Department of Computer Science, University of Otago

Technical Report OUCS-2013-11

Training and testing of a neural network model of motor control

Authors:

Jeremy Lee-Hand and Alistair Knott

Department of Computer Science, University of Otago, New Zealand

---

Department of Computer Science,
University of Otago, PO Box 56, Dunedin, Otago, New Zealand

http://www.cs.otago.ac.nz/research/techreports.php

Training and testing of a neural network model of motor control

Jeremy Lee-Hand and Alistair Knott

September 12, 2013

Abstract

This paper is an appendix to Lee-Hand and Knott (in submission). It describes the training and testing of the network model of motor control presented in that paper.

1 Objects

We built several objects in the simulation environment which could serve as targets for hand actions. One is a simple object (a cylinder) which serves as the target for three simple motor actions: grasping, punching and slapping. Three others are articulated objects which can undergo various changes in interal configuration. One is a lever which can pivot around a joint, and can be bent; one is a hinged door in a plane, which can be pushed open; one is a pair of horizontal plates connected by a spring, which can be 'squashed' by pushing down on the top plate. These objects are illustrated in Figure 1.

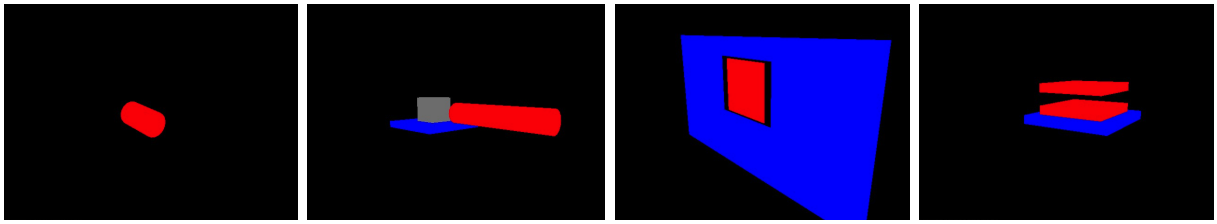


Figure 1: Objects created for the simulations. From left: a cylinder (for grasping, punching and slapping); a lever (for bending); a door (for opening); a compressable object (for squashing).

2 The tactile signal classification system

The simple motor programs our model learns are grasp, slap and punch. In our model, these motor programs are defined by distinctive patterns of haptic feedback recorded on the hand. In the pattern associated with a grasp, the touch sensors on the pads of the fingers and thumb are active concurrently while the hand is relatively closed, indicating the presence of an object within the hand's opposition space. Grasp patterns can be of different qualities; the best are those where the finger pads are deformed, and register no 'slip'. (The way the model registers light touch, skin deformation and slip is discussed in Neumegen, 2013.) In the pattern associated with a slap, the sensors on the palm are active concurrently with those on the finger pads while the hand is relatively open, indicating contact with a surface; the best patterns are those where contact is registered over a large surface area. In the pattern associated with a punch, sensors on the back of the fingers are active while the hand makes a fist. Up to a certain threshold, the best patterns are those registering the most forceful contact; beyond this threshold, contact is registered as pain, and is not rewarding.

In our model, these three distinctive haptic patterns are simply hardwired to deliver reward signals. In humans, we envisage there is some component of hardwiring, to encourage development of core useful motor programs like grasping. (Grasping has no adaptive benefit by itself, but it contributes to many adaptive behaviours like eating, so there is some evolutionary pressure to make it intrinsically rewarding.) But we also envisage there may be an element of self-organisation to haptic patterns. When we interact with objects, the resulting patterns are likely to fall into certain classes; a learning system that can detect commonly occurring patterns is likely to be useful in developing a core set of simple motor routines.

3 The external action classification system

To learn causative actions, the network contains a module for recognising actions taking place in objects in the world. We do not implement a realistic model of action recognition; instead the module uses oracles with direct access to the underlying physics engine, that detect changes in the configuration of the target object. The three external actions the module can recognise are bending, opening and squashing. A bend action is detected in the lever object when it rotates through an angle of 45°. An open action is detected in the door object when the door panel rotates behind the wall it is attached to. A squash action is detected in the squashable object when its top plate makes contact with its bottom plate.

Again, while we have hardwired an ability to recognise our chosen external actions, we assume the origin of external action categories in humans is due to a mixture of hardwiring and self-organisation. The perceptual system certainly seems hardwired to detect certain types of external action—for instance there is low-level circuitry specialised for detecting objects moving in relation to their background (e.g. Britten *et al.*, 1996). If the mirror

system hypothesis is correct, there are also special circuits training agents to recognise actions from their own motor repertoire being performed by external agents (see e.g. Oztop and Arbib, 2002). ('Bending' is conceivably learned through this kind of circuitry, as a visual pattern associated with the bending movements of the agent's own limbs, which then generalises to inanimate objects like levers.) But we assume the perceptual system is also able to form action categories representing arbitrary commonly-occurring patterns of motion detected in external objects. Opening and squashing are better thought of as categories of this type.

Crucially, although we predefine the external actions we want to be able to cause, we do not predefine the motor actions that bring these effects about. These must be learned.

4 The plan activation network

The model also includes a network that selects a planned action sequence in each trial. This network takes a localist encoding of the category of object presented to the system in this trial, and produces as output a planned action sequence in the action planning system. The action planning system holds localist encodings of all actions which can be executed or perceived by the agent. It stores a sequence of actions by activating all actions in the sequence in parallel, with activity levels proportional to their position in the sequence, the action to be executed first being most active, as found in Averbeck *et al.*'s 2002 study of prefrontal sequence plans. (A detailed model of this prefrontal planning medium is given in Takac and Knott, 2013.) The plan activation network is a placeholder for a much more complex network that selects a plan based not only on current sensory inputs but also on elaborate internal representations of 'the current context'. Its purpose in the current model is just to learn which actions are appropriate for which object types.

5 Training the networks

The first network that is trained is the reach network. A single object (the cylinder) is presented in each training trial in any location within the space reachable by the arm. The retinotopic location of this object is computed, and provided as input to the reach network, which generates an output goal motor state. Initially this output is annealed with noise, so the goal motor state is essentially random. A feedback controller then brings the hand towards the goal motor state. If the hand happens to make contact with the object (i.e. if a tactile signal is received), the current motor state is logged as training data for the reach network, paired with the retinotopic location of the object. After each trial, the reach network is trained on all the training data logged so far. During learning, the noise applied to the network's output is progressively reduced to zero. The training algorithm is displayed in more detail in Algorithm 1.

Algorithm 1: Learning Goal Arm states for reach actions

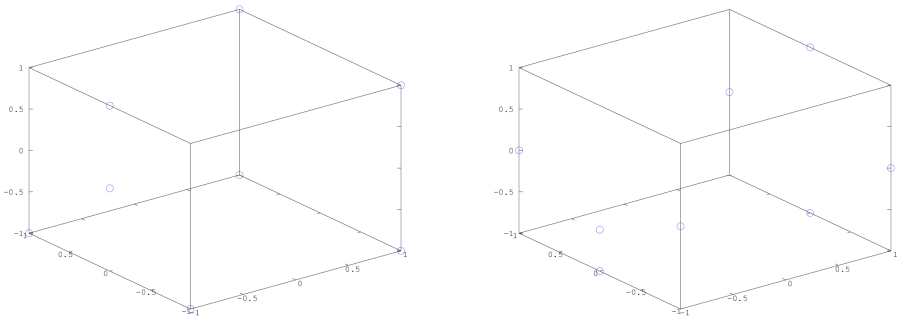
Data: $o = (o_x, o_y, o_z)$ (object centroid in retinal coordinates), $\theta = (\theta_{c1}, \theta_{c2}, \theta_{c3})$ (current arm joint angles)

begin

- o input into reach network to produce $\theta_g = (\theta_{g1}, \theta_{g2}, \theta_{g3})$ (goal motor state);
- Random noise added to θ_g;
- **while** *Maximum time not exceeded* **do**
  - Calculate force applied to arm using PID controller (a function of $\theta - \theta_g$);
  - **if** *Tactile feedback occurs* **then**
    - Store touch score paired with θ;
- **if** *Touch data recorded* **then**
  - Log the maximum touch score and corresponding θ, paired with o_r, as a new training item for the reach network;
  - Maximum possible random noise reduced;
  - **if** *number of training items exceeds* 200 **then**
    - Discard oldest training item;
  - Reach network trained on training data;

There is one further point to make about the training of the reach network (and those that follow). It is of course unrealistic to assume a storage medium where a large number of training items can be logged. In a more plausible online learning scheme, the network being trained would interleave self-generated pseudo-training items with new training data arriving sequentially (see e.g. Robins, 1995). To approximate an online scheme, we do however impose various constraints to ensure that the quality of stored training data improves during the course of training. Firstly, the learning rate used by the network for any given logged training item is a function of a **score** associated with the haptic feedback pattern that the hand received (recall some patterns are better than others). This means that trajectories which result in higher scores influence the behavior of the network more than trajectories producing lower scores. In addition to this, there is a minimum threshold score needed in order for an action to be logged as training data, which is increased as learning proceeds. Finally, we only retain the most recent 200 logged training items to use for training.

The next network to be trained is the simple action network. In each training trial a single object (the cylinder) is presented in one of a small number of training positions, as shown in Figure 2a. The trained reach network generates a goal motor state as usual; this is passed as input to the simple action network, along with a randomly selected simple action category (grasp, slap or punch). The output of the network is a perturbation, which is applied to the goal motor state. This output is again annealed with noise, which is reduced to zero as training progresses. The motor contoller then moves the hand in the direction of the perturbed goal motor state, and when the perturbation is removed, in the direction of the actual goal motor state. If the resulting movement activates one of the predefined



(a) (b)

Figure 2: Possible locations of target objects (a) during training; (b) during testing

classes of rich tactile signal, a training item is logged, mapping the actual goal motor state providing input to the netork, plus the simple action category corresponding to the activated tactile signal, onto the perturbation that was applied. After each trial, the network is trained on all the logged training items. This training algorithm is shown in Algorithm 2.

Algorithm 2: Learning simple motor programs

Data: $o = (o_x, o_y, o_z)$ (object center in retinal coordinates), $\theta_c(\theta_{c1}, \theta_{c2}, \theta_{c3})$ (current arm joint angles), d (perturbation removal distance)

begin

Reach algorithm and network used to determine $\theta_g = (\theta_{g1}, \theta_{g2}, \theta_{g3})$ (goal motor state);

while *Maximum time not exceeded* **do**

θ_g input into reach network to produce $\Delta = (\Delta_1, \Delta_2, \Delta_3)$ (perturbation of motor state);

$\theta_p = (\theta_{p1}, \theta_{p2}, \theta_{p3}) \leftarrow \theta_g + \Delta$ (perturbed goal motor state);

Random noise added to θ_p;

if *distance to object* $< d$ **then**

$\theta_p \leftarrow \theta_g$;

Calculate force applied to arm using PID controller (a function of $\theta_p - \theta_c$);

if *correct tactile feedback occurs* **then**

Store touch score, paired with θ_p;

if *Touch data recorded* **then**

Log the maximum touch score and corresponding θ_p, paired with o_r, as a new training item for the simple action network;

Maximum possible random noise reduced;

if *number of training items exceeds* 200 **then**

Discard oldest training item;

Simple action network trained on training data;

Finally the causative action network is trained. In each training trial either the squashable, bendable or openable object is presented in one of the locations given in Figure 2a. The plan activation network maps the object's category onto a planned sequence of two actions (a pair of two actions in the action planning system). The causative action network maps the the current goal motor state and the planned action sequence onto a sequence of two perturbations. (These outputs are again annealed with noise, reduced to zero during training.) The two perturbations drive the hand/arm along a particular trajectory. If this trajectory happens to result in the target object undergoing an action (bend, open or squash), the action recognition system will activate the relevant action category, and a training item is logged, mapping the current goal motor state, plus a unit in the action planning system corresponding to the recognised action category, onto the perturbation sequence. At the same time, the plan activation network learns to map the category of the target object onto the sequence 'cause', followed by the perceived action. This training regime is detailed in Algorithm 3.

Algorithm 3: Learning causative actions

Data: $o = (o_x, o_y, o_z)$ (object center in retinal coordinates),
$c \in \{lever, door, squashable\_object\}$ (object category), $\theta = (\theta_{c1}, \theta_{c2}, \theta_{c3})$ (current arm joint angles), d (perturbation removal distance)

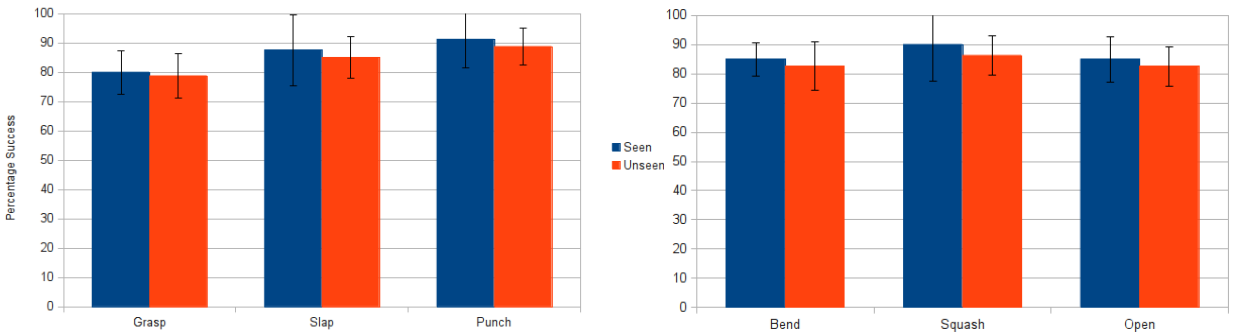
begin

Reach network maps o onto $\theta_g = (\theta_{g1}\theta_{g2}, \theta_{g3})$ (goal motor state);
Plan activation network maps c onto action plan $P = [P_1, P_2]$ $(P_1, P_2 \in \{squash, bend, open, cause\})$;
Causative action network maps θ_g, P onto $\Delta = (\Delta_1, \Delta_2, \Delta_3)$ (perturbation of motor state);
$\theta_p = (\theta_{p1}, \theta_{p2}, \theta_{p3}) \leftarrow \theta_g + \Delta$ (perturbed goal motor state);
Random noise added to θ_p;
while $\theta \neq \theta_p$ **do**
    Calculate force applied to arm using PID controller (a function of $\theta_p - \theta_c$);
Store θ_p as p_1;
θ_g input into causative action network to produce Δ;
$\theta_p \leftarrow \theta_g + \Delta$;
Random noise added to θ_p;
while *Maximum time not exceeded* **do**
    if *distance to object* $< d$ **then**
        $\theta_p \leftarrow \theta_g$;
    Calculate force applied to arm using PID controller (a function of $\theta_p - \theta_c$);
    if *Action recognition system detects an action a* **then**
        Store action score with $(\theta_{p1}, \theta_{p2}, \theta_{p3})$ as $p2$ paired with $p1$;
if *An action a was recognised* **then**
    Identify unit in action planning system a' corresponding to a;
    Retrieve maximum stored action score s_{max};
    Log a training item mapping a', θ onto the perturbation sequence (p_1, p_2) with learning constant s_{max};
    Reduce maximum possible random noise;
    if *Number of Training items* > 200 **then**
        Discard oldest training item;
    Causative action network trained on training data;
    Plan activation network trained to map c onto the planned sequence $[cause, a']$;

6 Testing the system's ability to learn simple/causative actions

Testing of the reach network is described in Lee-Hand *et al.* (2012). In this section we describe experiments testing the simple action and causative action networks.

We trained the two networks as described in Section 5. They were then tested in a series of trials; in each trial, a single target object was presented either in one of several possible unseen locations (see Figure 2b) or in one of the seen locations used during training (see Figure 2a). The trained simple action network was tested by presenting a cylinder at a selected location, activating a simple motor program at random (grasp, slap or punch), and observing how often the tactile stimulus associated with this motor program was produced. Results from these tests are summarised in Figure 3a. The causative action network was



(a) (b)

Figure 3: (a) Results from testing the simple action network. (b) Results from testing the causative action network. Error bars show 1 standard deviation.

tested in a similar way. In each trial, one of the articulated objects was presented at a selected location, and the causative network attempted to produce the causative action appropriate for this category of object. Results of these tests are presented in Figure 3b.

In general, the system was quite successful in producing motor actions with the expected perceptual consequences. The motor program network produced actions resulting in the expected tactile stimuli for X% of seen target locations and Y% of unseen locations; the causative action network produced actions resulting in the target undergoing the expected action in X% of seen locations and Y% of unseen locations. Illustrations of representative successful action of each type are shown in Figure 4. The cases where actions do not result in the expected sensory consequences can be accounted for by two main factors. Most failures are due to the simplicity of the feedback motor controller that moves the hand towards a goal state. The hand/arm system is subject to complex Coriolis forces when in motion, and there are limits to how precisely it can be controlled by a simple feedback controller. A few failures result from difficulties generalising from training locations to unseen locations, but in general the networks do this quite well.

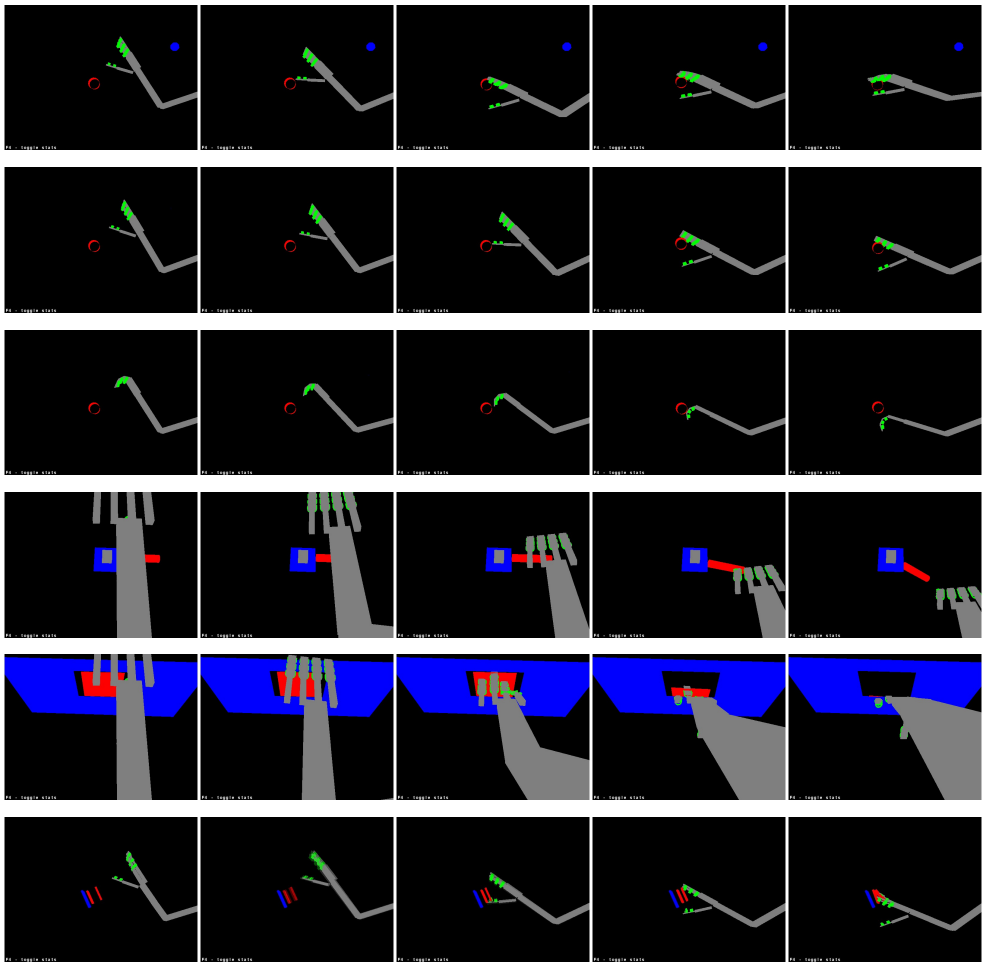


Figure 4: Learned actions. From top: grasping, slapping and punching a cylinder; bending a lever, opening a door and squashing a sprung plate. These sequences are taken from the latter stages of each action, when the hand makes contact with the target.

References

Averbeck, B., Chafee, M., Crowe, D., and Georgopoulos, A. (2002). Parallel processing of serial movements in prefrontal cortex. *PNAS*, **99**(20), 13172–13177.

Britten, K., Newsome, W., Shadlen, M., Celebrini, S., and Movshon, J. (1996). A relationship between behavioral choice and the visual responses of neurons in macaque mt. *Visual Neuroscience*, **13**(1), 87–100.

Lee-Hand, J. and Knott, A. (in submission). A neural network model of causative motor actions and causative alternation. Manuscript.

Lee-Hand, J., Neumegen, T., and Knott, A. (2012). Representing reach-to-grasp trajecto-

ries using perturbed goal motor states. In *Proceedings of the Pacific Rim Conference on Artificial Intelligence (PRICAI)*, pages 250–261.

Neumegen, T. (2013). A computational platform for simulating reach-to-grasp actions: modelling physics, touch receptors and motor control mechanisms. MSc thesis, Dept of Computer Science, University of Otago.

Oztop, E. and Arbib, M. (2002). Schema design and implementation of the grasp-related mirror neuron system. *Biological Cybernetics*, **87**, 116–140.

Robins, A. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, **7**, 301–329.

Takac, M. and Knott (2013). A neural network model of working memory for episodes. In *Proceedings of the 35th Annual Meeting of the Cognitive Science Society*, Berlin.