Bayesian Active Learning With Basis Functions

Ilya O. Ryzhov and Warren B. Powell
Operations Research and Financial Engineering
Princeton University
Princeton, NJ 08544

***Abstract*—A common technique for dealing with the curse of dimensionality in approximate dynamic programming is to use a parametric value function approximation, where the value of being in a state is assumed to be a linear combination of basis functions. Even with this simplification, we face the exploration/exploitation dilemma: an inaccurate approximation may lead to poor decisions, making it necessary to sometimes explore actions that appear to be suboptimal. We propose a Bayesian strategy for active learning with basis functions, based on the knowledge gradient concept from the optimal learning literature. The new method performs well in numerical experiments conducted on an energy storage problem.**

I. Introduction

Consider an infinite-horizon dynamic programming problem where the value $V(S)$ of a state $S \in \mathcal{S}$ is given by Bellman's equation,

$$V(S) = \max_{x \in \mathcal{X}} \mathbb{E}\left[C(S,x,W) + \gamma V(S') \mid S, x\right] \qquad (1)$$

and the optimal policy consists of making the decision that maximizes the right-hand side of (1) for the current state. The quantity $C(S,x,W)$ denotes a reward (possibly depending on a random variable W) received for taking action x while in state S. The state $S' = S^M(S,x,W)$ denotes the next state of the dynamic program after taking this action; the transition function S^M is sometimes referred to by the name "system model" in the literature. The expectation in (1) is thus taken over the distribution of W. If the number of states and actions is small, we can use the value iteration algorithm [1] to find the true value function V by iteratively solving

$$V^n(S) = \max_{x \in \mathcal{X}} \mathbb{E}\left[C(S,x,W) + \gamma V^{n-1}(S') \mid S, x\right] \quad \forall S \in \mathcal{S}. \qquad (2)$$

However, it is easy to envision problems where this algorithm is computationally intractable. We are particularly interested in problems where the state space $\mathcal{S}$ is continuous. This makes it impossible to solve (2) for all $S \in \mathcal{S}$, but it may also be impossible to compute the expectation over the distribution of W, even for a fixed state S.

Consider a simple energy storage problem where we can buy energy at the spot price, use it to charge a battery, then wait until the price goes up and discharge back to the grid. Energy prices are famously volatile, and there is an entire literature that uses stochastic differential equations (essentially stock price processes) to model them [2]. The state variable in this problem must consist of all the information we need to make a decision, which necessarily must include the current price of energy. The price is a continuous quantity; furthermore, the transitions between prices are also continuous (if we use an SDE model). Thus, even the simplest version of this problem immediately runs into the well-known curse of dimensionality.

In this situation, we can make use of techniques from reinforcement learning [3], neuro-dynamic programming [4], or approximate dynamic programming [5], [6]. One very popular class of methods of this sort is known as approximate value iteration. Given that we are in a state S^n, we first compute an approximate observation

$$\hat{v}^n = \max_x C(S^n, x) + \gamma V^{n-1}\left(S^{M,x}(S^n, x)\right) \qquad (3)$$

where V^{n-1} is an approximation of the true value function V, and $S^{x,n} = S^{M,x}(S^n, x)$ is the *post-decision state*. This concept, first introduced by [7] and extensively discussed in [6], describes the change in our state immediately after we have made a decision, but before any exogenous information has been observed. Thus, $S^{M,x}$ is a deterministic function of state and action that allows us to avoid having to compute the expectation in (2). Having calculated (3), we view $\hat{v}^n$ as an observation of the value of being in the previous post-decision state $S^{x,n-1}$, which we visited just before the transition to S^n.

In this paper, we consider a particular class of value function approximations (VFAs), namely the parametric representation

$$V(S) = \sum_{i=1}^{F} \theta_i \phi_i(S) \qquad (4)$$

for appropriately chosen *basis functions* $\phi_i : \mathcal{S} \mapsto \mathbb{R}$. The basis functions themselves can be arbitrary (though a theoretical analysis may require some additional conditions), but the value of the state is assumed to be linear in the parameter vector θ. This is one of the most enduring and popular classes of value function approximations (see [8] and [9] for early treatments), and still offers modeling and theoretical challenges to this day [10], [11]. Its main advantage is its ability to reduce the problem of approximating a value function to the problem of estimating a finite and relatively small number of parameters.

Despite this benefit, approximate value iteration is still vulnerable to the *exploration/exploitation* dilemma. The choice of x that maximizes (3) depends on the current approximation V^{n-1}. If the approximation is inaccurate, we run the risk of making poor decisions by relying on it. It may be occasionally necessary to take an action that seems to be suboptimal, on the chance that it might turn out better than expected, or that it might take us to a new state that we have not visited before. In this case, we sacrifice some immediate gain, but we

collect information that allows us to make better decisions later on. Popular heuristics for exploration include interval estimation [12], as well as the ε-greedy and soft-max methods [3]. More sophisticated, information-based heuristics include the ℓ-learning method of [13].

The field of optimal learning focuses on the problem of collecting information efficiently. In the ranking and selection problem [14] and its online variant, the multi-armed bandit problem [15], there is a finite set of alternatives whose values are unknown. The goal is to find the best alternative, possibly maximizing rewards in the process. We use a Bayesian belief structure to represent our uncertainty about the unknown values, and we have the ability to refine our beliefs by collecting noisy samples of the values of individual alternatives. We are able to choose which alternatives we want to sample; the goal is to make these measurement decisions efficiently.

Because these models focus exclusively on a changing state of belief, they do not allow for a physical state variable (such as the amount of energy in a battery) that affects our ability to make decisions now and in the future, a key feature of dynamic programming problems. However, there have been attempts to incorporate optimal learning into ADP, the first being the work by [16]. Two types of Bayesian approaches emerged: one that places a Dirichlet prior on the transition probabilities of a dynamic program [17]–[20], and one that places a Gaussian prior on the true value function itself [21]–[23]. The second approach tends to result in faster computation times. Additionally, if a multivariate Gaussian or Gaussian process prior is used, we are able to account for similarities between states. For example, in the energy storage problem, an observation of a single state should also provide some information about other states with similar price values.

In this paper, we apply a particular type of optimal learning technique known as the knowledge gradient (KG) method (studied by [24], [25] for ranking and selection and by [26], [27] for multi-armed bandits) to ADP with the parametric value function approximation given in (4). The KG technique has been used by [28] for Dirichlet priors on transition probabilities, and by [29] for multivariate Gaussian priors on the value function, performing well in both settings. However, [29] uses a lookup table for the value function approximation, which incurs high computational costs when the state space is large. We extend this idea to the case of parametric approximations by making a connection with [30], which uses a parametric model for ranking and selection. The resulting method performs well, while being substantially faster to compute than the original KG-ADP method of [29].

Section II describes the Bayesian model used to learn the parameters of the VFA. Section III lays out the KG algorithm for this setting. In Section IV, we present numerical results testing KG on a version of the energy storage problem described above. Finally, Section V concludes the paper.

II. Bayesian model for parametric VFAs

We consider a classical infinite-horizon problem with state space $\mathcal{S}$, action space $\mathcal{X}$, and a discount factor $\gamma \in (0,1)$. We have the standard objective function

$$\sup_{\pi} \sum_{n=0}^{\infty} \gamma^n C\left(S^n, X^{\pi,n}\left(S^n\right)\right),$$

where $X^{\pi,n}$ is the decision rule associated with the policy π. The post-decision state $S^{x,n} = S^{M,x}\left(S^n, x\right)$ is a deterministic function of state and action. The next pre-decision state $S^{n+1} = S^{M,W}\left(S^{x,n}, W^{n+1}\right)$ is determined randomly using the exogenous information W^{n+1}. Let $\mathcal{S}^x$ be the space of post-decision states. We define our value function approximation (and basis functions) around the post-decision state:

$$V\left(S^{x,n}\right) = \sum_{i=1}^{F} \theta_i \phi_i\left(S^{x,n}\right) = \phi\left(S^{x,n}\right)^T \theta.$$

We assume that the space $\mathcal{S}^x$ is smaller than $\mathcal{S}$, but still continuous. By modeling our state space in this way, we avoid the difficult expectation in (2) by using a deterministic optimization problem (3) to make decisions.

Recall that the quantity $\hat{v}^n$ from (3) is treated as an observation of the unknown value $V\left(S^{x,n-1}\right)$ of the previous post-decision state $S^{x,n-1}$. Let $Y^n = \left(\hat{v}^1, ..., \hat{v}^n\right)$ be a vector of the observations made up to time n, and define a matrix

$$X^n = \begin{pmatrix} \phi_1\left(S^{x,0}\right) & ... & \phi_F\left(S^{x,0}\right) \\ ... & ... & ... \\ \phi_1\left(S^{x,n-1}\right) & ... & \phi_F\left(S^{x,n-1}\right) \end{pmatrix}$$

containing the corresponding basis function values for each post-decision state that we observed. In classical linear regression, the best estimate θ^n is given by

$$\theta^n = \left[\left(X^n\right)^T X^n\right]^{-1} \left(X^n\right)^T Y^n.$$

Letting $B^n = \left[\left(X^n\right)^T X^n\right]^{-1}$, we can also compute θ^n recursively [6] using the equations

$$\theta^n = \theta^{n-1} - \frac{\hat{v}^n - \phi\left(S^{x,n-1}\right)^T \theta^{n-1}}{1 + \phi\left(S^{x,n-1}\right)^T B^{n-1} \phi\left(S^{x,n-1}\right)} B^{n-1} \phi\left(S^{x,n-1}\right) \quad (5)$$

and

$$B^n = B^{n-1} - \frac{B^{n-1} \phi\left(S^{x,n-1}\right) \phi\left(S^{x,n-1}\right)^T B^{n-1}}{1 + \phi\left(S^{x,n-1}\right)^T B^{n-1} \phi\left(S^{x,n-1}\right)}. \quad (6)$$

In our Bayesian model, we place a multivariate Gaussian prior with mean vector θ^0 and covariance matrix C^0 on the vector θ of unknown parameters. This induces a multivariate Gaussian distribution of belief on the value function V, such that $\mathbb{E}V\left(S\right) = \phi\left(S\right)^T \theta^0$ and $Cov\left(V\left(S\right), V\left(S'\right)\right) = \phi\left(S\right)^T C^0 \phi\left(S'\right)$. We now make a critical assumption about the observations $\hat{v}^n$ used to fit our regression model.

Assumption 1: The observation $\hat{v}^n$ follows the distribution $\mathcal{N}\left(V\left(S^{x,n-1}\right), \sigma_\varepsilon^2\right)$ and is independent of past observations.

It is easy to see that this assumption does not hold in most practical applications. In fact, $\hat{v}^n = \max_x C\left(S^n, x\right) + \gamma \phi\left(S^{x,n}\right)^T \theta^{n-1}$ explicitly depends on the previous approximation θ^{n-1}, which introduces bias into the observation. This

quantity is also unlikely to have a Gaussian distribution, and the variance σ_ε^2 of the noise is certainly not known in most cases. However, Assumption 1 is standard in previous work on Bayesian dynamic programming [21]. Even non-Bayesian ADP methods often make independence assumptions (e.g. the OSA stepsize rule of [31]). We follow this assumption in order to get easily computable updating rules for our parameters. We treat σ_ε^2 as a tunable parameter.

From Assumption 1, it follows that the covariance matrix of Y^n is given by $\sigma_\varepsilon^2 I$, where I is the identity matrix. Consequently, by the Gauss-Markov theorem [32], the posterior distribution of θ given the observations $\hat{v}^1, ..., \hat{v}^n$ is multivariate Gaussian with the mean vector given by the best linear unbiased estimator $\theta^n = B^n \left(X^n\right)^T Y^n$ and the covariance matrix given by $C^n = \sigma_\varepsilon^2 B^n$. We can now easily rewrite (5) and (6) to obtain the Bayesian updating equations

$$\theta^n = \theta^{n-1} - \frac{\hat{v}^n - \phi\left(S^{x,n-1}\right)^T \theta^{n-1}}{\sigma_\varepsilon^2 + \phi\left(S^{x,n-1}\right)^T C^{n-1} \phi\left(S^{x,n-1}\right)} C^{n-1} \phi\left(S^{x,n-1}\right) \quad (7)$$

and

$$C^n = C^{n-1} - \frac{C^{n-1}\phi\left(S^{x,n-1}\right)\phi\left(S^{x,n-1}\right)^T C^{n-1}}{\sigma_\varepsilon^2 + \phi\left(S^{x,n-1}\right)^T C^{n-1}\phi\left(S^{x,n-1}\right)}. \quad (8)$$

For simplicity, we suggest starting with a diagonal matrix for C^0, where the entries represent our beliefs about a possible range of values for the true parameters. Even if the parameters are assumed to be independent initially, the updating equations (7) and (8) will quickly populate the off-diagonal entries with empirical covariances. The quantity $V^n\left(S^x\right) = \phi\left(S^x\right)^T \theta^n$ represents our estimate of the value of a post-decision state given the VFA at time n.

III. The knowledge gradient algorithm for parametric VFAs

Under the Bayesian model, we know that every decision will change our beliefs (θ^n, C^n) through (7) and (8). The knowledge gradient (KG) concept can be viewed as a way to incorporate the impact of a decision on our beliefs into the decision-making process itself. While we do not know the exact outcome of our decision in advance, our Bayesian model gives us a way to represent our uncertainty about that outcome, conditional on the information we already have at our disposal. The KG idea uses this precise representation of uncertainty to compute the value of the new information contributed by a decision to our ability to solve the problem.

If we make our decision based solely on the existing approximation (a strategy known as pure exploitation), we compute the decision rule

$$X^{Exp,n}\left(S^n\right) = \arg\max_x Q^n\left(S^n, x\right) \quad (9)$$

where

$$Q^n\left(S^n, x\right) = C\left(S^n, x\right) + \gamma\phi\left(S^{x,n}\right)^T \theta^n.$$

The KG decision rule replaces the downstream value $\phi\left(S^{x,n}\right)^T \theta^n$ with an expectation of the future estimate of the downstream value that we will obtain as a result of making the decision x, resulting in the decision rule

$$X^{KG,n}\left(S^n\right) = \arg\max_x Q^{KG,n}\left(S^n, x\right) \quad (10)$$

where

$$Q^{KG,n}\left(S^n, x\right) = C\left(S^n, x\right) + \gamma \mathbb{E}_x^n V^{n+1}\left(S^{x,n}\right). \quad (11)$$

In this expression, $\mathbb{E}_x^n$ is a conditional expectation given the current VFA (characterized by θ^n and C^n), as well as the decision to take action x next.

We will first summarize the KG logic from [29], and then show how it can be adapted to deal with the parametric case. Note that the change in our beliefs from θ^n to θ^{n+1} will take place only after we have observed the quantity $\hat{v}^{n+1}$ from the pre-decision state S^{n+1}. We can expand (11) into an expression

$$Q^{KG,n}\left(S^n, x\right) = C\left(S^n, x\right) + \gamma \sum_{S^{n+1}} P\left(S^{n+1} | S^{x,n}\right) \mathbb{E}_x^n \max_{x'} Q^{n+1}\left(S^{n+1}, x'\right) \quad (12)$$

The KG approach assumes that the next decision x at time n will be the last decision to impact our beliefs. In this case, all decisions starting at time $n+1$ should be made using the final regression estimate θ^{n+1}. However, from the point of view of time n, this estimate is random, and so Q^{n+1} is a random variable as well. It can be shown that the conditional distribution of the vector θ^{n+1} at time n can be written as

$$\theta^{n+1} \sim \theta^n + \frac{C^{n-1}\phi\left(S^{x,n}\right)}{\sqrt{\sigma_\varepsilon^2 + \phi\left(S^{x,n}\right)^T C^n \phi\left(S^{x,n}\right)}} Z, \quad (13)$$

where $Z \sim \mathcal{N}(0,1)$. To see this, we can observe that θ^{n+1} is conditionally Gaussian by Assumption 1 together with (7), then compute the conditional mean vector and covariance matrix. Note that Z is a scalar random variable, common to the conditional distributions of all the components of θ^{n+1}. The conditional distribution of any $S^x \in \mathcal{S}^x$ can be found by multiplying both sides in (13) by $\phi\left(S^x\right)$, yielding

$$V^{n+1}\left(S^x\right) \sim V^n\left(S^x\right) + \frac{\phi\left(S^x\right)^T C^{n-1}\phi\left(S^{x,n}\right)}{\sqrt{\sigma_\varepsilon^2 + \phi\left(S^{x,n}\right)^T C^n \phi\left(S^{x,n}\right)}} Z. \quad (14)$$

The quantity $\phi\left(S^x\right)^T C^{n-1}\phi\left(S^{x,n}\right)$ is precisely $Cov^n\left(S^x, S^{x,n}\right)$. Note that Z is still scalar and common to all conditional distributions of $V\left(S^x\right)$ for $S^x \in \mathcal{S}^x$.

Using equation (14), the last term in (12) can be written as

$$\mathbb{E}_x^n \max_{x'} Q^{n+1}\left(S^{n+1}, x'\right) = \mathbb{E} \max_{x'} \left(a_{x'}^n + b_{x'}^n Z\right) \quad (15)$$

where

$$\begin{aligned} a_{x'}^n &= C\left(S^{n+1}, x'\right) + \gamma V^n\left(S^{M,x}\left(S^{n+1}, x'\right)\right), \\ b_{x'}^n &= \gamma \frac{\phi\left(S^{M,x}\left(S^{n+1}, x'\right)\right)^T C^{n-1}\phi\left(S^{x,n}\right)}{\sqrt{\sigma_\varepsilon^2 + \phi\left(S^{x,n}\right)^T C^n \phi\left(S^{x,n}\right)}}. \end{aligned}$$

We then apply the work by [33] to compute the right-hand side of (15) exactly via the formula

$$\begin{aligned}\mathbb{E}\max_{x'}\left(a^n_{x'}+b^n_{x'}Z\right) &= \left(\max_{x'}a^n_{x'}\right)\\ &+ \sum_{y\in A}\left(b^n_{y+1}-b^n_y\right)f\left(-\left|c_y\right|\right).\end{aligned}$$

In this expression, A is the set of all y for which there exist numbers $c_{y-1}<c_y$ satisfying $y=\arg\max_{x'}a^n_{x'}+b^n_{x'}z$ for $z\in(c_{y-1},c_y)$. In other words, the numbers c_y are the breakpoints of the piecewise linear function inside the expectation in (15), arranged in increasing order, and the numbers b_y are the corresponding slopes of that piecewise linear function. The function f is defined as $f(z)=z\Phi(z)+\phi(z)$, where ϕ and Φ denote the standard Gaussian pdf and cdf.

The *knowledge gradient* is defined to be

$$\begin{aligned}\nu^{KG,n}\left(S^{x,n},S^{n+1}\right) &= \mathbb{E}\max_{x'}\left(a^n_{x'}+b^n_{x'}Z\right)-\max_{x'}a^n_{x'}\\ &= \sum_{y\in A}\left(b^n_{y+1}-b^n_y\right)f\left(-\left|c_y\right|\right).\end{aligned}$$

Since $a^n_{x'}=Q^n\left(S^{n+1},x'\right)$, this quantity can be viewed as the expected improvement in our estimate of the value of being in state S^{n+1}, obtained as a result of making a transition from $S^{x,n}$ to S^{n+1}. We can substitute the definition of the knowledge gradient back into (12) to obtain

$$\begin{aligned}&\sum_{S^{n+1}}P\left(S^{n+1}|S^{x,n}\right)\mathbb{E}^n_x\max_{x'}Q^{n+1}\left(S^{n+1},x'\right)\\ =\;&\sum_{S^{n+1}}P\left(S^{n+1}|S^{x,n}\right)\max_{x'}Q^n\left(S^{n+1},x'\right)\\ +\;&\sum_{S^{n+1}}P\left(S^{n+1}|S^{x,n}\right)\nu^{KG,n}\left(S^{x,n},S^{n+1}\right).\end{aligned}$$

In ADP, the value of a post-decision state is defined to be the expected value of the next pre-decision state, with the expectation taken over the random transition from $S^{x,n}$ to S^{n+1}. Therefore, we can write

$$\sum_{S^{n+1}}P\left(S^{n+1}|S^{x,n}\right)\max_{x'}Q^n\left(S^{n+1},x'\right)\approx V^n\left(S^{x,n}\right)$$

and (10) reduces to

$$\begin{aligned}X^{KG,n}\left(S^n\right)=&\max_x C\left(S^n,x\right)+\gamma V^n\left(S^{x,n}\right)\\ &+\sum_{S^{n+1}}P\left(S^{n+1}|S^{x,n}\right)\nu^{KG,n}\left(S^{x,n},S^{n+1}\right)\end{aligned}\quad(16)$$

This final form looks remarkably similar to the pure exploitation rule of (9). We maximize the combination of the one-period reward plus the downstream value, but now we add a new term representing the expected value of information. Under the KG policy, we are more likely to choose a decision that has a high value according to our current approximation, but we are also more likely to choose a decision with a high value of information (typically corresponding to high uncertainty, that is, high values of $\phi\left(S^{x,n}\right)^T C^n\phi\left(S^{x,n}\right)$ and relevant covariances).

The value of information for a fixed transition to S^{n+1} can be computed exactly using the procedure developed in [33]. In our setting, the number of breakpoints in the piecewise linear function in (15) depends on the size of the action space, not the size of the state space. As long as we can compute the basis vector $\phi\left(S^x\right)$ inexpensively (for example, if the basis functions are continuous in the state variable), we can construct the components $a^n_{x'}$ and $b^n_{x'}$ of the KG formula as we need them, with no need for a basis matrix (denoted by X in [30]) where the number of rows equals the number of states. We are able to handle continuous state spaces, provided that the action space is reasonably small.

If the state space is continuous, the sum in (16) becomes an integral that is difficult to compute. In many problems, we may not even know the probability distribution of the transition from $S^{x,n}$ to S^{n+1}, making it difficult to find transition probabilities. As in [29], we suggest using Monte Carlo simulation to overcome this issue. This approach is suited to a model-based dynamic programming setting, where it is possible to simulate the real-world problem, and the complexity of the model and state space is the main obstacle to finding an optimal policy. We can simulate K transitions out of $S^{x,n}$, then calculate an approximate KG factor

$$\begin{aligned}&\sum_{S^{n+1}}P\left(S^{n+1}|S^{x,n}\right)\nu^{KG,n}\left(S^{x,n},S^{n+1}\right)\\ \approx\;&\frac{1}{K}\sum_{k=1}^{K}\nu^{KG,n}\left(S^{x,n},S^{n+1}_k\right),\end{aligned}\quad(17)$$

where $S^{n+1}_k=S^{M,W}\left(S^{x,n},W^{n+1}\left(\omega_k\right)\right)$ for the kth sample path generated. Finally, the policy specified by

$$X^{Off,n}\left(S^n\right)=\arg\max_x\sum_{S^{n+1}}P\left(S^{n+1}|S^{x,n}\right)\nu^{KG,n}\left(S^{x,n},S^{n+1}\right)$$

represents a variation of KG that is more suitable for offline problems, where we can train the VFA in a simulator, with no economic cost for making mistakes (thus with more flexibility to explore), before using it in the real world.

The algorithm proposed above uses the KG concept first proposed in [29]. The computational steps are largely the same, and the parametric model comes into play only when we compute the components $a^n_{x'}$ and $b^n_{x'}$ used in the KG formula. However, this difference has a profound effect on the computational cost of the algorithm. By using a parametric model where the number of parameters is typically much smaller than the number of states, we only need to maintain a covariance matrix of size $F\times F$. In particular, the updating equations (7) and (8) will require much less effort to compute. Furthermore, we no longer need to discretize the state space, because our VFA is continuous, and we also do not need to specify a full covariance structure when we create the prior C^0. Instead, we can start with a diagonal covariance matrix, and our updating procedure will add covariances automatically. We discuss and illustrate these issues in the next section.

IV. Numerical experiments

We evaluate the performance of the parametric KG policy on the energy storage problem considered in [29]. Suppose that we have a storage device, such as a battery. We define R^n to be the amount of energy in the battery (the "charge level"), as a percentage of total capacity, which is taken to be 35 MWh. We allow R^n to take integer values between 0 and 100. The spot price P^n of energy follows a geometric Ornstein-Uhlenbeck process [34], where

$$\log \frac{P^{n+1}}{P^n} = -\alpha \log \frac{P^n}{P^0} + \sigma Z^{n+1} \tag{18}$$

with $Z^{n+1} \sim \mathcal{N}(0,1)$. We use the values $\alpha = 0.0633$ and $\sigma = 0.2$, along with an initial price $P^0 = 30$. The decision x^n is an integer from -50 to 50 representing how much to charge ($x^n \geq 0$) or discharge ($x^n < 0$) the storage device. Not every value from -50 to 50 is available for a given state; for example, if the battery is empty, the possible actions range from 0 to 50. The post-decision state is given by

$$\begin{aligned} R^{x,n} &= R^n + x^n, \\ P^{x,n} &= P^n, \end{aligned}$$

whereas the next pre-decision state is obtained by letting $R^{n+1} = R^{x,n}$ and generating P^{n+1} using (18). The single-period reward is given by $C(S^n, x^n) = -P^n x^n$, the cost incurred or revenue obtained as a result of our decision. Our objective is to use our storage device to play the spot market and maximize discounted revenue ($\gamma = 0.99$) over time.

The spot price P^n is continuous, and thus we cannot solve the problem exactly. We use a parametric VFA with polynomial basis functions given by $\phi(S^{x,n}) = \left(1, R^{x.n}, (R^{x,n})^2, P^{x,n}, (P^{x,n})^2, R^{x,n}P^{x,n}\right)$. This representation is much more compact than the discretization of the state space used in [29]. We run the KG policy with the Monte Carlo approximation from (17), with $K = 20$.

For each sample path, we begin with a constant prior $\theta_1^0 = 15000$, $\theta_2^0, ..., \theta_6^0 = 0$ and a diagonal covariance matrix C^0 with all diagonal elements equal to 500^2. We chose an optimistic value for θ^0 heuristically, to reduce the likelihood of getting stuck in a subset of the state space (see Sec. 4.7 of [6]). The measurement noise was chosen to be $\sigma_\varepsilon^2 = 2000^2$. We found that the behaviour of the learning model was sensitive to the relationship between C^0 and σ_ε^2. Though we obtained good results with the parameter values given above, other parameter settings caused the values of θ^n to oscillate in a volatile manner. This problem is not specific to the KG policy (we observed it even for a generic ε-greedy policy), and appears to be endemic to parametric VFAs. Despite the risk of divergence, parametric VFAs remain extremely popular due to their ability to handle large state spaces [35].

We compared both offline and online variants of KG with a tuned ε-greedy policy with $\varepsilon = 0.05$. The online variant is the one given in (16), where we balance the existing VFA against the value of information when we make a decision. The offline variant simply chooses the action with the greatest value of information. Offline decision-making is more suitable in a setting where we can use a simulator to train our VFA for some time before applying it to the real-world problem. If there is no real economic cost for making a decision, we can afford to do more exploration. However, if we are attempting to train our VFA while solving the problem in real time, we can use the more conservative online policy. In addition to offline and online KG with basis functions, we also implemented a version of KG with a lookup-table approximation, originally proposed in [29], where this policy was shown to perform well against other lookup-table methods. We use this VFA (together with a distance-based covariance structure) as a benchmark for our parametric model.

TABLE I: Means and standard errors for the storage problem.

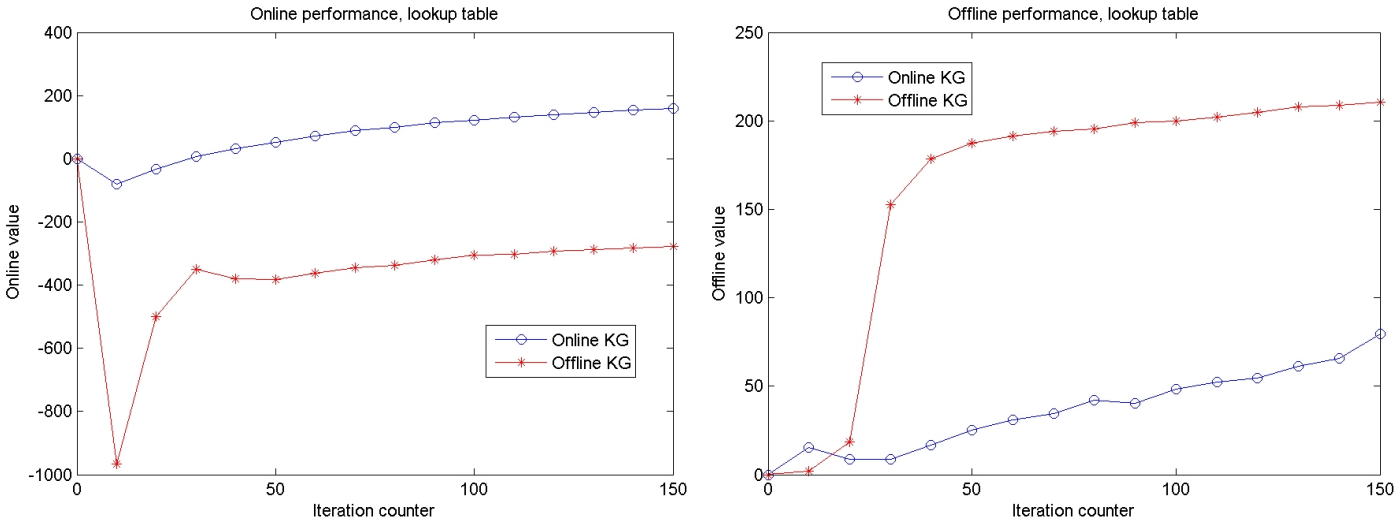
	Offline objective		Online objective	
	Mean	Avg. SE	Mean	Avg. SE
Offline KG (lookup)	210.4283	0.33	-277.3786	15.90
Online KG (lookup)	79.3624	0.23	160.2816	5.43
Offline KG (param.)	1136.2027	3.54	-342.2321	19.96
Online KG (param.)	871.1285	3.15	44.58	27.71
ε-greedy (param.)	-475.5430	2.30	-329.0319	25.3107

Table I reports the online and offline performance of the different policies. Although the parametric model is sensitive to the choice of measurement noise and prior variance, tuning these parameters allows us to obtain much better offline performance than with a lookup table approximation. Even online version of KG, which does less exploration than the offline version, outperforms its lookup-table counterpart by an order of magnitude in the offline setting. Offline KG achieves even better offline performance. The ε-greedy policy loses money in both online and offline settings, showing that intelligent exploration continues to be important even in the parametric model, where any decision updates the entire parameter vector.

Furthermore, using a parametric VFA allows for substantial computational savings. Table II reports the time in seconds required to compute KG factors for all possible decisions in the first time step, as well as the time required for Bayesian updating, using both lookup-table and parametric VFAs. We see that KG factors are more expensive to compute in the parametric case, because the covariances in (14) are now calculated via matrix multiplications. In the case of lookup tables, they are stored in a covariance matrix, requiring no additional calculations to access. However, the lookup-table case requires us to update a large covariance matrix in each time step, whose size grows if we make finer discretizations of the state space. In the parametric case, we only update a small $F \times F$ covariance matrix, making the Bayesian update much faster. Altogether, the computational cost of the KG algorithm

TABLE II: Computational effort required for KG and Bayesian updating.

	KG	Updating
Lookup	0.0247s	0.6817s
Param.	0.2048s	0.0588s



(a) Online case.

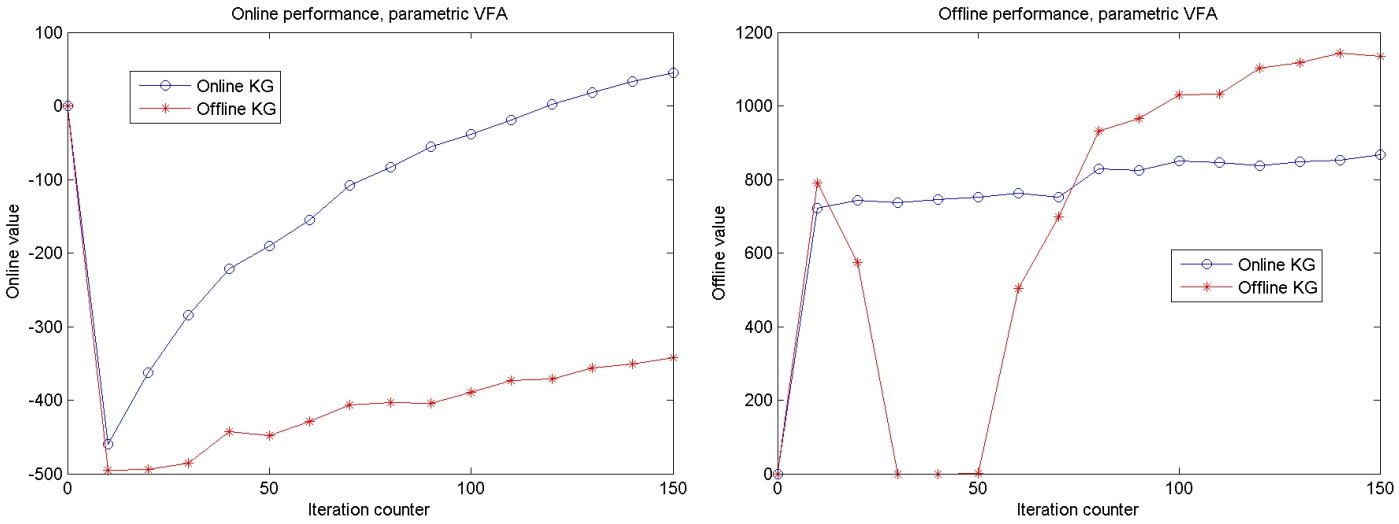
(b) Offline case.

Fig. 1: Performance of KG with a lookup table VFA.

with parametric VFAs is only 37% of the cost of KG with lookup tables. The parametric case also requires us to store much less data in memory.

At the same time, Table I suggests that KG performs better with the lookup table approximation in an online setting. We can examine this issue more closely by considering the performance of KG as a function of the number N of time steps, in both offline and online settings, for both types of VFAs. Figure 1 shows the performance of KG with a lookup table, and Figure 2 covers the parametric case. The parametric case induces much more exploration in the early stages of the problem. In the first 10 iterations, parametric KG suffers a sharp drop in online performance. However, the information collected in these iterations allows us to make much better decisions later on. The rate of improvement in the online performance of online KG is much greater in the parametric case (Figure 2a) than for the lookup table (Figure 1a), but the heavy exploration in the beginning incurs a greater cost.

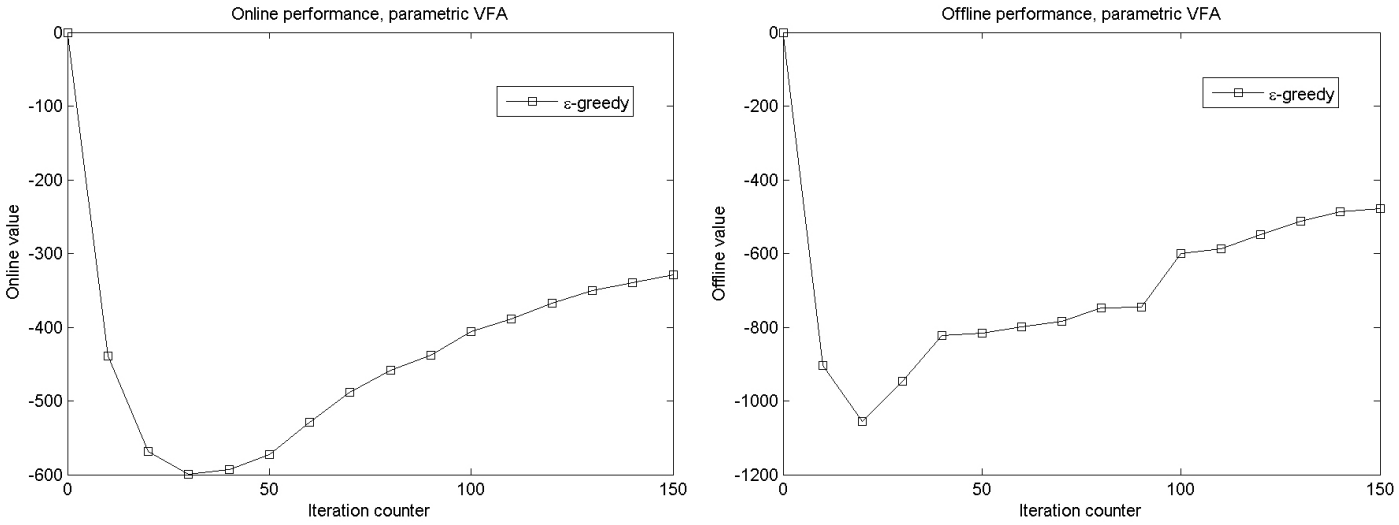
Because online KG explores heavily in the first 10 iterations in the parametric case, it also produces a good VFA. For the first 70 iterations, it actually yields better offline performance than offline KG. Offline KG spends this time making drastically different decisions, resulting in volatile changes in the VFA and the offline performance. However, after 70 iterations, the information collected in this manner begins to pay off, resulting in better offline performance.



(a) Online case.

(b) Offline case.

Fig. 2: Performance of KG with a parametric VFA.



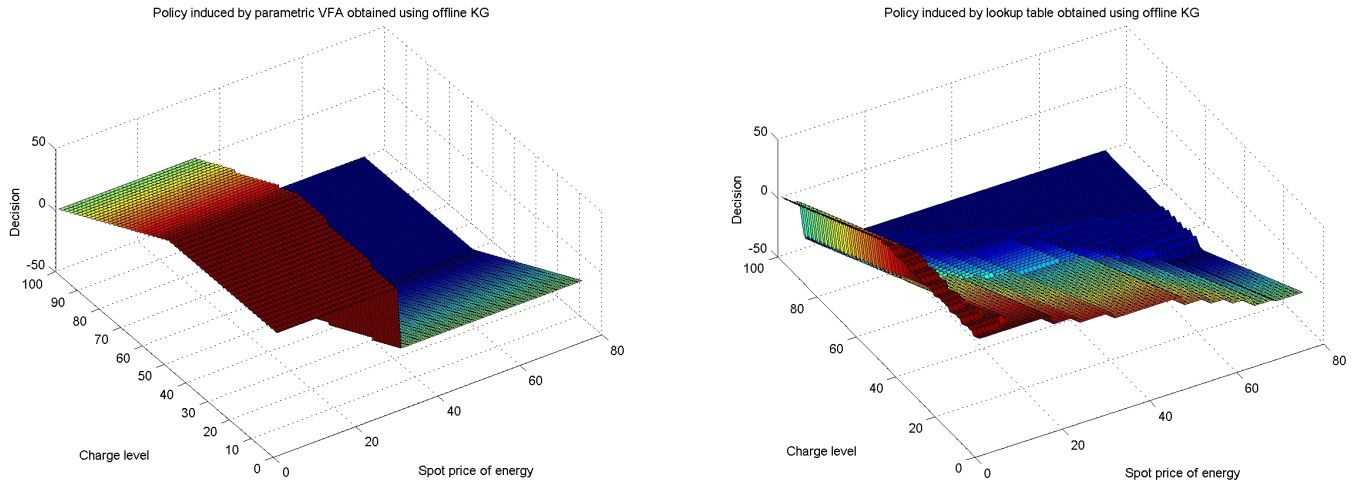
(a) Online case.

(b) Offline case.

Fig. 3: Performance of ε-greedy with a parametric VFA.

Figure 3 shows the performance of ε-greedy. The trajectory of both offline and online performance is similar to that of KG. However, ε-greedy is slower to collect useful information. We see that online performance hits its minimum at 30 iterations, whereas for both versions of KG this minimum occurs at 10 iterations. Afterward, performance steadily improves, though at a slower rate than online KG. The offline performance of ε-greedy is also characterized by a sharp drop at the beginning, whereas offline KG never produces a policy that loses money, even in the early exploration phase. These results indicate that KG not only explores, but does so more efficiently and systematically than the ε-greedy heuristic.

Finally, Figure 4 shows the policy induced by offline KG after 150 training iterations as a function of charge level and price for both types of VFA. The decision rules are somewhat similar: we always discharge the battery if both the charge level and the price are high, and we tend to charge (if possible) when the price is low. However, the lookup table VFA sets a much lower threshold for charging the battery. For $P^n = 20$ and an empty battery, we charge much more under the parametric VFA. In fact, the parametric VFA determines a very clear and intuitive threshold: we charge if the price is below the mean of 35, and discharge if the price is higher. In comparison, the lookup table VFA appears to be too eager to discharge the battery when the price is too low, leading to lower revenues.



(a) Parametric VFA.

(b) Lookup table.

Fig. 4: Policy induced by offline KG for each type of VFA.

These results suggest that parametric VFAs, when properly tuned, can capture the problem behaviour more accurately and quickly than lookup tables. An important advantage of this approach is that the covariance structure is automatically created using the information that we collect, and we do not need to rely on heuristic distance-based covariances, as in the lookup table case.

V. Conclusion

We have proposed a policy for Bayesian active learning with parametric value function approximations. We show how a Bayesian belief structure can be used in conjunction with a parametric VFA, and apply the knowledge gradient concept from [29] to this setting. The resulting ADP technique requires less disk space and computational time than a lookup table approach, and yields good performance on an energy storage problem. Our experimental results show that the KG policy finds a good balance between exploration and exploitation more efficiently than a traditional ε-greedy heuristic. We believe that the extension to parametric VFAs underscores the versatility of the KG concept as a general approach to the exploration/exploitation challenge. Future work will consider non-parametric approximations, which may be useful for problems where it is difficult to find suitable basis functions.

Acknowledgment

This work was supported in part by AFOSR contract FA9550-08-1-0195 through the Center for Dynamic Data Analysis.

References

[1] M. L. Puterman, *Markov Decision Processes*. New York: John Wiley & Sons, 1994.

[2] H. Geman and A. Roncoroni, "Understanding the Fine Structure of Electricity Prices," *The Journal of Business*, vol. 79, no. 3, 2006.

[3] R. Sutton and A. Barto, *Reinforcement Learning*. Cambridge, Massachusetts: The MIT Press, 1998.

[4] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[5] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, Eds., *Handbook of Learning and Approximate Dynamic Programming*. New York: IEEE Press, 2004.

[6] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. New York: John Wiley and Sons, 2007.

[7] B. Van Roy, D. Bertsekas, Y. Lee, and J. Tsitsiklis, "A neuro-dynamic programming approach to retailer inventory management," in *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 4, 1997, pp. 4052–4057.

[8] G. Tesauro, "Practical Issues in Temporal Difference Learning," *Machine Learning*, vol. 8, no. 3-4, pp. 257–277, 1992.

[9] P. Sabes, "Approximating Q-values with basis function representations," in *Proceedings of the Fourth Connectionist Models Summer School, Hillsdale, NJ*, M. Mozer, D. Touretzky, and P. Smolensky, Eds., 1993, pp. 264–271.

[10] I. Menache, S. Mannor, and N. Shimkin, "Basis function adaptation in temporal difference reinforcement learning," *Annals of Operations Research*, vol. 134, no. 1, pp. 215–238, 2005.

[11] R. Sutton, H. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora, "Fast gradient-descent methods for temporal-difference learning with linear function approximation," in *Proceedings of the 26th International Conference on Machine Learning*, 2009, pp. 993–1000.

[12] L. P. Kaelbling, *Learning in Embedded Systems*. Cambridge, MA: MIT Press, 1993.

[13] K. Iwata, K. Ikeda, and H. Sakai, "A new criterion using information gain for action selection strategy in reinforcement learning," *IEEE Transactions on Neural Networks*, vol. 15, no. 4, pp. 792–799, 2004.

[14] R. Bechhofer, T. Santner, and D. Goldsman, *Design and Analysis of Experiments for Statistical Selection, Screening and Multiple Comparisons*. New York: John Wiley and Sons, 1995.

[15] J. Gittins, *Multi-Armed Bandit Allocation Indices*. New York: John Wiley and Sons, 1989.

[16] M. Duff and A. Barto, "Local bandit approximation for optimal learning problems," *Advances in Neural Information Processing Systems*, vol. 9, pp. 1019–1025, 1996.

[17] R. Dearden, N. Friedman, and D. Andre, "Model-based Bayesian Exploration," in *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, 1999, pp. 150–159.

[18] M. Strens, "A Bayesian framework for reinforcement learning," in *Proceedings of the 17th International Conference on Machine Learning*, 2000, pp. 943–950.

[19] M. Duff, "Design for an optimal probe," in *Proceedings of the 20th International Conference on Machine Learning*, 2003, pp. 131–138.

[20] P. Poupart, N. Vlassis, J. Hoey, and K. Regan, "An analytic solution to discrete Bayesian reinforcement learning," in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 697–704.

[21] R. Dearden, N. Friedman, and S. Russell, "Bayesian Q-learning," in *Proceedings of the 15th National Conference on Artificial Intelligence*, 1998, pp. 761–768.

[22] Y. Engel, S. Mannor, and R. Meir, "Bayes Meets Bellman: The Gaussian Process Approach to Temporal Difference Learning," in *Proceedings of the 20th International Conference on Machine Learning*, 2003, pp. 154–161.

[23] ——, "Reinforcement learning with Gaussian processes," in *Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 208–215.

[24] S. Gupta and K. Miescke, "Bayesian look ahead one-stage sampling allocations for selection of the best population," *Journal of Statistical Planning and Inference*, vol. 54, no. 2, pp. 229–244, 1996.

[25] P. I. Frazier, W. B. Powell, and S. Dayanik, "A knowledge gradient policy for sequential information collection," *SIAM Journal on Control and Optimization*, vol. 47, no. 5, pp. 2410–2439, 2008.

[26] I. O. Ryzhov and W. B. Powell, "The knowledge gradient algorithm for online subset selection," in *Proceedings of the 2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, Nashville, TN*, 2009, pp. 137–144.

[27] I. O. Ryzhov, W. B. Powell, and P. I. Frazier, "The knowledge gradient algorithm for a general class of online learning problems," *Submitted for publication*, 2009.

[28] I. O. Ryzhov, M. R. Valdez-Vivas, and W. B. Powell, "Optimal Learning of Transition Probabilities in the Two-Agent Newsvendor Problem," in *Proceedings of the 2010 Winter Simulation Conference*, B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, Eds., 2010.

[29] I. O. Ryzhov and W. B. Powell, "Approximate Dynamic Programming With Correlated Bayesian Beliefs," in *Proceedings of the 48th Allerton Conference on Communication, Control and Computing*, 2010.

[30] D. Negoescu, P. Frazier, and W. Powell, "The Knowledge-Gradient Algorithm for Sequencing Experiments in Drug Discovery," *INFORMS J. on Computing (to appear)*, 2010.

[31] A. George and W. B. Powell, "Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming," *Machine Learning*, vol. 65, no. 1, pp. 167–198, 2006.

[32] J. Scharf, *Statistical signal processing*. Addison-Wesley, 1991.

[33] P. I. Frazier, W. B. Powell, and S. Dayanik, "The knowledge-gradient policy for correlated normal rewards," *INFORMS J. on Computing*, vol. 21, no. 4, pp. 599–613, 2009.

[34] E. Schwartz, "The stochastic behavior of commodity prices: Implications for valuation and hedging," *Journal of Finance*, vol. 52, no. 3, pp. 923–973, 1997.

[35] R. Sutton, C. Szepesvári, and H. Maei, "A convergent O(n) algorithm for off-policy temporal-difference learning with linear function approximation," *Advances in Neural Information Processing Systems*, vol. 21, pp. 1609–1616, 2008.