

---

# A Kernel Loss for Solving the Bellman Equation

---

Yihao Feng<sup>1</sup> Lihong Li<sup>2</sup> Qiang Liu<sup>1</sup>

## Abstract

Value function learning plays a central role in many state-of-the-art reinforcement-learning algorithms. Many popular algorithms like Q-learning do not optimize any objective function, but are fixed-point iterations of some variant of Bellman operator that is not necessarily a contraction. As a result, they may easily lose convergence guarantees, as can be observed in practice. In this paper, we propose a novel loss function, which can be optimized using standard gradient-based methods without risking divergence. The key advantage is that its gradient can be easily approximated using sampled transitions, avoiding the need for double samples required by prior algorithms like residual gradient. Our approach may be combined with general function classes such as neural networks, on either on- or off-policy data, and is shown to work reliably and effectively in several benchmarks.

## 1. Introduction

The goal of a reinforcement learning (RL) agent is to optimize the policy to maximize the long-term return through repeated interaction with an external environment. The interaction is often modeled as a Markov decision process, whose value functions are the unique fixed points of their corresponding *Bellman operators*. Many state-of-the-art algorithms, including TD( $\lambda$ ), Q-learning and actor-critic, have value function learning as a key component (Sutton & Barto, 2018).

A fundamental property of the Bellman operator is that it is a contraction in the value function space in the  $\ell_\infty$ -norm (Puterman, 1994). Therefore, starting from any bounded initial function, with repeated applications of the operator, the value function converges to the correct value function. A number of algorithms are inspired by this property, such

as temporal difference (Sutton, 1988) and its many variants (Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 2018). Unfortunately, when function approximation such as neural networks is used to represent the value function in large-scale problems, the critical property of contraction is generally lost (e.g., Boyan & Moore, 1995; Baird, 1995; Tsitsiklis & Van Roy, 1997), except in rather restricted cases (e.g., Gordon, 1995; Tsitsiklis & Van Roy, 1997). This instability issue is not only one of the core theoretical challenges in RL, but also has broad practical significance, given the growing popularity of algorithms like DQN (Mnih et al., 2015), A3C (Mnih et al., 2016) and their many variants (e.g., Gu et al., 2016; Schulman et al., 2016; Wang et al., 2016; Wu et al., 2017), whose stability largely depends on the contraction property. The instability becomes even harder to avoid, when training data (transitions) are sampled from an off-policy distribution, a situation known as the *deadly triad* (Sutton & Barto, 2018, Sec. 11.3).

The brittleness of Bellman operator’s contraction property has inspired a number of works that aim to reformulate value function learning as an optimization problem, where standard algorithms like stochastic gradient descent can be used to minimize the objective, without the risk of divergence (under mild assumptions). One of the earliest attempts is residual gradient, or RG (Baird, 1995), which relies on minimizing squared temporal differences. The algorithm is convergent, but its objective is not necessarily a good proxy due to a well-known “double sample” problem. As a result, it may converge to an inferior solution; see Sections 2 and 6 for further details and numerical examples. This drawback is inherited by similar algorithms like PCL (Nachum et al., 2017; 2018).

Another line of work seeks alternative objective functions, the minimization of which leads to desired value functions (Sutton et al., 2009; Maei, 2011; Liu et al., 2015; Dai et al., 2017). Most existing works are either for linear approximation, or for evaluation of a fixed policy. An exception is the SBEED algorithm (Dai et al., 2018), which transforms the Bellman equation to a saddle-point problem. While SBEED is provably convergent under fairly standard conditions, it relies on solving a minimax problem, whose optimization can be rather challenging in practice, especially with nonconvex approximation classes like neural networks.

---

\*Equal contribution <sup>1</sup>Department of Computer Science, University of Texas, Austin <sup>2</sup>Google Research. Correspondence to: Yihao Feng <yihao@cs.utexas.edu>.

In this paper, we propose a novel loss function for value function learning. It avoids the double-sample problem (unlike RG), and can be easily estimated and optimized using sampled transitions (in both on- and off-policy scenarios). This is made possible by leveraging an important property of integrally strictly positive definite kernels (Stewart, 1976; Sriperumbudur et al., 2010). This new objective function allows us to derive simple yet effective algorithms to approximate the value function, without risking instability or divergence (unlike TD algorithms), or solving a more sophisticated saddle-point problem (unlike SBEED). Our approach also allows great flexibility in choosing the value function approximation classes, including nonlinear ones like neural networks. Experiments in several benchmarks demonstrate the effectiveness of our method, for both policy evaluation and optimization problems. We will focus on the batch setting (or the growing-batch setting with a growing replay buffer), and leave the online setting for future work.

## 2. Background

This section starts with necessary notation and background information, then reviews two representative algorithms that work with general, nonlinear (differentiable) function classes.

**Notation.** A Markov decision process (MDP) is denoted by  $M = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$ , where  $\mathcal{S}$  is a (possibly infinite) state space,  $\mathcal{A}$  an action space,  $P(s' | s, a)$  the transition probability,  $R(s, a)$  the average immediate reward, and  $\gamma \in (0, 1)$  a discount factor. The value function of a policy  $\pi : \mathcal{S} \mapsto \mathbb{R}_+^{\mathcal{A}}$ , denoted  $V^\pi(s) := \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_t \sim \pi(\cdot, s_t)]$ , measures the expected long-term return of a state. It is well-known that  $V = V^\pi$  is the unique solution to the *Bellman equation* (Puterman, 1994),  $V = \mathcal{B}_\pi V$ , where  $\mathcal{B}_\pi : \mathbb{R}^{\mathcal{S}} \rightarrow \mathbb{R}^{\mathcal{S}}$  is the *Bellman operator*, defined by

$$\mathcal{B}_\pi V(s) := \mathbb{E}_{a \sim \pi(\cdot | s), s' \sim P(\cdot | s, a)}[R(s, a) + \gamma V(s') | s].$$

While we develop and analyze our approach mostly for  $\mathcal{B}_\pi$  given a fixed  $\pi$  (policy evaluation), we will also extend the approach to policy optimization, where the corresponding Bellman operator is

$$\mathcal{B}V(s) := \max_a \mathbb{E}_{s' \sim P(\cdot | s, a)}[R(s, a) + \gamma V(s') | s, a].$$

The unique fixed point of  $\mathcal{B}$  is known as the optimal value function, denoted  $V^*$ ; that is,  $\mathcal{B}V^* = V^*$ .

Our work is built on top of an alternative to the fixed-point view above: given some fixed distribution  $\mu$  whose support is  $\mathcal{S}$ ,  $V^\pi$  is the unique minimizer of the *squared Bellman error*:

$$L_2(V) := \|\mathcal{B}_\pi V - V\|_\mu^2 = \mathbb{E}_{s \sim \mu}[(\mathcal{B}_\pi V(s) - V(s))^2].$$

Denote by  $\mathcal{R}_\pi V := \mathcal{B}_\pi V - V$  the Bellman error operator. With a set  $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{1 \leq i \leq n}$  of transitions where  $a_i \sim \pi(\cdot | s_i)$ , the Bellman operator in state  $s_i$  can be approximated by *bootstrapping*:  $\hat{\mathcal{B}}_\pi V(s_i) := r_i + \gamma V(s'_i)$ . Clearly,  $\mathbb{E}[\hat{\mathcal{B}}_\pi V(s_i) | s_i] = \mathcal{B}_\pi V(s_i)$ . In the literature,  $\hat{\mathcal{B}}_\pi V(s_i) - V(s_i)$  is also known as the temporal difference or TD error, whose expectation is the Bellman error. For notation, all distributions are equalized to their probability density functions in the work.

**Basic Algorithms.** We are interested in estimating  $V^\pi$ , from a parametric family  $\{V_\theta : \theta \in \Theta\}$ , from data  $\mathcal{D}$ . The *residual gradient* algorithm (Baird, 1995) minimizes the *squared TD error*:

$$\hat{L}_{\text{RG}}(V_\theta) := \frac{1}{n} \sum_{i=1}^n \left( \hat{\mathcal{B}}_\pi V_\theta(s_i) - V_\theta(s_i) \right)^2, \quad (1)$$

with gradient descent update  $\theta_{t+1} = \theta_t - \epsilon \nabla_\theta \hat{L}_{\text{RG}}(V_{\theta_t})$ , where

$$\nabla_\theta \hat{L}_{\text{RG}}(V_\theta) = \frac{2}{n} \sum_{i=1}^n \left( (\hat{\mathcal{B}}_\pi V_\theta(s_i) - V_\theta(s_i)) \cdot \nabla_\theta (\hat{\mathcal{B}}_\pi V_\theta(s_i) - V_\theta(s_i)) \right).$$

However, the objective in (1) is a biased and inconsistent estimate of the squared Bellman error. This is because  $\mathbb{E}_{s \sim \mu}[\hat{L}_{\text{RG}}(V)] = L_2(V) + \mathbb{E}_{s \sim \mu}[\text{var}(\hat{\mathcal{B}}_\pi V(s) | s)] \neq L_2(V)$ , where there is an extra term that involves the conditional variance of the empirical Bellman operator, which does not vanish unless the state transitions are deterministic. As a result, RG can converge to incorrect value functions (see also Section 6). With random transitions, correcting the bias requires double samples (i.e., at least two independent samples of  $(r, s')$  for the same  $(s, a)$  pair) to estimate the conditional variance.

More popular algorithms in the literature are instead based on fixed-point iterations, using  $\hat{\mathcal{B}}_\pi$  to construct a target value to update  $V_\theta(s_i)$ . An example is *fitted value iteration*, or FVI (Bertsekas & Tsitsiklis, 1996; Munos & Szepesvári, 2008), which includes as special cases the empirically successful DQN and variants, as well as serves a key component in many state-of-the-art actor-critic algorithms. In its basic form, FVI starts from an initial  $\theta_0$ , and iteratively updates the parameter by

$$\theta_{t+1} = \arg \min_{\theta \in \Theta} \left\{ \hat{L}_{\text{FVI}}^{(t+1)}(V_\theta) := \frac{1}{n} \sum_{i=1}^n \left( V_\theta(s_i) - \hat{\mathcal{B}}_\pi V_{\theta_t}(s_i) \right)^2 \right\}. \quad (2)$$

Different from RG, when gradient-based methods are applied to solve (2), the current parameter  $\theta_t$  is treated as a constant:  $\nabla_\theta \hat{L}_{\text{FVI}}^{(t+1)}(V_\theta) = \frac{1}{n} \sum_{i=1}^n (V_\theta(s_i) - \hat{\mathcal{B}}_\pi V_{\theta_t}(s_i)) \nabla_\theta V_\theta(s_i)$ . TD(0) (Sutton, 1988) may be viewed as a stochastic version of FVI, where a single sample (i.e.,

$n = 1$ ) is drawn randomly (either from a stream of transitions or from a replay buffer) to estimate the gradient of (2).

Being fixed-point iteration methods, FVI-style algorithms *do not* optimize any objective function, and their convergence is guaranteed only in rather restricted cases (e.g., Gordon, 1995; Tsitsiklis & Van Roy, 1997; Antos et al., 2008). Such divergent behavior is well-known and empirically observed (Baird, 1995; Boyan & Moore, 1995); see Section 6 for more numerical examples. It creates substantial difficulty in parameter tuning and model selection in practice.

### 3. Kernel Loss for Policy Evaluation

Much of the algorithmic challenge described earlier lies in the difficulty in estimating squared Bellman error from data. In this section, we address this difficulty by proposing a new loss function that is more amenable to statistical estimation from empirical data. Proofs are deferred to the appendix.

Our framework relies on an *integrally strictly positive definite (ISPD) kernel*  $K: \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ , which is a symmetric bi-variate function that satisfies  $\|f\|_K^2 := \int_{\mathcal{S}^2} K(s, \bar{s}) f(s) f(\bar{s}) ds d\bar{s} > 0$ , for any non-zero  $L_2$ -integrable function  $f$ . We call  $\|f\|_K$  the  $K$ -norm of  $f$ . Many commonly used kernels, such as Gaussian RBF kernel  $K(s, \bar{s}) = \exp(-\|s - \bar{s}\|_2^2/h)$  is ISPD. More discussion on ISPD kernels can be found in Stewart (1976) and Sriperumbudur et al. (2010).

#### 3.1. The New Loss Function

Recall that  $\mathcal{R}_\pi = \mathcal{B}_\pi V - V$  is the Bellman error operator. Our new loss function is defined by

$$\begin{aligned} L_K(V) &= \|\mathcal{R}_\pi V\|_{K, \mu}^2 \\ &:= \mathbb{E}_{s, \bar{s} \sim \mu} [K(s, \bar{s}) \cdot \mathcal{R}_\pi V(s) \cdot \mathcal{R}_\pi V(\bar{s})], \end{aligned} \quad (3)$$

where  $\mu$  is any positive density function on states  $s$ , and  $s, \bar{s} \sim \mu$  means  $s$  and  $\bar{s}$  are drawn i.i.d. from  $\mu$ . Here,  $\|\cdot\|_{K, \mu}$  is regarded as the  $K$ -norm under measure  $\mu$ . It is easy to show that  $\|f\|_{K, \mu} = \|f\mu\|_K$ . Note that  $\mu$  can be either the visitation distribution under policy  $\pi$  (the *on-policy* case), or some other distribution (the *off-policy* case). Our approach handles both cases in a unified way. The following theorem shows that the loss  $L_K$  is consistent:

**Theorem 3.1.** *Let  $K$  be an ISPD kernel and assume  $\mu(s) > 0, \forall s \in \mathcal{S}$ . Then,  $L_K(V) \geq 0$  for any  $V$ ; and  $L_K(V) = 0$  if and only if  $V = V^\pi$ . In other words,  $V^\pi = \arg \min_V L_K(V)$ .*

The next result relates the kernel loss to a “dual” kernel norm of the value function error,  $V - V^\pi$ .

**Theorem 3.2.** *Under the same assumptions as Theorem 3.1, we have  $L_K(V) = \|V - V^\pi\|_{K^*, \mu}^2$ , where  $\|\cdot\|_{K^*, \mu}$  is the  $K^*$ -norm under measure  $\mu$  with a “dual” kernel  $K^*(s, \bar{s})$ , defined by*

$$\begin{aligned} K^*(s', \bar{s}') &:= \mathbb{E}_{s, \bar{s} \sim d_{\pi, \mu}^*} \left[ K(s', \bar{s}') + \gamma^2 K(s, \bar{s}) \right. \\ &\quad \left. - \gamma (K(s', \bar{s}) + K(s, \bar{s}')) \mid s', \bar{s}' \right], \end{aligned}$$

and the expectation notation is shorthand for  $\mathbb{E}_{s \sim d_{\pi, \mu}^*} [f(s) \mid s'] = \int f(s) d_{\pi, \mu}^*(s \mid s') ds$ , with

$$d_{\pi, \mu}^*(s \mid s') := \sum_a \pi(a \mid s) P(s' \mid s, a) \mu(s) / \mu(s').$$

The norm involves a quantity,  $d_{\pi, \mu}^*(s \mid s')$ , which may be heuristically viewed as a “backward” conditional probability of state  $s$  conditioning on observing the next state  $s'$  (but note that  $d_{\pi, \mu}^*(s \mid s')$  is not normalized to sum to one unless  $\mu = d_\pi$ ).

**Empirical Estimation** The key advantage of the new loss  $L_K$  is that it can be easily estimated and optimized from observed transitions, without requiring double samples. Given a set of empirical data  $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{1 \leq i \leq n}$ , a way to estimate  $L_K$  is to use the so-called *V-statistics*,

$$\hat{L}_K(V_\theta) := \frac{1}{n^2} \sum_{1 \leq i, j \leq n} K(s_i, s_j) \cdot \hat{\mathcal{R}}_\pi V_\theta(s_i) \cdot \hat{\mathcal{R}}_\pi V_\theta(s_j). \quad (4)$$

Similarly, the gradient  $\nabla_\theta L_K(V_\theta) = 2\mathbb{E}_\mu [K(s, \bar{s}) \mathcal{R}_\pi V_\theta(s) \nabla_\theta (\mathcal{R}_\pi V_\theta(\bar{s}))]$  can be estimated by

$$\nabla_\theta \hat{L}_K(V_\theta) := \frac{2}{n^2} \sum_{1 \leq i, j \leq n} K(s_i, s_j) \cdot \hat{\mathcal{R}}_\pi V_\theta(s_i) \cdot \nabla_\theta \hat{\mathcal{R}}_\pi V_\theta(s_j).$$

**Remark** An alternative approach is to use the *U-statistics*, which removes the diagonal ( $i = j$ ) terms in the pairwise average in (4). Following standard statistical approximation theory (e.g., Serfling, 2009), both U/V-statistics provide *consistent* estimation of the expected quadratic quantity given the sample is weakly dependent and satisfies certain mixing condition (e.g., Denker & Keller, 1983; Beutner & Zähle, 2012); this often amounts to saying that  $\{s_i\}$  forms a Markov chain that converges to its stationary distribution  $\mu$  sufficiently fast. In the case of i.i.d. samples, it is known that U-statistics forms an unbiased estimate, but may have higher variance than the V-statistics. In our experiments, we observe that V-statistics works better than U-statistics.

**Remark** Another advantage of our kernel loss is that we have  $L_K(V) = 0$  iff  $V = V^\pi$ . Therefore, the magnitude of the empirical loss  $\hat{L}_K(V)$  reflects the closeness of  $V$  to the true value function  $V^\pi$ . In fact, by using methods from kernel-based hypothesis testing (e.g., Gretton et al.,

2012; Liu et al., 2016; Chwialkowski et al., 2016), one can design statistically calibrated methods to test if  $V = V^\pi$  has been achieved, which may be useful for designing efficient exploration strategies. In this work, we focus on estimating  $V^\pi$  and leave it as future work to test value function proximity.

### 3.2. Interpretations of the Kernel Loss

We now provide some insights into the new loss function, based on two interpretations.

**Eigenfunction Interpretation** Mercer’s theorem implies the following decomposition

$$K(s, \bar{s}) = \sum_{i=1}^{\infty} \lambda_i e_i(s) e_i(\bar{s}), \quad (5)$$

of any continuous positive definite kernel on a compact domain, where  $\{e_i\}_{i=1}^{\infty}$  is a countable set of orthonormal eigenfunctions w.r.t.  $\mu$  (i.e.,  $\mathbb{E}_{s \sim \mu}[e_i(s)e_j(s)] = \mathbf{1}\{i=j\}$ ), and  $\{\lambda_i\}_{i=1}^{\infty}$  are their eigenvalues. For ISPD kernels, all the eigenvalues must be positive:  $\forall i, \lambda_i > 0$ .

The following shows that  $L_K$  is a squared projected Bellman error in the space spanned by  $\{e_i\}_{i=1}^{\infty}$ .

**Proposition 3.3.** *If (5) holds, then  $L_K(V) = \sum_{i=1}^{\infty} \lambda_i (\mathbb{E}_{s \sim \mu} [\mathcal{R}_\pi V(s) \times e_i(s)])^2$ . Furthermore, if  $\{e_i\}$  is a complete orthonormal basis of  $L_2$ -space under measure  $\mu$ , then the  $L_2$  loss is  $L_2(V) = \sum_{i=1}^{\infty} (\mathbb{E}_{s \sim \mu} [\mathcal{R}_\pi V(s) \times e_i(s)])^2$ . Therefore,  $L_K(V) \leq \lambda_{\max} L_2(V)$ , where  $\lambda_{\max} := \max_i \{\lambda_i\}$ .*

This result shows that the eigenvalue  $\lambda_i$  controls the contribution of the projected Bellman error to the eigenfunction  $e_i$  in  $L_K$ . It may be tempting to have  $\lambda_i \equiv 1$ , in which  $L_K(V) = L_2(V)$ , but the Mercer expansion in (5) can diverge to infinity, resulting in an ill-defined kernel  $K(s, \bar{s})$ . To avoid this, the eigenvalues must decay to zero fast enough such that  $\sum_{i=1}^{\infty} \lambda_i < \infty$ . Therefore, the kernel loss  $L_K(V)$  can be viewed as prioritizing the projections to the eigenfunctions with larger eigenvalues. In typical kernels such as Gaussian RBF kernels, these dominant eigenfunctions are Fourier bases with low frequencies (and hence high smoothness), which may intuitively be more relevant than the higher frequency bases for practical purposes.

**RKHS Interpretation** The squared Bellman error has the following variational form:

$$L_2(V) = \max_f (\mathbb{E}_{s \sim \mu} [\mathcal{R}_\pi V(s) \times f(s)])^2 \quad \text{s.t.} \quad \mathbb{E}_{s \sim \mu} [(f(s))^2] \leq 1, \quad (6)$$

which involves finding a function  $f$  in the unit  $L_2$ -ball, whose inner product with  $\mathcal{R}_\pi V(s)$  is maximal. Our kernel

loss has a similar interpretation, with a different unit ball.

Any positive kernel  $K(s, \bar{s})$  is associated with a Reproducing Kernel Hilbert Space (RKHS)  $\mathcal{H}_K$ , which is the Hilbert space consisting of (the closure of) the linear span of  $K(\cdot, s)$ , for  $s \in \mathcal{S}$ , and satisfies the reproducing property,  $f(x) = \langle f, K(\cdot, x) \rangle_{\mathcal{H}_K}$ , for any  $f \in \mathcal{H}_K$ . RKHS has been widely used as a powerful tool in various machine learning and statistical problems; see [Berlinet & Thomas-Agnan \(2011\)](#); [Muandet et al. \(2017\)](#) for overviews.

**Proposition 3.4.** *Let  $\mathcal{H}_K$  be the RKHS of kernel  $K(s, \bar{s})$ , we have*

$$L_K(V) = \max_{f \in \mathcal{H}_K} \left\{ (\mathbb{E}_{s \sim \mu} [\mathcal{R}_\pi V(s) \times f(s)])^2 : \|f\|_{\mathcal{H}_K} \leq 1 \right\}. \quad (7)$$

Since RKHS is a subset of the  $L_2$  space that includes smooth functions, we can again see that  $L_K(V)$  emphasizes more the projections to smooth basis functions, matching the intuitive from [Theorem 3.3](#). It also draws a connection to the recent primal-dual reformulations of the Bellman equation ([Dai et al., 2017; 2018](#)), which formulate  $V^\pi$  as a saddle-point of the following minimax problem:

$$\min_V \max_f \mathbb{E}_{s \sim \mu} [2\mathcal{R}_\pi V(s) \times f(s) - f(s)^2], \quad (8)$$

This is equivalent to minimizing  $L_2(V)$  as (6), except that the  $L_2$  constraint is replaced by a quadratic penalty term. When only samples are available, the expectation in (8) is replaced by the empirical version. If the optimization domain of  $f$  is *unconstrained*, solving the empirical (8) reduces to the empirical  $L_2$  loss (1), which yields inconsistent estimation. Therefore, existing works propose to further constrain the optimization of  $f$  in (8) to either RKHS ([Dai et al., 2017](#)) or neural networks ([Dai et al., 2018](#)), and hence derive a minimax strategy for learning  $V$ . Unfortunately, this is substantially more expensive than our method due to the cost of updating another neural network  $f$  jointly; the minimax procedure may also make the training less stable and more difficult to converge in practice.

### 3.3. Connection to Temporal Difference (TD) Methods

We now instantiate our algorithm in the *tabular* and *linear* cases to gain further insights. Interestingly, we show that our loss coincides with previous work, and as a result leads to the same value function as several classic algorithms. Hence, the approach developed here may be considered as their strict extensions to the much more general nonlinear function approximation classes.

Again, let  $\mathcal{D}$  be a set of  $n$  transitions sampled from distribution  $\mu$ , and linear approximation be used:  $V_\theta(s) = \theta^\top \phi(s)$ , where  $\phi : \mathcal{S} \rightarrow \mathbb{R}^d$  is a feature function, and  $\theta \in \mathbb{R}^d$  is the parameter to be learned. The TD solution,  $\hat{\theta}_{\text{TD}}$ , for

either on- and off-policy cases, can be found by various algorithms (e.g., Sutton, 1988; Boyan, 1999; Sutton et al., 2009; Dann et al., 2014), and its theoretical properties have been extensively studied (e.g., Tsitsiklis & Van Roy, 1997; Lazaric et al., 2012).

**Corollary 3.5.** *When using a linear kernel of form  $k(s, \bar{s}) = \phi(s)^\top \phi(\bar{s})$ , minimizing the kernel objective (4) gives the TD solution  $\hat{\theta}_{\text{TD}}$ .*

**Remark** The result follows from the observation that our loss becomes the Norm of the Expected TD Update (NEU) in the linear case (Dann et al., 2014), whose minimizer coincides with  $\hat{\theta}_{\text{TD}}$ . Moreover, in finite-state MDPs, the corollary includes tabular TD as a special case, by using a one-hot vector (indicator basis) to represent states. In this case, the TD solution coincides with that of a model-based approach (Parr et al., 2008) known as *certainty equivalence* (Kumar & Varaiya, 1986). It follows that our algorithm includes certainty equivalence as a special case in finite-state problems.

## 4. Kernel Loss for Policy Optimization

There are different ways to extend our approach to policy optimization. One is to use the kernel loss (3) inside an existing algorithm, as an alternative to RG or TD to learn  $V^\pi(s)$ . For example, our loss fits naturally into an actor-critic algorithm, where we replace the critic update (often implemented by  $\text{TD}(\lambda)$  or its variant) with our method, and the actor updating part remains unchanged. Another, more general way is to design a kernelized loss for  $V(s)$  and policy  $\pi(a|s)$  jointly, so that the policy optimization can be solved using a single optimization procedure. Here, we take the first approach, leveraging our method to improve the critic update step in Trust-PCL (Nachum et al., 2018).

Trust-PCL is based on a temporal/path consistency condition resulting from policy smoothing (Nachum et al., 2017). We start with the smoothed Bellman operator, defined by

$$\mathcal{B}_\lambda V(s) = \max_{\pi(\cdot|s) \in \mathcal{P}_A} \mathbb{E}_\pi[R(s, a) + \gamma V(s') + \lambda H(\pi | s) | s], \quad (9)$$

where  $\mathcal{P}_A$  is the set of distributions over action space  $\mathcal{A}$ ; the conditional expectation  $\mathbb{E}_\pi[\cdot|s]$  denotes  $a \sim \pi(\cdot|s)$ , and  $\lambda > 0$  is a smoothing parameter;  $H$  is a state-dependent entropy term:  $H(\pi | s) := -\sum_{a \in \mathcal{A}} \pi(a|s) \log \pi(a|s)$ . Intuitively,  $\mathcal{B}_\lambda$  is a smoothed approximation of  $\mathcal{B}$ . It is known that  $\mathcal{B}_\lambda$  is a  $\gamma$ -contraction (Fox et al., 2016), so has a unique fixed point  $V_\lambda^*$ . Furthermore, with  $\lambda = 0$  we recover the standard Bellman operator, and  $\lambda$  smoothly controls  $\|V_\lambda^* - V^*\|_\infty$  (Dai et al., 2018).

The entropy regularization above implies the following path consistency condition. Let  $\pi_\lambda^*$  be an optimal policy in (9)

for  $\mathcal{B}_\lambda$ , which yields  $V_\lambda^*$ . Then,  $(V, \pi) = (V_\lambda^*, \pi_\lambda^*)$  uniquely solves

$$\forall (s, a) \in \mathcal{S} \times \mathcal{A}: \quad V(s) = R(s, a) + \gamma \mathbb{E}_{s'|s, a} [V(s')] - \lambda \log \pi(a|s).$$

This property inspires a natural extension of the kernel loss (3) to the controlled case:

$$L_K(V) = \mathbb{E}_{s, \bar{s} \sim \mu, a \sim \pi(\cdot|s), \bar{a} \sim \pi(\cdot|\bar{s})} [K([s, a], [\bar{s}, \bar{a}]) \cdot \mathcal{R}_{\pi, \lambda} V(s, a) \cdot \mathcal{R}_{\pi, \lambda} V(\bar{s}, \bar{a})],$$

where  $\mathcal{R}_{\pi, \lambda} V(s, a)$  is given by

$$\mathcal{R}_{\pi, \lambda} V(s, a) = R(s, a) + \gamma \mathbb{E}_{s'|s, a} [V(s')] - \lambda \log \pi(a|s) - V(s).$$

Given a set of transitions  $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{1 \leq i \leq n}$ , the objective can be estimated by

$$\hat{L}_K(V_\theta) = \frac{1}{n^2} \sum_{1 \leq i, j \leq n} [K([s_i, a_i], [s_j, a_j]) \hat{\mathcal{R}}_i \hat{\mathcal{R}}_j],$$

with  $\hat{\mathcal{R}}_i = r_i + \gamma V_\theta(s'_i) - \lambda \log \pi_\theta(a_i|s_i) - V_\theta(s_i)$ . The U-statistics version and the multi-step bootstrapping can be similarly obtained (Nachum et al., 2017).

## 5. Related Work

In this work, we studied value function learning, one of the most studied and fundamental problems in reinforcement learning. The dominant approach is based on fixed-point iterations (Bertsekas & Tsitsiklis, 1996; Szepesvári, 2010; Sutton & Barto, 2018), which can risk instability and even divergence when function approximation is used, as discussed in the introduction.

Our approach exemplifies more recent efforts that aim to improve stability of value function learning by reformulating it as an optimization problem. Our key innovation is the use of a kernel method to estimate the squared Bellman error, which is otherwise hard to estimate directly from samples, thus avoids the double-sample issue unaddressed by prior algorithms like residual gradient (Baird, 1995) and PCL (Nachum et al., 2017; 2018). As a result, our algorithm is *consistent*: it finds the true value function with enough data, using sufficiently expressive function approximation classes). Furthermore, the solution found by our algorithm minimizes the projected Bellman error, as in prior works when specialized to the same settings (Sutton et al., 2009; Maei et al., 2010; Liu et al., 2015; Macua et al., 2015). However, our algorithm is more general: it allows us to use nonlinear value function classes and can be naturally implemented for policy optimization. Compared to nonlinear GTD2/TDC (Maei et al., 2009), our method is simpler (without having to do a local linear expansion) and empirically more effective (as demonstrated in the next section).

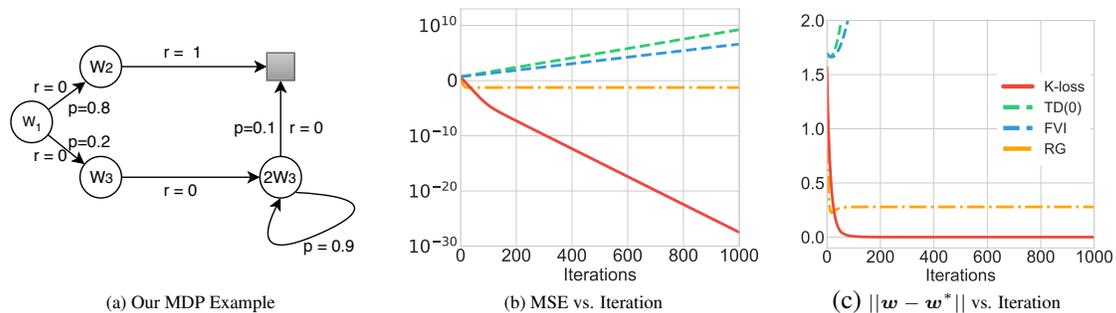


Figure 1. Modified example of (Tsitsiklis &amp; Van Roy, 1997).

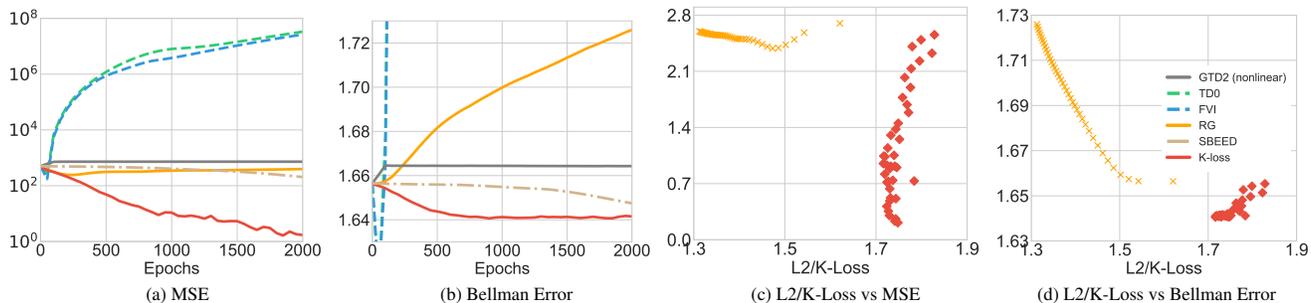


Figure 2. Results on Puddle World.

As discussed in Section 3, our method is related to the recently proposed SBEED algorithm (Dai et al., 2018) which shares many advantages with this work. However, SBEED requires solving a minimax problem that can be rather challenging in practice. In contrast, our algorithm only needs to solve a minimization problem, for which a wide range of powerful methods exist (e.g., Bertsekas, 2016). Note that there exist other saddle-point formulations for RL, which so far have focused on finite-state MDPs or linear value function approximation (Chen et al., 2018; Wang, 2017).

Finally, the kernel method has been widely used in machine learning (e.g., Schölkopf & Smola, 2001; Muandet et al., 2017). In RL, authors have used kernels either to model transition probabilities (Ormoneit & Sen, 2002) or to represent the value function (e.g., Xu et al., 2005; 2007; Taylor & Parr, 2009). These works differ significantly from our method in that they use kernels to specify the function class of value functions or transition models. In contrast, we leverage kernels for *designing proper loss functions* to address the double-sampling problem, while putting no constraints on which approximation classes to represent the value function. Our approach is thus expected to be more flexible and scalable in practice, especially when combined with neural networks.

## 6. Experiments

We compare our method (labelled ‘‘K-loss’’ in all experiments) with several representative baselines in both classic examples and popular benchmark problems, for both policy

evaluation and optimization.

### 6.1. Modified Example of Tsitsiklis & Van Roy

Fig. 1 (a) shows a modified problem of the classic example by Tsitsiklis & Van Roy (1997), by making transitions stochastic. It consists of 5 states, including 4 nonterminal (circles) and 1 terminal states (square), and 1 action. The arrows represent transitions between states. The value function estimate is linear in the weight  $\mathbf{w} = [w_1, w_2, w_3]$ : for example, the leftmost and bottom-right states’ values are  $w_1$  and  $2w_3$ , respectively. Furthermore, we set  $\gamma = 1$ , so  $V(s)$  is exact with the optimal weight  $\mathbf{w}^* = [0.8, 1.0, 0]$ . In the experiment, we randomly collect 2 000 transition tuples for training. We use a linear kernel in our method, so that it will find the TD solution (Corollary 3.5).

Fig. 1 (b&c) show the learning curves of mean squared error ( $\|V - V^*\|^2$ ) and weight error ( $\|\mathbf{w} - \mathbf{w}^*\|$ ) of different algorithms over iterations. Results are consistent with theory: our method converges to the true weight  $\mathbf{w}^*$ , while both FVI and TD(0) diverge, and RG converges to a wrong solution.

### 6.2. Policy Evaluation with Neural Networks

While popular in recent RL literature, neural networks are known to be unstable for a long time. Here, we revisit the classic divergence example of Puddle World (Boyan & Moore, 1995), and demonstrate the stability of our method. Experimental details are found in Appendix B.1.

Fig. 2 summarizes the result using a neural network as value

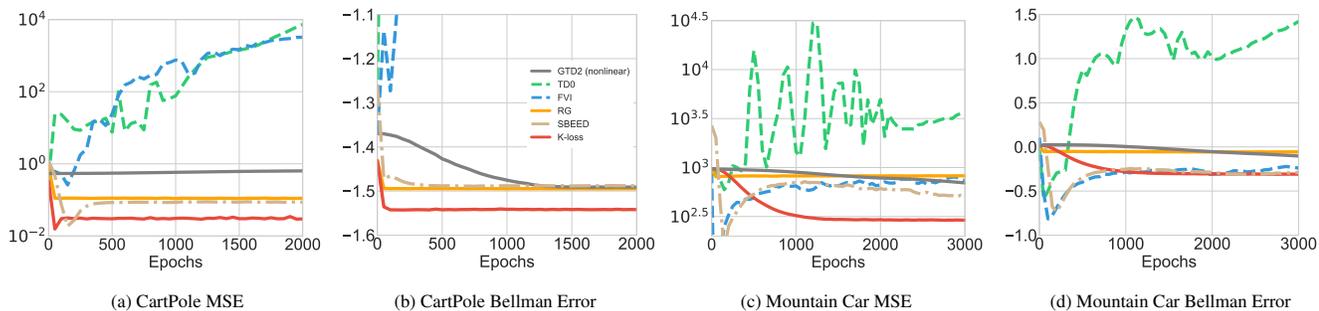


Figure 3. Policy evaluation results on CartPole and Mountain Car.

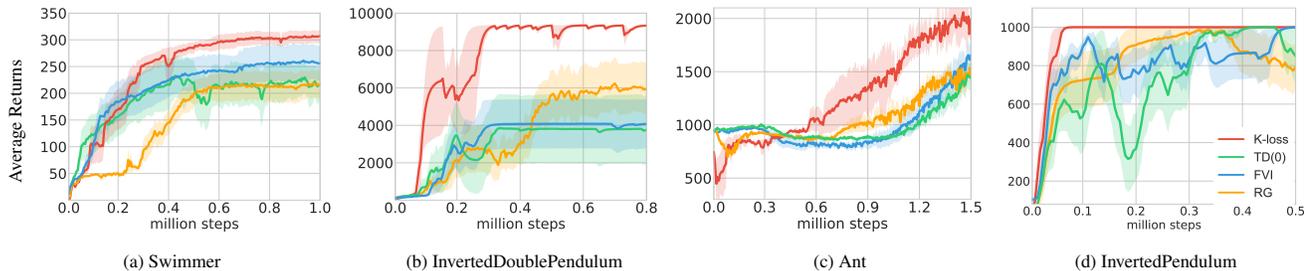


Figure 4. Results of various variants of Trust PCL on Mujoco Benchmark.

function for two metrics:  $\|V - V^*\|_2^2$  and  $\|BV - V\|_2^2$ , both evaluated on the training transitions. First, as shown in (a-b), our method works well while residual gradient converged to inferior solutions. In contrast, FVI and TD(0) exhibit unstable/oscilating behavior, and can even diverge, which is consistent with past findings (Boyan & Moore, 1995). In addition, non-linear GTD2 (Maei et al., 2009) and SBEED (Dai et al., 2017; 2018), which do not find a better solution than our method in terms of MSE.

Second, Fig. 2(c&d) show the correlation between MSE, empirical Bellman error of the value function estimation and an algorithm’s training objective respectively. Our kernel loss appears to be a good proxy for learning the value function, for both MSE and Bellman error. In contrast, the L2 loss (used by residual gradient) does not correlate well, which also explains why residual gradient has been observed not to work well empirically.

Fig. 3 shows more results on value function learning on CartPole and Mountain Car, which again demonstrate that our method performs better than other methods in general.

### 6.3. Policy Optimization

To demonstrate the use of our method in policy optimization, we combine it with Trust-PCL, and compare with variants of Trust-PCL combined with FVI, TD0 and RG. To fairly evaluate the performance of all these four methods, we use Trust-PCL (Nachum et al., 2018) framework and the public code for our experiments. We only modify the training of  $V_\theta(s)$  for each of the method and keep rest same

as original release. Experimental details can be found in Appendix B.2.1.

We evaluate the performance of these four methods on Mujoco benchmark and report the best performance of these four methods in Figure 4 (averaged on five different random seeds). K-loss consistently outperforms all the other methods, learning better policy with fewer data. Note that we only modify the update of value functions inside Trust PCL, which can be implemented relatively easily. We expect that we can improve many other algorithms in similar ways by improving the value function using our kernel loss.

## 7. Conclusion

This paper studies the fundamental problem of solving Bellman equations with parametric value functions. A novel kernel loss is proposed, which is easy to be estimated and optimized using sampled transitions. Empirical results show that, compared to prior algorithms, our method is convergent, produces more accurate value functions, and can be easily adapted for policy optimization. These promising results open the door to many interesting directions for future work, including finite-sample analysis, adaptation to online RL, and uncertainty estimation for exploration.

## References

Antos, A., Szepesvári, C., and Munos, R. Learning near-optimal policies with Bellman-residual minimizing based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008.

- Baird, L. C. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 30–37, 1995.
- Berlinet, A. and Thomas-Agnan, C. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.
- Bertsekas, D. P. *Nonlinear Programming*. Athena Scientific, 3rd edition, 2016.
- Bertsekas, D. P. and Tsitsiklis, J. N. *Neuro-Dynamic Programming*. Athena Scientific, September 1996.
- Beutner, E. and Zähle, H. Deriving the asymptotic distribution of U- and V-statistics of dependent data using weighted empirical processes. *Bernoulli*, pp. 803–822, 2012.
- Boyan, J. A. Least-squares temporal difference learning. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 49–56, 1999.
- Boyan, J. A. and Moore, A. W. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems 7*, pp. 369–376, 1995.
- Chen, Y., Li, L., and Wang, M. Scalable bilinear  $\pi$ -learning using state and action features. In *Proceedings of the Thirty-Fifth International Conference on Machine Learning*, pp. 833–842, 2018.
- Chwialkowski, K., Strathmann, H., and Gretton, A. A kernel test of goodness of fit. *JMLR: Workshop and Conference Proceedings*, 2016.
- Dai, B., He, N., Pan, Y., Boots, B., and Song, L. Learning from conditional distributions via dual embeddings. In *Artificial Intelligence and Statistics*, pp. 1458–1467, 2017.
- Dai, B., Shaw, A., Li, L., Xiao, L., He, N., Liu, Z., Chen, J., and Song, L. SBEED: Convergent reinforcement learning with nonlinear function approximation. In *Proceedings of the Thirty-Fifth International Conference on Machine Learning*, pp. 1133–1142, 2018.
- Dann, C., Neumann, G., and Peters, J. Policy evaluation with temporal differences: A survey and comparison. *Journal of Machine Learning Research*, 15(1):809–883, 2014.
- Denker, M. and Keller, G. On U-statistics and v. mise’ statistics for weakly dependent processes. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 64(4): 505–522, 1983.
- Farahmand, A. M., Ghavamzadeh, M., Szepesvári, C., and Mannor, S. Regularized policy iteration. In *Advances in Neural Information Processing Systems 21*, pp. 441–448, 2008.
- Fox, R., Pakman, A., and Tishby, N. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, 2016.
- Gordon, G. J. Stable function approximation in dynamic programming. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 261–268, 1995.
- Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.
- Gu, S., Lillicrap, T. P., Sutskever, I., and Levine, S. Continuous deep Q-learning with model-based acceleration. In *Proceedings of the Thirty-third International Conference on Machine Learning*, pp. 2829–2838, 2016.
- Kumar, P. and Varaiya, P. *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice Hall, 1986.
- Lazaric, A., Ghavamzadeh, M., and Munos, R. Finite-sample analysis of least-squares policy iteration. *Journal of Machine Learning Research*, pp. 3041–3074, 2012.
- Liu, B., Liu, J., Ghavamzadeh, M., Mahadevan, S., and Petrik, M. Finite-sample analysis of proximal gradient TD algorithms. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pp. 504–513, 2015.
- Liu, Q., Lee, J., and Jordan, M. A kernelized Stein discrepancy for goodness-of-fit tests. In *International Conference on Machine Learning*, pp. 276–284, 2016.
- Macua, S. V., Chen, J., Zazo, S., and Sayed, A. H. Distributed policy evaluation under multiple behavior strategies. *IEEE Transactions on Automatic Control*, 60(5): 1260–1274, 2015.
- Maei, H. R. *Gradient Temporal-Difference Learning Algorithms*. PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 2011.
- Maei, H. R., Szepesvári, C., Bhatnagar, S., Precup, D., Silver, D., and Sutton, R. S. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems 22*, pp. 1204–1212, 2009.
- Maei, H. R., Szepesvári, C., Bhatnagar, S., and Sutton, R. S. Toward off-policy learning control with function approximation. In *Proceedings of the Twenty-Seventh*

- International Conference on Machine Learning*, pp. 719–726, 2010.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.
- Mnih, V., Adrià, Badia, P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *Proceedings of the Thirty-third International Conference on Machine Learning*, pp. 1928–1937, 2016.
- Muandet, K., Fukumizu, K., Sriperumbudur, B., Schölkopf, B., et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017.
- Munos, R. and Szepesvári, C. Finite-time bounds for sampling-based fitted value iteration. *Journal of Machine Learning Research*, 9:815–857, 2008.
- Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems 30*, pp. 2772–2782, 2017.
- Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Trust-PCL: An off-policy trust region method for continuous control. In *International Conference on Learning Representations*, 2018.
- Ormoneit, D. and Sen, Š. Kernel-based reinforcement learning. *Machine Learning*, 49:161–178, 2002.
- Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., and Littman, M. L. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning*, pp. 752–759, 2008.
- Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, New York, 1994.
- Schölkopf, B. and Smola, A. J. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2001.
- Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations*, 2016.
- Serfling, R. J. *Approximation theorems of mathematical statistics*, volume 162. John Wiley & Sons, 2009.
- Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B., and Lanckriet, G. R. Hilbert space embeddings and metrics on probability measures. *Journal of Machine Learning Research*, 11(Apr):1517–1561, 2010.
- Stewart, J. Positive definite functions and generalizations, an historical survey. *The Rocky Mountain Journal of Mathematics*, 6(3):409–434, 1976.
- Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.
- Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press, 2nd edition, 2018.
- Sutton, R. S., Maei, H., Precup, D., Bhatnagar, S., Szepesvári, C., and Wiewiora, E. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning*, pp. 993–1000, 2009.
- Szepesvári, C. *Algorithms for Reinforcement Learning*. Morgan & Claypool, 2010.
- Taylor, G. and Parr, R. Kernelized value function approximation for reinforcement learning. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning*, pp. 1017–1024, 2009.
- Tsitsiklis, J. N. and Van Roy, B. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42:674–690, 1997.
- Wang, M. Primal-dual  $\pi$  learning: Sample complexity and sublinear run time for ergodic Markov decision problems, 2017. CoRR abs/1710.06100.
- Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., and de Freitas, N. Dueling network architectures for deep reinforcement learning. In *Proceedings of the Third International Conference on Machine Learning*, pp. 1995–2003, 2016.
- Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., and Ba, J. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. In *Advances in Neural Information Processing Systems 30*, pp. 5285–5294, 2017.
- Xu, X., Xie, T., Hu, D., and Lu, X. Kernel least-squares temporal difference learning. *International Journal of Information and Technology*, 11(9):54–63, 2005.

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

## Appendix

### A. Proofs for Section 3

#### A.1. Proof of Theorem 3.1

The assertion that  $L_K(V) \geq 0$  for all  $V$  is immediate from definition. For the second part, we have

$$\begin{aligned}
 L_K(V) = 0 &\iff \|\mathcal{R}_\pi V\|_{K,\mu} = 0 \quad (\text{since } K \text{ is an ISPD kernel}) \\
 &\iff \|\mathcal{R}_\pi V \cdot \mu\|_K = 0 \\
 &\iff \forall s \in \mathcal{S} : \mathcal{R}_\pi V(s)\mu(s) = 0 \\
 &\iff \forall s \in \mathcal{S} : \mathcal{R}_\pi V(s) = 0 \\
 &\iff V = V^\pi.
 \end{aligned}$$

#### A.2. Proof of Theorem 3.2

Define  $\delta = V - V^\pi$  to be the value function error. Furthermore, let  $\mathcal{I}$  be the identity operator ( $\mathcal{I}V = V$ ), and

$$\mathcal{P}_\pi V(s) := \mathbb{E}_{a \sim \pi(\cdot|s), s' \sim P(\cdot|s,a)}[\gamma V(s') \mid s]$$

the state-transition part of Bellman operator without the local reward term  $R(s, a)$ .

Note that  $\mathcal{R}_\pi V^\pi = \mathcal{B}_\pi V^\pi - V^\pi = 0$  by the Bellman equation, so

$$\mathcal{R}_\pi V = \mathcal{R}_\pi V - \mathcal{R}_\pi V^\pi = (\mathcal{P}_\pi V - V) - (\mathcal{P}_\pi V^\pi - V^\pi) = (\mathcal{P}_\pi - \mathcal{I})(V - V^\pi) = (\mathcal{P}_\pi - \mathcal{I})\delta.$$

Therefore,

$$\begin{aligned}
 L_K(V) &= \mathbb{E}_\mu[\mathcal{R}_\pi V(s) \cdot \mathcal{R}_\pi V(\bar{s}) \cdot K(s, \bar{s})] \\
 &= \mathbb{E}_\mu[(\mathcal{I} - \mathcal{P}_\pi)\delta(s) \cdot (\mathcal{I} - \mathcal{P}_\pi)\delta(\bar{s}) \cdot K(s, \bar{s})] \\
 &= \mathbb{E}_{(s,s'),(\bar{s},\bar{s}') \sim d_{\pi,\mu}}[(\delta(s) - \gamma\delta(s')) \cdot (\delta(\bar{s}) - \gamma\delta(\bar{s}')) \cdot K(s, \bar{s})],
 \end{aligned}$$

where  $\mathbb{E}_{d_{\pi,\mu}}[\cdot]$  denotes the expectation under the joint distribution

$$d_{\pi,\mu}(s, s') := \mu(s) \sum_{a \in \mathcal{A}} \pi(a|s) P(s'|s, a).$$

Expanding the quadratic form above, we have

$$\begin{aligned}
 L_K(V) &= \mathbb{E}_{d_{\pi,\mu}}[(\delta(s)K(s, \bar{s})\delta(\bar{s}) - \gamma\delta(s')\delta(\bar{s})K(s, \bar{s}) - \gamma\delta(\bar{s}')\delta(s)K(s, \bar{s}) + \gamma^2\delta(s')\delta(\bar{s}')K(s, \bar{s})] \\
 &= \mathbb{E}_\mu[\delta(s')K^*(s', \bar{s}')\delta(\bar{s}')],
 \end{aligned}$$

where  $K^*(s', \bar{s}')$  is as defined in the theorem statement:

$$K^*(s', \bar{s}') = \mathbb{E}_{d_{\pi,\mu}^*} [K(s', \bar{s}') - \gamma(K(s', \bar{s}) + K(s, \bar{s}')) + \gamma^2 K(s, \bar{s}) \mid (s', \bar{s}')] ]$$

with the expectation w.r.t. the following ‘‘backward’’ conditional probability

$$d_{\pi,\mu}^*(s \mid s') := \frac{\sum_{a \in \mathcal{A}} \pi(a|s) P(s'|s, a) \mu(s)}{\mu(s')},$$

which can be heuristically viewed as the distribution of state  $s$  conditioning on observing its next state  $s'$  when following  $d_{\pi,\mu}(s, s')$ .

### A.3. Proof of Proposition 3.3

Using the eigen-decomposition (5), we have

$$\begin{aligned} L_K(V) &= \mathbb{E}_\mu[\mathcal{R}_\pi V(s)K(s, \bar{s})\mathcal{R}_\pi V(\bar{s})] \\ &= \mathbb{E}_\mu[\mathcal{R}_\pi V(s) \sum_{i=1}^{\infty} \lambda_i e_i(s) e_i(\bar{s}) \mathcal{R}_\pi V(\bar{s})] \\ &= \sum_{i=1}^{\infty} \lambda_i (\mathbb{E}_\mu[\mathcal{R}_\pi V(s) e_i(s)])^2. \end{aligned}$$

The decomposition of  $L_2(V)$  follows directly from Parseval's identity.

### A.4. Proof of Proposition 3.4

The reproducing property of RKHS implies  $f(s) = \langle f, K(s, \cdot) \rangle_{\mathcal{H}_K}$  for any  $f \in \mathcal{H}_K$ . Therefore,

$$\begin{aligned} \mathbb{E}_\mu[\mathcal{R}_\pi V(s) f(s)] &= \mathbb{E}_\mu[\mathcal{R}_\pi V(s) \langle f, K(s, \cdot) \rangle_{\mathcal{H}_K}] \\ &= \langle f, \mathbb{E}_\mu[\mathcal{R}_\pi V(s) K(s, \cdot)] \rangle_{\mathcal{H}_K} \\ &= \langle f, f^* \rangle_{\mathcal{H}_K}. \end{aligned}$$

where we have defined  $f^*(\cdot) := \mathbb{E}_\mu[\mathcal{R}_\pi V(s) K(s, \cdot)]$ . Maximizing  $\langle f, f^* \rangle$  subject to  $\|f\|_{\mathcal{H}_K} := \sqrt{\langle f, f \rangle_{\mathcal{H}_K}} \leq 1$  yields that  $f = f^* / \|f^*\|_{\mathcal{H}_K}$ . Therefore,

$$\max_{f \in \mathcal{H}_K: \|f\|_{\mathcal{H}_K} \leq 1} (\mathbb{E}_s[\mathcal{R}_\pi V(s) f(s)])^2 = \left( \left\langle \frac{f^*}{\|f^*\|_{\mathcal{H}_K}}, f^* \right\rangle_{\mathcal{H}_K} \right)^2 = \|f^*\|_{\mathcal{H}_K}^2.$$

Further, we can show that

$$\begin{aligned} \|f^*\|_{\mathcal{H}_K}^2 &= \langle f^*, f^* \rangle_{\mathcal{H}_K} \\ &= \langle \mathbb{E}_\mu[\mathcal{R}_\pi V(s) K(s, \cdot)], \mathbb{E}_\mu[\mathcal{R}_\pi V(\bar{s}) K(\bar{s}, \cdot)] \rangle_{\mathcal{H}_K} \\ &= \mathbb{E}_\mu[\mathcal{R}_\pi V(s) K(s, \bar{s}) \mathcal{R}_\pi V(\bar{s})], \end{aligned}$$

where the last step follows from the reproducing property,  $K(s, \bar{s}) = \langle K(s, \cdot), K(\bar{s}, \cdot) \rangle_{\mathcal{H}_K}$ . This completes the proof, by definition of  $L_K(V)$ .

### A.5. Proof of Corollary 3.5

Under the conditions of the corollary, the kernel loss becomes the Norm of the Expected TD Update (NEU), whose minimizer coincides with the TD solution (Dann et al., 2014). For completeness, we provide a self-contained proof.

Since we are estimating the value function of a fixed policy, we ignore the actions, and the set of transitions is  $\mathcal{D} = \{(s_i, r_i, s'_i)\}_{1 \leq i \leq n}$ . Define the following vector/matrices:

$$\begin{aligned} r &= [r_1; r_2, \dots; r_n] \in \mathbb{R}^{n \times 1}, \\ X &= [\phi(s_1); \phi(s_2); \dots; \phi(s_n)] \in \mathbb{R}^{n \times d}, \\ X' &= [\phi(s'_1); \phi(s'_2); \dots; \phi(s'_n)] \in \mathbb{R}^{n \times d}, \end{aligned}$$

and  $Z = X - \gamma X'$ , where  $d$  is the feature dimension. Then, the TD solution is given by

$$\hat{\theta}_{\text{TD}} = (X^T Z)^{-1} X^T r.$$

Note that the above includes both the on-policy case as well as the off-policy case as in many previous algorithms with linear value function approximation (Dann et al., 2014), where the difference is in whether  $s_i$  is sampled from the state occupation distribution of the target policy or not.

Define  $\delta \in \mathbb{R}^{n \times 1}$  to be the TD error vector; that is,  $\delta = r - Z\theta$ , where

$$\delta_i = r_i + \gamma V(s'_i) - V(s_i) = r_i + \theta^\top (\gamma \phi(s'_i) - \phi(s_i)).$$

With a linear kernel, our objective function becomes:

$$\ell(\theta) = \frac{1}{n^2} \sum_{i,j} \delta_i K(s_i, s_j) \delta_j = \frac{1}{n^2} \delta^\top X X^\top \delta = \frac{1}{n^2} (r - Z\theta)^\top X X^\top (r - Z\theta).$$

Its gradient is given by

$$\begin{aligned} \nabla \ell &= \nabla \left( \frac{1}{n^2} (\theta^\top Z^\top X X^\top Z \theta - 2r^\top X X^\top Z \theta + \text{constant}(\theta)) \right) \\ &= \frac{2}{n^2} (Z^\top X X^\top Z \theta - Z^\top X X^\top r). \end{aligned}$$

Letting  $\nabla \ell = 0$  gives the solution obtained by minimizing our kernel loss:<sup>1</sup>

$$\hat{\theta}_{\text{KBE}} = (Z^\top X X^\top Z)^{-1} Z^\top X X^\top r.$$

Therefore,

$$\begin{aligned} \hat{\theta}_{\text{KBE}} - \hat{\theta}_{\text{TD}} &= ((Z^\top X X^\top Z)^{-1} Z^\top X - (X^\top Z)^{-1}) X^\top r \\ &= ((Z^\top X X^\top Z)^{-1} Z^\top X (X^\top Z) - I) (X^\top Z)^{-1} X^\top r \\ &= (I - I) (X^\top Z)^{-1} X^\top r = 0. \end{aligned}$$

## B. Experiment Details

### B.1. Policy Evaluation

We compare our method with representative policy evaluation methods including TD(0), FVI, RG, nonlinear GTD2 (Maei et al., 2009) and SBEED (Dai et al., 2017; 2018) on three different stochastic environments: Puddle World, CartPole and Mountain Car. Followings are the detail of the policy evaluation experiments.

**Network Structure** We parameterize the value function  $V_\theta(s)$  using a fully connected neural network with one hidden layer of 80 units, using `relu` as activation function. For test function  $f(s)$  in SBEED, we use a small neural network with 10 hidden units and `relu` as activation function.

**Data Collection** For each environment, we randomly collect 5000 independent transition tuples with states uniformly drawn from state space using a policy  $\pi$  learned by policy optimization, for which we want to learn the value function  $V^\pi(s)$ .

**Estimating the true value function  $V^\pi(s)$**  To evaluate and compare all methods, we approximate the true value function by finely discretizing the state space and then applying tabular value iteration on the discretized MDP. Specifically, we discretize the state space into  $25 \times 25$  grid for Puddle World,  $20 \times 25$  discrete states for CartPole, and  $30 \times 25$  discrete states for Mountain Car.

**Training Details** For each environment and each policy evaluation method, we train the value function  $V_\theta(s)$  on the collected 5000 transition tuples for 2000 epochs (3000 for Mountain Car), with a batch size  $n = 150$  in each epoch using Adam optimizer. We search the learning rate in  $\{0.003, 0.001, 0.0003\}$  for all methods and report the best result averaging over 10 trials using different random seeds. For our method, we use a Gaussian RBF kernel  $K(s_i, s_j) = \exp(-\|s_i - s_j\|_2^2 / h^2)$  and take the bandwidth to be  $h = 0.5$ . For FVI, we update the target network at the end of each epoch training. For SBEED, we perform 10 times gradient ascent updates on the test function  $f(s)$  and 1 gradient descent update on  $V_\theta(s)$  at each iteration. We fix the discount factor to  $\gamma = 0.98$  for all environments and policy evaluation methods.

<sup>1</sup>For simplicity, assume all involved matrices of size  $d \times d$  are non-singular, as is typical in analyzing TD algorithms. Without this assumption, we may either add  $L_2$ -regularization to  $X X^\top$  (Farahmand et al., 2008), for which the same equivalence between TD and ours can be proved, or show that the solutions lie in an affine space in  $\mathbb{R}^d$  but the corresponding value functions are identical.

## B.2. Policy Optimization

In this section we describe in detail the experimental setup for policy optimization regarding implementation and hyperparameter search. The code of Trust-PCL is available at [github](https://github.com)<sup>2</sup>. Algorithm 1 describes details in pseudocode, where the main change compared to Trust-PCL is highlighted. Note that as in previous work, we use the  $d$ -step version of Bellman operator, an immediate extension to the  $d = 1$  case described in the main text.

### B.2.1. NETWORK ARCHITECTURES

We use fully-connected feed-forward neural network to represent both policy and value network. The policy  $\pi_\theta$  is represented by a neural network with  $64 \times 64$  hidden layers with `tanh` activations. At each time step  $t$ , the next action  $a_t$  is sampled randomly from a Gaussian distribution  $\mathcal{N}(\mu_\theta(s_t), \sigma_\theta)$ . The value network  $V_\theta(s)$  is represented by a neural network with  $64 \times 64$  hidden layers with `tanh` activations. At each time step  $t$ , the network is given the observation  $s_t$  and it produces a single scalar output value. All methods share the same policy and value network architectures.

### B.2.2. TRAINING DETAILS

We average over the best 5 of 6 randomly seeded training runs and evaluate each method using the mean  $\mu_\theta(s)$  of the diagonal Gaussian policy  $\pi_\theta$ . Since Trust-PCL is off-policy, we collect experience and train on batches of experience sampled from the replay buffer. At each training iteration, we will first sample  $T = 10$  timestep samples and add them to the replay buffer, then both the policy and value parameters are updated in a single gradient step using the Adam optimizer with a proper learning rate searched, using a minibatch randomly sampled from replay buffer. For Trust-PCL using FVI updating  $V_\theta(s)$ , which requires a target network to estimate the final state for each path, we use an exponentially moving average, with a smoothing constant  $\tau = 0.99$ , to update the target value network weights as common in the prior work (Mnih et al., 2015). For Trust-PCL using TD(0), we will directly use current value network  $V_\theta(s)$  to estimate the final states except we do not perform gradient update for the final states. For Trust-PCL using RG and K-loss, which has an objective loss, we will directly perform gradient descent to optimize both policy and value parameters.

### B.2.3. HYPERPARAMETER SEARCH

We follow the same hyperparameter search procedure in Nachum et al. (2018) for FVI, TD(0) and RG based Trust-PCL<sup>3</sup>. We search the maximum divergence  $\epsilon$  between  $\pi_\theta$  and  $\pi_{\hat{\theta}}$  among  $\in \{0.001, 0.0005, 0.002\}$ , and parameter learning rate in  $\{0.001, 0.0003, 0.0001\}$ , and the rollout length  $d \in \{1, 5, 10\}$ . We also searched with the entropy coefficient  $\lambda$ , either keeping it at a constant 0 (thus, no exploration) or decaying it from 0.1 to 0.0 by a smoothed exponential rate of 0.1 every 2500 training iterations. For each hyper-parameter setting, we average best 5 of 6 seeds and report the best performance for these methods.

For our proposed K-loss, we also search the maximum divergence  $\epsilon$  but keep the learning rate as 0.001. Additionally, for K-loss we use a Gaussian RBF kernel  $K([s_i, a_i], [s_j, a_j]) = \exp(-(\|s_i - s_j\|_2^2 + \|a_i - a_j\|_2^2)/h)$ , and take the bandwidth to be  $h = (\alpha \times \text{med})^2$ , where we search  $\alpha \in \{0.1, 0.01, (1/\sqrt{\log B})\}$ , and  $B = 64$  is the gradient update batch size. We fix the discount to  $\gamma = 0.995$  for all environments and batch size  $B = 64$  for each training iteration.

<sup>2</sup>[https://github.com/tensorflow/models/tree/master/research/pcl\\_rl](https://github.com/tensorflow/models/tree/master/research/pcl_rl)

<sup>3</sup>Please check readme file in [https://github.com/tensorflow/models/tree/master/research/pcl\\_rl](https://github.com/tensorflow/models/tree/master/research/pcl_rl)

---

**Algorithm 1** K-Loss for PCL

---

**Input:** rollout step  $d$ , batch size  $B$ , coefficient  $\lambda$ ,  $\tau$ .

Initialize  $V_\theta(s)$ ,  $\pi_\phi(a|s)$ , and empty replay buffer  $RB(\beta)$ . Set  $\tilde{\phi} = \phi$ .

**repeat**

  // Collecting Samples

  Sample  $P$  steps  $s_{t:t+P} \sim \pi_\phi$  on ENV.

  Insert  $s_{t:t+P}$  to  $RB(\beta)$ .

  // Train

  Sample batch  $\{s_{t:t+d}^{(k)}, a_{t:t+d}^{(k)}, r_{t:t+d}^{(k)}\}_{k=1}^B$  from  $RB(\beta)$  to contain a total of  $Q$  transitions ( $B \approx Q/d$ ).

$$\Delta\theta = \nabla_\theta \frac{1}{B^2} \sum_{1 \leq i, j \leq B} [K([s_i, a_i], [s_j, a_j]) \hat{\mathcal{R}}_i \hat{\mathcal{R}}_j],$$

$$\Delta\phi = -\frac{1}{B} \sum_{1 \leq i \leq B} [\hat{\mathcal{R}}_i \sum_{t=0}^{d-1} \nabla_\phi \log \pi_\phi(a_{t+i}|s_{t+i})], \text{ where}$$

$$\hat{\mathcal{R}}_i = -V_\theta(s_i) + \gamma^d V_\theta(s_{i+d}) + \sum_{t=0}^{d-1} \gamma^t (r_{i+t} - (\lambda + \tau) \log \pi_\phi(a_{t+i}|s_{t+i}) + \tau \log \pi_{\tilde{\phi}}(a_{t+i}|s_{t+i})).$$

Update  $\theta$  and  $\phi$  using ADAM with  $\Delta\theta$ ,  $\Delta\phi$ .

  // Update auxiliary variables

  Update  $\tilde{\phi} = \alpha \tilde{\phi} + (1 - \alpha)\phi$ .

**until** Convergence

---

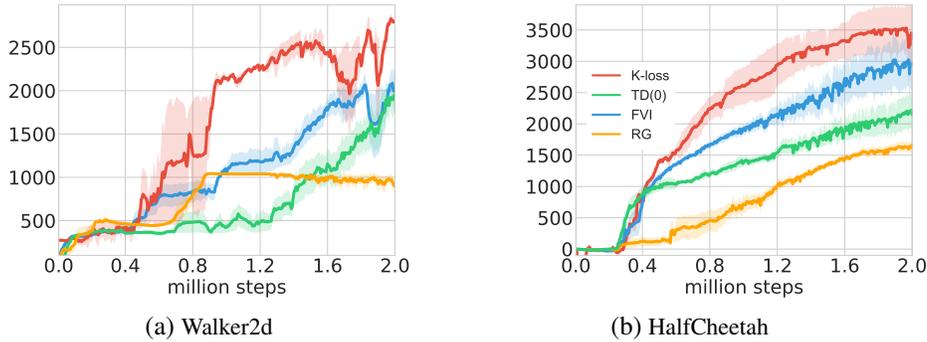


Figure 5. More results of various variants of Trust PCL on Mujoco Benchmark (on top of Figure 4).