Proceedings of the 24th IASTED International Multi-Conference
ARTIFICIAL INTELLIGENCE AND APPLICATIONS
February 13-16, 2006, Innsbruck, Austria

KERNEL REWARDS REGRESSION: AN INFORMATION EFFICIENT BATCH POLICY ITERATION APPROACH

Daniel Schneegaß and Steffen Udluft
Information & Communications, Learning Systems
Siemens AG, Corporate Technology
D-81739 Munich, Germany
{daniel.schneegass.ext,steffen.udluft}@siemens.com

Thomas Martinetz
Institute for Neuro- and Bioinformatics
University at Luebeck
D-23538 Luebeck, Germany
martinetz@informatik.uni-luebeck.de

ABSTRACT
We present the novel Kernel Rewards Regression (KRR) method for Policy Iteration in Reinforcement Learning on continuous state domains. Our method is able to obtain very useful policies observing just a few state action transitions. It considers the Reinforcement Learning problem as a regression task for which any appropriate technique may be applied. The use of kernel methods, e.g. the Support Vector Machine, enables the user to incorporate different types of structural prior knowledge about the state space by redefining the inner product. Furthermore KRR is a completely Off-policy method. The observations may be constructed by any sufficiently exploring policy, even the fully random one. We tested the algorithm on three typical Reinforcement Learning benchmarks. Moreover we give a proof for the correctness of our model and an error bound for estimating the Q-functions.

KEY WORDS
Machine Learning, Intelligent Control, Reinforcement Learning, Policy Iteration, Kernel-based Learning, Learning Theory

1 Introduction

Reinforcement Learning [1] is one of the most challenging current research topics in Learning Theory with many interesting open problems [2, 3, 4]. Reinforcement Learning concerns planning of how to move successfully within a given state space. In the general case the goal is to maximise a given objective function. Information about the objective function is collected in form of rewards during moving through this space.

Often in Data Mining and related applications one has got a fixed measured data set which needs to be analysed. It might be created by a random walk through the state space. In practice, many data sets contain just a few data points, because they are hard to simulate or expensive to measure. In this paper we therefore present a new batch learning method, which is able to deal very successfully with this kind of situation. We will show that for popular benchmark problems with short random walks very good results can be achieved. Our Kernel Rewards Regression (KRR) method tries to exploit as much information as possible. An advantage of our approach is however that any regression method can be used, so that we can profit from all the powerful methods in this field.

The use of kernel methods for Reinforcement Learning is already considered in earlier work. We especially want to mention the work of Ormoneit and Sen [5], and Dietterich and Wang [6]. Ormoneit and Sen used a smoothing kernel like the Gaussian and performed a kind of dynamic programming on the samples where some of them are supporting points and hence the kernel centers. The Value Function at a certain point within the state space is finally given as the weighted average of the Value Functions at the supporting points. Dietterich and Wang used an approach incorporating Support Vector models directly. They presented different possibilities to construct optimisation problems with appropriate constraints holding properties of typical Reinforcement Learning tasks such as the validity of the Bellman Equation.

Least Squares Policy Iteration was also considered in earlier work. Here we want to point out Lagoudakis and Parr [7, 8]. They presented a so-called LSQ method which is derived from the Least Squares Temporal Difference Learning [9] and combined it with a Policy Iteration. The latter is in turn derived from TD-Learning. But LSQ is restricted to finite state spaces and an unconstrained standard Least Squares regression method. Moreover the method does not converge to a stable solution in general and, most important, does not use the power of kernel methods.

In Supervised Learning kernels are a mighty tool to incorporate prior knowledge about the input space (e.g. invariances [10, 11]) in a variety of different types [12]. Corresponding tools can be used for the KRR method in a straightforward way. Additionally we will show an upper bound for the error of the Q-function estimated by our method. We tested our method on typical Reinforcement Learning benchmarks to validate its performance.

2 Reinforcement Learning

In Reinforcement Learning the main goal is to achieve a policy optimally moving an agent within an environment which is defined by a Markov Decision Process (MDP) [1]. It is generally given by a state space S, a set of actions A to choose in the different states, and the dynamics, e.g.

defined by a relation $T \subset S \times A \times S \times \mathbb{R} \times [0,1]$ depending on the current state, the chosen action, the successor state, a so-called reward R which the agent achieves while moving from current to successor state using the specified action, and the probability for reaching the successor state and achieving the certain reward given the current state and the action.

In most Reinforcement Learning tasks one is interested in maximising the discounting Value Function

$$V^\pi(s) = \mathbf{E}_\mathbf{s}^\pi \sum_{i=0}^{\infty} \gamma^i R(s^{(i)}, \pi(s^{(i)}), s^{(i+1)})$$

for all possible states s where $0 < \gamma < 1$ is the discount factor, s' the successor state of s, $\pi : S \to A$ the used policy, and $\mathbf{s} = \{s', s'', \ldots, s^{(i)}, \ldots\}$. R is the expected reward. Since the dynamics of the given state-action space cannot be modified by construction one has to maximise V over the policy space. Due to the fact that it is a very difficult task to achieve a good policy immediately by simulating the Markov Decision Process one usually takes one intermediate step and constructs a so-called Q-function depending on the current state and the chosen action holding

$$Q^\pi(s,a) = \mathbf{E}_{s'}(R(s,a,s') + \gamma Q^\pi(s', \pi(s'))).$$

By convention we define V as the Value Function of the optimal policy and Q respectively. These are the functions we want to estimate. They are defined as

$$\begin{aligned} Q(s,a) &= \mathbf{E}_{s'}(R(s,a,s') + \gamma V(s')) \\ &= \mathbf{E}_{s'}\left(R(s,a,s') + \gamma \max_{a'} Q(s',a')\right) \end{aligned}$$

which is called the Bellman Equation. Therefore the best policy is apparently the one using the action maximising the (best) Q-function, that is

$$\pi(s) = \arg\max_a Q(s,a).$$

For details we refer to Sutton and Barto [1]. This is a very general definition of the Reinforcement Learning problem. We use a slightly simplified version where we assume a discrete set of actions while the set of states remains continuous and the dynamics probabilistic.

3 Function Approximation with Ridge Regression and Support Vector Regression

In order to use a paradigm that incorporates prior knowledge about the structure of the state space we especially mention kernalized Ridge Regression [13, 14] and Support Vector Regression [15, 16, 17]. The trick is to redefine the inner product within the state space S in order to get a solvable linear regression task. So in Supervised Learning one is searching for the best configuration of coefficients α and bias b minimising a loss function, e.g.

$$\begin{aligned} L_{SL}(\alpha, b) &= \sum_{i=1}^{l} \|f_{\alpha,b}(\mathbf{x}_i) - y_i\|^2 \\ &= \min \end{aligned}$$

where

$$f_{\alpha,b}(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$$

with K as the kernel which is the modified inner product and l the number of observations containing input vectors $\mathbf{x}_i$ and output values y_i. Implicitly the weight vector $\mathbf{w} = \sum_{i=1}^{l} \alpha_i \phi(\mathbf{x}_i)$ with $K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$ performing a linear mapping within the feature space is given. The kernel has to be chosen appropriately and, as far as it is really an inner product, symmetric and positive definite. By redefining $K'(\mathbf{x}, \mathbf{z}) = K(\mathbf{x}, \mathbf{z}) + \frac{\delta_{\mathbf{x},\mathbf{z}}}{C}$, where δ is the Kronecker symbol, one gets a regularized Regression problem in order to minimise additionally $\|\mathbf{w}\|^2 = \sum_{i=1}^{l} \alpha_i \sum_{j=1}^{l} \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$. This leads to a flatter solution of the Least Squares problem given above. It has been proven to generalise better [15, 14]. Moreover in Support Vector Regression one defines additionally a tolerance error ϵ which leads to a more biased solution. We want to point out that there are many more linear regression methods based on kernels, e.g. with different error models. Any of them can be used for our task.

4 The Kernel Rewards Regression Approach

Finally we come to the main point of our approach where we basically do nothing more than connecting the Bellman Equation to any linear kernel regression method. As we use the sum of discounted rewards as achieved value we know that

$$\mathbf{E}_{s'} R(s,a,s') = Q(s,a) - \mathbf{E}_{s'} \gamma V(s').$$

The observed rewards r_i and successor states s_{i+1} are unbiased estimates for the expected ones using any sufficiently exploring policy including the fully random one with no strategic prior knowledge. Therefore we can use (s_i, a_i, s_{i+1}) as input variables and r_i as output variables for a Supervised Learning problem mapping $Q(s_i, a_i) - \gamma V(s_{i+1})$ on r_i.

Now we restrict the set of allowed Q-functions in a way that

$$Q(s,a) = \sum_{i=1}^{l} \alpha_i \hat{K}(s, a, s_i, a_i) + b_a$$

with action dependent biases b_a and

$$\hat{K}(s, a, \tilde{s}, \tilde{a}) = \delta_{a,\tilde{a}} K(s, \tilde{s}).$$

And thus

$$V(s) = \max_a \left(\sum_{i=1, a_i = a}^{l} \alpha_i K(s, s_i) + b_a \right).$$

As above the kernel K describes a transformation of the state space into an appropriate feature space. Note that from a state-action point of view the kernel $\hat{K}$ is constructed in a limited way. Each action spans its own subspace within the feature space. Of course $\hat{K}(s,a,\tilde{s},\tilde{a}) = \hat{K}'\left(\begin{pmatrix} s \\ a \end{pmatrix}, \begin{pmatrix} \tilde{s} \\ \tilde{a} \end{pmatrix}\right)$ remains symmetric and positive definite. Similar to [7, 8, 9] we now convert the Reinforcement Learning problem into a regression problem. All information we got for sure are the states, actions, and achieved rewards. We do not use any assumptions about the state-action-values. Due to the linearity of our model we obtain the rewards function as an estimate of the expected rewards

$$\begin{aligned} R(s_i, a_i, s_{i+1}) \quad = \quad & \sum_{j=1}^{l} \alpha_j \left(\delta_{a_i,a_j} K(s_i, s_j)\right. \\ & \left. - \gamma \delta_{a_{i+1,\max},a_j} K(s_{i+1}, s_j)\right) \\ & + b_{a_i} - \gamma b_{a_{i+1,\max}}, \end{aligned}$$

where $a_{i,\max}$ is the action reaching the maximal value on s_i. Suppose we already know these actions. Then the solution of a Least Squares regression problem is given as the α and $\mathbf{b}$ holding

$$\begin{aligned} L_{RL}(\alpha, \mathbf{b}) &= \left\| \mathbf{r} - (K' \quad \Delta) \begin{pmatrix} \alpha \\ \mathbf{b} \end{pmatrix} \right\| \\ &= \min \\ K'_{i,j} &= \delta_{a_i,a_j} K(s_i, s_j) \\ & \quad -\gamma \delta_{a_{i+1,\max},a_j} K(s_{i+1}, s_j) \\ \Delta_{i,j} &= \delta_{a_i,j} - \gamma \delta_{a_{i+1,\max},j}, j \in \{1, \ldots, |A|\}. \end{aligned}$$

Of course any more sophisticated regression solution [15, 16, 17, 18, 19] can be calculated as mentioned above. The attentive reader could remark that unfortunately the constructed kernel K', even if it is regularized, is no more symmetric and positive definite in general. Therefore the KRR input space cannot be interpreted as an inner product space in contrast to the one of the Q-function, and thus local minima might exist as for instance in Support Vector Regression. But e.g. the Sigmoid Kernel [16] and the Tangent Distance Kernel [11] are not positive definite as well [20, 21] and are often used for Support Vector Machine Learning. Additionally we want to mention e.g. the work of Ong et al. [22], Mangasarian and Musicant [23], and Guo and Schuurmans [21] who presented foundations on and different methods using indefinite kernels.

4.1 Interpretation

It is most important to point out, that the presented method only uses the states and actions observed as input variables and the rewards as output variables. The hypothesis space for R is constructed in such a way that the Q-function can be obtained by decomposition of R and holds the Bellman Equation as constraint. It is not necessary to integrate over any probabilities and make suppositions about it like in Planning and Learning methods. Moreover we do not need interim solutions of the Q-function, which will eventually move the final solution in a direction dependent on the initialization like in Q-Learning.

Our approach differs from Ormoneit and Sen's [5] in many aspects. E.g. the factors α are degrees of freedom in KRR while their work set α_i to an estimate of $Q(s_i, a_i)$. They are hence limited to distance based kernels. The method uses kernels just as smoothing functions to consider the continuity of the state space. In the Support Vector philosophy, which we promote, kernels are any functions transforming a nonlinear problem into a linear one and therefore simplifying it.

The work of Dietterich and Wang [6] requires full knowledge about the concerned policy. They look for the optimal Support Vector representation of the policy. However, from a certain perspective their work has the most similarities to ours because they combine Reinforcement Learning requirements with Support Vector Machine optimisation techniques in an immediate way.

Lagoudakis and Parr [7, 8] actually solve another, but similar regression problem (also within the feature space) which is restricted to a special type of the standard Least Squares approach. Nevertheless, their algorithm iterates in a comparable way, but does not profit from the advantages offered by kernels. However, our KRR can be seen as an extension and combination of the work of Ormoneit and Sen as well as Lagoudakis and Parr.

4.2 Upper Bound on the Error of Q

Due to the fact that only the correct Q-function holds the Bellman Equation, the correct estimate for the rewards of course leads to the correct estimate of the Q-function [9]. But if we have a small error on the rewards regression, then this could lead to a Q-function with an error obtaining a completely different policy. It is indeed not apparent that the error of the Q-function remains bounded.

Theorem 1 *Let S be a compact set of states and A a finite set of actions. Let further be R, Q the real expected rewards and Q-function, and $\hat{R}$, $\hat{Q}$ our estimates of them which satisfy*

$$\hat{R}(s, a, s') \quad = \quad \hat{Q}(s, a) - \gamma \max_{a'} \hat{Q}(s', a')$$

for all $s, s' \in S$ and $a \in A$, $0 < \gamma < 1$ the discount factor. Then the following holds true. If for all $s, s' \in S$ and $a \in A$ the error

$$|R(s, a, s') - \hat{R}(s, a, s')| \quad < \quad \epsilon$$

is bounded, then the relation

$$|Q(s, a) - \hat{Q}(s, a)| \quad < \quad \frac{\epsilon}{1 - \gamma}$$

holds true as well.

Proof: Suppose the converse of the claim. Then there exists at least one (s^*, a^*) such that

$$\frac{\epsilon}{1-\gamma} \quad \leq \quad |(Q-\hat{Q})(s^*, a^*)|.$$

Due to the validity of the Bellman Equation, for Q, R and by construction for $\hat{Q}$, $\hat{R}$, there is further at least one (s, a) which maximises this error of the Q-function, that is

$$\begin{aligned} \frac{\epsilon}{1-\gamma} + c &= |(Q-\hat{Q})(s,a)| \\ &= |\mathbf{E}_{\hat{s}}((R-\hat{R})(s,a,\hat{s}) + \gamma(V-\hat{V})(\hat{s}))| \end{aligned}$$

with $c > 0$. Then there exists at least one s' such that

$$\frac{\epsilon}{1-\gamma} + c \quad \leq \quad |(R-\hat{R})(s,a,s') + \gamma(V-\hat{V})(s')|.$$

Moreover from $|(V-\hat{V})(s')| = \beta$ it follows $\exists a' \in A : |(Q-\hat{Q})(s',a')| \geq \beta$. Therefore we obtain immediately

$$\begin{aligned} \frac{\epsilon}{1-\gamma} + c &\leq |(R-\hat{R})(s,a,s') + \gamma(Q-\hat{Q})(s',a')| \\ &\leq |(R-\hat{R})(s,a,s')| + \gamma|(Q-\hat{Q})(s',a')| \\ &< \epsilon + \gamma\left(\frac{\epsilon}{1-\gamma} + c\right) \\ &= \frac{\epsilon}{1-\gamma} + \gamma c. \end{aligned}$$

This is the contradiction and proves the theorem.

□

And thus any error bounds, both theoretical ones from Learning Theory and practical ones, e.g. by Cross Validation on the rewards, lead immediately to an error bound for the Q-function using any approach fulfilling the Bellman Equation. Because of the kernel structure of the KRR method all theoretical bounds for e.g. kernalized Ridge Regression and Support Vector Regression can be applied.

Furthermore, we want to point out that theorem 1 is not limited to linear methods for regression. Any regression method that fulfills the structure of the Bellman Equation can be used to estimate the rewards and implicitly the Q-function.

5 The Algorithm

As given above one uses $a_{i,\max}$ to solve the KRR problem which one does usually not know. Hence we define the function

$$\begin{aligned} F_{K,\mathbf{r},\rho}(\mathbf{z},h) &= (H(g,\mathbf{z},h), \rho h), 0 < \rho < 1 \\ g &= \arg\max_{\mathbf{u}} \|(K^{\mathbf{u}} A(M,\mathbf{r}) + \mathbf{b_u}(M,\mathbf{r}))\| \\ M &= \big(\delta_{a_i,a_j} K(s_i,s_j) \\ &\quad -\gamma \mathbf{z}_{i+1,a_j} K(s_{i+1},s_j)\big)_{l,l} \end{aligned}$$

as before. $A(K',\mathbf{r}) = \alpha$ and $\mathbf{b_u} = (b_{u_1} \quad \cdots \quad b_{u_l})^T$ solve the regression task. The fixed point of F, for which $F(\mathbf{z}) = (\mathbf{z}^*, 0)$ with $\mathbf{z}^*_{i,j} = 1$ for $\mathbf{a}_{i,\max} = j$ and $\mathbf{z}^*_{i,j} = 0$ otherwise leads to the correct solution.

Algorithm 1 The Reinforcement Learning Kernel Rewards Regression Algorithm

```
Require: given set of transitions $T = \{(s_1,a_1,s_2,r_1),\ldots,(s_l,a_l,s_{l+1},r_l)\}$, kernel $K$, regression parameters (e.g. $C$ and $\epsilon$), discount factor $0 < \gamma < 1$, and action change factor $0 < \rho \leq 1$
Ensure: calculates a policy choosing best actions for the Reinforcement Learning problem given by the representation $T$
  set $\mathbf{z} \in \mathbb{R}^{l \times |A|}$ randomly constrained by $\forall i : \sum_{j=1}^{|A|} \mathbf{z}_{i,j} = 1$
  set $t \leftarrow 0$
  while the desired precision is not reached do
    set $t \leftarrow t+1$
    set $\forall i,j : K'_{i,j} = \delta_{a_i,a_j} K(s_i,s_j) - \gamma \mathbf{z}_{i+1,a_j} K(s_{i+1},s_j)$
    solve the regression task $K'\alpha = \mathbf{y}$ w.r.t. the regression parameters (e.g. $C$ and $\epsilon$)
    find $\forall i : \mathbf{u}_{i,\max} = \arg\max_a \sum_{j=1}^{l} \delta_{a,a_j} K(s_i,s_j)$
    set $\forall i : \mathbf{z}_{i,\mathbf{u}_{i,\max}} = \mathbf{z}_{i,\mathbf{u}_{i,\max}} + \rho^t(1 - \mathbf{z}_{i,\mathbf{u}_{i,\max}})$
    set $\forall i, j \neq \mathbf{u}_{i,\max} : \mathbf{z}_{i,j} = \mathbf{z}_{i,j}\left(1 - \rho^t \left(\sum_{k=1}^{|A|} \mathbf{z}_{i,k}\right)^{-1}\right)$
  end while
  set $\pi(s) = \arg\max_{a'} \sum_{i=1}^{l} \alpha_i \delta_{a',a_i} K(s,s_i)$
```

6 Benchmarks

We compared the results of our method with the results of the random and the optimal policies, which were obtained by human analysis (modified Mountain-Car, Pole-Balancing [1]) or Genetic algorithms with millions of observations (Wet-Chicken [25]). In the Pole-Balancing problem we additionally included results of a variant of the Adaptive Heuristic Critic (AHC) [1] which is highly tuned for this special problem and hence works very well. Furthermore the AHC benefits from its own online exploration. For a comparison using random exploration we also show results achieved by Prioritized Sweeping. One can see that the results are similar. For our benchmarks we used kernalized Ridge Regression together with different types of kernels.

- Polynomial Kernel: $K_p(\mathbf{x},\mathbf{z}) = (\mathbf{x}^T\mathbf{z} + 1)^p$
- Gaussian Kernel: $K_\sigma(\mathbf{x},\mathbf{z}) = e^{\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}}$
- Linear Limits Kernel: $K_{\text{limits}}(\mathbf{x},\mathbf{z}) = \phi_{\text{limits}}(\mathbf{x})^T \phi_{\text{limits}}(\mathbf{z})$ with $\phi_{\text{limits}}(\mathbf{x}) = \big(|\mathbf{x}| \quad |\mathbf{x} - \text{limits}_1| \quad \cdots \quad |\mathbf{x} - \text{limits}_{\dim(\text{limits})}|\big)^T$

We used the Gaussian and Polynomial Kernel as standard variants in Kernel Machines. The parameters were each preselected solving smaller sized problems. We chose especially the Linear Limits Kernel because of our knowledge about the Q-Functions structure. As it is often the case in Reinforcement Learning the policies are fairly simple and unspectacular on a wide range of states, whereas within certain areas the optimal actions change frequently. In these regions nodes limits_i are set with higher density. So we are able to keep the VC-Dimension small, but take the different attractivities of certain parts of the state space into account. As can be seen below, the good results, however, are comparable to the ones using the Gaussian Kernel with appropriate variances.

We have tested our method using different popular Reinforcement Learning benchmarks. For details we refer to the literature [25, 1]. The results are each averaged over several trials and are given together with its error analysis. It can be seen that very few observations suffice to achieve quite good policies, in almost every setting with upward tendency for an increasing number of observations.

Table 1. Results for the Wet-Chicken problem [25]

Policy/Kernel	**Samples**	**Average Reward**	**Relative to Optimum**
Random		**7.2**	**41 %**
Gaussian	250	12.3 ± 0.3	$70.7 \pm 1.6\%$
($\sigma = 1$)	500	12.5 ± 0.2	$71.7 \pm 1.1\%$
	1000	13.6 ± 0.3	$78.0 \pm 1.5\%$
Gaussian	250	13.5 ± 0.4	$77.4 \pm 1.6\%$
($\sigma = 2.5$)	500	13.6 ± 0.4	$78 \pm 2\%$
	1000	13.9 ± 0.3	$80 \pm 2\%$
Gaussian	250	12.6 ± 0.4	$72 \pm 2\%$
($\sigma = 5$)	500	12.8 ± 0.3	$73 \pm 2\%$
	1000	13.3 ± 0.4	$76 \pm 2\%$
Linear Limits	250	12.3 ± 0.3	$71 \pm 2\%$
	500	12.5 ± 0.2	$72.0 \pm 1.2\%$
	1000	13.5 ± 0.3	$77.3 \pm 1.5\%$
Optimal		**17.4**	**100 %**

Table 2. Results for the Mountain-Car problem [1]

Policy/Kernel	**Samples**	**Steps until Crest**	**Relative to Optimum**
Random		**100**	**24 %**
Gaussian	250	59 ± 5	$41 \pm 4\%$
($\sigma = 0.05$)	500	57 ± 6	$42 \pm 4\%$
	1000	41 ± 6	$65 \pm 5\%$
Gaussian	250	29.4 ± 1.1	$82 \pm 3\%$
($\sigma = 0.25$)	500	26.1 ± 0.6	$92 \pm 2\%$
	1000	25.8 ± 0.6	$93 \pm 2\%$
Polynomial	250	41 ± 2	$58 \pm 3\%$
($p = 4$)	500	35.2 ± 1.2	$68 \pm 2\%$
	1000	32.2 ± 1.0	$75 \pm 2\%$
Polynomial	250	38 ± 2	$63 \pm 3\%$
($p = 5$)	500	33.7 ± 1.1	$71 \pm 2\%$
	1000	26.1 ± 0.6	$92 \pm 2\%$
Linear Limits	250	46 ± 3	$53 \pm 3\%$
	500	42 ± 2	$58 \pm 3\%$
	1000	39 ± 2	$62 \pm 4\%$
Optimal		**24**	**100 %**

Table 3. Results for the Pole-Balancing problem [1]

Policy/Kernel	**Samples**	**Steps until Failure**
Random		**22**
Gaussian	250	41 ± 5
($\sigma = 0.25$)	500	74 ± 6
	1000	90 ± 7
Polynomial	250	22 ± 2
($p = 4$)	500	35 ± 6
	1000	30 ± 4
Polynomial	250	22 ± 2
($p = 5$)	500	28 ± 3
	1000	36 ± 5
Linear Limits	250	58 ± 10
	500	89 ± 8
	1000	145 ± 11
Prioritized Sweeping	250	30
(Random Exploration)	500	58
	1000	82
AHC	250	53
(Boltzmann Exploration)	500	97
	1000	155
Optimal		∞

7 Conclusion

The main goal of our current research is to build up Reinforcement Learning methods using the existing information as efficient as possible. In an intrinsic way we construct the Q-function as linear in the samples within an appropriate feature space using the kernel trick. Therefore it is possible to incorporate structural prior knowledge about the state space. On the other hand, no strategic, thus policy-concerned prior knowledge is necessary. We proved the correctness of our model and presented an algorithm that works well, even on a few observed samples. Still there is future work to do. The fixed point iteration is constructed such that convergence is always guaranteed, at least to any point. The convergence proof with an asymptotically slower decreasing h is still open. Furthermore we want to combine an iterative regression method with the fixed point iteration over the solutions and the maximal actions. In each step the maximal actions could be recalculated much faster and the two iterations could support each other. Such incremental Policy Iteration could be based on the SMO algorithm [26] or the MMO approach [27] for Support Vector Machines. Such an algorithm could be used online and hence efficiently as On Policy method. Moreover an appropriate exploration strategy has to be discovered.

Acknowledgments

We are very grateful to Bernhard Lang, Peter Mayer, Anton Schaefer, Volkmar Sterzing, and Volker Tresp for fruitful discussions and useful comments during the preparation of this paper.

References

[1] Richard S. Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, 1998.

[2] Richard S. Sutton. Open theoretical questions in reinforcement learning. In *EuroCOLT*, pages 11–17, 1999.

[3] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. In *Discrete Event Dynamic Systems*, 2003.

[4] Anton Schaefer and Steffen Udluft. Solving partially observable reinforcement learning problems with recurrent neural networks. In *Workshop Proc. of the European Conference on Machine Learning*, 2005.

[5] D. Ormoneit and S. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2-3):161–178, 2002.

[6] T. Dietterich and X. Wang. Batch value function approximation via support vectors. In *NIPS*, pages 1491–1498, 2001.

[7] Michail G. Lagoudakis and Ronald Parr. Model-free least-squares policy iteration. In *Advances in Neural Information Processing Systems*, volume 14, 2002.

[8] Michail G. Lagoudakis, Ronald Parr, and Michael L. Littman. Least-squares methods in reinforcement learning for control. In *SETN*, pages 249–260, 2002.

[9] Justin A. Boyan. Least-squares temporal difference learning. In *I. Bratko and S. Dzeroski, editors, Machine Learning: Proceedings of the Sixteenth International Conference*, volume 14, pages 49–56. Morgan Kaufmann, San Francisco, CA, 1999.

[10] O. Chapelle and B. Schoelkopf. Incorporating invariances in non-linear support vector machines. In *T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, Advances in Neural Information Processing Systems*, volume 14, pages 609–616. MIT Press, Cambridge, MA, 2002.

[11] B. Haasdonk and D. Keysers. Tangent distance kernels for support vector machines. In *Proceedings of the 16th ICPR*, pages 864–868, 2002.

[12] Bernhard Schoelkopf, Patrice Simard, Alex J. Smola, and Vladimir Vapnik. Prior knowledge in support vector kernels. In *NIPS*, 1997.

[13] AE Hoerl and RW Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, pages 55–68, 1970.

[14] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements Of Statistical Learning Theory, Data Mining, Inference, and Prediction*. Springer, New York, 2001.

[15] Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., New York, 1998.

[16] Nello Cristianini and John Shawe-Taylor. *Support Vector Machines And Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, 2000.

[17] Alex J. Smola and Bernhard Schoelkopf. A tutorial on support vector regression. *NeuroCOLT2 Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK*, 1998.

[18] Volker Tresp. Scaling kernel-based systems to large data sets. *Data Mining and Knowledge Discovery*, 2001.

[19] Volker Tresp. Mixtures of gaussian processes. In *NIPS*, volume 13, 2000.

[20] Matthias Hein and Olivier Bousquet. Kernels, associated structures and generalizations. *Max-Planck-Institut fuer biologische Kybernetik, Technical Report*, 2004.

[21] Yuhong Guo and Dale Schuurmans. Support vector machines on general confidence functions. *Department of Computing Science, University of Alberta, Edmonton, Canada, Technical Report*, 2005.

[22] C.S. Ong, X. Mary, S. Canu, and A.J. Smola. Learning with non-positive kernels. In *Proceedings of the 21st International Conference on Machine Learning*, pages 639–646, 2004.

[23] O. L. Mangasarian and David R.. Musicant. Data discrimination via nonlinear generalized support vector machines. *Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, Technical Report 99-03*, 1999.

[24] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proc. 16th Intl. Conf. on Machine Learning*, pages 278–287, 1999.

[25] Volker Tresp. The wet game of chicken. *Siemens AG, CT IC 4, Technical Report*, 1994.

[26] John C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods: support vector learning*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[27] T. Martinetz. Maxminover: A simple incremental learning procedure for support vector classification. *Proc. of the International Joint Conference on Neural Networks (IEEE Press)*, pages 2065–2070, 2004.