Skill reconstruction as induction of LQ controllers with subgoals

Dorian Suc
dorian.suc@fri.uni-lj.si
Faculty of Computer and Information Sciences,
University of Ljubljana,
Trzaska 25, 1000 Ljubljana,
Slovenia

Ivan Bratko
ivan.bratko@fri.uni-lj.si
Faculty of Computer and Information Sciences,
University of Ljubljana,
Trzaska 25, 1000 Ljubljana,
Slovenia

Abstract

Controlling a complex dynamic system, such as a plane or a crane, usually requires a skilled operator. Such a control skill is typically hard to reconstruct through introspection. Therefore an attractive approach to the reconstruction of control skill involves machine learning from operators' control traces, also known as behavioural cloning. In the most common approach to behavioural cloning, a controller is induced in the form of a rule set or a decision or regression tree that maps system states to actions. Unfortunately, induced controllers usually suffer from lack of robustness and lack typical elements of human control strategies, such as subgoals and substages of the control plan. In this paper we present a new approach to behavioural cloning which involves the induction of a model of the controlled system and enables the identification of subgoals that the operator is pursuing at various stages of the execution trace. The underlying formal basis for the present approach to behavioural cloning is the theory *of LQ* controllers. Experimental results show that this approach greatly improves the robustness of the induced controllers and also offers a new way of understanding the operator's subcognitive skill.

1 Introduction

Controllers can be designed by Machine Learning using different kinds of information available to the learning system. Approaches like reinforcement learning, genetic-algorithms and neural networks typically don't use prior knowledge about the system to be controlled.

Humans, however, rarely attempt to learn from scratch. They extract initial biases as well as strategies from their prior knowledge of the system or from demonstration of experienced operators. Control theory makes use of the former, but it doesn't consider operator's skill.

On the other hand, the idea of *behavioural cloning* (a term introduced by Donald Michie [Michie, 93]) is to make use of the operator's skill in the development of an automatic controller. A skilled operator's control traces are used as examples for machine learning to reconstruct the underlying control strategy that the operator executes subconsciously. The goal of behavioural cloning is not only to induce a successful controller, but also to achieve better understanding of the human operator's subconscious skill [Urbancic and Bratko, 94]. Behavioural cloning was successfully used in problem domains as pole balancing, production line scheduling, piloting [Sammut et al., 92] and operating cranes. These experiments are reviewed in [Bratko et al., 95]. Controllers were usually induced in the form of decision or regression trees.

Although successful clones have been induced in the form of trees or rule sets, the following problems have generally been observed with this approach:

- Typically, induced clones are brittle with respect to small changes in the control task.
- The clone induction process typically has low yield: the proportion of successful controllers among all the induced clones is low, typically well below 50%.
- Resulting clones are purely reactive and inadequately structured as conceptualisations of the human skill. They lack typical elements of human control strategies: goals, subgoals, phases and causality.

In this paper we propose a different approach to behavioural cloning which exploits some results from control theory. In particular, our clones take the form of LQ controllers. The approach also involves induction of approximate models of controlled systems. It is experimentally demonstrated that this approach dramatically improves both the clones' robustness with respect to the changes of the control task, and the yield of the cloning process. Also, the approach provides a way of interpreting the induced clones in terms of the operator's goals and subgoals.

In this paper, the reconstruction of human operator's skill exploits some elements of the theory of Linear Quadratic Regulator problems [Bertsekas, 87]. Both the dynamics of the system and the example behavioural trace are considered in the learning process. First, the system dynamics is learned using locally weighted regression. Then, a linear quadratic regulator which performs similar to the operator is constructed . Since the controller is goal-directed and takes into account the system dynamics, good performance and robustness are achieved. The parameters of induced controllers give some insight in the human control skill. This provides an interesting possibility of studying human control skill apart from the system dynamics.

Bradtke [Bradtke, 93] also explored the idea using LQR, but in his case in combination with a reinforcement learning method. However, he did not pay attention to the interpretation of the optimal policy as Q-function in terms of the underlying operator's control skill.

The structure of the paper is as follows. The next section reviews the basics of LQR theory that will be needed in Section 3, which describes the methods used to construct controllers from behavioural traces. In Section 4 the ideas are extended to *subgoal identification* in more complex control tasks. Section 5 gives some experimental results.

2 Linear Quadratic Regulation

In many systems it, is not always possible to reach the desired state in one step. In such systems, one must initially move away from the desired state in order to approach it later. Linear Quadratic Regulation facilitates such policies through minimization of a long term cost function. Let *x(t)* be the system state vector, *u(t)* the action vector and x_{des} the desired goal state. Then LQ-Cost is defined as:

$$C = \sum_{t=0}^{\inf} (x_t - x_{des})^T \, Q \, (x_t - x_{des}) + (u_t - u_0)^T \, R \, (u_t - u_0)$$

where *Q* is a positive definite matrix, *R* is a positive semi-definite matrix and u_0 is a user defined lowest cost action. Elements of matrices *Q* and *R* set the tradeoff between the cost of the action components and the error components. If, for example, *Q* and *R* were set to identity matrices, then the sum of squared deviations from desired performance would be penalized. Usually human supervisor specifies the relative importance of the state and action components by suitable adjustments to *Q* and *R*. An alternative to using explicit human specification of the importance of various state and action components is in automatic induction of matrices *Q* and *R* from operator's behavioural traces as described in Section 3.

Consider a linear, discrete-time and multivariable dynamic system

$$x_{k+1} = T(x_k, u_k) = Ax_k + Bu_k + c$$

where T is a state transition function specified by matrices A and B and vector c. The control task is to achieve and maintain a given setpoint x_{des}, which satisfies a condition

$$x_{des} = T(x_{des}, u_0)$$

At each discrete time step k, the controller observes the system to be in state x_k and performs an action u_k. Each pair (x_k, u_k) incurs a cost $r(x_k, u_k) \in R^+$ and the controller's objective is to select actions so as to drive the system to x_{des} with the minimum cumulative cost.

Heuristic search and dynamic programming tackle problems of this kind by assigning a numerical value $V(x)$ to each state. The natural measure of state's value is the expected sum of costs if the controller starts in state x and executes an optimal policy. Function $V(x)$ is usually called value function or *cost-to-go* function. Given the cost-to-go function and the state transition function optimal action in any particular state x can easily be found as

$$u^{opt} = \arg\min_u (r(x, u) + V(T(x, u)))$$

Let cost-to-go function $V(x)$ be the minimal possible sum of future costs, starting from state x. Then $V(x)$ can be defined inductively:

$$V(x) = \min_u (r(x, u) + V(T(x, u)))$$

Linear Quadratic Regulation assumes the immediate cost $r(x, u)$ penalizing mahalanobis distance from the desired performance in states and actions:

$$r(x) = (x - x_{des})^T Q (x - x_{des}) + (u - u_0)^T R (u - u_0)$$

In the case of linear system dynamics and under assumption that the system will run forever, $V^{LQ}(x)$ is quadratic in x [Bertsekas, 87] and therefore can be expressed as:

$$V^{LQ}(x) = (x - x_{des})^T P (x - x_{des})$$

where P is $n * n$ cost matrix, associated with immediate cost given in Q and R. If the matrices A, B, Q and R are known, P can be found as a unique positive definite solution to the Riccatti equation by initialising $P := Q$ and running the following iteration to convergence:

$$P := Q + A^T P (I + BR^{-1} * B^T * P)^{-1} * A$$

Optimal control theory gives the optimal action *u,* which minimizes this cost-to-go function, by a simple gain matrix A":

$$\begin{aligned} u &= u_0 - K(x - x_{des}) \\ K &= (R + B^T P B)^{-1} B^T P A \end{aligned}$$

Linear system dynamics is assumed. However empirical results often show good robustness of LQ-controllers even if the actual system is non-linear and approximated by a linear model.

3 Approximating the operator's value function

Our aproach to behavioural cloning in the LQR framework relies on the following observations:

1. An approximate model of the controlled system, in the form of a linear state transition function T, can be induced by locally weighted regression [Schaal and Atkeson, 94] from some available execution traces.
2. Generally, when (a) cost matrices *Q* and *R* are given and the execution cost is defined as LQ cost (function of *Q* and *R),* and (b) the system is linear, formulas from control theory can be used to compute the state value function *V(x)* and corresponding optimal actions *u(x).*
3. We assume that the operator's actions in the available execution traces are intended to optimise the LQ cost. So in the case of behavioral cloning we have to solve the inverse to the optimal control problem: given the actions, find the matrices *Q* and *R* for which the actions are (approximately) optimal.
4. The operator may be executing a more elaborate plan when he pursues different subgoals at different stages of the execution trace. In such cases, the trace can be broken into stages whose subgoals are reflected in different *Q* and *R* matrices.

In the following we develop the details of this idea.

A continous trace is sampled so that we have a sequence of pairs *(xk,uk),* *k*=1,2,... *N,* ordered according to time. Goal state x_{des} is known in advance. The state transition function *T(x,u)* is estimated using locally weighted regression. Assuming the linear dynamics of the system we can compute LQ cost-to-go function as

$$V^{LQ}(x_k, x_{des}, Q, R) = (x_k - x_{des})^T P (x_k - x_{des})$$

where *P* is computed as the solution of the Riccatti equation as explained erlier.

Define LQ-optimal action as an action which minimizes the LQ-cost for particular *Q** and *R**:

$$u_{opt} = \arg\min_u \left((u - u_0)^T R^* (u - u_0) + V^{LQ}(T(x_k, u))\right)$$

Assuming that the actions observed in the trace are LQ-optimal and assuming we know matrices Q^* and R^*, we can compute operator's cost-to-go function for every state x_k, $k=N, N-1, \ldots, 1$:

$$V_k(x_k, x_{des}, Q, R) = \sum_{j=k}^{N} \left((x_j - x_{des})^T Q (x_j - x_{des}) + (u_j - u_0)^T R (u_j - u_0)\right)$$

We are interested in finding such matrices Q^* and R^* that produce LQ-cost-to-go function for which the actions observed in the trace are in some way closest to LQ-optimal actions. Matrices Q^* and R^* can be found using local optimization, minimizing LQ-fit error:

$$e(1, N, x_{des}, Q, R) = \sum_{k=1}^{N} \left(\frac{V_k(x_k, x_{des}, Q, R) - V^{LQ}(x_k, x_{des}, Q, R)}{V^{LQ}(x_k, x_{des}, Q, R)} \right)^2$$

i.e, by finding "least-squares fit" of operator's cost-to-go function and LQ-controller cost-to-go function. When *Q** and *R** are found, the corresponding LQ-controller can be constructed. Matrices *Q** and *R** can also be used to explain how the operator is performing the task. Local optimization optimizes matrices of Q and R, so the number of optimization parameters grows as the square of the number of state variables and control variables.

A simplification is to assume *Q* and *R* are diagonal. In this case the number of optimization parameters grows linearly with the number of variables. This reduces the complexity of local optimization problem, and sometimes improves the comprehensibility of the induced controller.

4 Learning subgoals

In complex control tasks (like flying a plane for example) the operator's policy usually changes in time. To achieve the final goal, the operator must first achieve some subgoals. Those subgoals define stages of a policy. Each stage ends when its subgoal is reached. One possible approach is to break the execution trace into stages by hand, and learn a controller for each stage separately. However, it is not always possible to define stages by hand. An operator learns his skills by practice. The rules underlying these skills are typically opaque to the operator, and he is not able to describe them completely. The obvious alternative is to induce the subgoals from execution trace and construct a controller for each stage. We define a subgoal as a point in a state space approached by the operator and after which the operator's policy changes in the sense of LQ-cost. Accordingly we define a subgoal candidate xg(j) at time step *j* as the point in the state space with coordinates computed as mean of the most recent 5 states.

The change of a policy is detected as minimization of weighted LQ-fit error at a given subgoal candidate:

$$\begin{aligned} e(1,N,j) = & \quad \frac{1}{j}e(1,j,xg(j),Q_1,R_1) \\ & + \frac{1}{N-j}e(j+1,N,xg(N),Q_2,R_2) \end{aligned}$$

First the whole execution trace is searched for a subgoal $^{xg}(j)$ which minimizes weighted LQ-fit error. Given the subgoal *xg(j),* the execution trace is divided into two stages, one before subgoal is reached and the other after the subgoal is reached. Both stages are then recursively searched for subgoals. This is done until the error of the divided stage is greater than the error of the undivided stage. For every stage an LQ-controller is computed. When controller's subgoal is approached, the next controller is activated.

5 Experimental results

5.1 Container crane

In this section we describe experiments in LQR cloning in the domain of container cranes [Urbancic and Bratko, 94], systems with nonlinear dynamics.

To transport a container from a shore to a target position two operations are to be performed: positioning of the trolley, bringing it above the target load position, and rope operation, bringing the load to the desired height. The performance requirements include basic safety, stop-gap accuracy and as high capacity as possible. The last requirement means that the time for transportation is to be minimized. The most difficult aspect of the task is to control the swing of the rope. When the load is close to the goal position, the swing should ideally be zero.

A crane simulator was used in our experiments. The state of the system is specified by six variables: trolley position X and its velocity *X,* rope inclination angle Φ and its angular velocity Φ, rope length L and its velocity *L.* Two control forces are applied to the system: force XF to the trolley in the horizontal direction and force YF in the direction of the rope.

We used experimental data from manually controlling the crane from a previous study [Urbancic and Bratko, 94]. In that study, six students volunteered to learn to control the simulator. All of them succeeded to accomplish the task. However, remarkable individual differences were observed regarding the speed of controlling and the characteristics of the strategy they used. Some operators tended towards fast and less reliable operation, others were more conservative and slower, in order to avoid large rope oscillations. Our goal of cloning was also to reconstruct these individual differences between the operators in the style of driving the crane.

Previous experiments in this domain involve behavioural cloning with regression trees and combining skills from several operators. The main problem for regression trees was the swing control, i.e large rope oscillations when the trolley approached its goal position.

5.2 Experimental details

To complete the task successfully, the operator had to reach the goal position within 180 s. Each trace was sampled with frequency 4 Hz. A successful trace typically lasts between approximately 50 and 120 s, so such a trace gives about 200 to 480 state-action pairs. For local optimization Powell's method with discarding the direction of largest decrease was used. Matrices *Q* and *Ft* were restricted to diagonal.

Since state is a 6 dimensional and action is a 2 dimensional vector, 7 parameters were approximated by local optimization. It took about 15 sec. on Pentium 133 Hz to find matrices *Q* and *R* and construct the corresponding controller. When learning subgoals *S* was set to 10. To minimize time needed to find the best subgoal, a subgoal candidate was tested every 10 time steps. It took about, 8 minutes to find the best one among those candidates.

To learn a linear model locally weighted regression at the mean point of the stage was used. To learn a model we used several operators' execution traces: first, the trace used to approximate the value function, another two traces of the same operator, and typically one more trace from another operator. There was no particular reason for this choice of traces and we feel that other choices would produce similar results.

5.3 Experimental results

Experiments with clones based on LQR showed that the swing control is not as difficult as it seems to be for regression tree clones. Even when a controller accelerates and this causes a huge swing of the rope, the controller usually manages to stabilize the rope angle within a few seconds. Most experiments were done with controllers where matrices *Q* and *R* were restricted to diagonal matrices. The reason for this restriction was simply in smaller complexity of local optimization. Also, so restricted controllers are easier to comprehend.

Controllers without subgoals, i.e single LQ controllers usually failed to accomplish the task.

This observation lead us to conclusion that LQ controllers with subgoals should be sought. There is also a

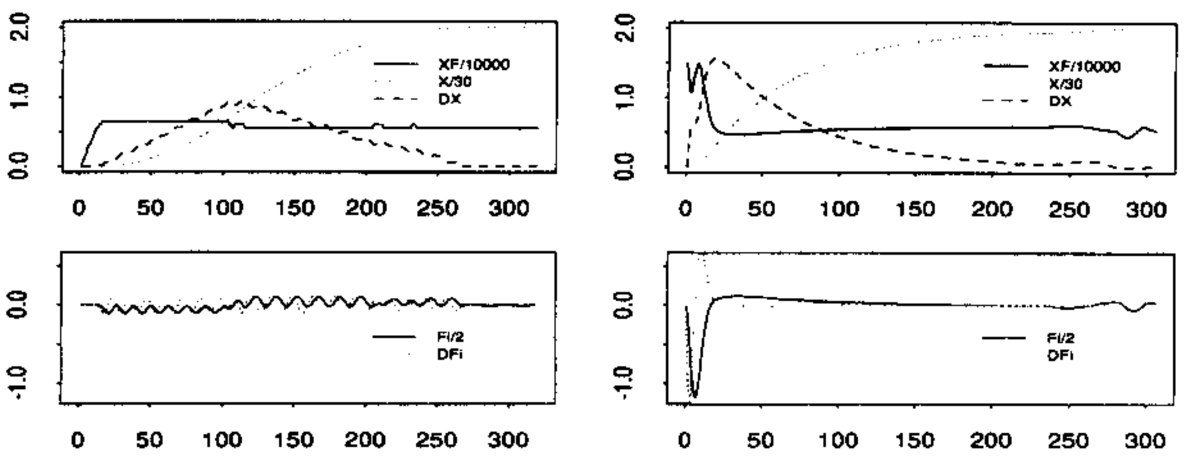

Figure 1: Trace 8 of subject S and trace of its clone

good intuitive reason for controllers with subgoals. Operators usually used one strategy when the trolley was far from the goal, and another strategy when the trolley was near the goal position. In the first stage, deviation in the horizontal speed and horizontal force is more important than in the second stage, since in the second stage there is no need for large horizontal speed.

In Figure 1 behavioural trace S8 (trace 8 of subject S) is shown on the left and that of an induced clone on the right. The clone consists of two LQ-controllers. Matrices Q and R are diagonal. The subgoal of the first minimizes the weighted LQ-fit error, and the subgoal of the second is the final goal. In the first stage the controller attempts to reach a point near the final goal. This point is: $xg(260) = (X{=}59.61,\ \dot{X}{=}0.07,\ \Phi{=}0.07,\ \dot{\Phi}{=}0.03,\ L{=}31.66,\ \dot{L}{=}0.00)$ The diagonals of the Q and R matrices at this stage are: $Q_X{=}1,\ Q_{\dot{X}}{=}28558,\ Q_\Phi{=}170,\ Q_{\dot{\Phi}}{=}135,\ Q_L{=}0.010, Q_{\dot{L}}{=}221,\ R_{XF}{=}40,\ R_{YF}{=}167.$

It is clear that the deviations from the desired horizontal velocity and angular velocity are penalized most. In the second stage only the final positioning is needed; deviations in all state components are to be respected. This can be seen from the goal and induced matrices for this stage: $x_{des} = (X{=}60.00,\ \dot{X}{=}0.00,\ \Phi{=}0.00,\ \dot{\Phi}{=}0.00,\ L{=}32.00,\ \dot{L}{=}0.00),\ Q_X{=}1,\ Q_{\dot{X}}{=}337,\ Q_\Phi{=}33,\ Q_{\dot{\Phi}}{=}22,\ Q_L{=}29,\ Q_{\dot{L}}{=}245,\ R_{XF}{=}0.285,\ R_{YF}{=}1291.$

In Figure 2 the behavioural trace V12 (trace 12 of subject V) is shown on the left and trace of induced clone on the right. The induced clone consists of three LQ-controllers, coresponding to the tree stages, and attains considerably better time than the original. The phenomenon of the clone surpassing the original, i.e. clean-up effect, can be observed.

It is interesting to interpret the differences in the control styles of the operators S and V. By comparing the subgoals and Q and R matrices of their clones we can see that S's strategy is very conservative (resulting in very cautions actions) wheras V tends to accelerate fast in the x direction at the price of large swing. Only at last stage V pays attention to the oscillations of the rope.

The most informative indicators that support this interpretation are the values $Q_{\dot{X}}$, R_{XF} and the goal $\dot{X}_{des}$ for the first stage. For operator S these values are: $Q_{\dot{X}}{=}28558,\ R_{XF}{=}40,\ \dot{X}_{des}{=}0.07.$

For operator V, these values are: $Q_{\dot{X}}{=}76899,\ R_{XF}{=}8.6,\ \dot{X}_{des}{=}0.87.$

5.4 Robustnes

The first observation is that the percentage of successful controllers among all the induced ones is above 60%. This is a significant improvement over the previous experiments with regression trees in the crane domain where the reported yield was less than 10%.

Robustness of controllers was tested by modifying two crane parameters: friction of the trolley on the trail and cargo weight. During the learning process these two parameters were as in the original simulator but were than changed at clone execution time. During testing both parameters were changed by -10%>, 0% and 10% with respect to their original values. For each of the nine combinations of parameter settings we get slightly different system dynamics. Twenty controllers which accomplished the original task were randomly selected and tested. Out of the total of 20 controllers induced from traces on the original system, betwen 13 and 18 controllers performed successfully also on the thus modified systems.

These results show that the controllers are rather robust against the change in the system dynamics.

The controllers were also tested for their robustness

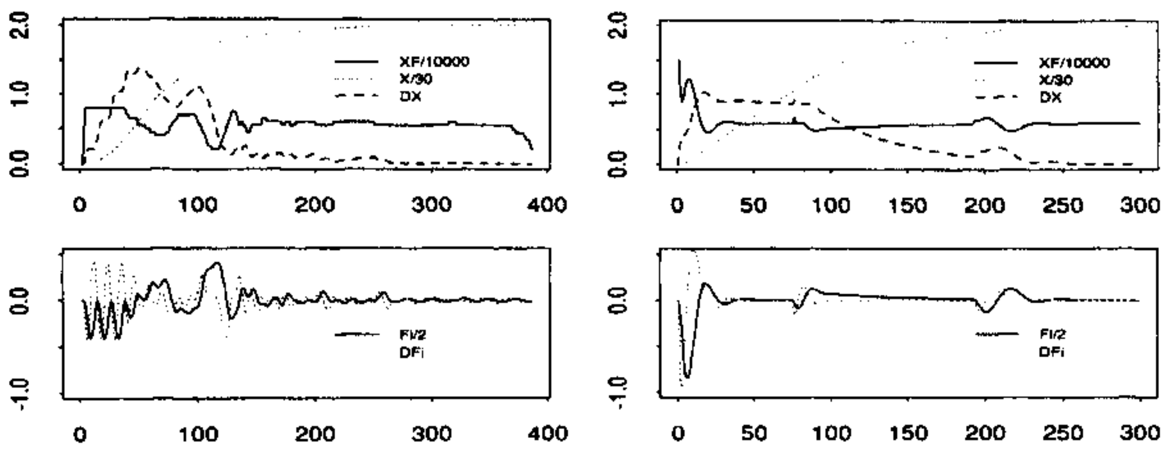

Figure 2: Trace 12 of subject V and trace of its clone

against changes in the start state, the test where the controllers in the form of decision or regression trees usually fail. Since LQ controllers are goal directed, the change of the start state didn't affect their performance significantly. Starting horizontal position was modified by 10% and 50%, and starting rope angle by -20%o, 0% and 20%. This amounts to six combinations - six different tasks. Nineteen of twenty controllers tested accomplished all six different tasks.

The results show that induced controllers perform very well and often accomplish the task faster than the original operator.

6 Conclusion

A new approach to behavioural cloning was presented in the paper. In previous work, the cloning problem was usually formulated as one of inducing a straightforward mapping from system states to actions. Our approach on the other hand also involves the induction of a model of the dynamics of the controlled system, and takes into account the *order* of the example state-action pairs. It in a way enables the identification of subgoals that the operator is pursuing at various stages of the execution trace. Experiments in the crane domain show that the (automatic) breakdown of the control traces into stages with their corresponding subgoals is essential for inducing robust controllers.

The underlying formal basis for the present approach to behavioural cloning is the theory of LQ controllers, and the cloning problem is here done as the inverse of the usual problem of designing an optimal controller. As experimental results show, this approach in comparison with most of the previous experiments greatly improves the robustness of induced controllers and the yield of the cloning process.

References

[Bertsekas, 87] D.P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models,* Prentice Hall, Englewood Cliffs, 1987

[Bradtke, 93] S.J.Bradtke, Reinforcement Learning Applied to Linear Quadratic Regulation, Hanson, J.S., Cowan,J.D., and Giles, C.I., (Eds.), *Advances in Neural Information Processing Systems 5, pp.295-302,* San Mateo, CA: Morgan Kaufmann, 1993

[Bratko et al., 95] I.Bratko, T.Urbancic, C.Sammut, Behavioural Cloning: phenomena, results and problems, *Automated Systems Based on Human Skills,* IFAC Symposioum, Berlin, 1995

[Michie, 93] D. Michie, Knowledge, learning and machine intelligence, *Intelligent Systems,* Plenum Press, New York, 1993

[Sammut et al., 92] C. Sammut, S. Hurst, D. Kedzier, and Michie, D., Learning to Fly, In *Proceedings of the Ninth International Workshop on Machine Learning,* Morgan Kaufmann, 1992

[Schaal and Atkeson, 94] S. Schaal and C.G. Atkeson, Assessing the quality of learned local models, *Advarices in Neural Information Processing Systems 6.,* Morgan Kaufmann, 1994

[Urbancic and Bratko, 94] T. Urbancic. and I. Bratko, Reconstructing Human Skill with Machine Learning, In *Proceedings of the 11th European Conference on Artificial Intelligence,* John Wiley & Sons, Ltd, 1994