Non-parametric Approximate Dynamic Programming via the Kernel Method

Nikhil Bhat
Graduate School of Business
Columbia University
email: nbhat15@gsb.columbia.edu

Vivek F. Farias
Sloan School of Management
Massachusetts Institute of Technology
email: vivekf@mit.edu

Ciamac C. Moallemi
Graduate School of Business
Columbia University
email: ciamac@gsb.columbia.edu

October 20, 2012

Abstract

This paper presents a novel and practical non-parametric approximate dynamic programming (ADP) algorithm that enjoys graceful, dimension-independent approximation and sample complexity guarantees. In particular, we establish both theoretically and computationally that our proposal can serve as a viable replacement to state of the art *parametric* ADP algorithms, freeing the designer from carefully specifying an approximation architecture. We accomplish this by 'kernelizing' a recent mathematical program for ADP (the 'smoothed' approximate LP) proposed by Desai et al. (2011). Our theoretical guarantees establish that the quality of the approximation produced by our procedure improves gracefully with sampling effort. Via a computational study on a controlled queueing network, we show that our non-parametric procedure outperforms the state of the art parametric ADP approaches and established heuristics.

1. Introduction

Problems of dynamic optimization in the face of uncertainty are frequently posed as Markov decision processes (MDPs). The central computational problem is then reduced to the computation of an optimal 'value' or 'cost-to-go' function that encodes the value garnered under an optimal policy starting from any given MDP state. MDPs for many problems of practical interest frequently have intractably large state spaces precluding exact computation of the cost-to-go function. Approximate dynamic programming (ADP) is an umbrella term for algorithms designed to produce good approximations to this function. Such approximations then imply a natural 'greedy' control policy.

ADP algorithms are, in large part, *parametric* in nature. In particular, the user specifies an 'approximation architecture' (i.e., a set of basis functions) and the algorithm then produces an approximation in the span of this basis. The strongest theoretical results available for such algorithms typically share the following two features:

- The quality of the approximation produced is comparable with the best possible within the basis specified.

- The sample complexity or computational effort required for these algorithms scales, typically polynomially, with the dimension of the basis.

These results highlight the importance of selecting a 'good' approximation architecture, and remain somewhat dissatisfying in that additional sampling or computational effort cannot remedy a bad approximation architecture.

In contrast, an ideal *non-parametric* approach would, in principle, free the user from carefully specifying a suitable low-dimensional approximation architecture. Instead, the user would have the liberty of selecting a very rich architecture (such as, say, the Haar basis). The quality of the approximation produced by the algorithm would then improve — gracefully — with the extent of computational or sampling effort expended, ultimately becoming exact. Unfortunately, existing non-parametric proposals for ADP fall short of this ideal on one or more fronts. In particular, the extant proposals include:

Non-parametric regression. The key computational step in approximate policy iteration methods is approximate policy evaluation. This step involves solving the projected Bellman equation, a linear stochastic fixed point equation. A numerically stable approach to this is to perform regression where the loss is the ℓ_2-norm of the Bellman error. This procedure is called least-squares temporal differences (Bertsekas, 2007), and is a popular ADP method. By substituting this parametric regression step with a suitable non-parametric regression procedure, Bethke et al. (2008), Engel et al. (2003), and Xu et al. (2007) come up with corresponding non-parametric algorithms. Unfortunately approximate policy iteration schemes have no convergence guarantees in parametric settings, and these difficulties remain in non-parametric variations. It is consequently difficult to characterize the computational effort or sample complexity required to produce a good approximation (if this is at all possible) with such approaches. More importantly, the practical performance or viability (given that many rounds of regression will typically be called for) of these methods is not clear.

Local averaging. Another idea has been to use kernel-based local averaging ideas to approximate the solution of an MDP with that of a simpler variation on a sampled state space (e.g., Ormoneit and Sen, 2002; Ormoneit and Glynn, 2002; Barreto et al., 2011). However, convergence rates for local averaging methods are exponential in the dimension of the problem state space. As in our setting, Dietterich and Wang (2002) construct kernel-based cost-to-go function approximations. These are subsequently plugged in to various ad hoc optimization-based ADP formulations. While their methods are closest in spirit to ours, they do not employ any regularization. This suggests potentially poor sample complexity, and, indeed, they do not provide theoretical justification or sample complexity results. Similarly, Ernst et al. (2005) replace the local averaging procedure used for regression by Ormoneit and Sen (2002) with non-parametric regression procedures such as the tree-based learning methods. This is done again without any theoretical justification.

Feature selection via ℓ_1-penalty (parametric). Closely related to our work, Kolter and Ng (2009) and Petrik et al. (2010) consider modifying the approximate linear program with an ℓ_1-regularization term to encourage sparse approximations in the span of a large, but necessarily *tractable* set of features. Along these lines, Pazis and Parr (2011) discuss a non-parametric method

that explicitly restricts the smoothness of the value function. However, sample complexity results for this method are not provided and it appears unsuitable for high-dimensional problems (such as, for instance, the queuing problem we consider in our experiments). In contrast to this line of work, our approach will allow for approximations in a potentially *full-dimensional* approximation architecture that capable of an *exact* representation of the value function, with a constraint on an appropriate ℓ_2-norm of the weight vector to provide regularization.

This paper presents what we believe is a practical, non-parametric ADP algorithm that enjoys non-trivial approximation and sample complexity guarantees. In particular, we establish both theoretically and computationally that our proposal can serve as a viable replacement to state-of-the-art parametric ADP algorithms based on linear programming, freeing the designer from carefully specifying an approximation architecture. In greater detail, we make the following contributions:

- **A new mathematical programming formulation.** We rigorously develop a kernel-based variation of the 'smoothed' approximate LP (SALP) approach to ADP proposed by Desai et al. (2011). The resulting mathematical program, which we dub the regularized smoothed approximate LP (RSALP), is distinct from simply substituting the local averaging approximation above in the SALP formulation. We develop a companion active set method that is capable of solving this mathematical program rapidly and with limited memory requirements.

- **Theoretical guarantees.**[1] Our algorithm can be interpreted as solving an approximate linear program in a (potentially infinite dimensional) Hilbert space. We provide a probabilistic upper bound on the approximation error of the algorithm relative to the best possible approximation one may compute in this space subject to a regularization constraint. We show that the number of samples grows polynomially as a a function of a regularization parameter. As this regularization parameter is allowed to grow, so does the set of permissible approximations, eventually permitting an exact approximation.

  The sampling requirements for our method are independent of the dimension of the approximation architecture. Instead, they grow with the desired complexity of the approximation. This result can be seen as the 'right' generalization of the prior parametric approximate LP approaches of de Farias and Van Roy (2003, 2004); Desai et al. (2011), where, in contrast, sample complexity grows with the dimension of the approximating architecture.

- **A computational study.** To study the efficacy of our approach, we consider an MDP arising from a challenging queueing network scheduling problem. We demonstrate that our method yields significant improvements over tailored heuristics and parametric ADP methods, all while using a generic high-dimensional approximation architecture. In particular, these results suggest the possibility of solving a challenging high-dimensional MDP using an entirely generic approach.

[1] These guarantees come under assumption of being able to sample from a certain idealized distribution. This is a common requirement in the analysis of ADP algorithms that enjoy approximation guarantees for general MDPs (de Farias and Van Roy, 2004; Van Roy, 2006; Desai et al., 2011).

The organization of the paper is as follows: In Section 2, we formulate an infinite dimensional LP for our problem, and present an effective approximate solution approach. Section 3 provides theoretical guarantees for the quality of the approximations computed via our non-parametric algorithm. Theorem 1 in that Section provides our main guarantee. In Section 4, we provide an active set method that can be used to efficiently solve the required quadratic optimization problem central to our approach while respecting practical memory constraints. We also establishing the correctness of our active set approach. Section 5 describes a numerical study for a criss-cross queueing system benchmarking our approach against approximate linear programming approaches and tailor made heuristics. Section 6 concludes.

2. Formulation

2.1. Preliminaries

Consider a discrete time Markov decision process with finite state space $\mathcal{S}$ and finite action space $\mathcal{A}$. We denote by x_t and a_t respectively, the state and action at time t. For notational simplicity, and without loss of generality, we assume that all actions are permissible at any given state. We assume time-homogeneous Markovian dynamics: conditioned on being at state x and taking action a, the system transitions to state x' with probability $p(x, x', a)$ independent of the past. A policy is a map $\mu\colon \mathcal{S} \to \mathcal{A}$, so that

$$J^{\mu}(x) \triangleq \mathsf{E}_{x,\mu}\left[\sum_{t=0}^{\infty} \alpha^t g_{x_t,a_t}\right]$$

represents the expected (discounted, infinite horizon) cost-to-go under policy μ starting at state x, with the discount factor $\alpha \in (0,1)$. Letting Π denote the set of all policies, our goal is to find an optimal policy μ^* such that $\mu^* \in \operatorname{argmax}_{\mu\in\Pi} J^{\mu}(x)$ for all $x \in \mathcal{S}$ (it is well known that such a policy exists). We denote the optimal cost-to-go function by $J^* \triangleq J^{\mu^*}$. An optimal policy μ^* can be recovered as a 'greedy' policy with respect to J^*,

$$\mu^*(x) \in \operatorname*{argmin}_{a\in\mathcal{A}} \; g_{x,a} + \alpha \mathsf{E}_{x,a}[J^*(X')],$$

where we define the expectation $\mathsf{E}_{x,a}[f(X')] \triangleq \sum_{x'\in\mathcal{S}} p(x,x',a) f(x')$, for all functions $f\colon \mathcal{S} \to \mathbb{R}$ on the state space.

Since in practical applications the state space $\mathcal{S}$ is often intractably large, exact computation of J^* is untenable. ADP algorithms are principally tasked with computing approximations to J^* of the form $J^*(x) \approx z^\top \Phi(x) \triangleq \tilde{J}(x)$, where $\Phi\colon \mathcal{S} \to \mathbb{R}^m$ is referred to as an 'approximation architecture' or a basis and must be provided as input to the ADP algorithm. The ADP algorithm computes a 'weight' vector $z \in \mathbb{R}^m$. One then employs a policy that is greedy with respect to the corresponding approximation $\tilde{J}$.

2.2. Primal Formulation

The approach we propose is based on the LP formulation of dynamic programming. It was observed by Manne (1960) that the optimal cost-to-go J^* can be obtained by solving the following linear program:

$$\begin{aligned} & \text{maximize} && \nu^\top J \\ & \text{subject to} && J(x) \leq g_{a,x} + \alpha \mathsf{E}_{x,a}[J(X')], \quad \forall\ x \in \mathcal{S},\ a \in \mathcal{A}, \\ & && J \in \mathbb{R}^{\mathcal{S}}, \end{aligned} \tag{1}$$

for any strictly positive state-relevance weight vector $\nu \in \mathbb{R}^{\mathcal{S}}_+$. Motivated by this, a series of ADP algorithms (Schweitzer and Seidman, 1985; de Farias and Van Roy, 2003; Desai et al., 2011) have been proposed that compute a weight vector z by solving an appropriate modification of (1). In particular, Desai et al. (2011) propose solving the following optimization problem:

$$\begin{aligned} & \text{maximize} && \sum_{x \in \mathcal{S}} \nu_x z^\top \Phi(x) - \kappa \sum_{x \in \mathcal{S}} \pi_x s_x \\ & \text{subject to} && z^\top \Phi(x) \leq g_{a,x} + \alpha \mathsf{E}_{x,a}[z^\top \Phi(X')] + s_x, \quad \forall\ x \in \mathcal{S},\ a \in \mathcal{A}, \\ & && z \in \mathbb{R}^m,\ s \in \mathbb{R}^{\mathcal{S}}_+. \end{aligned} \tag{2}$$

Here $\kappa > 0$ is a penalty parameter and $\pi \in \mathbb{R}^{\mathcal{S}}_+$ is a strictly positive distribution on $\mathcal{S}$. In considering the above program, notice that if one insisted that the slack variables s were precisely 0, the program (2) would be identical to (1), with the additional restriction to value function approximations of the form $J(x) = z^\top \Phi(x)$. This case is known as the approximate linear program (ALP), and was first proposed by Schweitzer and Seidman (1985). de Farias and Van Roy (2003) provided a pioneering analysis that, stated loosely, showed

$$\|J^* - z^{*\top} \Phi\|_{1,\nu} \leq \frac{2}{1-\alpha} \inf_z \ \|J^* - z^\top \Phi\|_\infty,$$

for an optimal solution z^* to the ALP. Desai et al. (2011) showed that these bounds could be improved upon by 'smoothing' the constraints of the ALP, i.e., permitting positive slacks. The resulting program (2) is called the smoothed approximate linear program (SALP). In both instances, in the general case of a large state space, one must solve a 'sampled' version of the above program.

Now, consider allowing Φ to map from $\mathcal{S}$ to a general (potentially infinite dimensional) Hilbert space $\mathcal{H}$. We use bold letters to denote elements in the Hilbert space $\mathcal{H}$, e.g., the weight vector is denoted by $\mathbf{z} \in \mathcal{H}$. We further suppress the dependence on Φ and denote the elements of $\mathcal{H}$ corresponding to their counterparts in $\mathcal{S}$ by bold letters. Hence, for example, $\mathbf{x} \triangleq \Phi(x)$ and $\mathbf{X} \triangleq \Phi(X)$. Denote the image of the state space under the map Φ by $\mathcal{X} \triangleq \Phi(\mathcal{S})$; $\mathcal{X} \subset \mathcal{H}$. The analogous value function approximation in this case would be given by

$$\tilde{J}_{\mathbf{z},b}(x) \triangleq \langle \mathbf{x}, \mathbf{z} \rangle + b = \langle \Phi(x), \mathbf{z} \rangle + b, \tag{3}$$

where b is a scalar offset corresponding to a constant basis function.[2] The following generalization of (2) — which we dub the regularized SALP (RSALP) — then essentially suggests itself:

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{x\in\mathcal{S}} \nu_x \langle \mathbf{x}, \mathbf{z}\rangle + b - \kappa \sum_{x\in\mathcal{S}} \pi_x s_x - \frac{\Gamma}{2}\langle \mathbf{z}, \mathbf{z}\rangle \\ \text{subject to} & \langle \mathbf{x}, \mathbf{z}\rangle + b \leq g_{a,x} + \alpha \mathsf{E}_{x,a}[\langle \mathbf{X}', \mathbf{z}\rangle + b] + s_x, \quad \forall\ x \in \mathcal{S},\ a \in \mathcal{A}, \\ & \mathbf{z} \in \mathcal{H},\ b \in \mathbb{R},\ s \in \mathbb{R}_+^{\mathcal{S}}. \end{array} \tag{4}$$

The only new ingredient in the program above is the fact that we regularize the weight vector $\mathbf{z}$ using the parameter $\Gamma > 0$. Penalizing the objective of (4) the according to the square of the norm $\|\mathbf{z}\|_{\mathcal{H}} \triangleq \sqrt{\langle \mathbf{z}, \mathbf{z}\rangle}$ anticipates that we will eventually resort to sampling in solving this program. In a sampled setting, regularization is necessary to avoid over-fitting and, in particular, to construct an approximation that generalizes well to unsampled states. This regularization, which plays a crucial role both in theory and practice, is easily missed if one directly 'plugs in' a local averaging approximation in place of $z^\top \Phi(x)$, as is the case in the earlier work of Dietterich and Wang (2002), or a more general non-parametric approximation as in the work of Ernst et al. (2005).

Since the RSALP of (4) can be interpreted as a regularized stochastic optimization problem, one may hope to solve it via its sample average approximation. To this end, define the likelihood ratio $w_x \triangleq \nu_x/\pi_x$, and let $\hat{\mathcal{S}} \subset \mathcal{S}$ be a set of N states sampled independently according to the distribution π. The sample average approximation of (4) is then

$$\begin{array}{ll} \text{maximize} & \displaystyle\frac{1}{N}\sum_{x\in\hat{\mathcal{S}}} w_x \langle \mathbf{x}, \mathbf{z}\rangle + b - \frac{\kappa}{N}\sum_{x\in\hat{\mathcal{S}}} s_x - \frac{\Gamma}{2}\langle \mathbf{z}, \mathbf{z}\rangle \\ \text{subject to} & \langle \mathbf{x}, \mathbf{z}\rangle + b \leq g_{a,x} + \alpha \mathsf{E}_{x,a}[\langle \mathbf{X}', \mathbf{z}\rangle + b] + s_x, \quad \forall\ x \in \hat{\mathcal{S}},\ a \in \mathcal{A}, \\ & \mathbf{z} \in \mathcal{H},\ b \in \mathbb{R},\ s \in \mathbb{R}_+^{\hat{\mathcal{S}}}. \end{array} \tag{5}$$

We call this program the sampled RSALP. Even if $|\hat{\mathcal{S}}|$ were small, it is still not clear that this program can be solved effectively. We will, in fact, solve the dual to this problem.

2.3. Dual Formulation

We begin by establishing some notation. Let $\mathcal{N}_{x,a} \triangleq \{x\} \cup \{x' \in \mathcal{S} \ : \ p(x, x', a) > 0\}$ denote the set of states that can be reached starting at a state $x \in \mathcal{S}$ given an action $a \in \mathcal{A}$. For any states $x, x' \in \mathcal{S}$ and action $a \in \mathcal{A}$, define $q_{x,x',a} \triangleq \mathbf{1}_{\{x=x'\}} - \alpha p(x, x', a)$. Now, define the symmetric positive semi-definite matrix $Q \in \mathbb{R}^{(\hat{\mathcal{S}}\times\mathcal{A})\times(\hat{\mathcal{S}}\times\mathcal{A})}$ according to

$$Q(x, a, x', a') \triangleq \sum_{\substack{y\in\mathcal{N}_{x,a} \\ y'\in\mathcal{N}_{x',a'}}} q_{x,y,a} q_{x',y',a'} \langle \mathbf{y}, \mathbf{y}'\rangle, \tag{6}$$

[2] Separating the scalar offset b from the linear term parameterized by $\mathbf{z}$ will permit us to regularize these two quantities differently in the sequel.

the vector $R \in \mathbb{R}^{\hat{\mathcal{S}} \times \mathcal{A}}$ according to

$$R(x,a) \triangleq \Gamma g_{x,a} - \frac{1}{N} \sum_{\substack{x' \in \hat{\mathcal{S}} \\ y \in \mathcal{N}_{x,a}}} w_{x'} q_{x,y,a} \langle \mathbf{y}, \mathbf{x}' \rangle, \tag{7}$$

and the scalar S as

$$S \triangleq - \sum_{x \in \hat{\mathcal{S}}} \sum_{y \in \hat{\mathcal{S}}} w_x w_y \langle \mathbf{x}, \mathbf{y} \rangle.$$

Notice that Q, R and S depend only on inner products in $\mathcal{X}$ (and other easily computable quantities). The dual to (5) is then given by:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\lambda^\top Q \lambda + R^\top \lambda + S \\ \text{subject to} & \sum_{a \in \mathcal{A}} \lambda_{x,a} \leq \frac{\kappa}{N}, \qquad \forall \;\; x \in \hat{\mathcal{S}}, \\ & \sum_{\substack{x \in \hat{\mathcal{S}} \\ a \in \mathcal{A}}} \lambda_{x,a} = \frac{1}{1-\alpha}, \\ & \lambda \in \mathbb{R}_+^{\hat{\mathcal{S}} \times \mathcal{A}}. \end{array} \tag{8}$$

Assuming that Q, R and S can be easily computed, this finite dimensional quadratic program, *is* tractable – its size is polynomial in the number of sampled states. We may recover a primal solution (i.e., the weight vector $\mathbf{z}^*$) from an optimal dual solution:

Proposition 1. *Programs* (5) *and* (8) *have equal (finite) optimal values. The optimal solution to* (8) *is attained at some* λ^**. The optimal solution to* (5) *is attained at some* (z^*, s^*, b^*) *with*

$$\mathbf{z}^* = \frac{1}{\Gamma} \left[\frac{1}{N} \sum_{x \in \hat{\mathcal{S}}} w_x \mathbf{x} - \sum_{x \in \hat{\mathcal{S}}, a \in \mathcal{A}} \lambda^*_{x,a} \Big(\mathbf{x} - \alpha \mathsf{E}_{x,a}[\mathbf{X}'] \Big) \right]. \tag{9}$$

Having solved this program, we may, using Proposition 1, recover our approximate cost-to-go function $\tilde{J}(x) = \langle \mathbf{z}^*, \mathbf{x} \rangle + b^*$ as

$$\tilde{J}(x) = \frac{1}{\Gamma} \left[- \sum_{\in \hat{\mathcal{S}}, a \in \mathcal{A}} \lambda^*_{y,a} (\langle \mathbf{y}, \mathbf{x} \rangle - \alpha \mathsf{E}_{y,a}[\langle \mathbf{X}', \mathbf{x} \rangle]) + \frac{1}{N} \sum_{y \in \hat{\mathcal{S}}} w_y \langle \mathbf{y}, \mathbf{x} \rangle \right] + b^*. \tag{10}$$

A policy greedy with respect to $\tilde{J}$ is not affected by constant translations, hence in (10), the value of b^* can be set to be zero arbitrarily. Again note that given λ^*, evaluation of $\tilde{J}$ only involves inner products in $\mathcal{X}$.

2.4. Kernels

As pointed out earlier, the sampled RSALP is potentially difficult to work with. Proposition 1 establishes that solving this program (via its dual) is a computation that scales polynomially in N so that it can be solved efficiently provided inner products in $\mathcal{H}$ can be evaluated cheaply. Alternatively, we may have arrived at a similar conclusion by observing that in any optimal solution to (5), we must have that $\mathbf{z}^* \in \text{span}\left\{\mathbf{x} \ : \ x \in \hat{\mathcal{S}} \cup \mathcal{N}(\hat{\mathcal{S}})\right\}$, where $\mathcal{N}(\hat{\mathcal{S}})$ denotes the set of states that can be reached from the sampled states of $\hat{\mathcal{S}}$ in a single transition. Then, one can restrict the feasible region of (5) to this subspace. In either approach, we observe that one need not necessarily have explicitly characterized the feature map $\Phi(\cdot)$; knowing $\langle \Phi(x), \Phi(y) \rangle$ for all $x, y \in \mathcal{S}$ would suffice and so the algorithm designer can focus on simply specifying these inner products. This leads us to what is popularly referred to as the 'kernel trick' which we discuss next without the assumption that $\mathcal{S}$ is necessarily a finite set.

A kernel is a map $K : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$; we will call such a kernel positive definite if for any finite collection of elements $\{x_i\}_{1 \leq i \leq n}$ in $\mathcal{S}$, the Gram matrix $G \in \mathbb{R}^{n \times n}$ defined by $G_{ij} \triangleq K(x_i, x_j)$ is symmetric and positive semi-definite. Given such a kernel, we are assured of the existence of a Hilbert space $\mathcal{H}$ and a map $\Phi \colon \mathcal{S} \to \mathcal{H}$ such that[3] $\langle \Phi(x), \Phi(y) \rangle = K(x, y)$. The kernel trick then allows us to implicitly work with this space $\mathcal{H}$ by replacing inner products of the form $\langle \mathbf{x}, \mathbf{y} \rangle \triangleq \langle \Phi(x), \Phi(y) \rangle$ in (6), (7), and (10) by $K(x, y)$. Of course, the quality of the approximation one may produce depends on $\mathcal{H}$ and consequently on the kernel employed.

One case of special interest is where $\mathcal{S}$ is finite and the Gram matrix corresponding to this set is positive definite. A kernel satisfying this condition is known as *full-dimensional* or *full-rank*. In this case, we may take $\mathcal{H}$ to be the $|\mathcal{S}|$-dimensional Euclidean space. Working with such a kernel would imply working with an approximation architecture where *any* function $J \colon \mathcal{S} \to \mathbb{R}$ can be exactly expressed in the form $J(x) = \langle \Phi(x), \mathbf{z} \rangle$, for some weight vector $\mathbf{z} \in \mathcal{H}$.

There are a number of kernels one can specify on the state space; we list a few examples here for the case where $\mathcal{S} \subset \mathbb{R}^n$. The polynomial kernel is defined according to

$$K(x, y) \triangleq (1 + x^\top y)^d.$$

A corresponding Hilbert space $\mathcal{H}$ is given by the span of all monomials of degree up to d on $\mathbb{R}^n$. A Gaussian kernel is defined according to

$$K(x, y) \triangleq \exp\left(-\frac{\|x - y\|^2}{\sigma}\right).$$

The Gaussian kernel is known to be full-dimensional (see, e.g., Theorem 2.18, Scholkopf and Smola, 2001), so that employing such a kernel in our setting would correspond to working with an infinite dimensional approximation architecture. A thorough exposition on this topic, along with many

[3] A canonical such space is the so-called 'reproducing kernel' Hilbert space, by the Moore-Aronszajn theorem. For certain sets $\mathcal{S}$, Mercer's theorem provides another important construction of such a Hilbert space.

more examples can be found in the text of Scholkopf and Smola (2001).

2.5. Overall Procedure

Our development thus far suggests the following non-parametric algorithm that requires as input a kernel K and the distributions π and ν. It also requires that we set the parameter κ and the number of samples N.

1. Sample N states from a distribution π.
2. Solve (8) with

$$Q(x,a,x',a') \triangleq \sum_{y\in\mathcal{N}_{x,a}} \sum_{y'\in\mathcal{N}_{x',a'}} q_{x,y,a} q_{x',y',a'} K(y,y'), \tag{11}$$

and

$$R(x,a) \triangleq \left(\Gamma g_{x,a} - \frac{1}{N} \sum_{y\in\hat{\mathcal{S}}} \sum_{x'\in\mathcal{N}_x,a} w_y q_{x,x',a} K(y,x') \right). \tag{12}$$

The value of S may be set to be zero.

3. Let λ^* be the dual optimal. Define an approximate value function by

$$\tilde{J}(x) \triangleq \frac{1}{\Gamma} \left[\frac{1}{N} \sum_{y\in\hat{\mathcal{S}}} w_y K(x,y) - \sum_{y\in\hat{\mathcal{S}},a\in\mathcal{A}} \lambda^*_{y,a} (K(x,y) - \alpha \mathsf{E}_{y,a}[K(X',x)]) \right]. \tag{13}$$

In the next section we will develop theory to characterize the sample complexity of our procedure and the approximation quality it provides. This theory will highlight the roles of the kernel K and the sampling distributions used.

3. Approximation Guarantees

3.1. Overview

Recall that we are employing an approximation $\tilde{J}_{\mathbf{z},b}$ of the form

$$\tilde{J}_{\mathbf{z},b} = \langle \mathbf{x}, \mathbf{z} \rangle + b$$

parameterized by the weight vector $\mathbf{z}$ and the offset parameter b. For the purpose of establishing theoretical guarantees, in this section, we will look at the following variation of the RSALP of (4):

$$
\begin{aligned}
& \text{maximize} && \sum_{x\in\mathcal{S}} \nu_x \langle \mathbf{x}, \mathbf{z}\rangle + b - \kappa \sum_{x\in\mathcal{S}} \pi_x s_x \\
(14) \qquad & \text{subject to} && \langle \mathbf{x}, \mathbf{z}\rangle + b \le g_{a,x} + \alpha \mathsf{E}_{x,a}[\langle \mathbf{X}', \mathbf{z}\rangle + b] + s_x, \quad \forall\, x \in \mathcal{S},\ a \in \mathcal{A}, \\
& && \|\mathbf{z}\|_{\mathcal{H}} \le C,\ |b| \le B, \\
& && \mathbf{z} \in \mathcal{H},\ b \in \mathbb{R},\ s \in \mathbb{R}_+^{\mathcal{S}}.
\end{aligned}
$$

Here, rather than regularizing by penalizing according to the weight vector $\mathbf{z}$ in the objective as in (4), we regularize by restricting the size of the weight vector as a constraint. This regularization constraint is parameterized by the scalar $C \ge 0$. It is easy to see that (14) is equivalent to the original problem (4), in the following sense: for any $\Gamma > 0$, there exists a $C \ge 0$ so that for all B sufficiently large, (4) and (14) have the same optimal solutions and value. Let $C(\Gamma)$ by any such value of C, corresponding to a particular Γ.

Now let

$$
\mathcal{C}\big(C(\Gamma), B\big) \triangleq \{(\mathbf{z}, b) \in \mathcal{H} \times \mathbb{R} \ :\ \|\mathbf{z}\|_{\mathcal{H}} \le C(\Gamma),\ |b| \le B\}\,.
$$

The best approximation to J^* within this set has ℓ_∞-approximation error

$$
\inf_{(\mathbf{z},b)\in\mathcal{C}\left(C(\Gamma),B\right)} \|J^* - \tilde{J}_{\mathbf{z},b}\|_\infty. \tag{15}
$$

Now observe that if $\text{span}\,\{\mathbf{x} \ :\ x \in \mathcal{S}\} \subset \mathcal{H}$ has dimension $|\mathcal{S}|$ (i.e., if the kernel is full-dimensional), there exists a $\tilde{\mathbf{z}} \in \mathcal{H}$ satisfying $\tilde{J}_{\tilde{\mathbf{z}},0} = J^*$. Consequently, for $C(\Gamma)$ sufficiently large and any value of $B \ge 0$, we have that $\tilde{\mathbf{z}} \in \mathcal{C}\big(C(\Gamma), B\big)$ — the approximation error in (15) reduces to zero. More generally, as $C(\Gamma)$ and B grow large, we expect the approximation error in (15) to monotonically decrease; the precise rate of this decrease will depend, of course, on the feature map Φ, which in turn will depend on the kernel we employ.

Loosely speaking, in this section, we will demonstrate that:

1. For any given $C(\Gamma)$ and B, exact solution of the RSALP (14) will produce an approximation with ℓ_∞-error comparable to the optimal ℓ_∞-error possible for approximations $\tilde{J}_{\mathbf{z},b}$ with $\|\mathbf{z}\|_{\mathcal{H}} \le C(\Gamma)$ and $b \le B$.

2. Under a certain set of idealized assumptions, solving a *sampled* variation of the RSALP (14) with a number of samples $N\big(C(\Gamma), B\big)$ that scales gracefully with $C(\Gamma)$, B, and other natural parameters, produces a near optimal solution to the RSALP with high probability. These sample complexity bounds will *not* depend on the dimension of the approximation architecture $\mathcal{H}$.

Since $C(\Gamma)$ and B are parameters of the algorithm designer's choosing, these results will effectively show that with the ability to use a larger number of samples, the designer may choose larger values of $C(\Gamma)$ and B and consequently produce approximations of improving quality. Under mild assumptions on the kernel, the approximation error can be made arbitrarily small in this fashion.

3.2. Preliminaries and an Idealized Program

Before proceeding with our analysis, we introduce some preliminary notation. For a vector $x \in \mathbb{R}^n$, and a 'weight' vector $v \in \mathbb{R}^n_+$, we denote by $\|x\|_v \triangleq \sum_{i=1}^n v_i|x_i|$ the weighted 1-norm of x. Let $\Psi = \{\psi \in \mathbb{R}^{\mathcal{S}} \ : \ \psi \geq \mathbf{1}\}$, be the set of all functions on the state space bounded from below by 1. For any $\psi \in \Psi$, let us define the weighted ∞-norm $\|\cdot\|_{\infty,1/\psi}$ by

$$||J||_{\infty,1/\psi} \triangleq \max_{x\in\mathcal{S}} \ |J(x)|/\psi(x).$$

Our use of such weighted norms will allow us to emphasize approximation error differently across the state space. Further, we define

$$\beta(\psi) \triangleq \max_{x\in\mathcal{X},\ a\in\mathcal{A}} \ \frac{\sum_{x'} p(x,x',a)\psi(x')}{\psi(x)}.$$

For a given ψ, $\beta(\psi)$ gives us the worst-case expected gain of the Lyapunov function ψ for any state action pair (x,a).

Finally, we define an idealized sampling distribution. Let ν be an arbitrary distribution over $\mathcal{S}$ and denote by μ^* and P_{μ^*} an optimal policy and the transition probability matrix corresponding to this policy, respectively. We define a distribution over $\mathcal{S}$, $\pi_{\mu^*,\nu}$ according to

$$\pi^\top_{\mu^*,\nu} \triangleq (1-\alpha)\nu^\top(I-\alpha P_{\mu^*})^{-1} = (1-\alpha)\sum_{t=0}^{\infty}\alpha^t\nu^\top P^t_{\mu^*}. \tag{16}$$

This idealized distribution will play the role of the sampling distribution π in the sequel.

It is notationally convenient for us introduce the Bellman operator defined by

$$(TJ)(x) \triangleq \min_{a\in\mathcal{A}} \ \Big[g(x,a) + \alpha\mathsf{E}_{x,a}[J(X')]\Big],$$

for all $x \in \mathcal{S}$ and $J\colon \mathcal{S} \to \mathbb{R}$. As before, let $\hat{\mathcal{S}}$ be a set of N states drawn independently at random from $\mathcal{S}$; here we pick a specific sampling distribution $\pi = \pi_{\mu^*,\nu}$. Given the definition of $\tilde{J}_{\mathbf{z},b}$ in (3), the 'idealized' sampled program we consider is:

$$\begin{array}{lll} \text{maximize} & \nu^\top\tilde{J}_{\mathbf{z},b} - \dfrac{2}{1-\alpha}\dfrac{1}{N}\displaystyle\sum_{x\in\hat{\mathcal{S}}} s_x & \\ \text{subject to} & \tilde{J}_{\mathbf{z},b}(x) \leq T\tilde{J}_{\mathbf{z},b}(x) + s_x, & \forall\ x \in \hat{\mathcal{S}}, \\ & \|\mathbf{z}\|_{\mathcal{H}} \leq C,\ |b| \leq B, & \\ & \mathbf{z} \in \mathcal{H},\ b \in \mathbb{R},\ s \in \mathbb{R}^{\hat{\mathcal{S}}}_+. & \end{array} \tag{17}$$

This program is a sampled variant of (14) and is closely related to the sampled RSALP (5) introduced in the last section. Before proceeding with an analysis of the quality of approximation afforded by solving this program, we discuss its connection with the sampled RSALP (5) of the previous section:

1. The distributions ν and π: We allow for an arbitrary distribution ν, but given this distribution require that $\pi = \pi_{\mu^*,\nu}$. In particular, the distribution ν might be chosen as the empirical distribution corresponding to N independent draws from $\mathcal{S}$ under some measure; one would then draw N independent samples under $\pi = \pi_{\mu^*,\nu}$ to construct the second term in the objective. In a sense, this is the only 'serious' idealized assumption we make here. Given the broad nature of the class of problems considered it is hard to expect meaningful results without an assumption of this sort, and indeed much of the antecedent literature considering parametric LP based approaches makes such an assumption. When the sampling distribution π is a close approximation to $\pi_{\mu^*,\nu}$ (say, the likelihood ratio between the two distributions is bounded), then it is possible to provide natural extensions to our results that account for how close the sampling distribution used is to the idealized sampling distribution.

2. Regularization: We regularize the weight vector $\mathbf{z}$ with an explicit constraint on $\|\mathbf{z}\|_{\mathcal{H}}$; this permits a transparent analysis and is equivalent to placing the 'soft' regularization term $\frac{\Gamma}{2}\|\mathbf{z}\|_{\mathcal{H}}$ in the objective. In particular, the notation $C(\Gamma)$ makes this equivalence explicit. In addition, we place a separate upper bound on the offset term B. This constraint is not binding if $B > 2KC + \|g\|_\infty/(1-\alpha)$ but again, permits a transparent analysis of the dependence of the sample complexity of solving our idealized program on the offset permitted by the approximation architecture.

3. The choice of κ: The smoothing parameter κ can be interpreted as yet another regularization parameter, here on the (one-sided) Bellman error permitted for our approximation. Our idealized program chooses a specific value for this smoothing parameter, in line with that chosen by Desai et al. (2011). Our experiments will use the same value of κ; experience with the parametric SALP suggests that this is a good choice of the parameter in practice.

3.3. The Approximation Guarantee

Let $(\hat{\mathbf{z}}, \hat{b})$ be an optimal solution to the idealized sampled program (17) and let $K \triangleq \max_{x\in\mathcal{S}} \|\mathbf{x}\|_{\mathcal{H}}$. Further define,

$$\Xi(C, B, K, \delta) \triangleq \left(4CK(1+\alpha) + 4B(1-\alpha) + 2\|g\|_\infty\right)\left(1 + \sqrt{\frac{1}{2}\ln(1/\delta)}\right).$$

Notice that $\Xi(C, B, K, \delta)^2$ scales as the square of C, K, and B and further is $O(\ln 1/\delta)$. The following result will constitute our main approximation guarantee:

Theorem 1. *For any $\epsilon, \delta > 0$, let $N \geq \Xi(C, B, K, \delta)^2/\epsilon^2$. If $(\hat{\mathbf{z}}, \hat{b})$ is an optimal solution to* (17)*,*

then with probability at least $1 - \delta - \delta^4$,

$$\left\| J^* - \tilde{J}_{\hat{\mathbf{z}},\hat{b}} \right\|_{1,\nu} \leq \inf_{\|\mathbf{z}\|_{\mathcal{H}} \leq C,\ |b| \leq B,\ \psi \in \Psi} \|J^* - \tilde{J}_{\mathbf{z},b}\|_{\infty,1/\psi} \left(\nu^\top \psi + \frac{2(\pi_{\mu^*,\nu}^\top \psi)(\alpha\beta(\psi) + 1)}{1 - \alpha} \right) + \frac{4\epsilon}{1 - \alpha}. \tag{18}$$

The remainder of this section is dedicated to parsing and interpreting this guarantee:

1. Optimal approximation error: Taking ψ to be the vector of all ones, we see that the right side of the approximation guarantee (18) is bounded above by

$$\frac{3 + \alpha}{1 - \alpha} \inf_{\|\mathbf{z}\|_{\mathcal{H}} \leq C,\ |b| \leq B} \|J^* - \tilde{J}_{\mathbf{z},b}\|_\infty + \frac{4\epsilon}{1 - \alpha}.$$

   In particular, ignoring the ϵ-dependent error term, we see that the quality of approximation provided by $(\hat{\mathbf{z}}, \hat{b})$ is essentially within a constant multiple (at most $(3 + \alpha)/(1 - \alpha)$) of the optimal (in the sense of ℓ_∞-error) approximation to J^* possible using a weight vector $\mathbf{z}$ and offsets b permitted by the regularization constraints. This is a 'structural' error term that will persist even if one were permitted to draw an arbitrarily large number of samples. It is analogous to the approximation results produced in parametric settings with the important distinction that *one allows comparisons to approximations in potentially full-dimensional sets* which might, as we have argued earlier, be substantially superior.

2. Dimension independent sampling error: In addition to the structural error above, one incurs an additional additive 'sampling' error that scales like $4\epsilon/(1 - \alpha)$. The result demonstrates that

$$\epsilon = O\left(\frac{(CK + B)\sqrt{\ln 1/\delta}}{\sqrt{N}} \right)$$

   This is an important contrast with existing parametric sample complexity guarantees. In particular, *existing guarantees typically depend on the dimension of the space spanned by the basis function architecture* $\{\mathbf{x} : x \in \mathcal{S}\}$. Here, this space may be full-dimensional, so that such a dependence would translate to a vacuous guarantee. Instead we see that the dependence on the approximation architecture is through the constants C, K and B. Of these K can, for many interesting kernels be upper bounded by a constant that is independent of $|\mathcal{S}|$, while C and B are user-selected regularization bounds. Put another way, the guarantee allows for arbitrary 'simple' (in the sense of $\|\mathbf{z}\|_{\mathcal{H}}$ being small) approximations in a rich feature space as opposed to restricting us to some a priori fixed, low-dimensional feature space. This yields some intuition for why we expect the approach to perform well even with a relatively general choice of kernel.

3. A non-parametric interpretation: As we have argued earlier, in the event that span $\{\mathbf{x} : x \in \mathcal{S}\}$ is full-dimensional, there exists a choice of C and B for which the optimal approximation error is, in fact, zero. A large number of kernels would guarantee such feature maps. More

generally, as C and B grow large, $\inf_{\|\mathbf{z}\|_{\mathcal{H}} \leq C,\ |b| \leq B} \|J^* - \tilde{J}_{\mathbf{z},b}\|_\infty$ will decrease. In order to maintain the sampling error constant, one would then need to increase N at a rate that is roughly $\Omega((CK + B)^2)$. In summary, by increasing the number of samples in the sampled program, we can (by increasing C and B appropriately) hope to compute approximations of increasing quality, approaching an *exact* approximation. In the event that the feature space permits good approximations that are 'simple' for that space (i.e., with $\|\mathbf{z}\|_{\mathcal{H}}$ small), the approach is capable of producing good approximations for a tractable number of samples.

3.4. Proof of Theorem 1

We prepare the ground for the proof by developing appropriate uniform concentration guarantees for appropriate function classes.

3.4.1. Uniform Concentration Bounds

We begin with defining the empirical Rademacher complexity of a class of functions $\mathcal{F}$ from $\mathcal{S}$ to R as

$$\hat{R}_n(\mathcal{F}) = \mathsf{E}\left[\sup_{f \in \mathcal{F}} \frac{2}{n} \sum_{i=1}^{n} \sigma_i f(X_i) \middle| X_1, X_2, \ldots, X_n\right],$$

where σ_i are i.i.d. random variable that take value 1 with probability $1/2$ and -1 with probability $1/2$. The X_i are i.i.d. $\mathcal{S}$-valued random variables drawn with the distribution π. We denote by $R_n(\mathcal{F}) \triangleq \mathsf{E}\hat{R}_n(\mathcal{F})$ the Rademacher complexity of $\mathcal{F}$.

We begin with the following abstract sample complexity result: let $\mathcal{F}$ be a class of functions mapping $\mathcal{S}$ to $\mathbb{R}$ that are uniformly bounded by some constant $\overline{B}$. Moreover denote for any function $f \in \mathcal{F}$, the empirical expectation $\hat{\mathsf{E}}_n f(X) \triangleq \frac{1}{n}\sum_{i=1}^{n} f(X_i)$, where the X_i are i.i.d. random draws from $\mathcal{S}$ as above. We then have the following sample complexity result:

Lemma 1.

$$\mathsf{P}\left(\sup_{f \in \mathcal{F}} \mathsf{E}f(X) - \hat{\mathsf{E}}_n f(X) \geq R_n(\mathcal{F}) + \sqrt{\frac{2\overline{B}^2 \ln(1/\delta)}{n}}\right) \leq \delta.$$

This result is standard; for completeness, the proof may be found in Appendix B. Next, we establish the Rademacher complexity of a specific class of functions. Fixing a policy μ, consider then the set of functions mapping $\mathcal{S}$ to $\mathbb{R}$ defined according to:

$$\mathcal{F}_{\mathcal{S},\mu} \triangleq \left\{x \mapsto \langle \mathbf{x}, \mathbf{z}\rangle - \alpha \mathsf{E}_{x,\mu(x)}[\langle \mathbf{X}', \mathbf{z}\rangle] \ :\ \|\mathbf{z}\|_{\mathcal{H}} \leq C\right\}.$$

We have:

Lemma 2. *For any policy μ,*

$$R_n\left(\mathcal{F}_{\mathcal{S},\mu}\right) \leq \frac{2CK(1+\alpha)}{\sqrt{n}}.$$

Proof. Observe that, due to triangle inequality

$$\|\mathbf{x} - \alpha \mathsf{E}_{x,\mu(x)}[\mathbf{X}']\|_{\mathcal{H}} \leq \|\mathbf{x}\|_{\mathcal{H}} + \alpha \mathsf{E}_{x,\mu(x)}[\|\mathbf{X}'\|_{\mathcal{H}}] \leq K(1+\alpha),$$

for all $x \in \mathcal{S}$. Now, let X_i be i.i.d. samples in $\mathcal{S}$ and $\mathbf{X}_i$ be the corresponding elements in $\mathcal{H}$,

$$\begin{aligned}
\hat{R}_n\left(\mathcal{F}_{\mathcal{S},\mu}\right) &= \frac{2}{n}\mathsf{E}\left[\sup_{z:\ \|z\|_{\mathcal{H}} \leq C} \left\langle \sum_i \sigma_i \left(\mathbf{X}_i - \alpha \mathsf{E}_{X_i,\mu(X_i)}[\mathbf{X}']\right), z \right\rangle \middle| X_1, \ldots, X_n\right] \\
&\leq \frac{2}{n}\mathsf{E}\left[\sup_{z:\ \|z\|_{\mathcal{H}} \leq C} \left\| \sum_i \sigma_i \left(\mathbf{X}_i - \alpha \mathsf{E}_{X_i,\mu(X_i)}[\mathbf{X}']\right)\right\|_{\mathcal{H}} \|z\|_{\mathcal{H}} \middle| X_1, \ldots, X_n\right] \\
&= \frac{2C}{n}\mathsf{E}\left[\left\| \sum_i \sigma_i \left(\mathbf{X}_i - \alpha \mathsf{E}_{X_i,\mu(X_i)}[\mathbf{X}']\right)\right\|_{\mathcal{H}} \middle| X_1, \ldots, X_n\right] \\
&\leq \frac{2C}{n}\sqrt{\sum_i \|\mathbf{X}_i - \alpha \mathsf{E}_{X_i,\mu(X_i)}[\mathbf{X}']\|_{\mathcal{H}}^2} \\
&\leq \frac{2CK(1+\alpha)}{\sqrt{n}}.
\end{aligned}$$

■

Now, consider the class of functions mapping $\mathcal{S}$ to $\mathbb{R}$, defined according to:

$$\overline{\mathcal{F}}_{\mathcal{S},\mu} \triangleq \left\{ x \mapsto \left(\tilde{J}_{\mathbf{z},b}(x) - (T_\mu \tilde{J}_{\mathbf{z},b})(x)\right)^+ \ : \ \|\mathbf{z}\|_{\mathcal{H}} \leq C,\ |b| \leq B \right\}.$$

Now, $\overline{\mathcal{F}}_{\mathcal{S},\mu} = \phi\left(\mathcal{F}_{\mathcal{S},\mu} + (1-\alpha)\mathcal{F}_B - g_\mu\right)$, where $\phi \triangleq (\cdot)^+$ and $\mathcal{F}_B \triangleq \{x \mapsto b \ : \ |b| \leq B\}$. It is easy to show that $R_n(\mathcal{F}_B) \leq 2B/\sqrt{n}$, so that with the previous lemma, the results of Bartlett and Mendelson (2002, Theorem 12, parts 4 and 5) allow us to conclude

Corollary 1.

$$R_n(\overline{\mathcal{F}}_{\mathcal{S},\mu}) \leq \frac{4CK(1+\alpha) + 4B(1-\alpha) + 2\|g_\mu\|_\infty}{\sqrt{n}} \triangleq \frac{\overline{C}}{\sqrt{n}}.$$

Now, define

$$\overline{\mathcal{F}}_{\mathcal{S}} \triangleq \left\{ x \mapsto \left(\tilde{J}_{\mathbf{z},b}(x) - (T\tilde{J}_{\mathbf{z},b})(x)\right)^+ \ : \ \|\mathbf{z}\|_{\mathcal{H}} \leq C, |b| \leq B \right\}.$$

We have:

Lemma 3. *For every $f \in \overline{\mathcal{F}}_{\mathcal{S}}$ we have that $\|f\|_\infty \leq \overline{C}/2$. Moreover,*

$$\mathsf{P}\left(\hat{\mathsf{E}}_N f(X) - \mathsf{E} f(X) \geq \epsilon\right) \leq \delta^4,$$

provided $N \geq \Xi(C, B, K, \delta)^2/\epsilon^2$.

The first claim above follows from routine algebra and the Cauchy-Schwartz inequality; the second is Hoeffding's inequality. Corollary 1, Lemma 1, and the first part of Lemma 3 yields the following sample complexity result:

Theorem 2. *Provided* $N \geq \Xi(C, B, K, \delta)^2/\epsilon^2$ *, we have*

$$\mathsf{P}\left(\sup_{f \in \bar{\mathcal{F}}_{\mathcal{S},\mu}} \mathsf{E}f(X) - \hat{\mathsf{E}}_N f(X) \geq \epsilon\right) \leq \delta.$$

Theorem 2 will constitute a crucial sample complexity bound for our main result; we now proceed with the proof of Theorem 1.

3.4.2. Proof of Theorem 1

Let $(\hat{\mathbf{z}}, \hat{b}, \hat{s})$ be the optimal solution to the sampled program (17). Define

$$\hat{s}_{\mu^*} \triangleq (\tilde{J}_{\hat{\mathbf{z}},\hat{b}} - T_{\mu^*}\tilde{J}_{\hat{\mathbf{z}},\hat{b}})^+.$$

Observe that we may assume, without loss of generality, that

$$\hat{s} = (\tilde{J}_{\hat{\mathbf{z}},\hat{b}} - T\tilde{J}_{\hat{\mathbf{z}},\hat{b}})^+,$$

so that $\hat{s} \geq \hat{s}_{\mu^*}$. Now, by definition, $\tilde{J}_{\hat{\mathbf{z}},\hat{b}} \leq T_{\mu^*}\tilde{J}_{\hat{\mathbf{z}},\hat{b}} + \hat{s}_{\mu^*}$, so that by Lemma 2 of Desai et al. (2011), we have that

$$\tilde{J}_{\hat{\mathbf{z}},\hat{b}} \leq J^* + \Delta^*\hat{s}_{\mu^*},$$

where $\Delta^* = (I - \alpha P_{\mu^*})^{-1}$. Let $\hat{\pi}_{\mu^*,\nu}$ be the empirical distribution obtained by sampling N states according to $\pi_{\mu^*,\nu}$. Now let $\mathbf{z}, b$ satisfying $\|\mathbf{z}\|_{\mathcal{H}} \leq C, |b| \leq B$ be given, and define $s_{\mathbf{z},b} \triangleq (T\tilde{J}_{\mathbf{z},b} - \tilde{J}_{\mathbf{z},b})^+$. Then, $(\mathbf{z}, b, s_{\mathbf{z},b})$ constitute a feasible solution to (17). Finally, let $\psi \in \Psi$ be arbitrary. We then have with probability at least $1 - \delta - \delta^4$,

$$\begin{aligned}
\|J^* - \tilde{J}_{\hat{\mathbf{z}},\hat{b}}\|_{1,\nu} &= \|J^* - \tilde{J}_{\hat{\mathbf{z}},\hat{b}} + \Delta^*\hat{s}_{\mu^*} - \Delta^*\hat{s}_{\mu^*}\|_{1,\nu} \\
&\leq \|J^* - \tilde{J}_{\hat{\mathbf{z}},\hat{b}} + \Delta^*\hat{s}_{\mu^*}\|_{1,\nu} + \|\Delta^*\hat{s}_{\mu^*}\|_{1,\nu} \\
&= \nu^\top\left(J^* - \tilde{J}_{\hat{\mathbf{z}},\hat{b}}\right) + 2\nu^\top\Delta^*\hat{s}_{\mu^*} \\
&= \nu^\top\left(J^* - \tilde{J}_{\hat{\mathbf{z}},\hat{b}}\right) + \frac{2}{1-\alpha}\pi_{\mu^*,\nu}^\top\hat{s}_{\mu^*} \\
&\leq \nu^\top\left(J^* - \tilde{J}_{\hat{\mathbf{z}},\hat{b}}\right) + \frac{2}{1-\alpha}\hat{\pi}_{\mu^*,\nu}^\top\hat{s}_{\mu^*} + \frac{2\epsilon}{1-\alpha} \\
&\leq \nu^\top\left(J^* - \tilde{J}_{\hat{\mathbf{z}},\hat{b}}\right) + \frac{2}{1-\alpha}\hat{\pi}_{\mu^*,\nu}^\top\hat{s} + \frac{2\epsilon}{1-\alpha} \\
&\leq \nu^\top\left(J^* - \tilde{J}_{\mathbf{z},b}\right) + \frac{2}{1-\alpha}\hat{\pi}_{\mu^*,\nu}^\top s_{\mathbf{z},b} + \frac{2\epsilon}{1-\alpha} \\
&\leq \nu^\top\left(J^* - \tilde{J}_{\mathbf{z},b}\right) + \frac{2}{1-\alpha}\pi_{\mu^*,\nu}^\top s_{\mathbf{z},b} + \frac{4\epsilon}{1-\alpha}. \\
&\leq (\nu^\top\psi)\|J^* - \tilde{J}_{\mathbf{z},b}\|_{\infty,1/\psi} + \frac{2}{1-\alpha}(\pi_{\mu^*,\nu}^\top\psi)\|T\tilde{J}_{\mathbf{z},b} - \tilde{J}_{\mathbf{z},b}\|_{\infty,1/\psi} + \frac{4\epsilon}{1-\alpha}.
\end{aligned} \tag{19}$$

The second equality follows by our observation that $\tilde{J}_{\hat{\mathbf{z}},\hat{b}} \leq J^* + \Delta^* \hat{s}_{\mu^*}$ and since $\Delta^* \hat{s}_{\mu^*} \geq 0$. The second inequality above holds with probability at least $1-\delta$ by virtue of Theorem 2 and the fact that $\hat{s}_{\mu^*} \in \overline{\mathcal{F}}_{\mathcal{S},\mu^*}$. The subsequent inequality follows from our observation that $\hat{s} \geq \hat{s}_{\mu^*}$. The fourth inequality follows from the assumed optimality of $(\hat{\mathbf{z}}, \hat{b}, \hat{s})$ for the sampled program (17) and the feasibility of $(\mathbf{z}, b, s_{\mathbf{z},b})$ for the same. The fifth inequality holds with probability $1-\delta^4$ and follows from the Hoeffding bound in Lemma 3 since $s_{\mathbf{z},b} \in \overline{\mathcal{F}}_{\mathcal{S}}$. The final in equality follows from the observation that for any $s \in \mathbb{R}^{\mathcal{S}}, \psi \in \Psi$ a and probability vector ν, $\nu^\top s \leq (\nu^\top \psi)\|s\|_{\infty,1/\psi}$.

Now the proof of Theorem 2 of Desai et al. (2011) establishes that for any $\psi \in \Psi$ and $J \in \mathbb{R}^{\mathcal{S}}$, we have,

$$\|TJ - J\|_{\infty,1/\psi} \leq (1 + \alpha\beta(\psi))\|J^* - J\|_{\infty,1/\psi}.$$

Applied to (19), this yields

$$\|J^* - \tilde{J}_{\hat{\mathbf{z}},\hat{b}}\|_{1,\nu} \leq \|J^* - \tilde{J}_{\mathbf{z},b}\|_{\infty,1/\psi} \left(\nu^\top \psi + \frac{2(\pi_{\mu^*,\nu}^\top \psi)(\alpha\beta(\psi)+1)}{1-\alpha} \right) + \frac{4\epsilon}{1-\alpha}.$$

Since our choice of $\mathbf{z}, b$ was arbitrary (beyond satisfying $\|\mathbf{z}\|_{\mathcal{H}} \leq C, |b| \leq B$), the result follows.

4. Numerical Scheme

This section outlines an efficient numerical scheme we use to solve the regularized SALP. In particular, we would like to solve the sampled dual problem (8), introduced in Section 2.3, in order to find an approximation to the optimal cost-to-go function. This approach requires solving a quadratic program (QP) with $N \times A$ variables, where $N \triangleq |\hat{\mathcal{S}}|$ is the number of sampled states and $A \triangleq |\mathcal{A}|$ is the number of possible actions. Furthermore, constructing the coefficient matrices Q and R for (8) requires $O(N^2A^2H^2)$ arithmetic operations, where H is the maximum number of states that can be reached from an arbitrary state-action pair, i.e.,

$$H \triangleq \max_{(x,a)\in\mathcal{S}\times\mathcal{A}} \left|\{x' \in \mathcal{S} \ : \ p(x,x',a) > 0\}\right|.$$

These computationally expensive steps may prevent scaling up solution of the QP to a large number of samples. Also, an off-the-shelf QP solver will typically store the matrix Q in memory, so that the memory required to solve our QP with an off-the-shelf solver effectively scales like $O(N^2A^2)$.

In this section, we develop an iterative scheme to solve the program (8) that, by exploiting problem structure, enjoys low computational complexity per iteration and attractive memory requirements. Our scheme is an active set method in the vein of the approaches used by Osuna et al. (1997) and Joachims (1999), among others, for solving large SVM problems. The broad steps of the scheme are as follows:

1. At the tth iteration, a subset $\mathcal{B} \subset \hat{\mathcal{S}} \times \mathcal{A}$ of the decision variables of (8) – the 'active set' – is chosen. Only variables in this set may be changed in a given iteration; these changes must preserve feasibility of the new solution that results. Our algorithm will limit the size of

the active set to two variables, i.e., $|\mathcal{B}| = 2$. The methodology for choosing this active set is crucial and will be described in the sequel.

2. Given the subset $\mathcal{B}$, we solve (8) for $\lambda^{(t)}$, where all variables except those in $\mathcal{B}$ must remain unchanged. In other words, $\lambda^{(t)}_{x,a} \triangleq \lambda^{(t-1)}_{x,a}$ for all $(x,a) \notin \mathcal{B}$. This entails the solution of a QP with only $|\mathcal{B}|$ decision variables. In our case, we will be able to solve this problem in closed form.

3. Finally, if the prior step does not result in a decrease in objective value we conclude that we are at an optimal solution; Proposition 2 establishes that this is, in fact, a correct termination criterion.

In the following section, we will establish an approach for selecting the active set at each iteration and show how Step 2 above can be solved while maintaining feasibility at each iteration. We will establish that steps one and two together need no more that $O(NA \log NA)$ arithmetic operations and comparisons, and moreover that the memory requirement for our procedure scales like $O(NA)$. Finally, we will establish that our termination criterion is correct: in particular, if no descent direction of cardinality at most two exists at a given feasible point, we must be at an optimal solution.

4.1. Subset Selection

The first step in the active set method is to choose the subset $\mathcal{B} \subset \hat{\mathcal{S}} \times \mathcal{A}$ of decision variables to optimize over. Given the convex objective in (8), if the prior iteration of the algorithm is at a sub-optimal point $\lambda \triangleq \lambda^{(t-1)}$, then there exists a direction $d \in \mathbb{R}^{\hat{\mathcal{S}} \times \mathcal{A}}$ such that $\lambda + \epsilon d$ is feasible with a lower objective value for $\epsilon > 0$ sufficiently small. To select a subset to optimize over, we look for such a descent direction d of low cardinality $\|d_0\|_0 \leq q$, i.e., a vector d that is zero on all but at most q components. If such a direction can be found, then we can use the set of non-zero indices of d as our set $\mathcal{B}$.

This problem of finding a 'sparse' descent direction d can be posed as

$$
\begin{aligned}
\text{minimize} \quad & h(\lambda)^\top d \\
\text{subject to} \quad & \sum_{a \in \mathcal{A}} d_{x,a} \leq 0, \quad \forall\ x \in \hat{\mathcal{S}} \text{ with } \sum_{x \in \mathcal{A}} \lambda_{x,a} = \frac{\kappa}{N}, \\
& d_{x,a} \geq 0, \qquad \forall\ (x,a) \in \hat{\mathcal{S}} \times \mathcal{A} \text{ with } \lambda_{x,a} = 0, \\
& \sum_{\substack{x \in \hat{\mathcal{S}} \\ a \in \mathcal{A}}} d_{x,a} = 0, \\
& ||d||_0 \leq q, \\
& ||d||_\infty \leq 1, \\
& d \in \mathbb{R}^{\hat{\mathcal{S}} \times \mathcal{A}}.
\end{aligned}
\tag{20}
$$

Here, $h(\lambda) \triangleq Q\lambda + R$ is the gradient of the objective of (8) at a feasible point λ, thus the objective

$h(\lambda)^\top d$ seeks to find a direction d of steepest descent. The first three constraints ensure that d is a feasible direction. The constraint $\|d\|_0 \leq q$ is added to ensure that the direction is of cardinality at most q. Finally, the constraint $\|d\|_\infty \leq 1$ is added to ensure that the program is bounded, and may be viewed as normalizing the scale of the direction d.

The program (20) is, in general, a challenging mixed integer program because of the cardinality constraint. Joachims (1999) discusses an algorithm to solve a similar problem of finding a low cardinality descent direction in an SVM classification setting. Their problem can be easily solved provided that the cardinality q is even, however no such solution seems to exist for our case. However, in our case, when $q = 2$, there is a tractable way to solve (20). We will restrict attention to this special case, i.e., consider only descent directions of cardinality two. In Section 4.3, we will establish that this is, in fact, sufficient: if the prior iterate λ is sub-optimal, then there will exist a direction of descent of cardinality two.

To begin, define the sets

$$\mathcal{P}_1 \triangleq \left\{(x,a) \in \hat{\mathcal{S}} \times \mathcal{A} \ : \ \lambda_{a,x} = 0\right\}, \qquad \mathcal{P}_2 \triangleq \left\{x \in \hat{\mathcal{S}} \ : \ \sum_{a \in \mathcal{A}} \lambda_{x,a} = \frac{\kappa}{N}\right\}.$$

Consider the following procedure:

1. Sort the set of indices $\hat{\mathcal{S}} \times \mathcal{A}$ according to their corresponding component values in the gradient vector $h(\lambda)$. Call this sorted list $\mathcal{L}_1$.

2. Denote by (x_1, a_1) the largest element of $\mathcal{L}_1$ such that $(x_1, a_1) \notin \mathcal{P}_1$, and denote by (x_2, a_2) the smallest element of $\mathcal{L}_1$ such that $x_2 \notin \mathcal{P}_2$. Add the tuple (x_1, a_1, x_2, a_2) to the list $\mathcal{L}_2$.

3. Consider all $x \in \mathcal{P}_2$. For each such x, denote by (x, a_1) the largest element of $\mathcal{L}_1$ such that $(x, a_1) \notin \mathcal{P}_1$, and denote by (x, a_2) the smallest element of $\mathcal{L}_1$. Add each tuple (x, a_1, x, a_2) to $\mathcal{L}_2$.

4. Choose the element $(x_1^*, a_1^*, x_2^*, a_2^*)$ of $\mathcal{L}_2$ that optimizes

$$\min_{(x_1,a_1,x_2,a_2) \in \mathcal{L}_2} h(\lambda)_{x_2,a_2} - h(\lambda)_{x_1,a_1}. \tag{21}$$

   Set $d_{x_1^*,a_1^*} = -1$, $d_{x_2^*,a_2^*} = 1$, and all other components of d to zero..

This procedure finds a direction of maximum descent of cardinality two by examining considering candidate index pairs (x_1, a_1, x_2, a_2) for which $h(\lambda)_{x_2,a_1} - h(\lambda)_{x_1,a_1}$ is minimal. Instead of considering at all N^2A^2 such pairs, the routine selectively checks only pairs with describe feasible directions with respect to the constraints of (20). Step 2 considers all feasible pairs with different states x, while Step 3 considers all pairs with the same state. It is thus easy to see that the output of this procedure is an optimal solution to (20), i.e., a direction of steepest descent of cardinality two. Further, if the value of the minimal objective (21) determined by this procedure is non-negative, then no descent direction of cardinality two exists, and the algorithm terminates.

In terms of computational complexity, this subset selection step requires us to first compute the gradient of the objective function $h(\lambda) \triangleq Q\lambda + R$. If the gradient is known at the first iteration, then we can update it at each step by generating two columns of Q, since λ only changes at two co-ordinates. Hence, the gradient calculation can be performed in $O(NA)$ time, and with $O(NA)$ storage (since it is not necessary to store Q). For Step 1 of the subset selection procedure, the component indices must be sorted in the order given by the gradient $h(\lambda)$. This operation requires computational effort of the order $O(NA \log NA)$. With the sorted indices, the effort required in the remaining steps to find the steepest direction is via the outlined procedure is $O(NA)$. Thus, our subset selection step requires a total of $O(NA \log NA)$ arithmetic operations and comparisons.

The initialization of the gradient requires $O(N^2A^2)$ effort. In the cases where such a computation is prohibitive, one can think of many approaches to approximately evaluate the initial gradient. For example, if we use the Gaussian kernel, the matrix Q will have most of its large entries close to the diagonal. In this case, instead of the expensive evaluation of Q, we can only evaluate the entries $Q(x, a, x', a')$ where $x = x'$ and the set the rest of them to zero. This block diagonal approximation might be used to initialize the gradient. Another approach is to sample from the distribution induced by λ to approximately evaluate $Q\lambda$. As the algorithm makes progress it evaluates new columns of Q. With a bit of book-keeping one could get rid of the errors associated with the gradient initialization. The convergence properties of the active set method with approximate initialization is an issue not tackled in this paper.

4.2. QP Sub-problem

Given a prior iterate $\lambda^{(t-1)}$, and a subset $\mathcal{B} \triangleq \{(x_1, a_1), (x_2, a_2)\}$ of decision variable components of cardinality two as computed in Section 4.1, we have the restricted optimization problem

$$
\begin{aligned}
& \text{minimize} && \sum_{(x,a)\in\mathcal{B}} \sum_{(x',a')\in\mathcal{B}} \lambda_{x',a'} Q(x, a, x', a') \lambda_{x,a} \\
& && + \sum_{(x,a)\in\mathcal{B}} \lambda_{x,a} \left(R(x, a) + 2 \sum_{(x',a')\notin\mathcal{B}} Q(x, a, x', a') \lambda^{(t-1)}_{x',a'} \right) \\
& \text{subject to} && \sum_{a:\ (x,a)\in\mathcal{B}} \lambda_{x,a} \leq \frac{\kappa}{N} - \sum_{a:\ (x,a)\notin\mathcal{B}} \lambda^{(t-1)}_{x,a}, && \forall\ x \in \{x_1, x_2\} \\
& && \sum_{(x,a)\in\mathcal{B}} \lambda_{x,a} = \frac{1}{1-\alpha} - \sum_{(x,a)\notin\mathcal{B}} \lambda^{(t-1)}_{x,a}, \\
& && \lambda \in \mathbb{R}^{\mathcal{B}}_{+}.
\end{aligned}
\tag{22}
$$

This sub-problem has small dimension. In fact, the equality constraint implies that $\lambda_{x_1,a_1} + \lambda_{x_2,a_2}$ is constant, hence, the problem is in fact a one-dimensional QP that can be solved in closed form. Further, to construct this QP, two columns of Q are required to be generated. This requires computation effort of order $O(NA)$.

Note that the subset $\mathcal{B}$ is chosen so that it is guaranteed to contain a descent direction, according to the procedure in Section 4.1. Then, the solution of (20) will produce an iterate $\lambda^{(t)}$ that is feasible

for the original problem (22) and has lower objective value than the prior iterate $\lambda^{(t-1)}$.

4.3. Correctness of Termination Condition

The following proposition establishes the correctness of our active set method: if the prior iterate $\lambda \triangleq \lambda^{(t-1)}$ is sub-optimal, then there must exist a direction of descent of cardinality two. Our iterative procedure will therefore improve the solution, and will only terminate when global optimality is achieved.

Proposition 2. *If $\lambda \in \mathbb{R}^{\hat{\mathcal{S}}\times\mathcal{A}}$ is feasible but sub-optimal for* (8) *then, there exists a descent direction of cardinality two.*

The proof of Proposition 2 requires the following lemma:

Lemma 4. *Suppose $x, y \in \mathbb{R}^n$ are vectors such that $\mathbf{1}^\top x = 0$ and $x^\top y < 0$. Then there exist co-ordinates $\{i, j\}$, such that $y_i < y_j$, $x_i > 0$, and $x_j < 0$.*

Proof. Define the index sets $S^+ \triangleq \{i \ : \ x_i > 0\}$ and $S^- \triangleq \{i \ : \ x_i < 0\}$. Under the given hypotheses, both sets are non-empty. Using the fact that $\mathbf{1}^\top x = 0$, define

$$Z \triangleq \sum_{i\in S^+} x_i = \sum_{i\in S^-} (x_i)^-,$$

where $(x_i)^- \triangleq -\min(x_i, 0)$ is the negative part of the scalar x_i. Observe that $Z > 0$. Further, since $x^\top y < 0$,

$$\frac{1}{Z}\sum_{i\in S^+} x_i y_i < \frac{1}{Z}\sum_{i\in S^-} (x_i)^- y_i.$$

Since the weighted average of y over S^- is more than its weighted average over S^+, we can pick an element in S^+, i and an element of S^-, j such that $y_i < y_j$. ■

Proof of Proposition 2. If λ is sub-optimal, since (8) is a convex quadratic program, there will exist some vector $d \in \mathbb{R}^{\hat{\mathcal{S}}\times\mathcal{A}}$ is a feasible descent direction at λ. Let $g \triangleq h(\lambda)$ be the gradient at that point. We must have that $g^\top d < 0$, so that it is a descent direction, and that d satisfies the first three constraints of (20), so that it is a feasible direction.

Define

$$\mathcal{T} \triangleq \left\{ x \in \hat{\mathcal{S}} \ : \ \sum_{a\in\mathcal{A}} \lambda_{x,a} = \frac{\kappa}{N}, \ \max_{a\in\mathcal{A}} \ d_{x,a} > 0, \ \min_{a\in\mathcal{A}} \ d_{x,a} < 0 \right\}, \quad \mathcal{P}_x \triangleq \{a \in \mathcal{A} \ : \ d_{x,a} \neq 0\}.$$

First, consider the case of $\mathcal{T} = \emptyset$. In this case, for all x such that $\sum_{a\in\mathcal{A}} \lambda_{x,a} = \kappa/N$, we have $d_{x,a} \leq 0$. Since d is a descent direction, we have $g^\top d < 0$. Lemma 4 can be applied to get a pair (x_1, a_1) and (x_2, a_2) such that $d_{x_1,a_1} > 0$ and $d_{x_2,a_2} < 0$, with $g_{x_1,a_1} < g_{x_2,a_2}$. Further x_1 is such that $\sum_{a\in\mathcal{A}} \lambda_{x_1,a} < \kappa/N$ and (x_2, a_2) is such that $\lambda_{x_2,a_2} > 0$. These conditions ensure that if (x_1, a_1) and (x_2, a_2) are increased and decreased respectively in the same amount, then the objective is

decreased. And hence we obtain a descent direction of cardinality 2. By assuming that $\mathcal{T} = \emptyset$, we have avoided the corner case of not being able to increase d_{x_1,a_1} due to $\sum_{a\in\mathcal{A}} \lambda_{x_1,a} = \kappa/N$.

For $\mathcal{T} \neq \emptyset$, without loss of generality, assume that $|\mathcal{P}_x| = 2$ for each $x \in \mathcal{T}$, i.e., d has exactly two non-zero components corresponding to the state x. This is justified at the end of the proof. Denote these indices by (x, a_+) and (x, a_-), so that $d_{x,a_+} > 0$ and $d_{x,a_-} < 0$. From the first constraint of (20), we must have that $d_{x,a_+} \leq (d_{x,a_-})^-$.

There are two cases:

(i) Suppose $g_{x,a_+} < g_{x,a_-}$, for some $x \in \mathcal{T}$.

Then, we can define a descent direction $\tilde{d} \in \mathbb{R}^{\hat{\mathcal{S}}\times\mathcal{A}}$ of cardinality two by setting $\tilde{d}_{x,a_+} = 1$, $\tilde{d}_{x,a_-} = -1$, and all other components of $\tilde{d}$ to zero.

(ii) Suppose that $g_{x,a_+} \geq g_{x,a_-}$, for all $x \in \mathcal{T}$.

For all $x \in \mathcal{T}$, define $\hat{d}_x \triangleq (d_{x,a_-})^- - d_{x,a_+} \geq 0$. Then, the fact that $\sum_{x,a} d_{x,a} = 0$ implies that

$$\sum_{\substack{x\notin\mathcal{T}\\ a\in\mathcal{A}}} d_{x,a} - \sum_{x\in\mathcal{T}} \hat{d}_x = 0. \tag{23}$$

At the same time, for all $x \in \mathcal{T}$, we have that

$$d_{x,a_+} g_{x,a_+} + d_{x,a_-} g_{x,a_-} = -\hat{d}_x g_{x,a_-} + d_{x,a_+}(g_{x,a_+} - g_{x,a_-}) \geq -\hat{d}_x g_{x,a_-}.$$

Then, since d is a descent direction, we have that

$$\sum_{\substack{x\notin\mathcal{T}\\ a\in\mathcal{A}}} d_{x,a} g_{x,a} - \sum_{x\in\mathcal{T}} \hat{d}_x g_{x,a_-} < 0. \tag{24}$$

Now, define the vector $\tilde{d} \in \mathbb{R}^{\hat{\mathcal{S}}\times\mathcal{A}}$ by

$$\tilde{d}_{x,a} = \begin{cases} d_{x,a} & \text{if } x \notin \mathcal{T}, \\ -\hat{d}_x & \text{if } x \in \mathcal{T} \text{ and } (x,a) = (x,a_-), \\ 0 & \text{otherwise.} \end{cases}$$

Applying Lemma 4 to (23) and (24), there must be a pair of indices (x_1, a_1) and (x_2, a_2) such that $\tilde{d}_{x_1,a_1} > 0$, $\tilde{d}_{x_2,a_2} < 0$ and $g_{x_1,a_1} < g_{x_2,a_2}$. For such (x_1, a_1) and (x_2, a_2) we have a descent direction where we can increase λ_{x_1,a_1} and decrease λ_{x_2,a_2} by the same amount and get a decrease in the objective. Note that since $\tilde{d}_{x_1,a_1} > 0$, we have that $d_{x_1,a_1} > 0$ and $x_1 \notin \mathcal{T}$, hence $\sum_a \lambda_{x_1,a} < \kappa/N$. Also, by construction, (x_2, a_2) is chosen such that $d_{x_2,a_2} < 0$, implying that $\lambda_{x_2,a_2} > 0$. Thus the specified direction is also feasible, and we have a feasible descent direction of cardinality two.

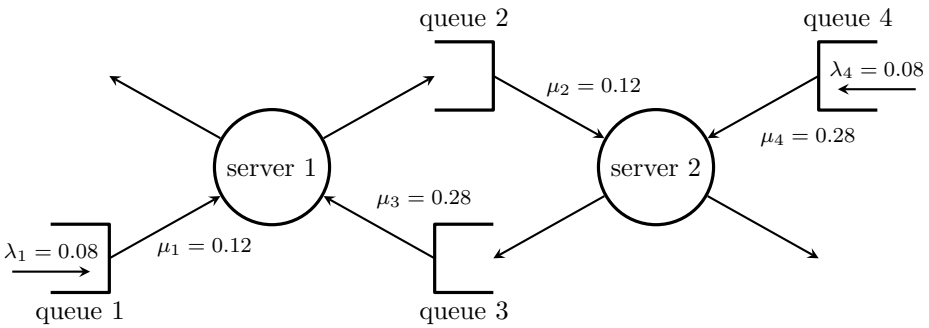

Figure 1: The queueing network example.

Finally, to complete the proof, consider the case where there are some $x \in \mathcal{T}$ with $|\mathcal{P}_x| > 2$, i.e., d has more than two non-zero components corresponding to the state x. For each $x \in \mathcal{T}$, define

$$\mathcal{A}_x^+ \triangleq \{a \in \mathcal{A} \ : \ d_{x,a} > 0\}, \qquad \mathcal{A}_x^- \triangleq \{a \in \mathcal{A} \ : \ d_{x,a} < 0\},$$

$$a_1 \in \operatorname*{argmin}_{a \in \mathcal{A}_x^+} \ g_{x,a}, \qquad a_2 \in \operatorname*{argmax}_{a \in \mathcal{A}_x^-} \ g_{x,a}.$$

Define a new direction $\tilde{d} \in \mathbb{R}^{\hat{\mathcal{S}} \times \mathcal{A}}$ by

$$\tilde{d}_{x,a} = \begin{cases} \sum_{a' \in \mathcal{A}_x^+} d_{x,a'} & \text{if } x \in \mathcal{T} \text{ and } (x,a) = (x,a_1), \\ \sum_{a' \in \mathcal{A}_x^-} d_{x,a'} & \text{if } x \in \mathcal{T} \text{ and } (x,a) = (x,a_2), \\ d_{x,a} & \text{otherwise.} \end{cases}$$

It is easy to verify that $\tilde{d}$ is also a feasible descent direction. Furthermore, $\tilde{d}$ has only two non-zero components corresponding to each start $x \in \mathcal{T}$. ■

5. Case Study: A Queueing Network

This section considers the problem of controlling the queuing network illustrated in Figure 1, with the objective of minimizing long run average delay. There are two 'flows' in this network: the first though server 1 followed by server 2 (with buffering at queues 1 and 2, respectively), and the second through server 2 followed by server 1 (with buffering at queues 4 and 3, respectively). Here, all inter-arrival and service times are exponential with rate parameters summarized in Figure 1.

This specific network has been studied by de Farias and Van Roy (2003); Chen and Meyn (1998); Kumar and Seidman (1990), for example, and closely related networks have been studied by Harrison and Wein (1989); Kushner and Martins (1996); Martins et al. (1996); Kumar and Muthuraman (2004). It is widely considered to be a challenging control problem. As such, a lot of thought has been invested in designing scheduling policies with networks of this type in mind. Our goal in this section will be two fold. First, we will show that the RSALP, used 'out-of-the-box' with a generic kernel, can match or surpass the performance of tailor made heuristics and state of

the art parametric ADP methods. Second, we will show that the RSALP can be solved efficiently.

5.1. MDP Formulation

Although the control problem at hand is nominally a continuous time problem, it is routinely converted into a discrete time problem via a standard uniformization device; see Moallemi et al. (2008), for instance, for an explicit such example. In the equivalent discrete time problem, at most a single event can occur in a given epoch, corresponding either to the arrival of a job at queues 1 or 4, or the arrival of a service token for one of the four queues with probability proportional to the corresponding rates. The state of the system is described by the number of jobs is each of the four queues, so that $\mathcal{S} \triangleq \mathbb{Z}_+^4$, whereas the action space $\mathcal{A}$ consists of four potential actions each corresponding to a matching between servers and queues. We take the single period cost as the total number of jobs in the system, so that $g_{x,a} = \|x\|_1$; note that minimizing the average number of jobs in the system is equivalent to minimizing average delay by Little's law. Finally, we take $\alpha = 0.9$ as our discount factor.[4]

5.2. Approaches

The following scheduling approaches were considered for the queueing problem:

RSALP (this paper). We solve (8) using the active set method outlined in Section 4, taking as our kernel the standard Gaussian radial basis function kernel $K(x,y) \triangleq \exp\left(-\|x-y\|_2^2/h\right)$, with the bandwidth parameter[5] $h \triangleq 100$. Note that this implicitly corresponds to an full-dimensional basis function architecture. Since the idealized sampling distribution, $\pi_{\mu^*,\nu}$ is unavailable to us, we use in its place the geometric distribution

$$\pi(x) \triangleq (1-\zeta)^4 \zeta^{\|x\|_1}, \tag{25}$$

with the sampling parameter ζ set at 0.9. This choice mimics that of de Farias and Van Roy (2003). The regularization parameter Γ was chosen via a line-search;[6] we report results for $\Gamma \triangleq 10^{-6}$. In accordance with the theory we set the constraint violation parameter $\kappa \triangleq 2/(1-\alpha)$, as suggested by the analysis of Section 3, as well as by Desai et al. (2011) for the SALP.

SALP (Desai et al., 2011). The SALP formulation (2), is, as discussed earlier, the parametric counterpart to the RSALP. It may be viewed as a generalization of the ALP approach proposed by de Farias and Van Roy (2003) and has been demonstrated to provide substantial performance benefits relative to the ALP approach. Our choice of parameters for the SALP mirrors those for the RSALP to the extent possible, so as to allow for an 'apples-to-apples' comparison. Thus, as earlier, we solve the sample average approximation of this program using the same sampling distribution

[4] Note that while we will solve a problem with a discounted infinite horizon optimality criterion, we will report long run average costs. This is in keeping with Desai et al. (2011) and de Farias and Van Roy (2003)

[5] The sensitivity of our results to this bandwidth parameter appears minimal.

[6] Again, performance does not appear to be very sensitive to Γ, so that a crude line-search appears to suffice. Specifically, we tried values of the form $\Gamma = 10^k$, for k between -12 and 2.

π as in (25), and we set $\kappa \triangleq 2/(1-\alpha)$. Being a parametric approach, one needs to specify an appropriate approximation architecture. Approximation architectures that span polynomials are known to work well for queueing problem. We use the basis functions used by de Farias and Van Roy (2003) for a similar problem modeled directly in discrete time. In particular, we use all monomials with degree at most 3, which we will call the *cubic* basis, as our approximation architectures.

Longest Queue First (generic). This is a simple heuristic approach: at any given time, a server chooses to work on the longest queue from among those it can service.

Max-Weight (Tassiulas and Ephremides, 1992). Max-Weight is a well known scheduling heuristic for queueing networks. The policy is obtained as the greedy policy with respect to a value function approximation of the form

$$\tilde{J}_{MW}(x) \triangleq \sum_{i=1}^{4} |x_i|^{1+\epsilon},$$

given a parameter $\epsilon > 0$. This policy has been extensively studied and shown to have a number of good properties, for example, being throughput optimal (Dai and Lin, 2005) and offering good performance for critically loaded settings (Stolyar, 2004). Via a line-search we chose to use $\epsilon \triangleq 1.5$ as the exponent for our experiments.

5.3. Results

Policies were evaluated using a common set of arrival process sample paths. The performance metric we report for each control policy is the long run average number of jobs in the system under that policy,

$$\frac{1}{T}\sum_{t=1}^{T} \|x_t\|,$$

where we set $T \triangleq 10000$. We further average this random quantity over an ensemble of 300 sample paths.

Further, in order to generate SALP and RSALP policies, state sampling is required. To understand the effect of the sample size on the resulting policy performance, the different sample sizes listed in Table 1 were used. Since the policies generated involve randomness to the sampled states, we further average performance over 10 sets of sampled states. The results are reported in Table 1 and have the following salient features:

1. *RSALP outperforms established policies:* Approaches such as the Max-Weight or 'parametric' ADP with basis spanning polynomials have been previously shown to work well for the problem of interest. We see that the RSALP with more than 300 samples achieves performance that is superior to these extant schemes.

2. *Sampling improves performance:* This is expected from the theory in Section 3. Ideally, as the sample size is increased one should relax the regularization. However, for our experiments

policy	performance									
Longest Queue	8.09									
Max-Weight	6.55									
sample size	1000		3000		5000		10000		15000	
SALP, cubic basis	7.19	(1.76)	7.89	(1.76)	6.94	(1.15)	6.63	(0.92)	6.58	(1.12)
RSALP, Gaussian kernel	6.72	(0.39)	6.31	(0.11)	6.13	(0.08)	6.04	(0.05)	6.02	(0.06)

Table 1: Performance results in the queueing example. For the SALP and RSALP methods, the number in the parenthesis gives the standard deviation across sample sets.

we noticed that the performance is quite insensitive to the parameter Γ. Nonetheless, it is clear that larger sample sets yield a significant performance improvement.

3. *RSALP in less sensitive to state sampling:* We notice from the standard deviation values in Table 1 that our approach gives policies whose performance varies significantly less across different sample sets of the same size.

In summary, we view these results as indicative of the possibility that the RSALP may serve as a practical and viable alternative to state-of-the-art parametric ADP techniques.

6. Conclusions

This paper set out to present a practical non-parametric algorithm for approximate dynamic programming building upon the success of linear programming based methods that require the user to specify an approximation architecture. We believe that the RSALP, presented and studied here, is such an algorithm. In particular, the theoretical guarantees presented here establish the 'non-parametric' nature of the algorithm by showing that increased sampling effort leads to improved approximations. On the empirical front, we have shown that our essentially 'generic' procedure was able to match the performance of tailor made heuristics as well as ADP approaches using pre-specified approximation architectures. Nevertheless, several interesting directions for future work are evident at this juncture:

- The choice of kernel: The choice of kernel matters in so much as the feature map it encodes allows for approximations with small Hilbert norm (i.e., small C). Thus, a good choice of kernel would require fewer samples to achieve a fixed approximation accuracy than a poor choice of kernel. That said, picking a useful kernel is an apparently easier task than picking a low-dimensional architecture — there are many full-dimensional kernels possible, and with sufficient sampling, they will achieve arbitrarily good value function approximations. Nevertheless, it would be valuable to understand the interplay between the choice of kernel and the corresponding value of C for specific problem classes (asking for anything more general appears rather difficult).

- The active set method: In future work, we would like to fine tune/ build more extensible software implementing our active set method for wider use. Given the generic nature of the approach here, we anticipate that this can be a technology for high-dimensional MDPs that can be used 'out of the box'.

References

A. M. S. Barreto, D. Precup, and J. Pineau. Reinforcement learning using kernel-based stochastic factorization. In *Advances in Neural Information Processing Systems*, volume 24, pages 720–728. MIT Press, 2011.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II.* Athena Scientific, 2007.

B. Bethke, J. P. How, and A. Ozdaglar. Kernel-based reinforcement learning using Bellman residual elimination. MIT Working Paper, 2008.

R. R. Chen and S. Meyn. Value iteration and optimization of multiclass queueing networks. In *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, volume 1, pages 50 –55 vol.1, 1998.

J. G. Dai and W. Lin. Maximum pressure policies in stochastic processing networks. *Operations Research*, 53(2):197–218, 2005.

D. P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003.

D. P. de Farias and B. Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29:462–478, 2004.

V. V. Desai, V. F. Farias, and C. C. Moallemi. Approximate dynamic programming via a smoothed linear program. To appear in *Operations Research*, 2011.

T. G. Dietterich and X. Wang. Batch value function approximation via support vectors. In *Advances in Neural Information Processing Systems*, volume 14, pages 1491–1498. MIT Press, 2002.

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*, pages 154–161. AAAI Press, 2003.

D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, April 2005.

J. M. Harrison and L. M. Wein. Scheduling network of queues: Heavy traffic analysis of a simple open network. *Queueing Systems*, 5:265–280, 1989.

T. Joachims. *Making large-scale support vector machine learning practical*, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

J. Z. Kolter and A. Y. Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 521–528. ACM, 2009.

P. R. Kumar and T. I. Seidman. Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *IEEE Transactions on Automatic Control*, 35(3): 289 –298, March 1990. ISSN 0018-9286.

S. Kumar and K. Muthuraman. A numerical method for solving singular stochastic control problems. *Operations Research*, 52(4):563–582, 2004.

H. J. Kushner and L. F. Martins. Heavy traffic analysis of a controlled multiclass queueing network via weak convergence methods. *SIAM J. Control Optim.*, 34(5):1781–1797, 1996.

D. G. Luenberger. *Optimization by Vector Space Methods*. John Wiley & Sons, Inc., 1997.

A. S. Manne. Linear programming and sequential decisions. *Management Science*, 6(3):pp. 259–267, 1960. ISSN 00251909. URL `http://www.jstor.org/stable/2627340`.

L. F. Martins, S. E. Shreve, and H. M. Soner. Heavy traffic convergence of a controlled multiclass queueing network. *SIAM J. Control Optim.*, 34(6):2133–2171, 1996.

C. C. Moallemi, S. Kumar, and B. Van Roy. Approximate and data-driven dynamic programming for queueing networks. Working Paper, 2008.

D. Ormoneit and P. Glynn. Kernel-based reinforcement learning in average cost poblems. *IEEE Transactions on Automatic Control*, 47(10):1624–1636, 2002.

D. Ormoneit and S. Sen. Tree-based batch mode reinforcement learning. *Machine Learning*, 49(2): 161–178, 2002.

E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Neural Networks for Signal Processing, Proceedings of the 1997 IEEE Workshop*, pages 276 –285, sep 1997.

J. Pazis and R. Parr. Non-parametric approximate linear programming for MDPs. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

M. Petrik, G. Taylor, R. Parr, and S. Zilberstein. Feature selection using regularization in approximate linear programs for Markov decision processes. In *Proceedings of the 27th Annual International Conference on Machine Learning*, pages 871–879. ACM, 2010.

B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2001.

P. Schweitzer and A. Seidman. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110:568–582, 1985.

A. L. Stolyar. Maxweight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic. *The Annals of Applied Probability*, 14:1–53, 2004.

L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, December 1992.

B. Van Roy. Performance loss bounds for approximate value iteration with state aggregation. *Mathematics of Operations Research*, 31(2):234–244, 2006.

X. Xu, D. Hu, and X. Lu. Kernel-based least squares policy iteration for reinforcement learning. *IEEE Transactions on Neural Networks*, 18(4):973–992, 2007.

A. Duality of the Sampled RSALP

Proof of Proposition 1. We begin with a few observations about the primal program, (5):

1. Because the objective function is coercive,[7] weight vector $\mathbf{z}$ can be restricted without loss of generality to some finite ball in $\mathcal{H}$. The optimal value of the primal is consequently finite.

2. The primal has a feasible interior point: consider setting

$$\mathbf{z} \triangleq \mathbf{0}, \quad b \triangleq 0, \quad s_x \triangleq \max(-\min_a g_{x,a}, \epsilon),$$

   for some $\epsilon > 0$.

3. The optimal value of the primal is achieved. To see this, we note that it suffices to restrict $\mathbf{z}$ to the finite dimensional space spanned by the vectors $\{\mathbf{x} : x \in \hat{\mathcal{S}} \cup \mathcal{N}(\hat{\mathcal{S}})\}$, where $\mathcal{N}(\hat{\mathcal{S}})$ denotes the set of states that can be reached from the sampled states of $\hat{\mathcal{S}}$ in a single transition. Then, the feasible region of the primal can be restricted, without loss of optimality, to a compact subset of $\mathcal{H} \times \mathbb{R}^{\hat{\mathcal{S}}} \times \mathbb{R}$. Since the objective function of the primal is continuous, we know that its optimal value must be achieved by the Weierstrass theorem.

[7] As $\|\mathbf{z}\|_{\mathcal{H}} \to \infty$, the objective value goes to $-\infty$.

We next derive the dual to (5). As in Luenberger (1997, Chapter 8), we define the Lagrangian:

$$\mathcal{L}(\mathbf{z}, b, s, \lambda) \triangleq \left\langle -\frac{1}{N}\sum_{x\in\hat{\mathcal{S}}} w_x \mathbf{x} + \sum_{(x,a)\in\hat{\mathcal{S}}\times\mathcal{A}} \lambda_{x,a}\left(\mathbf{x} - \alpha \mathsf{E}_{x,a}[\mathbf{X}']\right), \mathbf{z}\right\rangle + \frac{\Gamma}{2}||\mathbf{z}||_{\mathcal{H}}^2$$
$$+ \sum_{x\in\hat{\mathcal{S}}} s_x \left(\frac{\kappa}{N} - \sum_{a\in\mathcal{A})} \lambda_{x,a}\right) - b\left(1 - (1-\alpha)\sum_{(x,a)\in\hat{\mathcal{S}}\times\mathcal{A}} \lambda_{x,a}\right) - \sum_{(x,a)\in\hat{\mathcal{P}}} g_{x,a}\lambda_{x,a}.$$

and define the dual function $G(\lambda) \triangleq \inf_{(\mathbf{z},b,s)\in\mathcal{D}} \mathcal{L}(\mathbf{z}, b, s, \lambda)$ where we denote by $\mathcal{D}$ the feasible region of the primal problem. Now, observe that for any given λ, $\mathcal{L}(\mathbf{z}, b, s, \lambda)$ is (uniquely) minimized at

$$\mathbf{z}^*(\lambda) = \frac{1}{\Gamma}\left[\frac{1}{N}\sum_{x\in\hat{\mathcal{S}}} w_x \mathbf{x} - \sum_{x\in\hat{\mathcal{S}}, a\in\mathcal{A}} \lambda_{x,a}\Big(\mathbf{x} - \alpha \mathsf{E}_{x,a}[\mathbf{X}']\Big)\right], \tag{26}$$

for any finite b, s. This follows from the observation that for any $\overline{\mathbf{z}} \in \mathcal{H}$, $\langle \mathbf{z}, \mathbf{z}\rangle - \langle \bar{\mathbf{z}}, \mathbf{z}\rangle$ is minimized at $-\frac{1}{2}\overline{\mathbf{z}}$ by the Cauchy-Schwartz inequality. It follows immediately that on the set defining the feasible region of the program (8), we must have that

$$G(\lambda) = \frac{1}{2}\lambda^\top Q\lambda + R^\top\lambda + S$$

and moreover that $G(\lambda) = +\infty$ outside that set. This suffices to establish that the dual problem $\inf_{\lambda\geq 0} G(\lambda)$ is precisely program (8).

The first conclusion of Luenberger (1997, Theorem 1, pp. 224–225) and the first and second observations we made at the outset of our proof then suffice to establish that programs (5) and (8) have equal optimal values (i.e. strong duality holds) and that the optimal value of the dual program is achieved at some λ^*. Our third observation, (26), and the second conclusion of Luenberger (1997, Theorem 1, pp. 224–225) then suffice to establish our second claim. ■

B. Proof of Lemma 1

Proof of Lemma 1. We begin with some preliminaries: Define

$$Z(X_1, X_2, \ldots, X_n) \triangleq \sup_{f\in\mathcal{F}} \mathsf{E}f(X) - \hat{E}_n f(X).$$

Notice that

$$\left|Z(X_1, X_2, \ldots, X_i, \ldots, X_n) - Z(X_1, X_2, \ldots, X_i', \ldots, X_n)\right| \leq \frac{2\overline{B}}{n}.$$

McDiarmid's inequality (or equivalently, Azuma's inequality) then implies:

$$\mathsf{P}\left(Z - \mathsf{E}Z \geq \sqrt{\frac{2\overline{B}^2\ln(1/\delta)}{n}}\right) \leq \delta. \tag{27}$$

Now,

$$\begin{aligned}
\mathsf{E}Z &= \mathsf{E}\left[\sup_{f\in\mathcal{F}} \mathsf{E}f(X) - \hat{\mathsf{E}}_n f(X)\right] \\
&= \mathsf{E}\left[\sup_{f\in\mathcal{F}} \mathsf{E}\hat{\mathsf{E}}_n f(X) - \hat{\mathsf{E}}_n f(X)\right] \\
&\leq \mathsf{E}\left[\sup_{f\in\mathcal{F}} \hat{\mathsf{E}}_n f(X') - \hat{\mathsf{E}}_n f(X)\right] \\
&= \mathsf{E}\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i \left(f(X_i') - f(X_i)\right)\right] \\
&\leq \mathsf{E}\left[\sup_{f\in\mathcal{F}} \frac{2}{n}\sum_{i=1}^{n} \sigma_i \left(f(X_i)\right)\right] \\
&= R_n(\mathcal{F})
\end{aligned}$$

With (27), this immediately yields the result. ■