed. The base reward is 0 if the patient experiences symptoms and 1 if the patient does not. This is modified by subtracting $.75$ if the patient is prescribed medication, representing the side effects of the medication.

With this model in mind, the results of the following experiment are offered.

1. Generate a small amount of data from the original process.
2. Use Fitted Q-iteration to learn a first policy from this original data. Evaluate performance of this first policy.
3. Use the original data to fit a Marginalized Transition Model. From this estimated model, generate an additional amount of “synthetic” data.
4. Add the synthetic data to the original data, and use Fitted Q-iteration to learn a second policy.
5. Compare the performance of the two policies.

From the original process with the true MTM parameters, data consisting of 10 observations on each of 3 patients is simulated. The initial values of the covariates for time $t = 0$ are selected randomly and uniformly from their possible values. During this data generation, the physician has an 80% chance of dosing the patient when symptoms are present, and a 20% chance of dosing when symptoms are not present.

Step 2 uses this original data to learn a policy. In this application, the regression method used in Fitted Q-iteration is Nadaraya-Watson kernel smoothing [20] [12]. This method calculates the value of a state-action as a weighted average of nearby (according to some specified metric) observed state-action values.
The exact values of the weights are determined by the kernel function, which is a truncated Gaussian density. The kernel bandwidth, a parameter that controls the amount of smoothing, is tested at multiple values to find the appropriate level of smoothing.

Once a policy is produced, the next step is to evaluate its performance. 1000 patients are randomly initialized and treated according to the policy. Each patient is simulated for 100 time steps, resulting in a total of \(10^5\) observations. The rewards for each patient and time step are recorded and averaged to yield an estimate of long-term average performance.

In step 4, the original 30 observations are fed to a maximum likelihood algorithm that estimates MTM parameters. The parameters estimated from the data are \(\hat{\beta} = [0.174, 0.384, -0.634]\) and \(\hat{\alpha} = 1.196\). Using these estimated parameters, an additional 30 observations (10 time steps for each of 3 patients) are generated. We consider these observations to be “synthetic” because they do not come from the true process. These synthetic observations are combined with the original observations to form a data set of size 60.

The combined set of original and synthetic observations are passed to the same learning algorithm used in step 2, and the resultant policy is tested in the same manner. Figure 2 shows the long-term reward for the policies obtained using bandwidths in increments of 0.05. The horizontal axis is the value of the kernel bandwidth, which controls how much smoothing occurs when estimating state-action values. The vertical axis shows the long-term average reward when the policy is tested over \(10^5\) observations. The performance of the policy calculated using only the original data is marked with circles, and the performance of the policy calculated also using the synthetic data is marked by plus signs. One can see that the best performance is obtained at a bandwidth of 0.4 using the combination of original and synthetic data. This confirms our hypothesis.
that it is possible to improve policies obtained from reinforcement learning using synthetic data from a process such as Marginalized Transition Models.

For comparison, the graph displays the performance of two benchmark policies as well. The dashed line is the trivial policy of never dosing the patient, which performs poorly. The dashed-dotted line is the policy of always dosing the patient. Notice that only using the original data, the policy does no better than simply always prescribing medication. Only once the synthetic data has been added can the policy make prescriptions in a judicious way that beats the trivial policy.

It is clear from the graph that good policies are produced within a small range of bandwidth values, about .35 to .45. When the bandwidth is smaller, the function $\hat{Q}_n(s,a)$ is over-fitting the data. If the bandwidth is larger, it is over-smoothing; using data from states and actions is too distant and dissimilar to the point being approximated.

Because the original data consisting of 30 observations is quite sparse relative to the entire state-action space of 800 elements, it is expected that some state-actions will contain no observations within the kernel bandwidth. If this occurs, the kernel regression estimate at that point will involve a division by zero, resulting in a “NaN” (not a number) value. As the bandwidth increases, it seems reasonable that the number of NaN values would decrease, though at the risk of over-smoothing. One would also expect that when the additional 30 synthetic observations are added the number of NaNs would decrease for any particular bandwidth level. Figure 3 displays the relationship between bandwidth and the number of state-actions with undefined value estimates.

In this example, the best performance was found by producing 30 synthetic observations. Because an arbitrary amount of data can be produced, one may wonder why the researcher would not generate a very large amount of data to
pass to the learning algorithm. In our experiments, we found that this was actually detrimental to the performance of the resultant policy. In a separate experiment, we produced differing amounts of synthetic data. For each amount of synthetic data, we combined it with the original data, learned a policy, and then evaluated the performance of that policy. Figure 4 summarizes the result of this experiment. The horizontal axis is the amount of synthetic data produced from the approximate generative model, as measured in multiples of the number of original observations. The vertical axis is the long-term performance of the policy learned from the corresponding amount of synthetic data. Notice that the best performance comes from using the same amount of synthetic data as original data, and that large amounts of synthetic data result in policies with inferior performance. The reason is that synthetic data, coming not from a true generative model but instead an approximate generative model, represents a probability transition structure different from the structure that produces the original data. When too much synthetic data is used, the learning algorithm produces policies that are optimal for a problem with a different probability transition structure. The original data, representing the true probability transition structure, gets dwarfed by the synthetic data. Therefore, it seems best to use just enough synthetic data to aid in smoothing $Q$-values, and no more.

5 Conclusion

The generative model paradigm is useful in many disciplines that use reinforcement learning, but does not extend to the clinical setting and SMART designs. This paper has presented a method for taking advantage of a researcher’s partial knowledge about a system by estimating parameters for an assumed model for transition probabilities, and using that model as an approximate generative model to produce synthetic observations. In an example, we gave one possible
transition probability model, namely Marginalized Transition Models. When
the synthetic observations are combined with the original data, the learned pol-
icy is superior to the policy obtained when only the original data is used.

The exposition of this paper has emphasized concepts over technical details.
For readers interested in implementation, MATLAB code for the example pre-
sented in this paper is available at the author’s webpage\footnote{https://sites.google.com/a/georgiasouthern.edu/stephen-carden/research}. This code contains
routines for data generation from an MTM process, estimation of a value func-
tion with Fitted Q-iteration, and evaluation of the policy implicitly defined by
the estimated value function. The only routine not included is one for estimating
MTM parameters from data, for which we used the R package \texttt{mtm}.

The idea of using synthetic data from approximate generative models is new,
so future work in this area is open and we believe could be quite fruitful. This
paper has shown the most modest of results: that synthetic data can improve
the learned policy, and therefore approximate generative models are worthy
of further investigation. A logical next step would be a theoretical analysis
of the bias and variance of the long-term reward of the policy learned using
synthetic data. On the applied side, there is still the question of how to use
this technique to its fullest extent and narrow the gap between the learned
policy and the optimal policy. Open questions include the optimal amount of
synthetic data to generate, which portions of the state-action space synthetic
data should be generated from, robustness to misspecification of the transition
probability model, and the usefulness of transition probability models beyond
MTM, including models capable of dealing with multi-dimensional decisions.
References


Figure Captions

**Figure 1:** A decision tree illustrating the contribution of our method.

**Figure 2:** A graph showing policy performance at multiple bandwidths in intervals of .05.

**Figure 3:** A graph showing the number of “NaN” (not a number) values in the value-function at multiple bandwidths in intervals of .05.

**Figure 4:** A graph showing the performance of the learned policy across varying amounts of synthetic data. The amount of synthetic data is calculated as a multiple of the amount of the original data.
Do you know the system dynamics?

- **Yes**: Solve with dynamic programming.
- **No**: Do you have or can you obtain data about the system?
  - **Yes**: Is it enough data to produce a good policy?
    - **Yes**: Use standard reinforcement learning.
    - **No**: You need more information.
  - **No**: Do you have some knowledge about transition probabilities?
    - **Yes**: Our synthetic data idea may be helpful.
    - **No**: You need more information.

Fig. 1
Algorithm 1 Pseudocode for Fitted Q-iteration.

Input: $\mathcal{F}$, a set of $M$ four-tuples; a regression method; a stopping condition; $\gamma$, a discount factor.
Initialize $\hat{Q}_0(s, a) = 0$.
Initialize $n = 0$.
for $k=1:M$ do
  $i_k = (s_k, a_k)$.
end for
while Stopping condition not met do
  $n = n + 1$
  for $k=1:M$ do
    $o_k = r_k + \gamma \max_{a \in A} \hat{Q}_{n-1}(u_k, a)$.
  end for
  Build next regression model $\hat{Q}_n(\cdot)$ using \{$i_k | k = 1, \cdots, M$\} as explanatory data and \{$o_k | k = 1, \cdots, M$\} as response data.
end while
Table 1: Summary of variables and relation to Marginalized Transition Models and Reinforcement Learning.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Describes</th>
<th>Takes values</th>
<th>MTM Role</th>
<th>RL Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>Time since symptoms</td>
<td>${1, 2, \ldots, 9, 10}$</td>
<td>Explanatory</td>
<td>State</td>
</tr>
<tr>
<td>$X_2$</td>
<td>Time since prescription</td>
<td>${1, 2, \ldots, 9, 10}$</td>
<td>Explanatory</td>
<td>State</td>
</tr>
<tr>
<td>$X_3$</td>
<td>Prescription made</td>
<td>${0, 1}$</td>
<td>Explanatory</td>
<td>Action</td>
</tr>
<tr>
<td>$Y$</td>
<td>Symptoms present</td>
<td>${0, 1}$</td>
<td>Response</td>
<td>State</td>
</tr>
</tbody>
</table>
Table 2: The reward structure.

<table>
<thead>
<tr>
<th></th>
<th>No prescription made</th>
<th>Prescription made</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symptoms not present</td>
<td>$r = 1$</td>
<td>$r = .25$</td>
</tr>
<tr>
<td>Symptoms present</td>
<td>$r = 0$</td>
<td>$r = -.75$</td>
</tr>
</tbody>
</table>
Policy Performance vs. Kernel Bandwidth

Fig. 2
Number of NaNs in the Final Policy vs Bandwidth

Kernel Bandwidth

Number of NaNs

Original Data
Original+Synthetic Data

Fig. 3
Policy Performance vs Amount of Synthetic Data

Fig. 4