Noname manuscript No.
(will be inserted by the editor)

Regularized Feature Selection in Reinforcement Learning

Dean S. Wookey · George D. Konidaris

Received: date / Accepted: date

Abstract We introduce *feature regularization* during feature selection for value function approximation. Feature regularization introduces a prior into the *selection* process, improving function approximation accuracy and reducing overfitting. We show that the smoothness prior is effective in the incremental feature selection setting and present closed-form smoothness regularizers for the Fourier and RBF bases. We present two methods for feature regularization which extend the temporal difference orthogonal matching pursuit (OMP-TD) algorithm and demonstrate the effectiveness of the smoothness prior; smooth Tikhonov OMP-TD and smoothness scaled OMP-TD. We compare these methods against OMP-TD, regularized OMP-TD and least squares TD with random projections, across six benchmark domains using two different types of basis functions.

1 Introduction

Representing value functions in large or continuous state spaces requires function approximation. Linear function approximation is one common approach, where the value function is represented as a weighted sum of basis functions. It is often beneficial to keep the number of basis functions low (Kolter and Ng, 2009; Painter-Wakefield and Parr, 2012); however, the right set is usually domain-specific.

Recent work in *feature selection* (Kolter and Ng, 2009; Johns et al, 2010; Painter-Wakefield and Parr, 2012; Petrik et al, 2010; Ghavamzadeh et al, 2011; Geist et al, 2012) and *construction* (Ghavamzadeh et al, 2010; Sun et al, 2011; Parr et al, 2007; Mahadevan and Liu, 2010) reduces the size of the approximation set by selecting or constructing basis functions specifically for each domain, and

Dean S. Wookey
School of Computer Science
The University of Witwatersrand, Johannesburg
E-mail: dean.wookey@students.wits.ac.za

George D. Konidaris
Departments of Computer Science & Electrical and Computer Engineering
Duke University, Durham, NC 27708
E-mail: gdk@cs.duke.edu

are typically used in conjunction with a batch method (Bradtke and Barto, 1996; Lagoudakis and Parr, 2003; Ernst et al, 2005) to fit the weights. Of particular interest are greedy selection methods such as orthogonal matching pursuit TD (OMP-TD), and orthogonal matching pursuit with Bellman residual minimization (OMP-BRM) (Painter-Wakefield and Parr, 2012), which aim to select basis functions predicted to cause the largest reduction in approximation error post selection. These methods use a set of samples collected from a fixed policy, and fit the basis function weights to approximate the value function for that policy. Unlike OMP-BRM which assigns weights to basis functions only once as they are selected, OMP-TD recalculates weights using a batch method after each selection. Appropriate batch method choices for OMP-TD include linear TD (Sutton, 1988), least squares policy evaluation (LSPE) (Yu and Bertsekas, 2009) and least squares TD (LSTD) (Bradtke and Barto, 1996), of which LSTD is used in this paper. The result of either method is a tailored set of basis functions and their respective weights specific to the domain/task and current policy. We focus on learning the value function for a specific policy, however policy improvement can be achieved through policy iteration (Sutton and Barto, 1998) by using the value function of the old policy (found using OMP-TD for example) to generate a new policy.

Although greedy selection of basis functions using current methods does lead to steadily improving approximations, the goal is to improve the approximation as rapidly as possible. OMP-TD often performs better in practice than OMP-BRM (Painter-Wakefield and Parr, 2012), but even so can be slow to converge for large discount factors (Mahadevan and Liu, 2010). In our experiments, values as low as $\gamma = 0.8$ reduced performance noticeably on some domains. This problem occurs because OMP-TD aims to select functions which will improve the representational power of the approximation set, such that with optimal fitting, the value function will move closer to the true value function at least as much as applying the Bellman operator would move it. Since the Bellman operator is a γ-contraction in the weighted L_2 norm (Van Roy, 1998), the reduction in L_2 error depends heavily on γ, with large γ leading to slow convergence.

With large γ however, value functions of environments with sparse reward functions and local transitions tend to be smooth. This is due to the Bellman equation, where the value of a state is equal to the discounted expected value of the next state and the expected reward. With sparse reward functions, the values between consecutive states change little, and if those consecutive states are close in distance sufficiently often, the value function as a whole can be expected to be smooth. Many domains based on physical systems exhibit spatially local transitions (Jonschkowski and Brock, 2013), and sparse reward functions are common, making smoothness a reliably exploitable attribute of value functions.

We introduce *feature regularization* as a way to enhance performance during feature selection by giving preference to basis functions according to prior expectations of the value function. In standard regression, regularization is commonly used to avoid overfitting, where simpler (smoother) solutions—solutions with smaller weights—are preferred. This regularization is usually applied in the final fitting of weights with a predetermined set of basis functions. In the incremental feature selection setting, however, regularization can be applied to the selection of individual basis functions as they are selected. Such regularization could be used to reduce overfitting, or to introduce a prior, but unlike regularizing the fit of the entire function, weight-based regularization is inappropriate since incremental fea-

ture selection methods usually choose basis functions that have the largest weights (high impact on the value function).

Tikhonov regularization (Tikhonov, 1963) is one way to incorporate domain knowledge such as value function smoothness into feature selection. We present the general form of this regularization and use it to form the smooth Tikhonov OMP-TD (STOMP-TD) algorithm, which uses a smoothness regularizer on the selection step of the OMP-TD algorithm. We also present an almost parameter free heuristic algorithm utilising smoothness; smoothness scaled OMP-TD (SSOMP-TD).

We show empirically that basis function smoothness is an effective regularizer in many reinforcement learning domains, and compare these two methods against methods with similar goals, regularized OMP-TD (ROMP-TD) (Needell and Vershynin, 2009) and least squares TD with random projections (LSTD-RP) (Ghavamzadeh et al, 2010). We chose these methods because ROMP-TD focuses on the problem of overfitting in feature selection, and LSTD-RP deals with the problem of large feature spaces by projecting them onto a smaller space.

We empirically evaluate feature regularization across six benchmark domains using two different types of basis, showing that feature regularization can significantly lower prediction error. SSOMP-TD in particular performs reliably well with no increase in computational complexity and an easily chosen parameter.

2 Background

A reinforcement learning (RL) task is typically formalized as a Markov Decision Process (MDP) (Sutton and Barto, 1998). An MDP is defined as a tuple, $M = (S, A, P, R, \gamma)$, where S is the set of all possible states s, $A = \{a_1, a_2, \ldots, a_N\}$ is the set of all actions, P is the transition probability model where $P(s, a, s')$ is the probability of moving from state s to state s' with action a, R is a real-valued reward function where $r = R(s, a, s')$ is the reward for taking action a in state s and ending up in state s', and $\gamma \in [0, 1)$ is the discount factor (Sutton and Barto, 1998). We define our state as a vector of real values, i.e., $s = [x_1, \ldots, x_d]$ where d is the number of dimensions, and each x_j is scaled to be between 0 and 1.

The goal is to learn a policy, π, that maximizes the expected return (discounted sum of future rewards). π is a stochastic mapping of states to actions where the probability of choosing a particular action a in state s is given by $\pi(s, a)$. Learning a policy can be achieved by learning a *value function* V if P is known, or an *action-value function* Q if P is not known. $V^\pi(s) \in \mathbb{R}$ is the expected return an agent receives starting in state s by following policy π thereafter. The value function for a policy π satisfies the Bellman equation $V^\pi = R + \gamma P V^\pi$, where V^* is the value function for an optimal policy. We only consider V since Q is similar.

Often, the number of states is too large to store individual values. A standard technique for representing value functions in such cases is value function approximation. Linear function approximation is a common choice. The approximate value function at a state s is represented by a linear sum of real-valued basis functions $\phi_i(s) \in \mathbb{R}$ with associated weights $w_i \in \mathbb{R}$: $\bar{V}^\pi(s) = \sum_{i=1}^{k} w_i \phi_i(s) = \boldsymbol{\phi}(s) \cdot \boldsymbol{w}$, where $\boldsymbol{\phi}(s) = [\phi_1(s), \ldots, \phi_k(s)]^T$ and $\boldsymbol{w} = [w_1, \ldots, w_k]^T$. For multiple states $\boldsymbol{s} = [s_1, \ldots, s_q]^T$, the vector $\bar{V}^\pi(\boldsymbol{s}) = [\boldsymbol{\phi}(s_1) \cdot \boldsymbol{w}, \ldots, \boldsymbol{\phi}(s_q) \cdot \boldsymbol{w}]^T$ is more easily expressed as $\bar{V}^\pi(\boldsymbol{s}) = \boldsymbol{\Phi}(\boldsymbol{s})\boldsymbol{w}$ where $\boldsymbol{\Phi}(\boldsymbol{s})$ is a matrix with columns formed by the basis functions $\phi_1, \phi_2, \ldots, \phi_k$ and rows formed by the states such that

$\Phi(\boldsymbol{s})_{ij} = \phi_j(s_i)$. We further use the notation Φ_J, where $\Phi \in \mathbb{R}^{q \times v}$ is a matrix and $J \subset \{1, \ldots, v\}$ is an index set, to mean the submatrix of Φ containing only the columns with indices present in J.

We use two types of basis functions, radial basis functions (RBFs) and the Fourier basis. An RBF is a Gaussian with variance $\sigma^2 > 0 \in \mathbb{R}$ and center $b_i \in \mathbb{R}^d$: $\phi_i(s) = \frac{1}{\sigma^{d/2}\pi^{d/4}} e^{-||b_i - s||^2/2\sigma^2}$, where each element b_{ij} gives the RBF's center position in dimension $j \in \{1, \ldots, d\}$. Typically, for an order n RBF approximation, $n+1$ RBFs are tiled evenly across each dimension with an additional constant function for a total of $(n+1)^d + 1$ basis functions. The RBF centers b_i in each dimension are further parametrized, with $b_{ij} = (c_{ij})/n$, where $c_i = [c_{i1}, c_{i2}, \ldots, c_{id}], c_{ij} \in \{0, \ldots, n\}$. The entire tiling can be obtained by varying c, and while σ can vary, we use $\sigma^2 = 1/(2(n+1)^3 - 4(n+1)^2 + 2(n+1))$ as described in Benoudjit and Verleysen (2003).

An n^{th} order Fourier basis set (Konidaris et al, 2011) consists of cosine functions of the form: $\phi_i(s) = cos(\pi c_i \cdot s)$, where $c_i = [c_{i1}, \ldots, c_{id}], c_{id} \in \{0, \ldots, n\}$. By systematically varying c, the full order n set of $(n+1)^d$ basis functions can be obtained. In the context of the Fourier basis, we say ϕ_i is *independent* if it is univariate.

2.1 Feature Selection

The choice of features $\phi_1, \phi_2, \ldots, \phi_k$ can have a significant impact on the performance of RL algorithms. One way to choose these features is to use a complete fixed basis scheme, examples of which include CMAC tilings, RBF and Fourier basis sets. These sets must grow exponentially in size with the number of dimensions, quickly becoming impractical. An approximate sparse representation is often desirable (Kolter and Ng, 2009), and therefore, methods for intelligently selecting sparse sets of features are of practical importance.

One method for feature selection that performs well is OMP-TD (Painter-Wakefield and Parr, 2012). Despite lacking theoretical guarantees, its empirical performance is as good as or better than other feature selection methods such as LARS-TD (Kolter and Ng, 2009; Painter-Wakefield and Parr, 2012). For this reason, we only consider OMP-TD here, though our methods could be extended to other feature selection methods.

The OMP-TD algorithm is a variation of Matching Pursuit (MP) (Mallat and Zhang, 1993). Given sampled states $\boldsymbol{s} = [s_1, \ldots, s_q]^T$, weights $\boldsymbol{w} = [w_1, \ldots, w_k]^T$ and matrix $\Phi(\boldsymbol{s}) \in \mathbb{R}^{q \times k}, \Phi(\boldsymbol{s})_{ij} = \phi_j(s_i)$ made up of previously selected basis functions $\phi_1, \ldots, \phi_k$ evaluated on those states, the OMP-TD algorithm selects basis functions in the same greedy manner as MP. The basis function ϕ_{k+1} is selected from a dictionary D of candidate basis functions such that the correlation $\rho = |\langle \mathbf{R}_k, \boldsymbol{\phi}(\boldsymbol{x}) \rangle| / ||\boldsymbol{\phi}(\boldsymbol{x})||$ between the basis function and the residual error is maximised. OMP-TD uses the temporal difference (TD) error as the residual, $\mathbf{R}_{k+1} = \boldsymbol{r} + \gamma \Phi(\boldsymbol{s}')\boldsymbol{w} - \Phi(\boldsymbol{s})\boldsymbol{w}$, and maintains optimal weight values $w_1, \ldots, w_k$ using LSTD after each basis function is added. At each iteration, the residual is recalculated and the basis function in the dictionary with largest correlation to the residual is selected.

2.2 Regularization

A common solution in least squares regression to the problem of ill-posed problems and overfitting is regularization. Given a matrix X and target vector $\boldsymbol{y}$, the problem is to find a vector $\boldsymbol{w}$ which minimizes the residual subject to the regularizer $\lambda U(\boldsymbol{w})$, equivalent to minimizing

$$||\boldsymbol{y} - X\boldsymbol{w}||^2 + \lambda U(\boldsymbol{w}) \tag{1}$$

for some $\lambda \in \mathbb{R}^+$.

An often desirable side effect of regularization is a sparse solution (Kolter and Ng, 2009), which can be achieved directly with L_1 regularization by imposing a sparsity inducing regularizer $U(\boldsymbol{w}) = ||\boldsymbol{w}||_1$. L_2 regularization instead introduces a constraint U which favors certain solutions or makes a problem well posed. L_2 regularization has the advantage that solutions can be found analytically where L_1 regularization often requires linear programming techniques. The most common form of constraint for L_2 regularization, $U(\boldsymbol{w}) = ||\Gamma\boldsymbol{w}||^2$, is known as Tikhonov regularization (Tikhonov, 1963) or ridge regression, where Γ is a suitable regularization matrix (usually the identity matrix which encourages solutions with small weight norms). While it is common to choose a U which prefers low weights, U can also be chosen as a measure of the solution's smoothness. This expresses preference for simpler solutions that are less likely to overfit the data, and works in the incremental feature selection setting where basis functions with large weights are explicitly selected for.

One class of smoothness measures[1] for a function f acting on a state vector s is given by

$$U_m(f) = \int (f^{(m)}(s))^2 ds, \tag{2}$$

where $f^{(m)}$ is the m^{th} derivative of f with $m = 2$ being a common and intuitive choice (Wahba, 1990) since it gives the total squared rate of change in slope. In the case of linear function approximation, where $X \in \mathbb{R}^{q \times k}$ is a matrix of basis functions, $\phi_1, \ldots, \phi_k$, evaluated at sampled states $s_1, \ldots, s_q$ with $X_{ij} = \phi_j(s_i)$, the smoothness of the underlying function can be expressed independently of X as $U_m(\boldsymbol{\phi} \cdot \boldsymbol{w})$, where $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_k]^T$ is a vector of basis functions, $\boldsymbol{w} = [w_1, \ldots, w_k]^T$ is a vector of real-valued weights, and $\boldsymbol{\phi} \cdot \boldsymbol{w}$ is a real-valued function of s.

3 Feature Regularization

With OMP-TD, each basis function selected is guaranteed to improve the approximation provided it is sufficiently correlated with the Bellman error basis function (BEBF) (Parr et al, 2007), the basis function equal to the Bellman error. While correlation to the Bellman error is a reliable metric for feature selection, in practice OMP-TD can be slow to reduce prediction error when the value function does

[1] There are many ways to measure smoothness, many of which exhibit similarities, for example the total variation norm is similar to U_1, where instead of squaring the function's derivative, the absolute value is taken instead. Our smoothness measure was chosen for its intuitive simplicity and existence of analytic solutions.

not resemble the reward function (Mahadevan and Liu, 2010). Using the Neumann series expansion of the value function $V = R + \gamma PR + \gamma^2 P^2 R + ...$ it is easy to see as γ increases away from 0, the value function looks less like the reward function. This is important because the first k BEBFs which OMP-TD aims to select span the space of the first k terms of the Neumann series (Parr et al, 2008). Even values as low as $\gamma = 0.9$ can lead to slow convergence in practice (Mahadevan and Liu, 2010). Since OMP-TD chooses a basis function which approximates the BEBF at each step, it can be even slower to converge.

One way to improve performance could be to choose basis functions less likely to overfit the BEBF, a problem when there are insufficient samples and many basis functions to chose from. This problem is particularly acute in the RL setting where dimensionality can be high and interaction costly. Another possibility is to introduce prior knowledge to augment selection. The smoothness prior can be used when the underlying value function is expected to be smooth. We show in section 3.1 there is good reason to believe that value functions are smooth in general, making the smoothness prior a powerful tool for feature selection in RL.

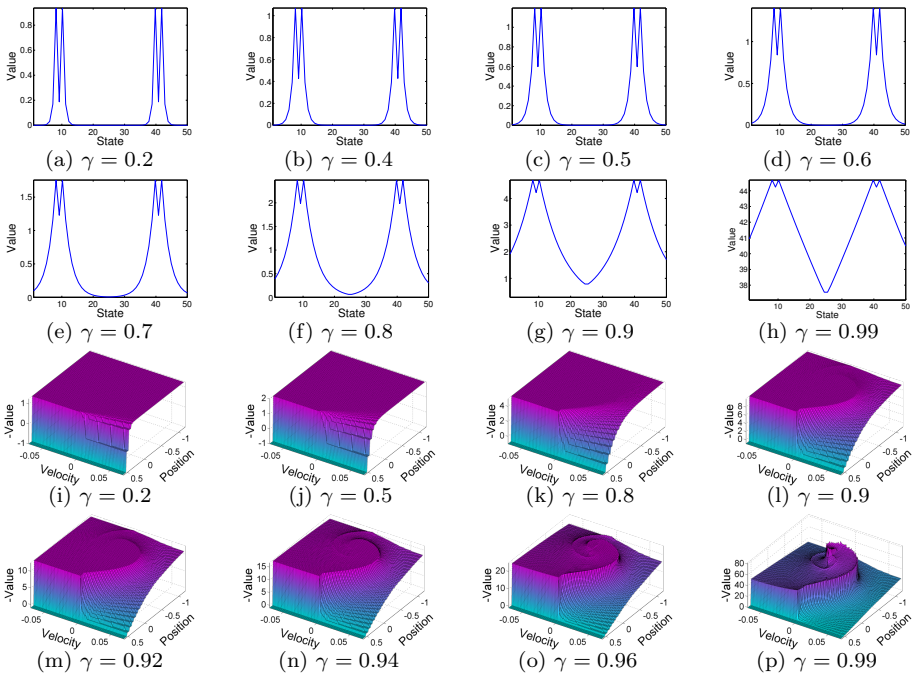
We present feature regularization as a way to improve feature selection in RL by using regularization techniques in the selection process to incorporate prior knowledge/reduce overfitting. We present methods based on the OMP-TD algorithm because its strong empirical results on benchmark domains and simplicity make it a good choice for feature selection (Painter-Wakefield and Parr, 2012). These methods could potentially be extended for use in other feature selection methods because they are decoupled from the LSTD fitting step in OMP-TD.

3.1 Regularized Selection for Smoothness

The idea that value functions are smooth in general is not new. Mahadevan and Maggioni (2007) note that value functions are typically smooth in Euclidean space. Their proto-value functions (PVFs), however, aim to take advantage of smoothness in the state graph. This type of smoothness derives from the fact that the value of a state is a function of "neighbouring" state values and associated rewards, where a state's neighbours are those reachable through transitions. Smoothness in Euclidean space is related, and coincides when neighbouring states are close in *distance* sufficiently often in conjunction with smooth or sparse reward functions. These conditions are often easy to fulfil as spatially local transitions are frequent in domains based on physical interactions (Jonschkowski and Brock, 2013) and sparse/smooth reward functions are common.

Value function smoothness is also related to the discount factor γ by the Bellman equation, since the value of each state is a sum of γ-discounted values of future states. Values of γ closer to 1 lead to increased smoothness in the value function along trajectories through the state space as successive values change less. This however, can also lead to larger differences in values between states which are close in space but not close in the state graph.

This can be seen in Fig. 1 where the chainwalk and negative mountain car value functions have been plotted against γ using the same policies used in the experiments in section 4. In mountain car with $\gamma = 0.99$, the policy follows the smooth spiral path down to the goal (position ≥ 0.5). The reward $= 0$ received at the goal is propagated along the spiral trajectory to the start position at $(0, 0)$.



(a) $\gamma = 0.2$ (b) $\gamma = 0.4$ (c) $\gamma = 0.5$ (d) $\gamma = 0.6$

(e) $\gamma = 0.7$ (f) $\gamma = 0.8$ (g) $\gamma = 0.9$ (h) $\gamma = 0.99$

(i) $\gamma = 0.2$ (j) $\gamma = 0.5$ (k) $\gamma = 0.8$ (l) $\gamma = 0.9$

(m) $\gamma = 0.92$ (n) $\gamma = 0.94$ (o) $\gamma = 0.96$ (p) $\gamma = 0.99$

Fig. 1 50-state chainwalk value functions for different values of γ (a-h). As γ increases, the value function gets smoother. Negative mountain car value functions for different values of γ (i-p). As γ increases, the value function gets smoother until $\gamma \approx 0.9$ and then rougher again.

Although the path following π is smoothest as $\gamma \to 1$, the value function as a whole is smoothest around $\gamma = 0.9$ due to sharp discontinuities forming for larger γ. Chainwalk gets smoother as $\gamma \to 1$.

These observations of value function smoothness naturally lead to a smoothness prior for feature selection. While value functions aren't necessarily smooth in Euclidean state space, smoothness is common, and unlike smoothness over the state graph, introducing a smoothness prior in Euclidean space does not require specific domain knowledge. In the case of OMP-TD with a fixed basis, the smoothness of each function in the dictionary $U_m(\phi_i)$ can be pre-calculated independently of the state graph since each ϕ_i is known. Using the Frobenius tensor norm, $||A|| = \sqrt{\sum_{j_1,j_2,\ldots,j_l} A^2_{j_1,j_2,\ldots,j_l}}$, where A is any order$-l$ tensor, we derive[2] the smoothness measures $U_m(\phi_i)$ using equation 2 in the general case where $m \geq 1$ for both the RBF and Fourier bases. For an RBF basis function ϕ_i in d dimensions with variance σ and arbitrary center position, the smoothness is:

$$U_m(\phi) = \frac{\prod_{j=1}^{m}(d + 2(j-1))}{2^m \sigma^{2m}}. \tag{3}$$

For a Fourier basis function ϕ_i with parameter vector c_i, the smoothness is:

$$U_m(\phi_i) = \frac{||c_i||^{2m} \pi^{2m}}{2}. \tag{4}$$

[2] Derivations for these two equations are given in the Appendix.

3.2 Smoothness Regularized Feature Selection Methods

We describe two ways of integrating the smoothness regularizer into the selection process, first using the standard Tikhonov regularization formulation with STOMP-TD, and second by directly modifying the basis function correlation formula with the SSOMP-TD selection algorithm.

The algorithm for each method follows the same steps as OMP-TD. Given a set of samples obtained using a fixed policy, and a dictionary of basis functions, an approximation set of basis functions is formed. This is initialized with at least the constant function. Weights are fitted to the approximation set using LSTD, and the resulting Bellman residual calculated. The correlation ρ of the Bellman residual to each basis function in the dictionary is calculated. This correlation is then modified by STOMP-TD (equation 6) or SSOMP-TD (equation 7) and the single most correlated basis function is selected and added to the approximation set. The weights are again calculated by LSTD and the process repeated until the Bellman residual is sufficiently small.

The algorithms can be prevented from selecting redundant features by using orthogonal basis sets in the dictionary, however even with redundant features, their correlation to the Bellman error will be small. This is because LSTD is the orthogonal projection of TV into the approximation set, and consequently Parr et al (2007) show that the Bellman residual $(TV - V)$ is orthogonal to the span of the approximation set.

3.2.1 Smooth Tikhonov OMP-TD

Tikhonov regularization (ridge regression) is a common type of regularization where the regularization term λU is added linearly to the minimization problem. For feature selection we are only interested in finding the single most correlated basis function, and hence the problem becomes finding the basis function ϕ_i and associated weight w_i which minimize the regularized problem at step k:

$$(\phi_i, w_i) = \arg \min_{\phi \in D, w \in \mathbb{R}} ||\mathbf{R}_{k-1} - w\phi(\mathbf{x})||^2 + \lambda U_m(w\phi), \tag{5}$$

where $U_m : C^1(\mathbb{R}^d, \mathbb{R}) \to \mathbb{R}$ is a suitable regularization function and $\lambda \in \mathbb{R}$ controls the harshness of the regularization with $\lambda = 0$ giving the unregularized case. Here $C^1(\mathbb{R}^d, \mathbb{R})$ is the class of continuously differentiable functions from $\mathbb{R}^d$ to $\mathbb{R}$. We use the smoothing regularization functions in equations 3 and 4 and find the solution to equation 5 by searching the dictionary for the basis function $\phi_i \in D$ with largest regularized correlation[3] ρ_i' where,

$$\rho_i' = \left| \frac{\langle \mathbf{R}_{k-1}, \phi_i(\mathbf{x}) \rangle}{\sqrt{||\phi_i(\mathbf{x})||^2 + \lambda U_m(\phi_i)}} \right|. \tag{6}$$

Using ρ_i' in place of ρ_i in the OMP-TD algorithm results in the Smooth Tikhonov OMP-TD (STOMP-TD) algorithm.

[3] The regularized correlation is derived in the Appendix.

3.2.2 Smoothness Scaled OMP-TD

The STOMP-TD algorithm introduces a new parameter λ. Methods for finding appropriate λ exist (Wahba, 1990), however we propose a parameter free heuristic approach which is less powerful than STOMP-TD but simpler to use in practice, called Smoothness Scaled OMP-TD (SSOMP-TD). The SSOMP-TD algorithm favors basis functions which are both smooth and highly correlated instead of focusing on correlation alone. In order to achieve this, the correlation ρ_i at step k of each basis function $\phi_i \in D$ is scaled by the square root of its smoothness $U_m(\phi_i)$, giving the regularized correlation ρ_i'':

$$\rho_i'' = \left| \frac{\langle \mathbf{R}_{k-1}, \phi_i(\mathbf{x}) \rangle}{||\phi_i(\mathbf{x})|| \sqrt{U_m(\phi_i)}} \right| = \frac{\rho_i}{\sqrt{U_m(\phi_i)}}, \tag{7}$$

where smooth basis functions have a low $U_m(\phi_i)$, increasing the correlation, and erratic/rough basis functions have a high $U_m(\phi_i)$, particularly in the case where $m = 2$. These two methods of regularizing the selection are similar; SSOMP-TD can be thought of as STOMP-TD with a large λ. To avoid division by 0, all basis functions with a smoothness of 0 are added to the approximation up front.

SSOMP-TD's performance is strongly related to the smoothness of V^* as shown in Fig. 2 where the performance of OMP-TD and SSOMP-TD $m = 2$ are plotted against the discount factor γ (as a way of controlling the function's smoothness). In chainwalk, the error in SSOMP-TD steadily decreases along with the smoothness as γ increases. In mountain car, the best performance of SSOMP-TD is reached when the value function is smoothest.

Both STOMP-TD and SSOMP-TD only modify the correlation calculation without increasing complexity and hence the running times are the same as OMP-TD's $O(Kq)$ for selection, and $O(k^3 + qk^2)$ for fitting with LSTD, where K is the size of the dictionary, q is the number of samples and k is the size of the approximation set.

3.3 ROMP-TD and LSTD-RP

We compare our methods to two methods with similar goals; the temporal difference version of the regularized orthogonal matching pursuit algorithm (ROMP) (Needell and Vershynin, 2009): ROMP-TD, and the least squares temporal difference with random projections algorithm (LSTD-RP) (Ghavamzadeh et al, 2010).

An adaptation to the OMP algorithm, ROMP has not yet been applied in the RL setting to the best of our knowledge. While the theoretical guarantees of the ROMP algorithm may still apply in the TD case, we leave the proofs of such for a more complete review of the ROMP algorithm in this setting. We include it as the only other algorithm we know of which employs regularization in the selection step of OMP. Unlike our OMP-TD based methods, ROMP selects a *set* of basis functions to add to the approximation at each step instead of a single function. The selected set J_0 fulfils two criteria: 1) the correlations of each basis function in the set are comparable,[4] and 2) the magnitude or energy $||\rho|_{J_0}||^2$ of the correlations

[4] The correlation ρ_i of basis function ϕ_i is comparable to the correlation ρ_j of ϕ_j if $|\rho_i| \leq 2|\rho_j|$ and $|\rho_j| \leq 2|\rho_i|$.

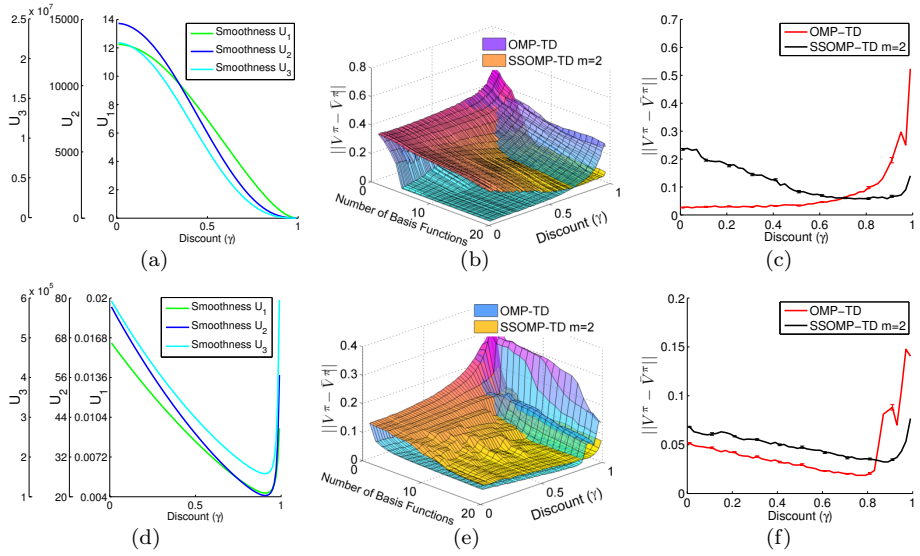


Fig. 2 The effect of discount factor γ on the normalised value function smoothness in chainwalk and mountain car (a) and (d). The normalised root mean squared error of OMP-TD and SSOMP-TD in relation to γ on the chainwalk (b,c) and mountain car domains (e,f) after each of the first 20 basis functions are selected (b,e) and after exactly 20 basis functions have been selected (c,f).

in the set is the largest amongst all sets which fulfil criteria 1. This forces the correlation of basis functions in the solution to be closer to uniform and thus more regular (Needell and Vershynin, 2009). Intuitively, it is better to have a set of good basis functions rather than one highly correlated basis function which may overfit the data. The selection step of the ROMP algorithm is decoupled from the OMP fitting step and can be inserted into OMP-TD without modification.

LSTD-RP deals with large feature sets differently. Instead of selecting features so that the set used in the fitting step is computationally manageable, LSTD-RP projects the entire set of features using a random projection matrix onto a low dimensional space. The low dimensional space results in a significant reduction in the size of the matrices used in LSTD, making it possible to compute value functions despite the curse of dimensionality.

The running time of the ROMP selection step is of the same order as OMP-TD's selection step[5]: $O(Kq)$ where K is the number of basis functions in the dictionary and q is the number of samples.

LSTD-RP has the advantage that it is not incremental and the LSTD calculation only has to be done once in $O(k^3 + qkK)$ where q is the number of samples, K is the size of the dictionary and k is the size of the low dimensional space which is the size of the approximation sets of the feature selection methods. Predicting the value of a state comes with a cost of $O(Kk)$ instead of $O(k)$ for feature selection methods.

[5] See the Appendix for the complete algorithm and analysis.

4 Experiments

Our goal is to compare the selection methods based on the ordering they impose on their candidate basis functions. We therefore show the performance of each method against the number of basis functions selected instead of the stopping condition parameter (β) described in the OMP-TD algorithm (Painter-Wakefield and Parr, 2012). Any setting of β would correspond to adding a vertical line at some point in the graph. Our experimental design is similar to that used in Painter-Wakefield and Parr (2012). For ROMP-TD, where sets of basis functions are selected,[6] we added basis functions from the set one at a time in descending order of correlation magnitude to make the methods comparable. For LSTD-RP, we projected the entire dictionary available to the other algorithms such that the matrices used in the fitting step of each method were the same size. We used the same amount of L_2 regularization as the other methods in the fit.

In the RBF experiments, the approximation set contained the constant function $\phi_1 = 1$ at the start. The dictionary contained basis function sets up to a maximum order. For example, with a maximum order of 4, the RBF basis sets of orders $n = 1, 2, 3$ and 4 would all be added to the dictionary, with each set using a different variation parameter $\sigma^2 = 1/(2(n+1)^3 - 4(n+1)^2 + 2(n+1))$ depending on its order, n. This would give the selection algorithm the choice between wider (more general/smoother) and thinner (more specific/sharper) basis functions where appropriate.

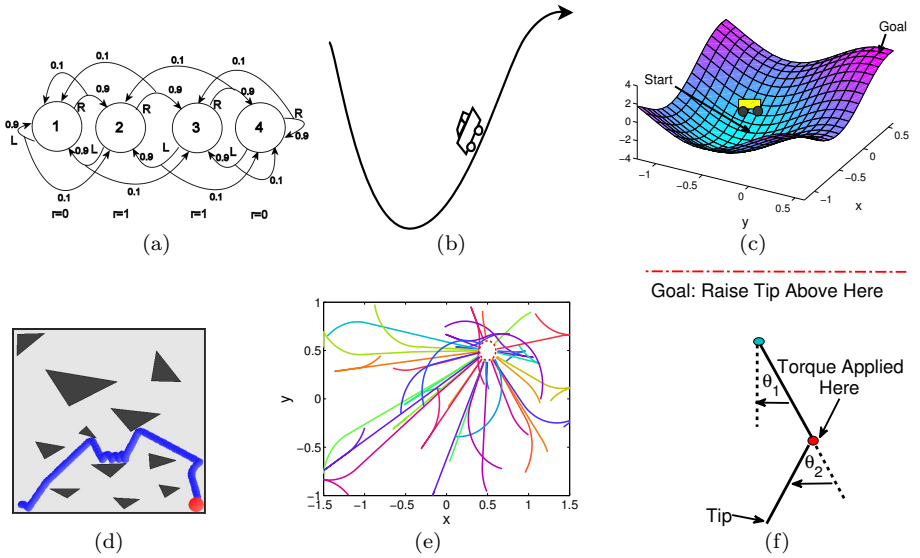
In the Fourier basis experiments, the approximation set contained independent basis functions up to a given order at the start. The dictionary contained all basis functions up to a maximum order. For the chain walk experiments, where all basis functions are independent, only the constant basis function was added to the approximation set at the start.

4.1 Domains

In each domain, training and test samples were collected using a fixed deterministic policy. In chainwalk, pairs of samples were collected by following the optimal policy from a random state to the next state. In the other domains, the given policy was followed from a random state until termination. The error in the approximation was obtained by comparing the approximate value $\bar{V}^\pi$ to the true value V^π using the root mean squared error. In chainwalk, V^π was calculated using dynamic programming. In the other domains, V^π was calculated using a single rollout (deterministic policy). The experiment setup for each domain with the RBF basis is given in Table 1 and with the Fourier basis in Table 2. We note that the high regularization for some of the RBF experiments was due to unstable behavior in OMP-TD and sometimes ROMP-TD. Painter-Wakefield and Parr (2012) also note this behavior. For the STOMP-TD experiments, we tried on average 10 different values of λ and only showed the best performing out of those. A poor choice significantly affects performance. Results were averaged over 100 trials.

50 State ChainWalk: The 50 state chainwalk domain (Lagoudakis and Parr, 2003) is a 1 dimensional domain with 50 states and rewards at states 10 and 41. We scaled the states between 0 and 1 for function approximation.

[6] We did not restrict the size of the selected set using the optional parameter ψ.



(a) (b) (c) (d) (e) (f)

Fig. 3 The domains used. A 4 state chainwalk with rewards at states 2 and 3 (a). Mountain car 2d (b). Mountain car 3d (c). The pinball configuration used in our experiments with sample trajectory (d). RC Car sample trajectories with random starts using our hand coded policy (e). Acrobot (f).

Domain	γ	Max Order	Initial BFs	Total BFs	Training Samples	Test Samples	L_2 Regularization
50 State Chainwalk	0.8	49	1	1276	500	1000	0.01
Mountain Car	0.99	9	1	385	5000	10000	0.1
Mountain Car 3D	0.99	4	1	978	20000	10000	1000.0
Pinball	0.99	4	1	978	20000	10000	0.1
RC Car	0.99	4	1	978	20000	10000	2500.0
Acrobot	0.99	4	1	978	20000	10000	5000.0

Table 1 Discount, initial dictionary and approximation set, samples collected and L_2 regularization for each experiment using the RBF basis.

Mountain Car: Mountain Car (Sutton and Barto, 1998) is a 2 dimensional domain where an underpowered car must escape a valley by driving up the sides, using its momentum to get to the top of the valley. We used the policy of accelerating in the direction of the current velocity or going right if at rest, used in Painter-Wakefield and Parr (2012).

Mountain Car 3D: Mountain Car 3D (Taylor et al, 2008) is a 4 dimensional version of the Mountain Car domain. We used a similar policy to the 2 dimensional case where the car always accelerates in the direction of maximum velocity, or in the closest direction towards the goal if stationary.

Pinball: The Pinball domain (Konidaris and Barto, 2009) is a 4 dimensional domain in which the agent controls a ball it must maneuver to the goal, bypassing or using the various obstacles. The configuration used is shown in Fig. 3. The policy was the greedy policy over a value function learned using Sarsa(λ) (γ =

Domain	γ	Max Order	Initial BFs	Total BFs	Training Samples	Test Samples	L_2 Regularization
50 State Chainwalk	0.8	300	1	301	500	1000	0.01
Mountain Car	0.99	10	21	121	5000	10000	0.01
Mountain Car 3D	0.99	5	21	1296	20000	10000	0.01
Pinball	0.99	5	21	1296	20000	10000	0.01
RC Car	0.99	5	21	1296	20000	10000	0.01
Acrobot	0.99	5	21	1296	20000	10000	0.01

Table 2 Discount, initial dictionary and approximation set, samples collected and L_2 regularization for each experiment using the Fourier basis.

$0.99, \lambda = 0.95, \epsilon = 0.01$) with an $O(5)$ Fourier basis and an automatic learning rate adjusting method (Dabney and Barto, 2012).

RC Car: The Remote Control Car domain (Geramifard, 2013) is a 4 dimensional domain with 9 actions in which a remote control car must be maneuvered to within a distance of 0.1 meters from the goal. The 9 possible actions are derived from combinations of 2 primitive action types: steering $\in \{\texttt{left}, \texttt{straight}, \texttt{right}\}$ and acceleration $\in \{\texttt{forward}, \texttt{coast}, \texttt{backward}\}$. The state is given by the x and y position of the car, its heading θ in radians and its velocity v. If the car collides with any of the walls in the 2×3 meter room it's in, its velocity is set to 0, otherwise the domain is frictionless. At each step a reward of -1 is given if the agent is not at the terminal goal state or 100 if it is. The policy π used simulates all possible actions at each state and chooses the one which minimises $\delta + 2o$, where δ is the Euclidean distance to the goal and o is the orthogonal distance to the goal along the current trajectory of the car.

Acrobot: The acrobot domain (Sutton and Barto, 1998) is a 4 dimensional domain where an agent controls the torque of the middle link of a 2 link pendulum. The goal is to raise the tip of the pendulum a certain height. The policy applies torque in the direction of the joints if they are moving in the same direction, else it reverses it. Occasionally it didn't terminate, and we discarded those samples.

4.2 Results

The experimental results are shown in Fig. 4 and Fig. 5. They show that standard OMP-TD performs well in general, but is beaten by SSOMP-TD and STOMP-TD on all but the Fourier acrobot experiment. Interestingly, with the settings of γ selected, the requirement for guaranteed improvement (the angle between ϕ and the BEBF must be less than $cos^{-1}(\gamma)$ (Parr et al, 2007)) was seldom, if ever, satisfied. Despite this, OMP-TD performs well, suggesting the existence of stronger theoretical results. The high regularization for the RBF experiments was due to unstable behavior in the LSTD fitting step. Interestingly SSOMP-TD and STOMP-TD did not suffer from this unstable behavior.

The SSOMP-TD algorithms performed reliably well, with the smoothing regularizer U_1 performing the worst of the three on average, and U_2 most often performing the best, or tied for best. SSOMP-TD performed its worst on acrobot with both basis sets. This could indicate that the underlying value function of our

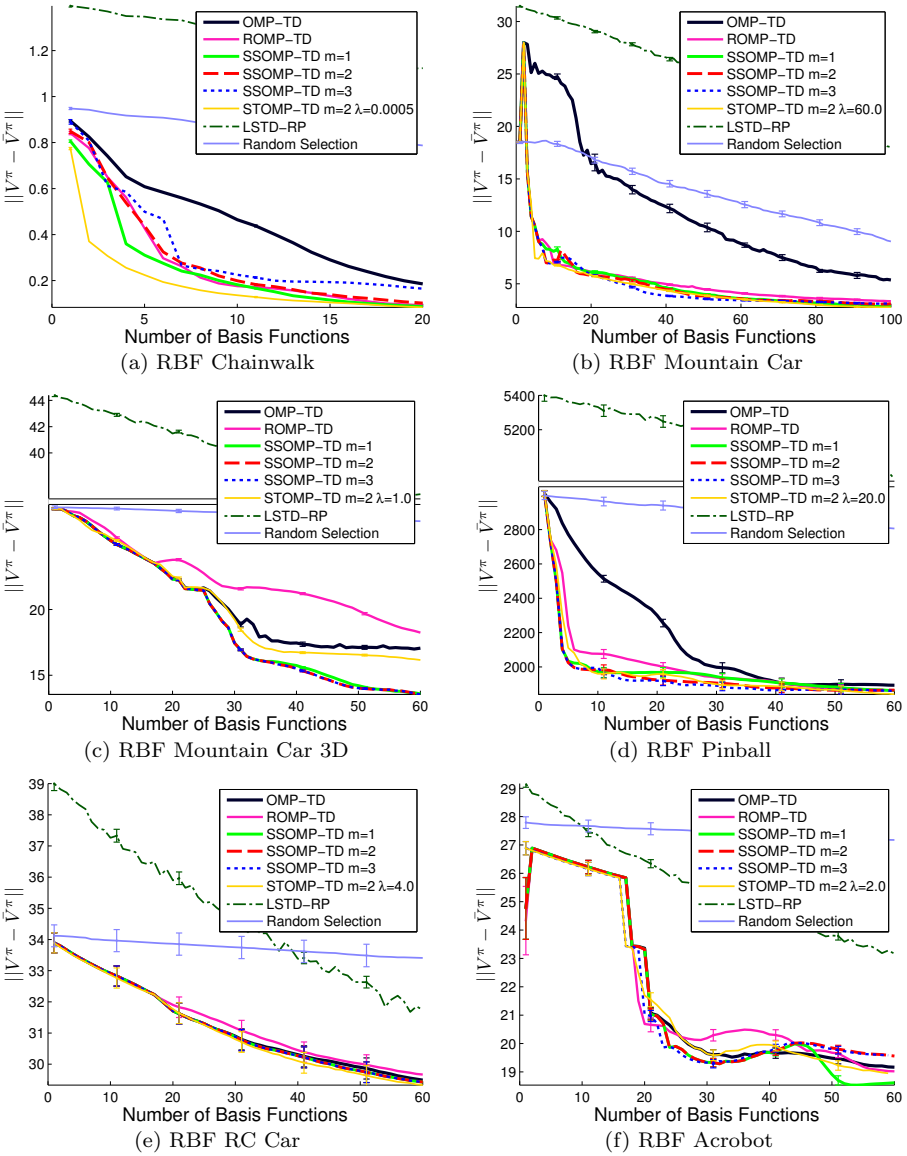


Fig. 4 Experimental results for the RBF basis on the chainwalk, mountain car, mountain car 3d, pinball, rc car and acrobot domains (a-f).

policy is not smooth. The algorithm seems to be at least as good as the others, if not better on the remaining domains taken as a whole.

The STOMP-TD algorithm performed well across all domains, however selecting an appropriate λ was difficult, and some settings resulted in performance significantly worse than OMP-TD. It is more powerful than SSOMP-TD as it can be adapted to use any prior, but it is also more difficult to use. In practice, we

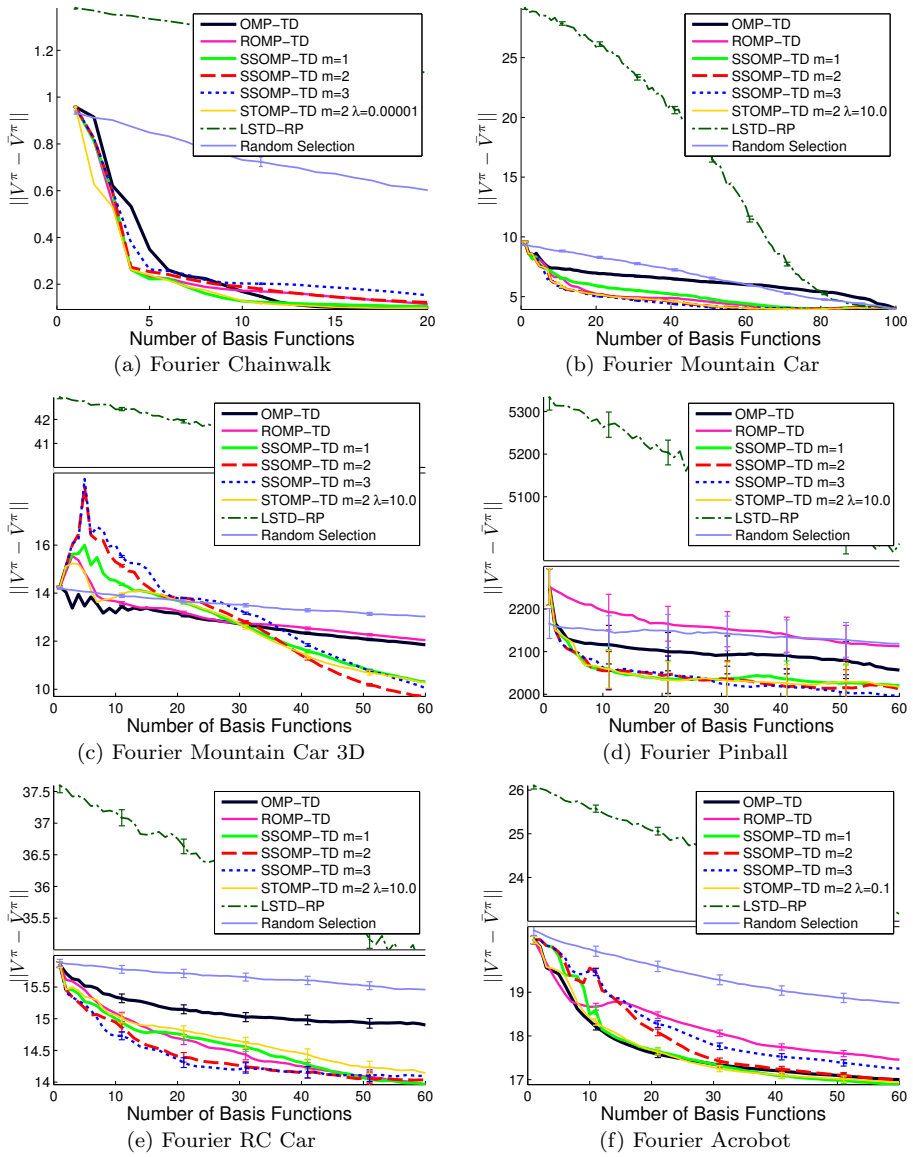


Fig. 5 Experimental results for the Fourier basis on the chainwalk, mountain car, mountain car 3d, pinball, rc car and acrobot domains (a-f).

would use SSOMP-TD as choosing the m parameter is significantly easier than choosing λ, and the results are comparable for the two methods.

ROMP-TD was never the best performer, and in multiple experiments performed the worst. ROMP-TD primarily tackles the problem of overfitting, and it is possible there were too many samples for overfitting to occur. This suggests that the effectiveness of SSOMP-TD and STOMP-TD is primarily due to the smoothness of the value function in these experiments. Interestingly in the Fourier

pinball experiment, selecting basis functions randomly outperformed ROMP-TD. It is possible that selecting batches of basis functions was too conservative and the other methods could take advantage of the changing residual after each selection. Even randomly chosen functions were better than a set of functions selected to approximate one residual well.

LSTD-RP performed poorly in all domains, only beating the random selection method in two of the RBF experiments. Its best performance with the Fourier basis was in mountain car, which was the only experiment where every basis function in the dictionary was selected. In that experiment its performance matched that of the other methods near the end, indicating that LSTD-RP is negatively affected by poor basis functions in the dictionary, which dilute the resulting projection.

The spikes in the graphs, particularly in figure 5 (c) are the result of fitting problems with LSTD. We verified this by fitting the basis set at each step to the estimates of the true value function obtained using rollouts. We used the same training and test set and plotted the results.[7] As expected, the error for each method decreased monotonically and the relative performance of the methods was similar.

5 Future Work and Conclusions

A major concern when using function approximation in high dimensional domains is the large computational costs involved. Feature selection, the automated process of selecting a subset of these functions to use in the approximation, has consequently begun to receive significant attention recently. However, current greedy selection techniques ignore characteristics of the underlying value function and select basis functions which best fit the sample data. This may lead to the selection of more basis functions than necessary. The introduction of prior knowledge can improve performance. One such prior is the smoothness prior which can be integrated into the selection process via regularization.

We have presented three ways to regularize feature selection in RL, ROMP-TD, STOMP-TD and SSOMP-TD. Our results using these methods show that feature regularization is a simple yet powerful tool for improving performance without significant computational overhead. Further, the smoothness prior generalizes the intuition that wide/low frequency basis functions are better when only a few basis functions can be selected, and shows continued empirical improvements well beyond the first few basis functions.

One possible direction for future study is to combine STOMP-TD and SSOMP-TD with state space restructuring methods such as Predictive Projections (Sprague, 2009), which projects the original state space in such a way that neighboring states in the state graph are closer together in the projected state space. There is also potential to use the smoothness prior in dictionary construction, to either reduce its size significantly in high dimensional domains, or to improve the quality of basis functions in a dictionary of fixed size. Randomly selected RBFs would be a natural choice, where a bias towards smooth RBFs could be included easily.

[7] Many of the plots involving the true error $||V^\pi - \Pi V^\pi||$ overlapped the others, so we have omitted them for clarity.

References

Benoudjit N, Verleysen M (2003) On the kernel widths in radial-basis function networks. Neural Processing Letters 18:139–154

Bradtke S, Barto A (1996) Linear least-squares algorithms for temporal difference learning. Machine Learning 22(1):33–57

Dabney W, Barto A (2012) Adaptive step-size for online temporal difference learning. In: Twenty-Sixth AAAI Conference on Artificial Intelligence, pp 872–878

Ernst D, Geurts P, Wehenkel L (2005) Tree-based batch mode reinforcement learning. In: Journal of Machine Learning Research, pp 503–556

Geist M, Scherrer B, Lazaric A, Ghavamzadeh M (2012) A Dantzig selector approach to temporal difference learning. In: Proceedings of the 29th International Conference on Machine Learning, pp 1399–1406

Geramifard A (2013) Rc-car domain. http://acl.mit.edu/RLPy/api/domains_misc.html#rccar

Ghavamzadeh M, Lazaric A, Maillard O, Munos R (2010) LSTD with random projections. In: Advances in Neural Information Processing Systems 23, pp 721–729

Ghavamzadeh M, Lazaric A, Munos R, Hoffman M (2011) Finite-sample analysis of Lasso-TD. In: Proceedings of the 28th International Conference on Machine Learning, pp 1177–1184

Johns J, Painter-Wakefield C, Parr R (2010) Linear complementarity for regularized policy evaluation and improvement. In: Advances in Neural Information Processing Systems 23, pp 1009–1017

Jonschkowski R, Brock O (2013) Learning task-specific state representations by maximizing slowness and predictability. 6th International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems (ERLARS)

Kolter J, Ng A (2009) Regularization and feature selection in least-squares temporal difference learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp 521–528

Konidaris G, Barto A (2009) Skill discovery in continuous reinforcement learning domains using skill chaining. In: Advances in Neural Information Processing Systems 22, pp 1015–1023

Konidaris G, Osentoski S, Thomas P (2011) Value function approximation in reinforcement learning using the Fourier basis. In: Proceedings of the Twenty-Fifth Conference on Artificial Intelligence, pp 380–385

Lagoudakis M, Parr R (2003) Least-squares policy iteration. Journal of Machine Learning Research 4:1107–1149

Mahadevan S, Liu B (2010) Basis construction from power series expansions of value functions. In: Advances in Neural Information Processing Systems 23, pp 1540–1548

Mahadevan S, Maggioni M (2007) Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. Journal of Machine Learning Research pp 2169–2231

Mallat S, Zhang Z (1993) Matching pursuits with time-frequency dictionaries. IEEE Transactions on Signal Processing 41(12):3397–3415

Needell D, Vershynin R (2009) Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. Foundations of computational mathematics 9(3):317–334

Painter-Wakefield C, Parr R (2012) Greedy algorithms for sparse reinforcement learning. In: Proceedings of the 29th International Conference on Machine Learning, pp 1391–1398

Parr R, Painter-Wakefield C, Li L, Littman M (2007) Analyzing feature generation for value-function approximation. In: Proceedings of the 24th international conference on Machine learning, pp 737–744

Parr R, Li L, Taylor G, Painter-Wakefield C, Littman M (2008) An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In: Proceedings of the 25th International Conference on Machine Learning, pp 752–759

Petrik M, Taylor G, Parr R, Zilberstein S (2010) Feature selection using regularization in approximate linear programs for Markov decision processes. In: Proceedings of the 27th International Conference on Machine Learning, pp 871–878

Sprague N (2009) Predictive projections. In: Proceedings of the 21st international joint conference on Artifical intelligence, pp 1223–1229

Sun Y, Gomez FJ, Ring MB, Schmidhuber J (2011) Incremental basis construction from temporal difference error. In: Proceedings of the 28th International Conference on Machine Learning, pp 481–488

Sutton R, Barto A (1998) Introduction to Reinforcement Learning. MIT Press, Cambridge, MA, USA
Sutton RS (1988) Learning to predict by the methods of temporal differences. Machine learning 3(1):9–44
Taylor M, Kuhlmann G, Stone P (2008) Autonomous transfer for reinforcement learning. In: The Seventh International Joint Conference on Autonomous Agents and Multiagent Systems, pp 283–290
Tikhonov A (1963) Solution of incorrectly formulated problems and the regularization method. In: Soviet Math. Dokl., vol 5, p 1035
Van Roy B (1998) Learning and value function approximation in complex decision processes. PhD thesis, Massachusetts Institute of Technology
Wahba G (1990) Spline models for observational data, vol 59. Society for industrial and applied mathematics
Yu H, Bertsekas DP (2009) Convergence results for some temporal difference methods based on least squares. Automatic Control, IEEE Transactions on 54(7):1515–1531

A Smoothness Measures

We use the Frobenius tensor norm, $||A|| = \sqrt{\sum_{j_1,j_2,\ldots,j_l} A^2_{j_1,j_2,\ldots,j_l}}$, where A is any order$-l$ tensor.

The smoothness measure $U_m(\phi)$ gives the smoothness of the function ϕ, or how far ϕ is from a smooth function. In general, different classes of smoothness take the form:

$$U_m(\phi) = \int (\phi^{(m)}(s))^2 dx_1 dx_2 ... dx_d, \tag{8}$$

where $s = [x_1, \ldots, x_d]$, d is the number of dimensions, and each x_j is scaled to be between 0 and 1.

A.1 Fourier Smoothness

Given a Fourier basis function $\phi(s) = cos(\pi c \cdot s)$ in d dimensions with parameter vector $c = [c_1, c_2, ..., c_d]$ defined on the interval $[0, 1]$, $s = [x_1, x_2, ...x_d]$, the smoothness measure for $m = 1$, $U_1(\phi)$ is as follows:

$$\begin{aligned} U_1(\phi) &= \int_0^1 ||\phi(s)'||^2 ds \\ &= \int_0^1 \left|\left|\left(\tfrac{\partial cos(\pi c \cdot s)}{\partial x_1} \;\; \tfrac{\partial cos(\pi c \cdot s)}{\partial x_2} \;\; \ldots \;\; \tfrac{\partial cos(\pi c \cdot s)}{\partial x_d}\right)\right|\right|^2 dx_1 dx_2 ... dx_d \\ &= \int_0^1 (c_1^2 + c_2^2 + ... + c_n^2)\pi^2 sin(\pi c \cdot s)^2 dx_1 dx_2 ... dx_d \\ &= \int_0^1 ||c||^2 \pi^2 \frac{(1 - cos(2\pi c \cdot s))}{2} dx_1 dx_2 ... dx_d. \end{aligned} \tag{9}$$

Since c is a vector of integer values, $sin(2\pi \sum_{i=1}^d c_i) = 0$, and hence the expression simplifies to:

$$U_1(\phi) = \frac{||c||^2 \pi^2}{2} \tag{10}$$

$$
\begin{aligned}
U_2(\phi) &= \int_0^1 ||\phi(s)''||^2 ds \\
&= \int_0^1 \left|\left| \begin{pmatrix} \frac{\partial^2 cos(\pi c \cdot s)}{\partial x_1^2} & \frac{\partial^2 cos(\pi c \cdot s)}{\partial x_2 \partial x_1} & \cdots & \frac{\partial^2 cos(\pi c \cdot s)}{\partial x_d \partial x_1} \\ \frac{\partial^2 cos(\pi c \cdot s)}{\partial x_1 \partial x_2} & \frac{\partial^2 cos(\pi c \cdot s)}{\partial x_2^2} & \cdots & \frac{\partial^2 cos(\pi c \cdot s)}{\partial x_d \partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 cos(\pi c \cdot s)}{\partial x_1 \partial x_d} & \frac{\partial^2 cos(\pi c \cdot s)}{\partial x_2 \partial x_d} & \cdots & \frac{\partial^2 cos(\pi c \cdot s)}{\partial x_d^2} \end{pmatrix} \right|\right|^2 dx_1 dx_2 ... dx_d \\
&= \int_0^1 (c_1^4 + 2c_1^2 c_2^2 + ... + c_d^4)\pi^4 cos(\pi c \cdot s)^2 dx_1 dx_2 ... dx_d \\
&= \int_0^1 ||c||^4 \pi^4 cos(\pi c \cdot s)^2 dx_1 dx_2 ... dx_d \\
&= \int_0^1 ||c||^4 \pi^4 \frac{(cos(2\pi c \cdot s) + 1)}{2} dx_1 dx_2 ... dx_d
\end{aligned} \tag{11}
$$

Since c is a vector of integer values, $sin(2\pi \sum_{i=1}^d c_i) = 0$, and hence the expression simplifies to:

$$
U_2(\phi) = \frac{||c||^4 \pi^4}{2} \tag{12}
$$

After the 3^{rd} derivative a pattern emerges and the general form for $m > 0, m \in \mathbb{Z}$ is:

$$
U_m(\phi) = \frac{||c||^{2m} \pi^{2m}}{2}. \tag{13}
$$

A.2 RBF Smoothness

For a radial basis function,

$$
\phi(s) = \frac{e^{\frac{-||c-s||^2}{2\sigma^2}}}{\pi^{d/4}\sigma^{d/2}}, \tag{14}
$$

in d dimensions with parameter vector $c = [c_1, c_2, ..., c_d]$ and variance σ defined on the interval $(-\infty, \infty)$, $s = [x_1, x_2, ...x_d]$. When taking the integral with respect to s from $-\infty$ to ∞, the location of the rbf center becomes irrelevant so c can be set to 0. The smoothness measure for $m = 1$, $U_1(\phi)$ is as follows:

$$
\begin{aligned}
U_1(\phi) &= \int_{-\infty}^{\infty} (\phi^{(1)}(s))^2 dx_1 dx_2 ... dx_d \\
&= \int_{-\infty}^{\infty} \left|\left| \begin{pmatrix} \frac{\partial \phi(s)}{\partial x_1} & \frac{\partial \phi(s)}{\partial x_2} & \cdots & \frac{\partial \phi(s)}{\partial x_d} \end{pmatrix} \right|\right|^2 dx_1 dx_2 ... dx_d \\
&= \int_{-\infty}^{\infty} \sum_{i=1}^{d} \left(\frac{x_i^2}{\pi^{d/2}\sigma^{d+4}} e^{\frac{-||c-s||^2}{\sigma^2}} \right) ds \\
&= \frac{1}{\pi^{d/2}\sigma^{d+4}} \left(\sum_{i=1}^{d} \int_{-\infty}^{\infty} x_i^2 e^{-x_i^2/\sigma^2} dx_i \prod_{j=1, j\neq i}^{d} \int_{-\infty}^{\infty} e^{-x_j^2/\sigma^2} dx_j \right) \\
&= \frac{1}{\pi^{d/2}\sigma^{d+4}} \left(\sum_{i=1}^{d} \sigma^3 \int_{-\infty}^{\infty} t^2 e^{-t^2} dt \prod_{j=1, j\neq i}^{d} \sigma \int_{-\infty}^{\infty} e^{-t^2} dt \right) \\
&= \frac{1}{\pi^{d/2}\sigma^{d+4}} \left(d\sigma^3 \int_{-\infty}^{\infty} t^2 e^{-t^2} dt \left(\sigma \int_{-\infty}^{\infty} e^{-t^2} dt \right)^{d-1} \right) \\
&= \frac{d\sigma^{d+2}}{\pi^{d/2}\sigma^{d+4}} \frac{\sqrt{\pi}}{2} (\sqrt{\pi})^{d-1} \\
&= \frac{d}{2\sigma^2}
\end{aligned} \tag{15}
$$

The smoothness measure for $m = 2$, $U_2(\phi)$ is derived as follows:
When $i = j$,

$$\begin{aligned}\frac{\partial^2\phi(s)}{\partial x_i x_j} &= \frac{\partial^2\phi(s)}{\partial x_i^2} \\ &= \left(\frac{x_i^2}{\sigma^4} - \frac{1}{\sigma^2}\right)\frac{e^{-(x_1^2+\ldots+x_d^2)/2\sigma^2}}{(\pi\sigma^2)^{d/4}} \\ &= \frac{1}{\pi^{d/4}\sigma^{d/2+4}}(x_i^2-\sigma^2)e^{-(x_1^2+\ldots+x_d^2)/2\sigma^2},\end{aligned} \tag{16}$$

and when $i \neq j$,

$$\begin{aligned}\frac{\partial^2\phi(s)}{\partial x_i x_j} &= \frac{x_i x_j}{\sigma^4}\frac{e^{-(x_1^2+\ldots+x_d^2)/2\sigma^2}}{(\pi\sigma^2)^{d/4}} \\ &= \frac{x_i x_j}{\pi^{d/4}\sigma^{d/2+4}}e^{-(x_1^2+\ldots+x_d^2)/2\sigma^2},\end{aligned} \tag{17}$$

then:

$$\begin{aligned}U_2(\phi) &= \int_{-\infty}^{\infty}(\phi^{(2)}(s))^2 dx_1 dx_2 \ldots dx_d \\ &= \int_{-\infty}^{\infty}\left|\left|\begin{pmatrix}\frac{\partial^2\phi(s)}{\partial x_1^2} & \frac{\partial^2\phi(s)}{\partial x_2\partial x_1} & \cdots & \frac{\partial^2\phi(s)}{\partial x_d\partial x_1}\\ \frac{\partial^2\phi(s)}{\partial x_1\partial x_2} & \frac{\partial^2\phi(s)}{\partial x_2^2} & \cdots & \frac{\partial^2\phi(s)}{\partial x_d\partial x_2}\\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2\phi(s)}{\partial x_1\partial x_d} & \frac{\partial^2\phi(s)}{\partial x_2\partial x_d} & \cdots & \frac{\partial^2\phi(s)}{\partial x_d^2}\end{pmatrix}\right|\right|^2 dx_1 dx_2 \ldots dx_d \\ &= \frac{1}{\pi^{d/2}\sigma^{d+8}}\int_{-\infty}^{\infty}\left(\sum_{i=1}^{d}(x_i^2-\sigma^2)^2 + \sum_{i=1}^{d}\sum_{j=1,i\neq j}^{d}x_i^2 x_j^2\right)e^{-(x_1^2+\ldots+x_d^2)/\sigma^2}dx_1 dx_2 \ldots dx_d \\ &= \frac{1}{\pi^{d/2}\sigma^{d+8}}\left\{\left(d\sigma^5\int_{-\infty}^{\infty}(t^2-1)^2 e^{-t^2}dt\right)\left(\sigma\int_{-\infty}^{\infty}e^{-t^2}dt\right)^{d-1}\right. \\ &\quad \left.+d(d-1)\left(\sigma^3\int_{-\infty}^{\infty}t^2 e^{-t^2}dt\right)^2\left(\sigma\int_{-\infty}^{\infty}e^{-t^2}dt\right)^{d-2}\right\} \\ &= \frac{1}{\pi^{d/2}\sigma^{d+4}}\left\{\left(3d\frac{\sqrt{\pi}}{4}\right)\pi^{(d-1)/2} + d(d-1)\frac{\pi}{4}\pi^{(d-2)/2}\right\} \\ &= \frac{1}{\sigma^4}\left\{\frac{3d}{4}+\frac{d(d-1)\pi}{4}\right\} \\ &= \frac{3d+d(d-1)}{4\sigma^4} \\ &= \frac{d(d+2)}{4\sigma^4}\end{aligned} \tag{18}$$

After the 4^{th} derivative a pattern emerges and the general form for $m > 0, m \in \mathbb{Z}$ is:

$$U_m(\phi_i) = \frac{\prod_{i=1}^{m}(d+2(i-1))}{2^m\sigma^{2m}}. \tag{19}$$

B Full ROMP Algorithm

The full ROMP selection algorithm is given in algorithm 1.

B.1 ROMP Running Time

The running time of the ROMP selection step is of the same order as OMP-TD's selection step: $O(Kq)$ where K is the number of basis functions in the dictionary and q is the number of samples. Both compute the correlations of each basis function in $O(Kq)$ time, however the ROMP algorithm sorts these correlations ($O(K\log(K))$) and selects the set with maximal energy which can be done in $O(K)$.

In the given algorithm, the inner while loop only runs forward and never runs over the same indices more than once, therefore despite being inside a for loop, the total executions of the instructions in the while loop are of $O(K)$. Additional proof of this is given in the original ROMP paper.

As long as $q > \log(K)$, the time complexity of the selection is the same as OMP-TD $O(Kq)$, otherwise the complexity is $O(K\log(K))$.

Algorithm 1 ROMPFeatureSelect

Input:
$\boldsymbol{\rho} \in \mathbb{R}^K$: Correlation of each basis function.
$I \subset \{1, ..., K\}$: The indices of currently selected basis functions.
$\psi \in [1, ..., K]$: The sparsity parameter.
Output:
Set of basis function indices J_0 to be added to I.

$c \leftarrow$ list of indices, $\bar{I}_i$, sorted in descending order of the $\rho_{\bar{I}_i}$ they correspond to.
$J \leftarrow \{c_1, ..., c_\psi\}$, the set of indices corresponding to the ψ largest values in ρ.
Select: Find all sets $J_0 \subset J$ such that $|\rho_i| \leq 2|\rho_j|$ for all $i, j \in J_0$. Choose set with the greatest $||\boldsymbol{\rho}_{J_0}||^2$ value.
$currTotal \leftarrow 0$, $bestTotal \leftarrow 0$, $bestStart \leftarrow 0$, $bestEnd \leftarrow 0$
$j \leftarrow 0$
for $i = 1$ to $size(J)$ **do**
  $currTotal \leftarrow currTotal + \rho^2_{J_i}$
  while $|\rho_{J_j}| > 2|\rho_{J_i}|$ **do**
    $currTotal \leftarrow currTotal - \rho^2_{J_j}$
    $j \leftarrow j + 1$
  end while
  if $currTotal >= bestTotal$ **then**
    $bestStart \leftarrow j$, $bestEnd \leftarrow i$, $bestTotal \leftarrow currTotal$
  end if
end for
$J_0 \leftarrow \{J_{bestStart}, J_{bestStart+1}, ..., J_{bestEnd}\}$
return J_0

C Tikhonov Derivation

For a single basis function ϕ_i we derive the Tikohonv regularized correlation ρ' at step k by first solving for the optimal α_i which minimizes the residual error vector $\mathbf{R}_k = \mathbf{R}_{k-1} + \alpha_i\phi_i(\mathbf{x})$:

$$\alpha_i = \arg\min_{\alpha_i \in \mathbb{R}} ||\mathbf{R}_{k-1} - \alpha_i\phi_i(\mathbf{x})||^2 + \lambda U_m(\alpha_i\phi_i) \tag{20}$$

by taking the derivative and setting to 0:

$$\begin{aligned} 0 &= \frac{\partial ||\mathbf{R}_{k-1} - \alpha_i \phi_i(\mathbf{x})||^2}{\partial \alpha_i} + \frac{\partial \lambda U_m(\alpha_i \phi_i)}{\partial \alpha_i} \\ 0 &= \frac{\partial ||\mathbf{R}_{k-1} - \alpha_i \phi_i(\mathbf{x})||^2}{\partial \alpha_i} + \frac{\partial \lambda \alpha_i^2 U_m(\phi_i)}{\partial \alpha_i} \\ 0 &= -2\langle \mathbf{R}_{k-1}, \phi_i(\mathbf{x})\rangle + 2\alpha_i ||\phi_i(\mathbf{x})||^2 + 2\lambda \alpha_i U \\ \alpha_i &= \frac{\langle \mathbf{R}_{k-1}, \phi_i(\mathbf{x})\rangle}{||\phi_i(\mathbf{x})||^2 + \lambda U}. \end{aligned} \tag{21}$$

Substituting the optimal α into the minimization problem for any given ϕ_i, the best $\phi \in D$ is the one that minimizes the following equation:

$$\begin{aligned} &||\mathbf{R}_{k-1} - \alpha\phi(\mathbf{x})||^2 + \lambda U_m(\alpha\phi) \\ &= \left|\left|\mathbf{R}_{k-1} - \frac{\langle \mathbf{R}_{k-1}, \phi(\mathbf{x})\rangle}{||\phi(\mathbf{x})||^2 + \lambda U_m(\phi)}\phi(\mathbf{x})\right|\right|^2 + \lambda U_m(\phi) \left|\left|\frac{\langle \mathbf{R}_{k-1}, \phi(\mathbf{x})\rangle}{||\phi(\mathbf{x})||^2 + \lambda U_m(\phi)}\right|\right|^2 \\ &= ||\mathbf{R}_{k-1}||^2 - 2\left(\frac{\langle \mathbf{R}_{k-1}, \phi(\mathbf{x})\rangle}{||\phi(\mathbf{x})||^2 + \lambda U_m(\phi)}\right)\langle \mathbf{R}_{k-1}, \phi(\mathbf{x})\rangle \\ &\quad + \left(\frac{\langle \mathbf{R}_{k-1}, \phi(\mathbf{x})\rangle}{||\phi(\mathbf{x})||^2 + \lambda U_m(\phi)}\right)^2 ||\phi(\mathbf{x})||^2 + \lambda U_m(\phi) \left|\left|\frac{\langle \mathbf{R}_{k-1}, \phi(\mathbf{x})\rangle}{||\phi(\mathbf{x})||^2 + \lambda U_m(\phi)}\right|\right|^2 \\ &= ||\mathbf{R}_{k-1}||^2 - 2\left(\frac{\langle \mathbf{R}_{k-1}, \phi(\mathbf{x})\rangle^2}{||\phi(\mathbf{x})||^2 + \lambda U_m(\phi)}\right) \\ &\quad + \left(\frac{\langle \mathbf{R}_{k-1}, \phi(\mathbf{x})\rangle}{||\phi(\mathbf{x})||^2 + \lambda U_m(\phi)}\right)^2 ||\phi(\mathbf{x})||^2 + \lambda U_m(\phi)\left(\frac{\langle \mathbf{R}_{k-1}, \phi(\mathbf{x})\rangle}{||\phi(\mathbf{x})||^2 + \lambda U_m(\phi)}\right)^2 \\ &= ||\mathbf{R}_{k-1}||^2 - \frac{\langle \mathbf{R}_{k-1}, \phi(\mathbf{x})\rangle^2}{||\phi(\mathbf{x})||^2 + \lambda U_m(\phi)}. \end{aligned} \tag{22}$$

Therefore in order to minimize equation 20,

$$\rho' = \left|\frac{\langle \mathbf{R}_{k-1}, \phi(\mathbf{x})\rangle}{\sqrt{||\phi(\mathbf{x})||^2 + \lambda U_m(\phi)}}\right|, \tag{23}$$

must be maximized.