Reinforcement Learning in Latent Space

Duo Li
Department of Computer Science
University of British Columbia
duoli@cs.ubc.ca

Abstract

We presents a new strategy to efficiently eliminate *the curse of dimensionality* problem for reinforcement learning. Instead of pruning the full dimensional searching space via sampling or approximation, we directly conduct reinforcement learning in a low dimensional latent space from a group of prior data via Gaussian Process Latent Variable Model (GPLVM), with an emphasis on continuity. Not only is this latent space able to represent existed full dimensional data with little errors, but it also provides a tool to synthesize new possible states. This characteristic makes applying reinforcement learning in such space become feasible. In this paper, we describe how to build an appropriate latent space, and how to embed this latent space into traditional reinforcement learning procedure. We also test our algorithm on a punching planning problem which contains up to 62 Degree of Freedoms (DoFs) for one state. Our experiment shows that such high dimensionality reinforcement learning problem can be solved in a short time with our approach.

1 Introduction

Reinforcement Learning (RL) has been widely studied in many areas such as robotics and computer animation due to its generality. However, RL in real world applications is to deal with high-dimensional state-action pairs, that immediately incurs *the curse of dimensionality*, in which computational cost increases exponentially with respect to the number of state-action variables. For example, suppose we represent a state with n features and each feature contains m discretized actions, the state-action pair will be up to m^n, which makes searching impractical. The situation turns out to be even worse if we consider a continuous action space for each features. To deal with this problem, strategies such as GP-UCB[1][2] proposed to exploit Gaussian Process Regression (GPR) for evaluating the expected reward for each step, and pruning low reward branches. However, such pruning behavior still works in full dimensional space, which is not suitable for multi-variable states. As an alternative, hierarchical RL[3] exploits a divide-and-conquer strategy to split the action space into different levels, but suffers from difficulties in option selections.

Rather than focusing on the full dimensional state space, we find out that generally most states can be divided into or be approximated with certain types once we have sufficient prior data. Take hand grasping as an example, although hand needs to be modeled with at least 26 variables, all behavior for grasping can be viewed as loop-closing of each finger. If we have sufficient sorts of loop-closing motions sequences for each finger, only a small group of parameters are enough for synthesizing the whole grasping behavior. This synthesis strategy is known as motion graph[4]. With this technique, we can conduct our learning process on this level instead of full dimensional data. However, in most cases, gathering all possible types of motion sequences is difficult if not impossible. Therefore, we need to build up a similar latent space with less data.

As a non-linear dimensionality reduction technique, GPLVM[5] offers an efficient tool to represent high dimensional data via pretty low dimensional manifold with little reconstruction errors. Usually

a 2 or 3 dimensional space is enough for expression. Moreover, we can even synthesize new states in this latent space to meet user-specific constrains. [6] exploit this characteristic of GPLVM to handle inverse kinematics problem which conduct an optimization process on latent space to synthesize new plausible poses. Our work most closes to their style-based inverse kinematics strategy. Nevertheless, their optimization strategy is hard to be directly applied into reinforcement learning due to the frequent interaction between agent and environment in RL. In other words, given an uncertain environment, attempting every possible state in data space with an optimization may cost too much time. Instead, we propose to directly search possible states in the latent space.

Nevertheless, our new strategy demands a strong continuity both for the mapping from latent space to data space, as well as the mapping from data space to latent space. Therefore, standard GPLVM is not suitable for this task because it only maintains the dissimilarity from data space to latent space. Thanks to the developing of GPLVM, recent Gaussian Process Dynamical Mode (GPDM)[8][9] serves as a reasonable choice for our task. Of course, one can exploits other branches of GPLVM-based algorithms such as back constraint technique[7] or topological persevered approach[11] to reinforce the continuity of mapping. For the sake of generality, we name our framework as constrained GPLVM-based RL.

This paper is organized as follows: first, we provide some background on RL and specifically on SARSA algorithm (Section 2); after outlining the main idea about GPLVM (Section 3.1) and discussing how to construct a suitable latent space for searching with an emphasis on continuity (Section 3.2), we describe how to directly incorporate GPLVM into RL (Section 4); finally, we verify our new strategy with a punching planning experiment (Section 5), and discuss open questions as well as potential extensions of this approach (Section 6).

2 Reinforcement Learning

Usually, underlying RL problem can be considered as a Markov Decision Process (MDP). An MDP is defined as a 5-tuple (S, A, P, R, γ) where $S = \{s_1, \cdots, s_n\}$ is a finite set of states; $A = \{a_1, \cdots, a_n\}$ is finite set of actions; P stands for the probability of making a transition to state s' via action a in state s; R describes the scalar reward value by taking action a in state s into state s'; and $\gamma \in [0, 1)$ is the discount factor for future rewards. Unlike supervised learning that exploits prior input/output pairs to predict new data, instead, RL focus on achieving satisfied task via maximizing accumulated rewards. Introducing a policy function $\pi(s) : S \to A$ to map action a in state s, the over goal for RL lies in find such a policy π to maximize discount return:

$$\pi = \arg\max V^{\pi}(s_0),$$

where

$$V^{\pi}(s) = \sum_{i=0}^{\infty} \gamma^i r_{i+1}$$

is the state value function for state s according to policy π, and $r_{i+1} \in R$ is the reward value gathered from environment int time step i+1. Sometimes, V is hard to be obtained directly, but we still want to choose actions this way. Define a state-action function $Q(s, a)$ that

$$Q^{\pi}(s, a) = r(s, a) + \gamma V^{\pi}(s),$$

we can find policy $\pi_{m+1}(s)$ that

$$\pi_{m+1}(s) = \arg\max_{a \in A} Q^{\pi_m}(s, a)$$

with a greedy policy instead.

Lots of methods(value iteration, policy iteration, policy search, etc.[13]) can be applied to search for the optimal policy. In this paper, we are specific on a popular approach named $SARSA(\lambda)$ due to its simplicity and good convergence behavior. Essentially, $SARSA(\lambda)$ combines eligibility traces as TD(λ) algorithm with classical $SARSA$ algorithm to update state-action pairs. The $SARSA(\lambda)$ algorithm is summarized in Figure 1. Details about this algorithm can be found in [13].

Note that, although we restrict ourselves with $SARSA(\lambda)$ in this paper, indeed, all other basic RL approaches as well as advanced improving methods can replace $SARSA(\lambda)$ in our GPLVM-base RL framework.

$SARSA(\lambda)$
Initialize $Q(s,a)$ arbitrarily, $e(s,a)$ = 0 for all s,a
Repeat until maximum episodes:
Initialize s,a
Repeat until maximum steps
Take action a, observe reward r and next state s
Choose a' from s using policy derived from Q (ϵ-greedy)
$\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
$e(s,a) \leftarrow e(s,a) + 1$
For all s,a:
$Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
$e(s,a) \leftarrow \gamma\lambda e(s,a)$
$s \leftarrow s'$, $a \leftarrow a'$
break if s meets terminal

Figure 1: $SARSA(\lambda)$ Algorithm

As above introduction, the advantage of RL lies in its low demands of assumptions and prior knowledge about the problem. Nevertheless, RL also 'pays a price' for its generality. As discussed before, for high dimension problems, the number of state-action pairs becomes too large to be tractable. Therefore, we seek to project the problem into a subspace in which 1) the state only contains a few variables, and 2) the candidate action can be easily obtained. Once achieved these two tasks, we can easily solve the RL problem equivalently.

3 Construction of Latent Space

Now we consider how to construct a suitable latent space for state-action searching.

3.1 Gaussian Process Latent Variable Model

The simplest way to achieve dimensionality reduction is to represent the data in a linear subspace of the observation space. Probabilistic principal components analysis (PPCA) linearly maps data set $\mathbf{Y} = [\mathbf{y_1}, \cdots, \mathbf{y_N}]^\mathbf{T}$ into latent variables $\mathbf{X} = [\mathbf{x_1}, \cdots, \mathbf{x_N}]^\mathbf{T}$ as

$$\mathbf{y_n} = \mathbf{W}\mathbf{x_n} + \eta,$$

where the mapping is given by weight coefficient matrix $\mathbf{W} \in \mathbb{R}^{D \times q}$ with D the dimension of data space and q the dimension of the latent space, as well as a noise term η. Generally, the noise is taken to be Gaussian distributed that,

$$p(\eta|\beta) = \mathbf{N}(\eta|\mathbf{0}, \beta^{-1}\mathbf{I}).$$

Suppose independence across data points, the conditional probability of the $\mathbf{Y}$ can be written as

$$p(\mathbf{Y}|\mathbf{W}, \mathbf{X}, \beta) = \prod_{\mathbf{n=1}}^{\mathbf{N}} \mathbf{N}(\mathbf{y_n}|\mathbf{W}\mathbf{x_n}, \beta^{-1}\mathbf{I}).$$

Instead of denoting a prior distribution over latent space, $p(\mathbf{X})$ and seek to marginalize out X, we can consider a dual form of PPCA that appoint a Gaussian distribution of weights,

$$p(\mathbf{W}) = \prod_{\mathbf{n=1}}^{\mathbf{D}} \mathbf{N}(\mathbf{w_{i,:}}|\mathbf{0}, \mathbf{I}),$$

therefore, by marginalizing out $\mathbf{W}$

$$p(\mathbf{Y}|\mathbf{X}, \beta) = \int \prod_{\mathbf{n=1}}^{\mathbf{D}} \mathbf{p}(\mathbf{y_n}|\mathbf{x_n}, \mathbf{W}, \beta)\mathbf{p}(\mathbf{W})\mathbf{dW},$$

we get

$$p(\mathbf{Y}|\mathbf{X},\beta) = \frac{\mathbf{1}}{(\mathit{2}\pi)^{\frac{DN}{2}}|\mathbf{K}|^{\frac{\mathbf{D}}{2}}}\exp(-\frac{\mathbf{1}}{\mathbf{2}}\mathbf{tr}(\mathbf{K^{-1}YY^T})), \tag{1}$$

where covariance matrix $\mathbf{K} = \mathbf{XX^T} + \beta^{-1}\mathbf{I}$. With the perspective of Gaussian Process, we can write equation 1 as a a product of D independent Gaussian process as

$$p(\mathbf{Y}|\mathbf{X},\beta) = \prod_{\mathbf{i=1}}^{\mathbf{D}} \frac{\mathbf{1}}{(\mathit{2}\pi)^{\frac{N}{2}}|\mathbf{K}|^{\frac{\mathbf{1}}{\mathbf{2}}}}\exp(-\frac{\mathbf{1}}{\mathbf{2}}(\mathbf{y}_{:,\mathbf{i}}^{\mathbf{T}}\mathbf{K^{-1}y}_{:,\mathbf{i}})). \tag{2}$$

However, linear covariance kernels are always unable to catch the non-liner characteristic of data. Instead, we can replace it as non-linear kernel such as RBF kernel that

$$k(\mathbf{x_i},\mathbf{x_j}) = \alpha\exp(-\frac{\gamma}{\mathit{2}}(\mathbf{x_i}-\mathbf{x_j})^{\mathbf{T}}(\mathbf{x_i}-\mathbf{x_j})) + \beta^{-1}\delta_{\mathbf{ij}},$$

where δ_{ij} is delta function.

Consider equation 2 as the likelihood for $\mathbf{Y}$ over $\mathbf{X}$, we can obtain $\mathbf{X}$ with Maximum Likelihood Estimation (MLE), which minimizes the corresponding negative log-likelihood L by making its gradient $\frac{\partial L}{\partial \mathbf{X}}$ close to zero.This can be achieved with chain rule that

$$\frac{\partial L}{\partial \mathbf{K}} = \mathbf{K^{-1}YY^TK^{-1}} - D\mathbf{K^{-1}}$$
$$\frac{\partial \mathbf{k}(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}} = -\gamma(\mathbf{x}-\mathbf{x}')\mathbf{k}(\mathbf{x},\mathbf{x}')$$

Besides, followed Gaussian Process, for each new point $\mathbf{x}^*$in latent space, we are also able to predict its corresponding mapping in data space with a mean value μ, as well as a variance value σ^2 for describing its confidence via posterior.

This is known as GPLVM, which provides a powerful tool to construct a low-dimensional state space to fulfill task 1.

3.2 Continuous Mapping

Given a motion sequence of punching, GPLVM reduces these 62 DoFs states into a 2 dimensional space as Figure 2. Notice that we can synthesize poses(left-upper pose) which never show up in prior dataset. However, even similar poses (e.g. the two poses in right side) may locate far away from each other with standard GPLVM reduction. This is because GPLVM only maintains dissimilarity from data space to latent space. Far part points in data space keep far part in latent space, but it is not necessarily vice versa. The situation becomes even worse when we add different patterns of motions(figure 3, left). In a word, this discontinuity demands our RL to either skip in the latent space or repeatedly visit similar poses, which all makes our framework inefficient or even unworkable.

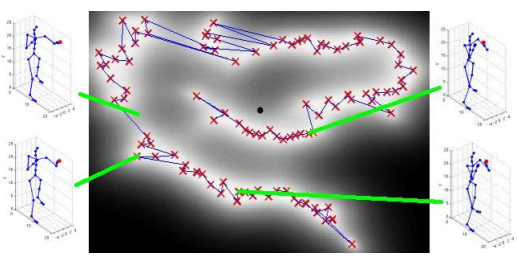

Figure 2: The latent space for 3 punching motion sequences(100 frames) with standard GPLVM. Red cross is the prior data points. Blue line shows the temporal order of data points. Lighter color stands for higher variances and thus higher confidence.

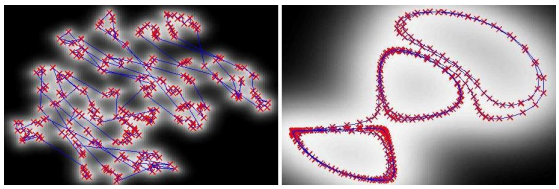

Figure 3: The latent spaces for 30 punching motion sequences(1200 frames) which can be categorized into 3 patterns. The left figure adopts standard GPLVM, the right one exploits GPDM. GPLVM projects the data as scatter subsequences. In contrast, GPDM successfully projects them into three circles.

Instead, we exploit a revised version of GPLVM named Gaussian Process Dynamical Model (GPDM)[8][9], which introduces a dynamical model as prior into GPLVM. GPDM uses two mappings from $\mathbf{X}$ to $\mathbf{Y}$,

$$\begin{aligned}\mathbf{x_t} &= f(\mathbf{x_{t-1}}; \mathbf{A}) + \mathbf{n_{x,t}},\\ \mathbf{y_t} &= g(\mathbf{x_t}; \mathbf{B}) + \mathbf{n_{y,t}},\end{aligned}$$

where $\mathbf{A}, \mathbf{B}$ are weights and $\mathbf{x_{t-1}}$ is the previous state for $\mathbf{x_t}$ with respect to time.

Moreover, GPDM denotes prior distribution of $\mathbf{X}$ as $p(\mathbf{X}|\alpha) = p(\mathbf{x_1}) \prod_{\mathbf{n=2}}^{\mathbf{N}} p(\mathbf{x_n}|\mathbf{x_{n-1}})$ according to Markov chain. It also assumes a Gaussian distribution over $p(\mathbf{x_n}|\mathbf{x_{n-1}})$. Rather than solving an intractable likelihood of $\mathbf{Y}$ as GPLVM, GPDM maximizes a posterior for X. The negative log-posterior is

$$\begin{aligned}L &= -\log p(\mathbf{X}, \alpha, \beta|\mathbf{Y})\\ &= -\log p(\mathbf{Y}|\mathbf{X}, \beta)\mathbf{p}(\mathbf{X}|\alpha)\mathbf{p}(\alpha)\mathbf{p}(\beta),\end{aligned}$$

where α, β are hyperparameters for $\mathbf{Y}$ and $\mathbf{X}$. Then we can calculate $\mathbf{X}$ by minimizing L. More details about GPDM can be found in [8][9].As an alternative, GPLVM with back-constraints[7] can achieve similar effects as GPDM for continuous mapping.

In sum, with GPDM, we are able to synthesize a group of potential new poses in a continuous latent space, which is suitable for Reinforcement Learning(figure 5).

4 Reinforcement Learning in Latent Space

After gathered all ingredients, we outline our constrained GPLVM-based RL framework in figure 4.

cGPLVM-RL (Y, R, q)
Preprocessing
Standardize Y
Construct continuous latent space X with GPLVM
Calculate predictions $Y^{'}$ for searching area $X^{'}$
Choose initial state x_0
Run-time
Repeat until achieve maximum trials
Repeat until maximum steps
Arrive state x via action a
Evaluate reward value with respect to y, the prediction of x
Update as classical chosen RL algorithm
Terminal if task q has been achieved
Select the best trace X^* with respect to rewards
Return corresponding predictions Y^*

Figure 4: constrained GPLVM-based Reinforcement Learning

Y is the prior expert data, R is a callback function about reward, and q stands for the desirable goal of this task. Specifically, we exploits GPDM and $SARSA(\lambda)$ in this paper. But one can choose any other RL methods to replace $SARSA$ in this framework; moreover, other GPLVM-based reduction algorithm can be used to replace GPDM once it is able to maintain the dual continuous mapping between observation space and latent space.

5 Experiments

We prove our constrained GPLVM-RL framework with 2 experiments for solving a punching planning problem, which needs the agent to punch specific target with left fist and avoid obstacle if necessarily. We model the human with 31 joints and 62 variables including the roots position and joint angles (figure 2, figure 5). Actually, even a simple punching motion demands the coordinating of each parts of the whole body (e.g. in figure 5). If one wants to punch a higher point, he/she needs to wriggle the waist and then swift the arm. Instead, for punching lower point, he/she has to first crouch and then swift the fist. In a word, punching planning needs the cooperation of the whole body, which is hard to be solved with traditional RL method due to its high dimensionality.

5.1 Experiment 1

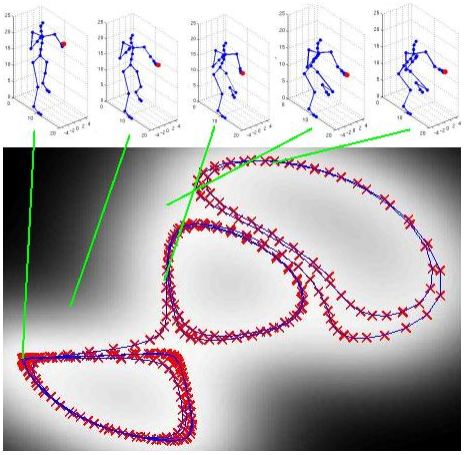

Figure 5: Three types of punching sequences have been mapped into latent space continuously via GPDM. A wide range of new poses can be easily modeled within this space.

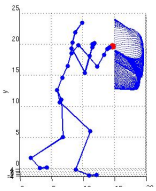

Figure 6: Reachable punching area in latent space are drawn as blue mesh in the right side. Note that we only record points with z values larger than 15 in this mesh.

In experiment 1, we show how to construct a suitable latent space for RL. Given 1200 punching frames with approximately 3 patterns, GPDM projects all these sequences into approximately 3 circles in a 2-dimensional space (figure 5). This means similar motions have been projected into close points in the latent space, and we can search for nearby full-dimensional states locally in this latent space. Moreover, although the prior dataset only contains several reachable end-points, we can synthesize a large range of reachable punching area with GP prediction (figure 6). These results

proves our analysis about approximating the unsolved question space with a low-dimensional latent space.

5.2 Experiment 2

Now we conduct $SARSA(\lambda)$ algorithm in this latent space. In experiment 2, the agent is demanded to punch targets in different locations. We first discretize the space into a 50×40 girds, in which the grid points stands for the corresponding states in $SARSA$. Due to the dual continuously mapping via GPDM, we can directly design the actions as moving to the nearby 8 neighbor points in a state. Also, we design our reward function as $r = r_{var} + r_{target}$, in which r_{var} punishes the states in low-variance area, r_{target} only can be obtained when the agent punches the target. Note that generally we offset the highest variances as 0, hence high confident points maintain a negative value close to 0, and low confident points will be denoted with a negative value with big magnitude. Since we initialize all $Q(s, a)$ as 0, this trick helps us to find solution more efficiently.

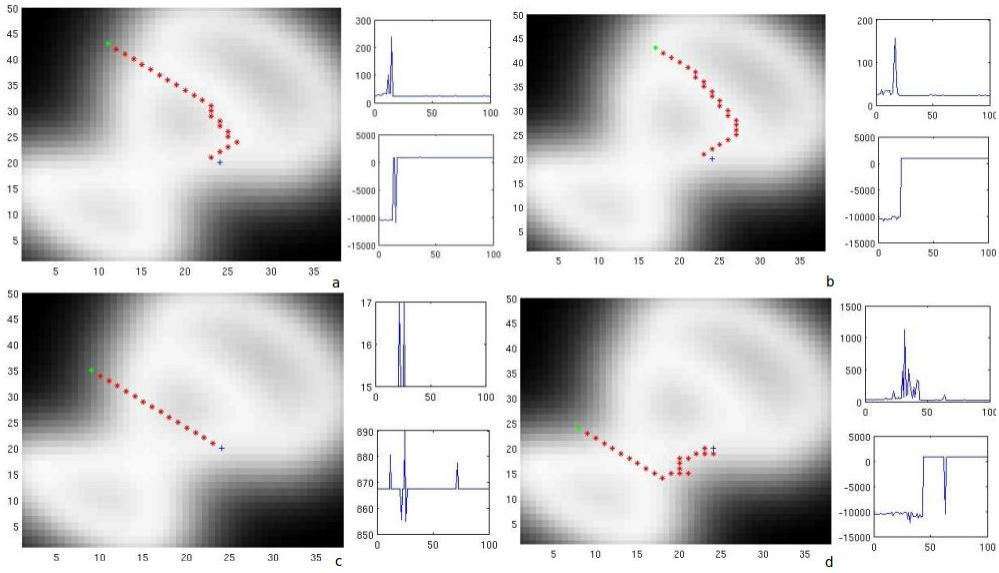

Figure 7: Optimal solutions for punching 4 targets. (a) places the target in (-2,12,18) to (2,15,23) with an obstacle in (-2,12,18) to (0,15,23); (b) puts the target in (-2,12,18) to (0,15,23) without obstacle; (c) puts the target in (-2,15,18) to (2,16,23); (d) places the target in (-2,18,18) to (0,20,23). The upper figure in each subfigure plot the steps for searching optimal path; and the lower figures record the corresponding rewards for each episode.

Just as figure 7, we test 4 targets with our framework. Both our target and obstacle are defined as an AABB box in data space, which is represented with the minimal and maximal points. Observing the optimal path for each trail, we can clearly find out that these paths are close to high confident areas, which thus always results in fluid simulation result subject to physical constraints. Moreover, the terminal point in (a), (c), (d) do not exist in prior dataset, which proves the validation of our latent searching space. Note that all trials converge after a small number of episodes and inner steps. This proves the efficiency of our algorithm. Of course, it is very easy to handle obstacles(constraints) with our algorithm. By placing an obstacle in the optimal terminal (the solution in (b)), our algorithm converges to other solutions dynamically (the solution in (a)). This proves the adaptivity of our algorithm.

6 Discussion and Extensions

To sum up, we have presented a constrained GPLVM based RL framework to overcome the curse of dimensionality problem of RL. Given a group of prior data and a target task, our framework is able to efficient solve this task in a continuous mapped latent space. Coincidentally, [10] came up with

a similar idea as our work a little earlier. But their framework is only able to handle one sequence due to the discontinuity mapping. Also, a number of potential improvements can be applied to our framework.

First extensions should be expanding the patterns of potential states with less data. Our work put an assumption that new poses are similar or between prior data due to some physical constraints. Thus, our framework is powerful once we have a groups of boundary data. Extend it to exploit some unknown poses, we may need to incorporate some sampling strategy to automatically generate some extreme states in data space and then plug them in during reduction.

Another possible extensions is to deal with the hybrid data sequences. Sometimes we want to project totally different sequences in a shared latent space and still maintain its continuity for searching. In that case, Topologically constrained model [11] would be a better choice.

Finally, a renewable latent space which dynamically plugs in new pose and searching results will be more adaptive for some tasks.

Acknowledgments

We would like to thank Neil D. Lawrence and Jack Wang for making their GPLVM/GPDM toolbox and MOCAP toolbox freely available. We would also like to thank CMU for publishing their motion capture databases online freely as well. Last but not least, we would like to thank Nando de Freitas for his valuable feedbacks for this project.

References

[1] Kroemer, O.& Detry, R.& Piater, J.& Peters, J.(2010) Combining Active Learning and Reactive Control for Robot Grasping, *Robotics and Autonomous Systems*, 58, 9, pp.1105-1116.

[2] Auer, P., Cesa-Bianchi, N.,& Fischer, P.,(2002) *Finite-time analysis of the multiarmed bandit problem.* Mach. Learn. 47 (2-3), 235-256.

[3] A. Barto, S. Mahadevan, (2003) *Recent advances in hierarchical reinforcement learning.* Discrete Event Systems (Special Issue on Reinforcement Learning) 13, 41-77.

[4] Lucas Kovar, Michael Gleicher & Frederic Pighin. (2002) *Motion Graphs*. ACM Transactions on Graphics 21(3) (Proceedings of SIGGRAPH 2002). July 2002.

[5] Lawrence, N.D. (2003). *Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data.* Proc. NIPS 2003, 2004.

[6] Keith Grochow, Steven L. Martin, Aaron Hertzmann. & Zoran Popovi, (2004) *Style-Based Inverse Kinematics.* ACM Transactions on Graphics Vol. 23 Num. 3 (SIGGRAPH 2004).

[7] Lawrence, N.D., Quionero Candela, J. (2006) *Local distance preservation in the GP-LVM through back constraints*. Proc. of the 23rd International Conference on Machine Learning 2006 pp. 513-520.

[8] Wang, J. M., Fleet, D. J.,& Hertzmann, A., (2006) *Gaussian Process Dynamical Models*, Proc. of NIPS 2005.

[9] Wang, J. M., Fleet, D. J., Hertzmann, A., (2008) *Gaussian Process Dynamical Models for Human Motion*. IEEE Trans. on Pattern Analysis and Machine Intelligence

[10] Bitzer, S., Howard, M., & Vijayakumar, S. (2010). *Using Dimensionality Reduction to Exploit Constraints in Reinforcement Learning*. In , IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)

[11] R. Urtasun, D. J. Fleet, A. Geiger, J. Popovic, T. Darrell & N. D. Lawrence. (2008) *Topologically-Constrained Latent Variable Models.* In International Conference in Machine Learning (ICML) Helsinki, Finland, July 2008

[12] Lawrence, N.D. & Moore, A. J. (2007). *Hierarchical gaussian process latent variable models*. In Proceedings of the International Conference in Machine Learning.

[13] Sutton, Richard S. & Andrew G. Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press. ISBN 0-262-19398-1.