Neuro-Fuzzy Inverse Forward Models

Brian Highfill
Stanford University
Department of Computer Science

***Abstract*- Internal cognitive models are useful methods for the implementation of motor control [1]. The approach can be applied more generally to any intelligent adaptive control problem where the dynamics of the system (plant) is unknown and/or changing. In particular, paired inverse-forward models have been shown successfully to control complex motor tasks using robotic manipulators [1]. One way this is accomplished is by training a pair of multi-layered neural networks to learn both the forward model of the plant dynamics and an inverse controller model simultaneously using an extension of the backpropagation algorithm. This paper explores a variation of the traditional multi-layered network used for teaching an inverse-forward model pair (IFMP). We investigate the use of a simple fuzzy-neural system for implementing both models.**

I. INTRODUCTION

An inverse-forward model pair (IFMP) is a modeling technique used for control which attempts to invert the dynamics of some unknown system. This allows for the indirect manipulation of system outputs (distal variables) [1]. IFMP's have been applied to nonlinear adaptive control systems to learn complex motor tasks using robotic manipulators. A common training technique used for such systems is backpropagation on a pair of multi-layered feedforward neural networks. Previous work on IFMP's indicates that learning can be extended to any supervised learning method which can learn in multiple layers [1]. The current paper explores the training of an IFMP on a fuzzy Adaptive Standard Additive Model (ASAM).

ASAM's are "shallow" (single-layered) networks which rely on a set of natural language-like fuzzy IF-THEN rules to map inputs to outputs. ASAM's, like multi-layered networks are universal approximators but have the added advantage of being "grey box". One can look "inside" a trained ASAM and interpret the induced fuzzy sets (unlike multilayered networks where hidden layers are difficult to decipher). By extending gradient descent to ASAMS in inverse-forward pairs, one would be able to solve complex adaptive control problems while also enabling insight to the structure of the learned control law and plant dynamics in an intuitive rule-based format.

II. METHODS

A. Controlling Distal Outputs

An environment can be thought of as a mapping from actions to observed outputs. If we could invert that mapping, we would be able to indirectly manipulate the environmental outputs as we please. This can be accomplished using an *inverse model*. An inverse model is a mapping from desired outputs to actions which, when composed with the environment, forms an identity relation. Sometimes it is desired (or useful) to model the dynamics of the environment as well. A *forward model* is a mapping from actions to outputs which aims to approximate the same mapping of a (real) environment.

Given some nonlinear (or possible non-stationary) environment (plant), our goal then, is to learn an inverse mapping of the environment from the desired "sensation" $y^*[n]$ to the observed sensation (output) $y[n]$ which enables it to be controlled. Figure 1 shows the general setup with the inverse model (controller or learner) in series with the environment.

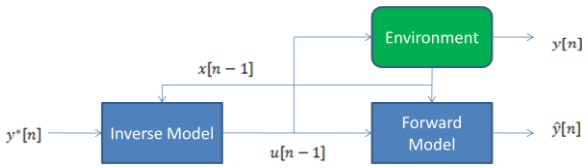

Fig. 1: Training the forward model from environmental response to actions.

We will assume that the full state of the environment $x[n-1]$ is observable. Using a forward model as in this setup helps us to avoid a subtle difficulty in learning inverse models. Figure 2 shows the convexity problem [1]. The environment is in general a surjective function which may map more than one action to a single output. If we were to directly model the inverse function by treating it as a standard supervised learning problem, the inverse model would effectively select a centroid of the inverse image for a desired sensation. However, this centroid need not be an element of the inverse image of the desired sensation. In other words, the inverse image is nonconvex [1].

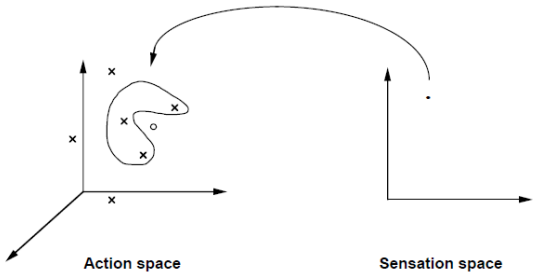

Fig. 2: The convexity problem. Multiple actions map to the same sensation. The inverse model must select a proper action from the inverse image of a desired sensation [1].

Modeling the inverse learning problem in the goal-direct form of figure 1 allows the inverse model to select a *single* action from the inverse image given a desired output. The particular action chosen, however, cannot be predicted as it depends on the order of the sequence of training examples [1].

B. Learning the Models

In order to learn the inverse model, we must first estimate the forward dynamics of the environment. This is accomplished by learning a forward model. We can learn a forward model by observing training pairs $\{u[n-1], y[n]\}$ through the setup of figure 3.

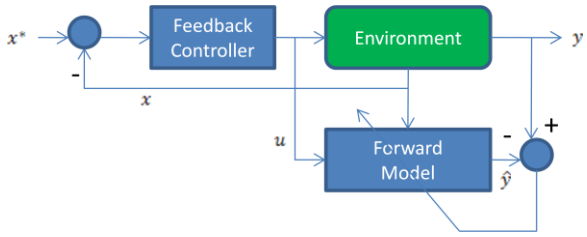

Fig. 3: Learning the forward model.

An auxiliary feedback controller is utilized to maintain stability of the dynamic system. The feedback controller is typically a simple proportional-derivative (PD) controller.

Assume, now, that we already have an accurate forward model which correctly maps from actions $u[n-1]$ to $y[n]$. By holding the forward model fixed, we can train the composite system of figure 1 to be the identity mapping from $p[n-1]$ to $y^*[n]$ by implicitly training the learner to be the inverse of the forward model. Figure 4 shows the generic setup. The backpropagation algorithm is often utilized to this end in multilayer networks by propagating errors backwards through the forward model (without changing it) and finally backwards through the learner.

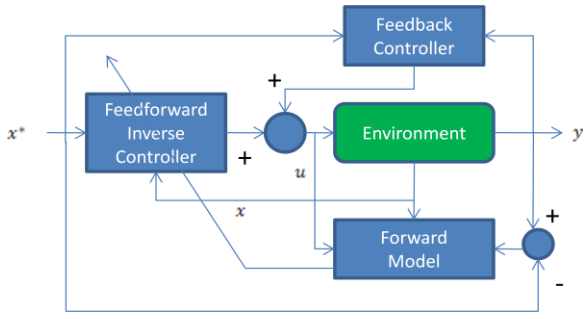

Fig. 4: Learning the inverse model (controller).

We can simultaneously learn the forward model while trying to learn the inverse controller. Logically, there are two iterative phases per training input:

first, the forward model is tuned by observing the actual response of the environment to actions (figure 3), then, holding the forward model fixed, the inverse model is tuned via backpropagation so that the composite inverse-forward pair realizes the identity map. To learn the forward model simultaneously, we can use the response of the (stabilized) inverse model to learn on actual desired trajectories [1].

C. Adaptive Standard Additive Model

The forward and inverse models have been often implemented using a standard feedforward hidden-layered neural network and trained via backpropagation. Although this method is powerful, these networks are black boxes and intuition about what they learn cannot be gained by simply observing the learned weights. Adaptive neuro-fuzzy systems are also powerful (they are universal approximators [2]) but have the added advantage that they are grey box. By looking at the learned fuzzy IF-THEN rules, one can understand the learned structure in a natural language way.

The neuro-fuzzy system presented is the adaptive standard additive model (ASAM). On each of the inputs, a set of m fuzzy IF-THEN rules (sets) are used to predict the output. Each fuzzy rule is activated *to a degree* in $[0,1]$ depending upon the input x. Rule activation functions can be chosen with many different shapes, for example, triangle or Gaussian (radial basis) functions (see figure 5) [2]. Once a given rule is activated, the corresponding fuzzy THEN-part set is asserted (which has its own shape). The final decision (output) can be made by adding together all the asserted THEN-part sets and finding the centroid.

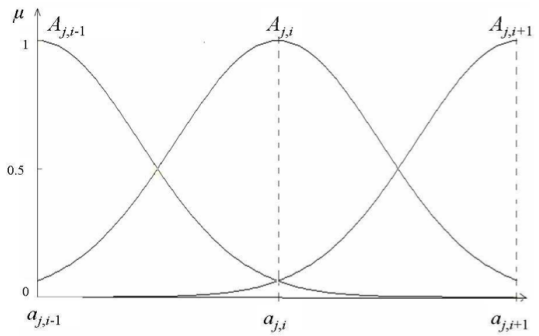

Fig. 5: Gaussian activation (set membership) functions on the input.

Let the SAM be a mapping $F: \mathbb{R}^n \to \mathbb{R}^p$, further, define $A_j \Leftrightarrow a_j: \mathbb{R}^n \to [0,1]$ be the fuzzy membership function for rule j, $B_j \Leftrightarrow b_j: \mathbb{R}^p \to [0,1]$ be the fuzzy THEN-part set for rule j, $V_j = volume(B_j)$ and $c_j = centroid(B_j)$, then:

$$F(x) = \frac{\sum_{j=1}^{m} a_j V_j c_j}{\sum_{j=1}^{m} a_j V_j}$$

Using gradient descent, an adaptive learning law is derived to adjust the parameters above by minimizing the squared error cost function over a set of training examples.

D. Gradient Decent Learning Laws

We now derive the learning laws for an IFMP for an environment with one continuous state variable and on continuous action and a single output. Each model will be an ASAM with a factorable Gaussian activation functions. Let the forward model be:

$$\hat{y} = \hat{f}(x, u, \theta') = \frac{\sum_{j=1}^{m'} a_j{}' V_j{}' c_j{}'}{\sum_{j=1}^{m'} a_j{}' V_j{}'}, with\ a_j^{i\prime} = a_j^{1\prime} a_j^{2\prime}$$

Also, define:

$$p_j' = \frac{a_j'(x,u) V_j{}'}{\sum_{i=1}^{2} a_i'(x,u) V_i{}'}$$

Define the squared error to be minimized as:

$$L_t = \frac{1}{2}(y_t - \hat{y}_t)^T (y_t - \hat{y}_t)$$

Then

$$\theta'_{t+1} = \theta'_t - \eta \nabla_{\theta'} L_t$$

$$\nabla_{\theta'} L_t = \frac{\partial L_t}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \theta'} = -\frac{\partial \hat{y}}{\partial \theta'}(y_t - \hat{y}_t)$$

$$\begin{aligned} c_j'(t+1) &= c_j'(t) - \eta_{c'} \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial c_j'} \\ &= c_j'(t) - \eta_{c'}\big(y(t) - \hat{y}(x_t, u_t)\big) p_j'(x_t, u_t) \end{aligned}$$

$$\begin{aligned} V_j'(t+1) &= V_j'(t) - \eta_{V'} \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial V_j'} \\ &= V_j'(t) \\ &\quad - \eta_{V'}\big(y(t) - \hat{y}(x_t, u_t)\big) \frac{p_j'(x_t, u_t)}{V_j'(t)} \Big(c_j' \\ &\quad - \hat{y}(x_t, u_t)\Big) \end{aligned}$$

$$\mu_j^{k'}(t+1) = \mu_j^{k'}(t) - \eta_{\mu'} \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_j'} \frac{\partial a_j'}{\partial a_j^{k'}} \frac{\partial a_j^{k'}}{\partial \mu_j^{k'}}$$

With:

$$\frac{\partial L}{\partial \hat{y}} = -\big(y(t) - \hat{y}(x_t, u_t)\big)$$

$$\frac{\partial \hat{y}}{\partial a_j'} = \frac{p_j'(x_t, u_t)}{a_j'(x_t, u_t)} \Big(c_j' - \hat{y}(x_t, u_t)\Big)$$

$$\frac{\partial a_j'}{\partial a_j^{k'}} = \begin{cases} a_j^{2'}, if\ k = 1 \\ a_j^{1'}, if\ k = 2 \end{cases}$$

$$\frac{\partial a_j^{k'}}{\partial \mu_j^{k'}} = a_j^{k'} \frac{(x^k - \mu_j^{k'})}{\left(\sigma_j^{k'}\right)^2}$$

Similarly, let the inverse model be:

$$\hat{u} = h(x, u, \theta) = \frac{\sum_{j=1}^m a_j V_j c_j}{\sum_{j=1}^m a_j V_j}, with\ a_j^i = a_j^1 a_j^2$$

Also, define:

$$p_j = \frac{a_j(x_t, y_t^*) V_j}{\sum_{i=1}^2 a_i(x_t, y_t^*) V_i}$$

Finally:

$$\sigma_j^{k'}(t+1) = \sigma_j^{k'}(t) - \eta_{\sigma'} \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_j'} \frac{\partial a_j'}{\partial a_j^{k'}} \frac{\partial a_j^{k'}}{\partial \sigma_j^{k'}}$$

$$\frac{\partial a_j^{k'}}{\partial \sigma_j^{k'}} = a_j^{k'} \frac{\left(x^k - \mu_j^{k'}\right)^2}{(\sigma_j^{k'})^3}$$

$$J_t = \frac{1}{2}(y_t^* - \hat{y}_t)^T (y_t^* - \hat{y}_t)$$

Then,

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J_t$$

$$\nabla_\theta J_t = \frac{\partial J_t}{\partial y_t} \frac{\partial \hat{y}}{\partial \theta} = \frac{\partial J_t}{\partial y_t} \frac{\partial \hat{y}}{\partial a_j'} \frac{\partial a_j'}{\partial a_j^{2'}} \frac{\partial a_j^{2'}}{\partial u} \frac{\partial u}{\partial \theta}$$

Where,

$$\frac{\partial J_t}{\partial y_t} = -\big(y_t^*(t) - y_t(x_t, u_t)\big)$$

$$\frac{\partial \hat{y}}{\partial a_j'} = -\frac{p_j'(x_t, u_t)}{a_j'(x_t, u_t)} \Big(c_j' - \hat{y}(x_t, u_t)\Big)$$

$$\frac{\partial a_j'}{\partial a_j^{2'}} = a_j^{1'}$$

$$\frac{\partial a_j^{2'}}{\partial u} = a_j^{2'}(u_t) \frac{u_t - u_j^{2'}}{\left(\sigma_j^{2'}\right)^2}$$

Now, for the centroid:

$$c_j(t+1) = c_j(t) - \eta_c \nabla_c J_t$$

$$\nabla_c J_t = \frac{\partial J_t}{\partial y_t} \frac{\partial \hat{y}}{\partial a_j'} \frac{\partial a_j'}{\partial a_j^{2'}} \frac{\partial a_j^{2'}}{\partial u} \frac{\partial u}{\partial c_j}$$

$$\frac{\partial u}{\partial c_j} = p_j(x_t, u_t)$$

For updating the volume:

$$V_j(t+1) = V_j(t) - \eta_V \nabla_V J_t$$

$$\nabla_V J_t = \frac{\partial J_t}{\partial y_t} \frac{\partial \hat{y}}{\partial a_j'} \frac{\partial a_j'}{\partial a_j^{2'}} \frac{\partial a_j^{2'}}{\partial u} \frac{\partial u}{\partial V_j}$$

$$\frac{\partial u}{\partial V_j} = \frac{p_j(x_t, u_t)}{V_j} \Big(c_j - u(x_t, y_t^*)\Big)$$

For updating the Gaussian means:

$$\mu_j^k(t+1) = \mu_j^k(t) - \eta_\mu \nabla_\mu J_t$$

$$\nabla_\mu J_t = \frac{\partial J_t}{\partial y_t} \frac{\partial \hat{y}}{\partial a_j'} \frac{\partial a_j'}{\partial a_j^{2'}} \frac{\partial a_j^{2'}}{\partial u} \frac{\partial u}{\partial a_j} \frac{\partial a_j}{\partial a_j^k} \frac{\partial a_j^k}{\partial \mu_j^k}$$

$$\frac{\partial u}{\partial a_j} = \frac{p_j(x_t, y_t^*)}{a_j(x_t, y_t^*)} \Big(c_j - u(x_t, y_t^*)\Big)$$

$$\frac{\partial a_j}{\partial a_j^k} = \begin{cases} a_j^2, if\ k = 1 \\ a_j^1, if\ k = 2 \end{cases}$$

$$\frac{\partial a_j^k}{\partial \mu_j^k} = a_j^k \frac{(x^k - \mu_j^k)}{(\sigma_j^k)^2}$$

For updating the Gaussian dispersions:

$$\sigma_j^k(t+1) = \sigma_j^k(t) - \eta_\sigma \nabla_\sigma J_t$$

$$\nabla_\sigma J_t = \frac{\partial J_t}{\partial y_t} \frac{\partial \hat{y}}{\partial a_j'} \frac{\partial a_j'}{\partial a_j^{2\prime}} \frac{\partial a_j^{2\prime}}{\partial u} \frac{\partial u}{\partial a_j} \frac{\partial a_j}{\partial a_j^k} \frac{\partial a_j^k}{\partial \sigma_j^k}$$

$$\frac{\partial u}{\partial a_j} = \frac{p_j(x_t, y_t^*)}{a_j(x_t, y_t^*)} \left(c_j - u(x_t, y_t^*)\right)$$

$$\frac{\partial a_j}{\partial a_j^k} = \begin{cases} a_j^2, if\ k = 1 \\ a_j^1, if\ k = 2 \end{cases}$$

$$\frac{\partial a_j^k}{\partial \sigma_j^k} = a_j^k \frac{(x^k - \mu_j^k)^2}{(\sigma_j^k)^3}$$

III. SIMULATIONS

We simulated learning the forward and inverse models separately. First, we assumed that the environment has the known dynamics (see figure 6) given by:

$$y(x, u) = \sin(x) \cos(u)$$

To simulate learning of the forward model, we used a Gaussian ASAM with 10 rules on each input (state and action). Using 10^-4 for all learning rates and allowing the forward model to sample each of the environment outputs in response to a uniform sampling of the input and control space over $\{x, u \mid 0 \leq x \leq 2\pi, 0 \leq u \leq 2\pi\}$ and for 3000 iterations, the forward model converged to a good approximation to y(x,u).

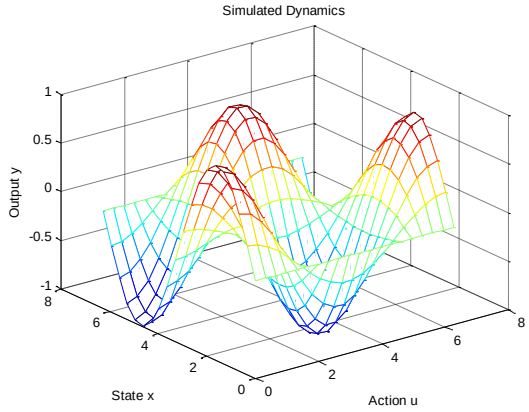

Fig. 6: Simulated environmental dynamics.

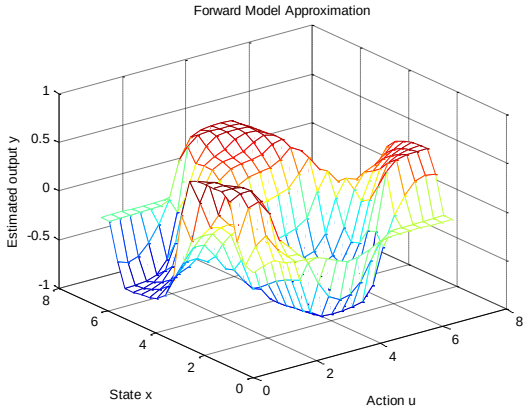

Fig. 7: Learned forward model.

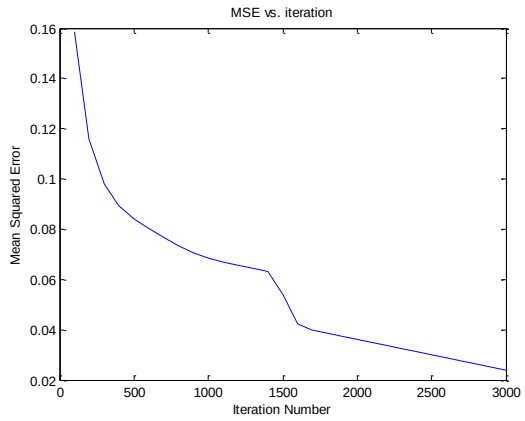

Fig. 8: Convergence to learned forward model.

We then trained the inverse model using the gradient descent learning law derived above

using the same type of ASAM and with 2*10^-5 for all learning rates. The inverse model converges (after 2000 iterations) to the model shown in figure 9.

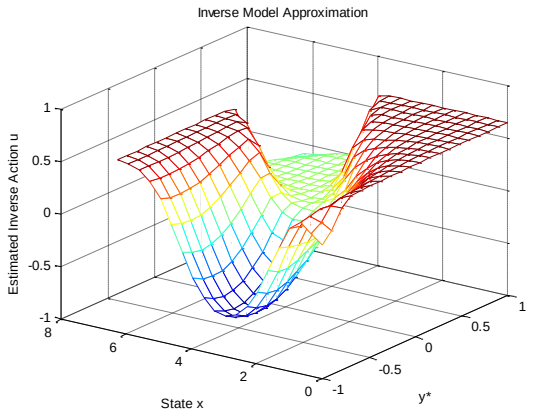

Fig. 9: Convergence to learned inverse model.

IV. CONCLUSION

The applications of this method of control are plentiful. It is especially useful for environments with non-linear or non-stationary dynamics where control is possible only through adaptive controller updates. Additionally, the benefit of utilizing a neuro-fuzzy controller under the IFMP paradigm is that the mystery of "what is learned" is partially removed. One can directly observe the learned fuzzy rules in the form of fuzzy if-then statements.

V. REFERENCES

[1] M. I. Jordan, D. E. Rumelhart, "*Forward models: Supervised Learning with a distal teacher",* Cognitive Science, 16, 307-354, 1992.

[2] B. Kosko, *Fuzzy Engineering,* Upper Saddle River, New Jersey: Prentice-Hall, 1996, pp. 97-104.

[3] T. M. Mitchell, *Machine Learning,* Boston, Massachusetts: McGraw-Hill, 1997, pp. 97-103.