Learning the Inverse Dynamics of a Robot Arm by Auto-Imitation

Karl Theodor KALVERAM
Cybernetical Psychology,
Institute of Experimental Psychology
University of Duesseldorf
40225 Duesseldorf, Germany
e-mail: kalveram@uni-duesseldorf.de

and

André SEYFARTH
Locomotion Laboratory,
Institute of Sport Science
University of Jena
07743 Jena, Germany
e-mail: oas@uni-jena.de

ABSTRACT

Auto-imitation is a type of unsupervised learning which enables a controller to quickly acquire the inverse dynamics of plants. The basics of the method are demonstrated using a one-segment robot arm. If embedded into a learner-operator schema, the inverse model can be updated while the robot performs purposive or even arbitrary movements. So, the proposed algorithm would be an alternative to supervised learning, for instance via feedback-error. It is presumed that auto-imitation like learning governs especially social learning in humans.

Keywords: inverse adaptive control, sensorimotor learning, inverse dynamics model, auto-imitation

1. INTRODUCTION

Control of human goal directed rotational forearm movements is commonly assumed to include feedforward through an internal inverse model of the arm’s dynamics, and negative feedback realized either through appropriately processed peripheral measurements [1-3], or through a tunable torsion spring [4,5]. The present paper focuses on how an artificial neural network can acquire the inverse dynamics with adequate precision.

Technically, two learning algorithms come into consideration: (a) Supervised learning, for instance back-propagation of feedback-error [6], and (b) non-supervised learning through auto-imitation [7]. Both algorithms have been tested by simulation as principally feasible. Feedback-error learning [6,8] is a sequential learning rule which, however, revealed as time consuming and “loose”. In contrast, auto-imitation is a non-error based learning algorithm which is applicable in parallel, and reveals as precise and fast. The root idea of this learning algorithm has been conceived to characterize sensorimotor learning as acquiring and extending the “sensorimotor self and its reverse” [9] in the framework of the re-afference principle [10]. Formally, the algorithm can be described as “direct inverse modeling” [11]. The name “auto-imitation” used later on for this strategy of “self-extension” [12,13] refers to the procedure, through which the controller explores and applies the dynamical properties of a part of a machine, respectively of a limb of the own body: stimulating the limb with arbitrary proof loads (forces), observing the reactive movements, and casting both into a law which predicts the proof loads from the reactions. The thus accumulated knowledge enables the controller to select that load which, when exerted to the limb, realizes a predefined (desired) movement. So far, a purposively performed movement looks like an imitation of a previously self-generated, but non-purposive movement.

The method, however, was generally considered as not practicable to be used in the organismic control of limb movements, because the re-wiring of the respective neural network when switching from learning to operating and vice versa was assessed as too difficult or even impossible to be applicable in biological systems [6,11]. As Fig. 2 shows, this objection has been overcome by introducing the learner-operator concept, in which the functions of learning and operating are assigned to different neural networks complementing each other [7], what makes re-wiring unnecessary.

Though applicable also to multi-jointed levers or arms, here we demonstrate the method using an one-segment lever. At first we give a description of the robot used for the demonstration, then we explicate inverse control and auto-imitation, followed by the description of the outcome if auto-imitation and inverse control are applied to the robot. Finally it is suggested to take auto-imitation together with inverse control as a model to explain also higher order types of behavior, for instance those in the social domain.
2. METHODS

Hardware
The ADAM (Adaptable Machine) device is a simple robot test bed built to test the effectiveness of different forcing functions applied to a lever. The version used in the current investigation is illustrated in Fig. 1 and comprises a DC-motor (Maxon RE30 310007) without gear, the shaft being aligned horizontally, and a freely rotating lever of about 0.2m length fixed at the front end’s shaft, which is considered a pointer. The motor is driven by an analogue power amplifier (Mattke MAR 24/6z) operating in the voltage-to-current mode. Hence, the current indicates the total torque \( Q \) exerted on the lever via the motor. SIMULINK Realtime software (fixed step sample time 1ms, solver ode5) organizes through AD- and DA-conversion (interface: Meilhaus ME 2600) the control of the power amplifier. A tachometer (Mattke TS05) fixed to the other end of the shaft measures the actual angular velocity. Derivative and integration of this signal via SIMULINK software provide the actual angular acceleration and position.

![Figure 1: Robotic arm used for the hardware test. Left: Front view. Right: Side view with adumbrated wiring. The “hand” is made of coarse cellular foam (usage with courtesy of Thomas Schinauer).](image)

Inverse control and auto-imitation
Fig. 2 depicts, how operating the arm via inverse control, and learning the arm’s inverse dynamics via auto-imitation, can be arranged simultaneously. The upper part of Fig. 2 sketches the control: The pattern generator transforms a sequence of discrete angular movement goals into the desired angular kinematics (desired trajectories of angular position, velocity and acceleration) which smoothly interpolate between the steps of the goals. These three streams of values are put into the inverse dynamics model which computes that trajectory of torques \( Q_{ff} \) which, when applied to the arm, realizes the desired kinematics values. In engineering, this type of control is also known as ‘computed torque’ \([14]\) or ‘inverse control’ \([15]\). A disturbing torque can be added upon \( Q \) via the input \( Q_x \). To preserve stability after a disturbance, the inverse model gets its state input as ‘state feedforward’ from the pattern generator \([4,5]\) and not as ‘state feedback’ from the periphery. Deviations between the desired and the actual positions are processed by proportional negative feedback control (gain factor \( K \)). The thus generated torque \( Q_{fb} \), however, vanishes if the inverse model is perfect and disturbances are absent. The lower part of Fig. 2 sketches auto-imitative learning of the arm’s inverse dynamics model. The basic idea is, that learner and operator rely on the same simplified general model of how the arm’s dynamics are physically describable: The arm, if regarded as an ideal gravity pendulum with viscous damping and friction around the pivot, is governed by the differential equation

\[
J \ddot{\theta} + B_d \dot{\theta} + B_f \text{sign} \dot{\theta} + M \cdot \sin(\theta_0 - \theta) = Q, \tag{1}
\]

where \( J \) denotes the arm’s inertia, \( B_d \) the coefficient of viscous damping, and \( B_f \) the coefficient of friction. \( M = m \cdot d \cdot g \) (\( m \): mass; \( d \): the distance of the center of mass from the pivot, \( g = 9.81 \text{N/kg} \)) and \( \theta_0 \) (offset, angular deviation of the rest position from the gravitational reference) relate to the gravitational induced torque.

![Figure 2: Auto-imitation embedded into a learner-operator scheme. The inverse model is split into two segregated networks, named the ‘learner’ and the ‘operator’, which have equal architecture, but different functions. The “learner” acquires the parameters (that are the synaptic weights of the underlying artificial neural network) of the inverse model through auto-imitation, but does not have an output to control the plant (here the arm). The “operator” gets the parameters from the learner and exerts control, but does not have a teaching input to update the parameters. Notice that the diagram is functionally oriented. Therefore, the physiological realization of the concept is left open.](image)
The lower part of Fig. 2 sketches auto-imitative learning of the arm’s inverse dynamics model. The basic idea is, that learner and operator rely on the same formula which physically describes the arm’s motion: The arm, if regarded as an ideal gravity pendulum with viscous damping and friction around the pivot, is governed by the differential equation

\[ J \ddot{\phi} + B_0 \dot{\phi} + B_f \text{sign} \phi + M \cdot \sin(\phi_0 - \phi) = Q, \]  
(2)

where \( J \) denotes the arm’s inertia, \( B_0 \) the coefficient of viscous damping, and \( B_f \) the coefficient of friction, whereas \( M = m \cdot d^2 \) (\( m \): mass; \( d \): the distance of the center of mass from the pivot, \( g = 9.81 \text{N/kg} \)) and \( \phi_0 \) (offset, angular deviation of the rest position from of gravitational reference) relate to the gravitational induced torque. \( Q \) is the total external torque acting upon the arm. If angular positions \( \phi \) are small enough, sin \( \phi \) can be replaced by \( \phi - \phi^3 / 6 \). In so doing, and omitting terms expected to get negligible, after some rearranging Eq. (2) becomes

\[ M \phi_0 + J \ddot{\phi} + B_0 \dot{\phi} + B_f \text{sign} \phi = M \dot{\phi} + \frac{M}{6} \phi^3 = Q. \]  
(3)

In general terms, the left side of Eq. (3) is expressed as a function \( f(x) \) which is approximated by a power series in four variables,

\[ f(x) = \sum_{i,j,k,l} w_i x_i x_j x_k x_l, \]  
(4)

where the first order variables \( x_i = \dot{\phi}, x_j = \ddot{\phi}, x_k = \text{sign} \phi, x_l = \theta \) form the vector \( x \). \( 0 \leq i,j,k,l \) denote non-negative integers, and the \( w_m \) (\( m = 0,1,2, \ldots \)) represent coefficients (weights) to be determined. Regarding Eq. (4) as power series truncated by the condition \( i+j+k+l \leq 3 \) (which makes Eq. (4) a power series of order 3), and additionally discarding a number of terms, Eq. (4) may be written as

\[ w_0 + w_1 \dot{\phi} + w_2 \ddot{\phi} + w_3 \text{sign} \phi \ddot{\phi} + w_4 \phi \ddot{\phi} + w_5 \phi^3 \approx Q. \]  
(5)

Compared to Eq. (2), this suggests to interpret the weights as \( w_0 = M \phi_0, \ w_1 = J, \ w_2 = B_0, \ w_3 = B_f, \ w_4 = M, \) and \( w_5 = M/6, \) provided Eq. (3) indeed governs the arm’s movements. The appearance of Eq. (5) is that of an inverse model of the arm’s dynamics, if \( x_i = \dot{\phi}, x_j = \ddot{\phi}, x_k = \text{sign} \phi, x_l = \theta \) are considered as inputs and \( Q \) as the output. So, the inverse model implemented in the operator can tentatively be written as

\[ \dot{a}_0 + \dot{a}_1 \dot{a}_0 + \dot{a}_2 \ddot{a}_0 + \dot{a}_3 \text{sign} \phi \ddot{a}_0 + \dot{a}_4 \phi \ddot{a}_0 + \dot{a}_5 \phi^3 \approx Q. \]  
(6)

whereby the last two terms are added to check the effect of surplus terms on the value of the weights. The computations described in Eq. (6) can be mimicked by a simple artificial neural network, the Power Net [16], the - for the present paper adapted - form of which is depicted in Fig. 3.

The weights \( w_m \) are determined through the “auto-imitative loop” outlined in Fig. 2: Firstly, at \( n \) points of time in an ongoing movement, \( n \) samples, each consisting of the actual acceleration \( \dot{a}_0 \), velocity \( \dot{a}_1 \), position \( a_0 \), and torque \( Q \), are collected. Then, as shown in Fig. 4, the values \( a_0, \ldots, a_7 \) of the seven variables occurring in Eq. (6) are inserted into matrix \( A \), and the measurements of \( Q \) into \( B \), \( a_0 \) is given the value 1. The matrix operation \( A*B \) with \( W \) being the column vector of yet unknown weights, is assumed to reproduce the matrix \( B \). For numbers \( n > 7 \), \( A*B = W \) can be considered an over-determined system of equations. The Matlab command \( W = B*A \) then gets the solution via the least squares method.

![Figure 4: Arrangement of n samples of measurements a0, ..., a7 of the left-side variables of Eq. (6) and the related torque values Q into matrices A and B. w0 is always set to 1, and B has one column. The MATLAB command W=B*A then provides a least squares approximation of the weights.](image-url)
Application of the concept to the robot arm and results

In Fig. 5, the sequence of discrete angular goals put into the pattern generator (see Fig. 2) is indicated by the broken gray (yellow) line, and the corresponding continuous desired angular positional trajectory interpolating between the steps of the goals by the thick black line. The actual angular trajectory produced by the robot arm is indicated by the thin black line. The gain $K$ in the positional negative feedback loop was set to 0.5Nm/rad. In a first movement of about 10s, all weights were set to zero, so control was limited to pure negative feedback control. As can be seen in Fig. 5a, the actual angular positions then deviate from the desired ones considerably. To yield the weights, at the end of this period the recorded kinematics and torque values were, according to Fig. 4, processed by the least squares method. The weights were then inserted into the operator’s inverse dynamics model. This results in a nearly perfect matching of the actual with the desired positional angular trajectory (see Fig. 5b) in the next run.

Principally, in each ongoing movement the weights of the inverse model can be updated to improve the controller’s performance in the next movement. The weights thus obtained in seven successive runs and their means are listed in tab.1. After the first run, essential further improvements are not achievable, but obviously also not necessary.

Though the weights in tab.1 exhibit some fluctuations, the values seem physically plausible. In any case, the weights referring to a run produce, if inserted into the operator’s network, an actual positional curve like that shown in Fig. 5b. From $w_0$ and $w_4$ one gets the angular offset $\phi_0 = \phi_4 \approx 0.0258 \text{rad} \cong 1.5^\circ$ which is probably due to an imperfect offset adjustment of the power amplifier. $w_2$ reflecting the coefficient of viscous damping $D_\phi$ is zero, whereas $w_3$ representing the coefficient $B_\phi$ of friction force is greater than zero. This is in line with the construction of the motor which uses laminated brushes for commutation, what produces friction, but no damping. The small values of $w_5$ and $w_7$ further indicate, that the corresponding terms in Eq. (6) are indeed dispensable.

3. DISCUSSION

Auto-imitation reveals as a technique for unsupervised learning of the inverse dynamics of a robot arm. The technique is quick, 10s suffice to program the respective network. Though demonstrated using an one-segmented lever, the algorithm is expected to be applicable at least also to a two-segmented real arm, a conclusion we extrapolate from the results of simulations [17].

In the present paper we use a simultaneous rule to acquire the weights of the artificial neural network representing the inverse dynamics. This rule is based on the least squares method applied to a great many of observed kinematics and torque data stored in memory. For auto-imitation, however, also a sequential rule is forthcoming, as could be demonstrated with a simulated two-segment arm, but a hardware test for this instance is yet lacking. The sequential rule is describable as modified Hebbian learning [18], what links auto-imitation with organismic learning. This leads to the motivation underlying the presented investigation, namely to test, whether the envisaged type of acquisition of an inverse dynamics model is applied also in human motor control. Regarding the test trilogy [4] 1 proposed to tackle such questions, simulation tests have already been performed, and a hardware test has been presented in this paper. The behavioral comparison test is pending; its results will be published elsewhere.

1 In this test trilogy, the simulation test probes whether a concept is logically sound. The hardware test employs a machine (robot) to check whether a successfully simulated theoretical concept is practicable also under real world conditions. The behavioral comparison test compares human data statistically with simulated data generated by checked models. With this test procedure it is possible to assess, which of several competing models is nearest to human data.
Humans explore the physical properties not only of their own limbs, but also of the external environment. Here, the same technique as described above can be applied: Arbitrary, but scaled, test signals have to be put into the part of the environment to be explored, the sensory signals encoding the related environmental changes have to be observed, and in an auto-imitative loop both types of signals have to be connected and condensed into a general law. The law then can be used to actively shape the environment towards a desired state. In everyday language, we call this ‘learning to handle a tool’ and ‘using a tool purposively’, in science and engineering ‘making an experiment’ and ‘solving a technical problem’. Considered psychologically, in interpersonal contacts each person is the other person’s social environment. In the framework of the so emerging intertwined auto-imitative social loops, each partner in a pair gets knowing and manipulating the other. So, the rules governing sensory motor behavior can be consulted how to investigate and explain also higher order human behavior.

Acknowledgements: This study was supported by grants SE1042/2 and KA417/18 of the German Research Foundation (DFG).

4. REFERENCES


Table 1: Weights of the inverse model determined by auto-imitation in seven subsequent runs.

<table>
<thead>
<tr>
<th>weight</th>
<th>run 1</th>
<th>run 2</th>
<th>run 3</th>
<th>run 4</th>
<th>run 5</th>
<th>run 6</th>
<th>run 7</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>w0 = Mϕ0</td>
<td>0.0033</td>
<td>0.0032</td>
<td>0.0031</td>
<td>0.0031</td>
<td>0.0032</td>
<td>0.0029</td>
<td>0.0021</td>
<td>0.0030</td>
</tr>
<tr>
<td>w1 = J</td>
<td>0.0019</td>
<td>0.0020</td>
<td>0.0019</td>
<td>0.0019</td>
<td>0.0019</td>
<td>0.0019</td>
<td>0.0019</td>
<td>0.0019</td>
</tr>
<tr>
<td>w2 = Bp</td>
<td>0.0000</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>-0.0002</td>
<td>-0.0001</td>
<td>0.0002</td>
<td>0.0001</td>
<td>-0.0000</td>
</tr>
<tr>
<td>w3 = Bp</td>
<td>0.0155</td>
<td>0.0159</td>
<td>0.0174</td>
<td>0.0178</td>
<td>0.0178</td>
<td>0.0137</td>
<td>0.0147</td>
<td>0.0161</td>
</tr>
<tr>
<td>w4 = M</td>
<td>0.1150</td>
<td>0.1167</td>
<td>0.1153</td>
<td>0.1173</td>
<td>0.1156</td>
<td>0.1166</td>
<td>0.1165</td>
<td>0.1161</td>
</tr>
<tr>
<td>w5 = M/6</td>
<td>-0.0168</td>
<td>-0.0193</td>
<td>-0.0171</td>
<td>-0.0246</td>
<td>-0.0226</td>
<td>-0.0196</td>
<td>-0.0195</td>
<td>-0.0199</td>
</tr>
<tr>
<td>w6 = w6</td>
<td>-0.0004</td>
<td>-0.0005</td>
<td>-0.0004</td>
<td>-0.0006</td>
<td>-0.0004</td>
<td>-0.0004</td>
<td>-0.0004</td>
<td>-0.0004</td>
</tr>
<tr>
<td>w7 = w7</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>-0.0001</td>
</tr>
</tbody>
</table>


