The motor-sensory primitives of random creatures

Emanuel Todorov, Zoubin Ghahramani
Gatsby Computational Neuroscience Unit
University College London
17 Queen Square, London WC1N 3AR, UK
{emo,zoubin}@gatsby.ucl.ac.uk

Abstract

The search for motor primitives has captured the attention of researches in both biological and computational motor control. Yet a theory of how to construct such primitives from first principles is lacking. Here we propose to do that by building a forward motor-sensory model of the world via unsupervised learning. The idea is applied to data collected from simulated creatures. We also propose a method of using such a forward model for control.

1 Introduction

Investigators of motor behavior have long been looking for motor primitives, or building blocks of movement. Candidates for such primitives include muscle synergies, spinal force fields, basis functions for representing internal models, small pieces of endpoint trajectories. So far the search has been purely data-driven: experimenters collect data (muscle activity, finger position, hand posture), apply some variant of principal components analysis, and declare anything that comes out to be a movement primitive. Unfortunately little independent evidence exists that the principal components correspond to anything real[1].

From a computational perspective building blocks of movement may seem less important, since in principle an optimal controller can be found using reinforcement learning, without any prior knowledge or control structure. In practice however such controllers cannot be found for high-dimensional continuous state systems. The only hope is to build some low-level control structure before applying reinforcement learning[3, 4, 5]. The problem is how to find that structure in the first place. It cannot come from considerations of optimal control, since it has to exist before optimal control can be attempted. Presently it comes from intuition about the specific problem - but what we really need is a way to automate that intuition and make it a part of the learning algorithm.

So in both biological and computational motor control, a theory is needed that explains how to find good motor primitives. The goal of this paper is to propose such a theory, and illustrate it with preliminary results. We will argue that the way

[1]Even if movements were generated by combining some low-level building blocks, this approach will not identify them unless the building blocks were sampled independently during the data collection period. But that is very unlikely, since the common behavioral goal will introduce dependencies. For example, we have found that the principal components of hand posture for the same subject can be very different depending on the task[2].

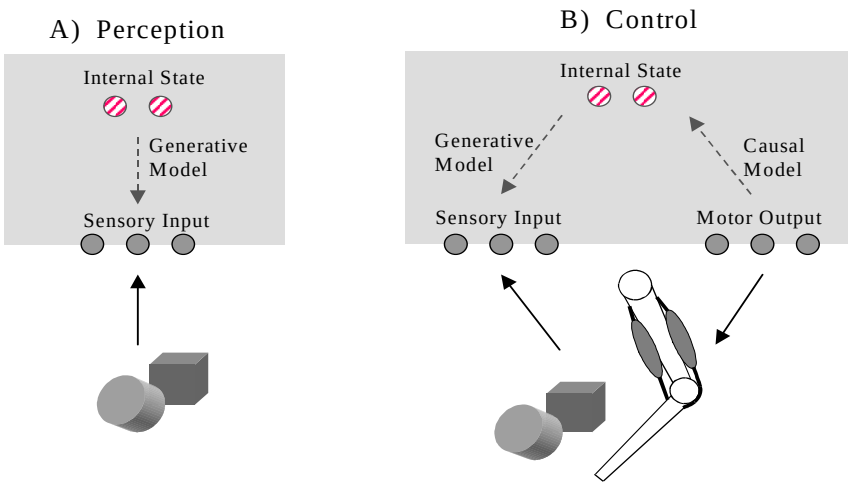

Figure 1: Models of unsupervised learning in the perceptual (A) and motor (B) systems. The gray box represents the nervous system, the solid arrows - events in the physical world.

to construct motor primitives is via unsupervised learning, where the system is not attempting to achieve any goals or maximize rewards, but is simply collecting data about the world and extracting the regularities present in that data.

The idea is best illustrated by comparison with the perceptual system, where the importance of unsupervised learning has been appreciated for a long time. The perceptual analog to a motor primitive is a receptive field: a basic element used to decompose and represent the sensory input[2]. Receptive fields can be obtained by collecting sensory data and fitting a "generative model" (Fig 1A). Note that the generative model as such is rather useless, since the job of the perceptual system is to perform exactly the opposite transformation.

We propose that motor primitives can be learned using the same principle - building an internal mirror image of physical world (Fig 1B). The only difference from perceptual learning is that we now have an input-output forward model, where the inputs (motor output) are under the control of the nervous system. The goal of learning is to find an internal state representation that explains the observed data. As in perception, the forward model is exactly the opposite to what the motor system needs, but the hope is that learning such a model first will enable the system to do its job later[3].

The motor output in Fig 1B is simply the sequence of muscle activations. What is the relevant sensory input? While the common task studied in motor control is reaching, and vision is typically the only sensory modality considered, we believe the approach needed here is completely different. The sensory input most directly related to motor behavior comes from muscle spindles (signalling muscle length and velocity), Golgi tendon organs (signalling force), joint capsule receptors (signalling joint limits), and tactile skin receptors. These sensory inputs are represented in

[2]As in the motor system, the receptive fields needed for perception cannot be easily derived from considerations of how an optimal perceptual system works. For example, the receptive field properties of V1 cells were described 40 years ago, yet we still don't know how to build a functional visual system - with or without those receptive fields. Instead the best available models are based on unsupervised learning (e.g. [6]).

[3]Unsupervised motor learning was previously proposed [1] in the opposite direction: as a control rather than forward model. Thus training data from the already functional nervous system was needed.

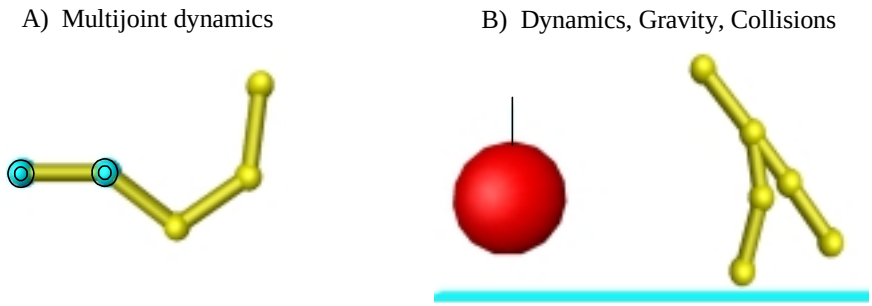

Figure 2: Two of the creatures we simulated.

a similar coordinate frame to the motor output, they enter the nervous system in the spinal cord and make numerous projections to spinal interneurons involved in muscle activation. Then they reach S1, which is densely interconnected with M1 - providing another direct link to motor output. Therefore we will forget about endpoint trajectories and visual targets for the time being, and focus on learning the more basic forward model described below.

2 Data

In order to learn a forward motor-sensory model, the nervous system needs sequences of muscle activation and the corresponding sensory feedback. In building a theory of how the nervous system learns, we would ideally have access to the same sequences available to the system. Such data is easily collected for models of visual or auditory learning - by walking around with a camera or a microphone. For our purposes, collecting "natural" data requires the following absurd experiment: anesthetize an animal, implant lots of stimulating electrodes in muscles and lots of recording electrodes in muscle, joint, and tactile sensors, stimulate randomly (with sufficient currents to evoke a wide range of body movements), and record sensor activity while the animal is jumping around[4]. Fortunately there is another way to collect "natural" data. We can simulate an "animal" equipped with realistic muscles, sensors, and a multijoint body interacting with a physical environment.

2.1 Physical simulation

The physical simulator was build with the MathEngine Toolkit. The bodies of our animals are made of cylinders with spherical ends, connected with hinge joints (see Figure 2). For now they are restricted to a 2D plane[5]. Some of the spheres can be fixed in the world (Figure 2A). The world can include gravity, ground, and external objects, i.e. spheres attached to spring-dampers whose other end is fixed (Figure 2B).

At the beginning of each (10 msec) time step the joint torques resulting from muscle activation (see below) are computed. All colliding pairs of objects are found, and the corresponding forces added. We use soft collisions (allowing some penetration) and friction in the directions orthogonal to the normal. The simulation is evolved using a semi-implicit integrator.

[4]Recording from awake behaving animals defeats the purpose - we need data that the nervous system presumably uses during learning, not data generated by the system after it has learned.

[5]The physical simulation is actually in 3D, and we are producing extra forces to avoid deviations from the plane. The 2D constraint is introduced for easier visualiztion, and also because we have not yet settled on a reasonable model of 3D muscle geometry.

2.2 Muscle simulation

We simulated an agonist-antagonist pair of muscles around each joint. The muscle model had constant moment arms, constant stiffness, and damping present only for shortening. Each muscle included one slow (50msec) and one fast (10msec) motor unit, modeled as second order linear filters. The fast motor unit had a fatigue level that increased in proportion to activation, and decreased exponentially with time. The two motor units were controlled by the activity of two (alpha) motor outputs - Slow and Fast. The Fast unit had bigger maximum force.

2.3 Sensor simulation

Each simulated muscle was equipped with one static spindle (measuring length), one dynamic spindle (measuring velocity), one Golgi tendon organ (measuring force). The spindle sensitivity was adjusted by two (gamma) motor outputs: Static and Dynamic. Each joint had 2 limit sensors. Each sphere and cylinder (Fig 2) had one tactile sensor on each side, which responded whenever a contact occurred on the corresponding surface. Since contacts involved some penetration, the sensor had graded response.

2.4 Algorithms

At each point in time we have controls $\mathbf{u}$, sensory feedback $\mathbf{y}$, and we want to infer a hidden state $\mathbf{x}$ (i.e. we are not interested in direct mappings from $\mathbf{u}$ to $\mathbf{y}$). One way to build the model is to represent the dynamics of $\mathbf{x}$ on the same time scale as the simulation. The problem with this approach is that it will not provide any temporal abstraction, and the true dynamics of $\mathbf{x}$ is not likely to be identified by existing algorithms anyway. Instead we will fit a stationary model, where the inputs and outputs contain data from several consecutive time steps. The simplest algorithm we can use is Canonical Correlation Analysis. Below we introduce two additional algorithms.

2.4.1 Conditional Factor Analysis

This is a generalization of factor analysis with inputs. The generative model is

$$\mathbf{x} = B\mathbf{u} + \mathbf{w}$$

$$\mathbf{y} = C\mathbf{x} + \mathbf{v}$$

where $\mathbf{w}$ and $\mathbf{v}$ are zero-mean Gaussian noise vectors with covariance Q, and R respectively. Like in ordinary factor analysis R is assumed to be diagonal. It is not difficult to derive an EM algorithm for learning B, C, Q and R.

E-step: The mean given $\mathbf{u}$ and $\mathbf{y}$ is $\hat{\mathbf{x}} = B\mathbf{u} + K(\mathbf{y} - C\mathbf{u})$, and the covariance matrix is $\Sigma = Q - KCQ$, where $K = QC^\top(R + CQC^\top)^{-1}$.

M-step: The model parameters are updated as $\hat{C} = (\sum_i \mathbf{y}_i \hat{\mathbf{x}}_i^\top)\Sigma^{-1}$ and $\hat{B} = (\sum_i \hat{\mathbf{x}}_i \mathbf{u}_i^\top)(\sum_i \mathbf{u}_i \mathbf{u}_i^\top)^{-1}$. Similarly updates can be obtained for Q and R as well.

2.4.2 Sparse Coding Model

We define a cost function (closely related to [6])

$$\mathcal{C} = (\mathbf{x} - B\mathbf{u})^\top(\mathbf{x} - B\mathbf{u}) + (\mathbf{y} - C\mathbf{x})^\top(\mathbf{y} - C\mathbf{x}) + \lambda s(\mathbf{x}) + \rho r(B) + \rho r(C)$$

The first two terms are proportional to the negative log likelihood terms under a Gaussian model for $\mathbf{x}$ given $\mathbf{u}$, and $\mathbf{y}$ given $\mathbf{x}$, as in conditional factor analysis (with $Q = I$ and $R = I$). The third term is an additional log prior on the hidden activities which can be used to code sparse representations. For example, $s(\mathbf{x}) = \sum_i |x_i|$ is the log of a Laplace prior. We used $s(\mathbf{x}) = \sum_i \log(\cosh(\beta x_i))/\beta$ corresponding to a cosh prior [7] which is smooth but can approximate the absolute value function for

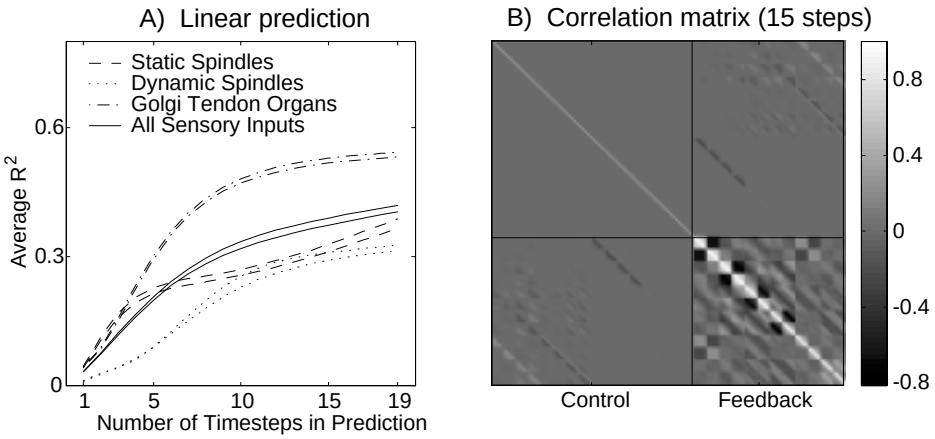

Figure 3: A) Linear regression of sensory feedback on sequences of previous control signals with different length. B) Cross-correlation matrix.

large β. The last two terms are log priors on the weights. Again we used the cosh prior.

Differentiating $\mathcal{C}$ we obtain coupled nonlinear differential equations for $\mathbf{x}$, B and C:

$$\dot{\mathbf{x}} = 2C^\top(\mathbf{y} - C\mathbf{x}) - 2(\mathbf{x} - B\mathbf{u}) - \lambda \tanh(\beta\mathbf{x})$$

$$\dot{B} = 2(\mathbf{x} - B\mathbf{u})\mathbf{u}^\top - \rho \tanh(\beta B)$$

$$\dot{C} = 2(\mathbf{y} - C\mathbf{x})\mathbf{x}^\top - \rho \tanh(\beta C)$$

Following these dynamics, $\mathbf{x}$ converges to the maximum a posteriori (MAP) estimate of the hidden states. This can be viewed as an approximation of the E step of EM. B and C will converge to approximate MAP estimates of the weight matrices, where the approximation comes from having used the MAP estimates of $\mathbf{x}$ rather than averaging over the posterior distribution of $\mathbf{x}$.

3 Results

We simulated different creatures driven by random control signals - 4 per muscle, sampled at each time step from a uniform distribution. The low-pass filtering properties of muscles and the system impedance (inertia of the skeleton and muscle viscoelasticity) transformed the noise activation sequence into relatively smooth movements. The simulation continued for about 20000 seconds of the life of each creature (much less in CPU time).

Since two of our models are linear, we computed how (linearly) predictable the sensory feedback is given the recent control signals. From Fig 3A we decided that 15 time steps (150 msec) is a reasonable amount of time to include in the stationary model.

Typical weight matrices are shown in Fig 4. The Canonical Correlation Analysis rarely found any meaningful structure - probably because of the orthogonality constraint. The sparse model found sparse structure on the sensory side (histograms of weights and hidden states confirmed that), but presently the input weight matrices are less sparse[6]. The Conditional Factor Analysis always produced nice patterns, many of which we could interpret. For example, notice the agonist-antagonist structure for both motor units and tactile receptors. This corresponds to the fact that a pair of muscles around a joint pull in opposite directions, and that contacts usually

[6]This appears to be due to scaling problems which we hope to fix.

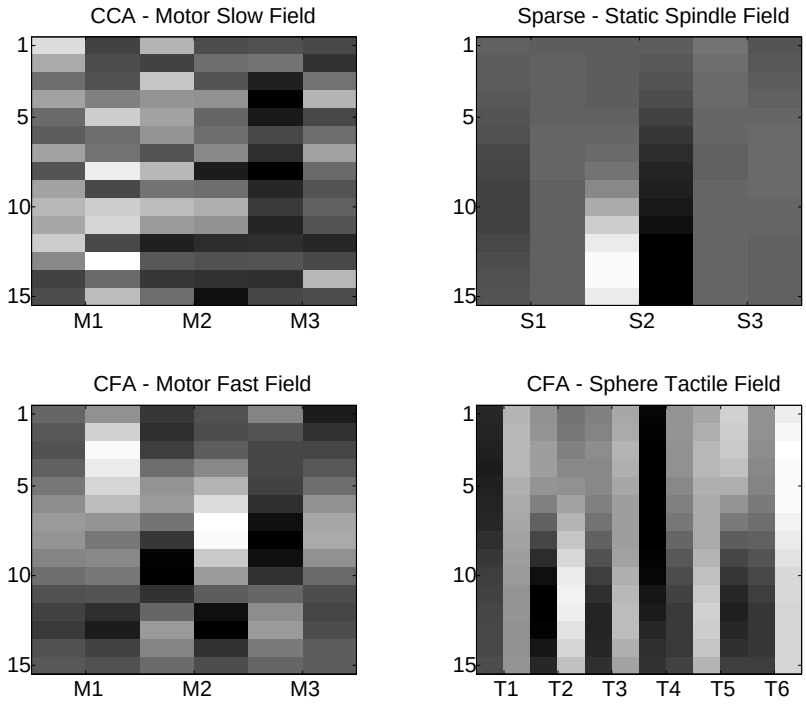

Figure 4: Typical weight matrices found by the different algorithms.

occur on one side or the other, but not both. Also, the motor weights were bigger at early time steps while sensory weights were bigger at late time steps - which captures temporal causality. The smooth progression of weights in the vertical direction corresponds to the fact that muscle action is temporally correlated (due to filtering) and so is sensory feedback. The weights for the fast muscles changed faster (in time) than the slow muscles, etc.

We are just beginning to analyze the results, and hoping to improve the sparse model. But it is already clear that a lot of structure can be captured by even a simple linear model such as conditional factor analysis, when applied to somewhat realistic data. Our hope is that some robust and unexpected prediction will emerge from this analysis, that would be testable.

4 From motor-sensory to sensory-motor primitives

Finally, how could the motor system use such a forward model to control behavior? One obvious possibility is to use the state estimate in higher levels of control (instead of processing again the raw sensory data). This would perhaps be useful, but the higher level controller would still have to deal with very detailed low level muscle commands.

Instead we propose the following control scheme. The forward model is used to set up a low-level sensory motor feedback loop, that can effectively transform the system into a qualitatively different one (the technique of feedback linearization is a good example - we are thinking of a more general feedback transformation). The resulting feedback loop is tunable; the high-level controller receives the state estimate as input, and sends control signals as desired states in the hidden state space. Then the low-level controller uses the forward model to compute the control signals most likely to achieve those states.

More formally, our conditional factor analysis model defines a probability distribu-

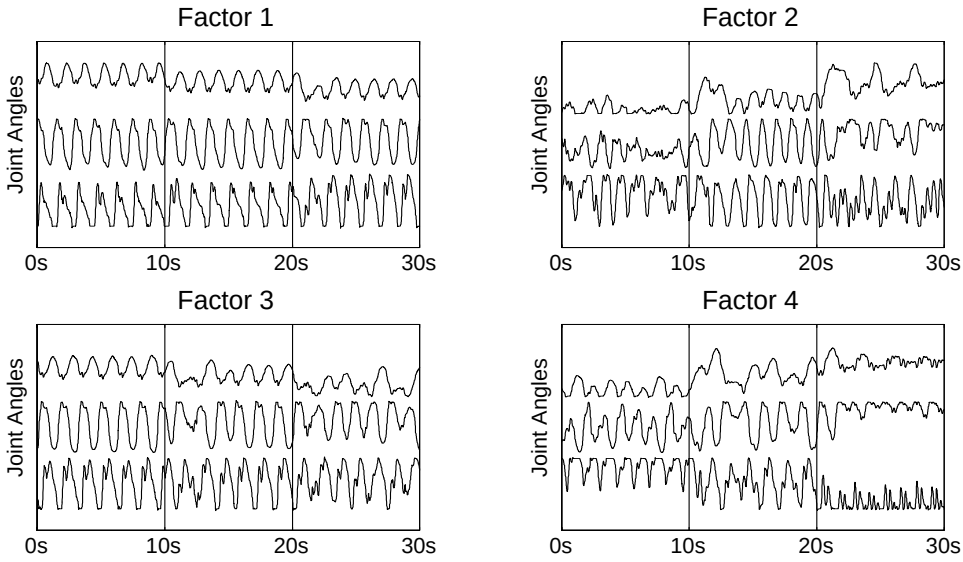

Figure 5: Effects of running different factors as feedback controllers, on the creature from Fig 2A. In each plot, we have clampled the value of the corresponding factor to a constant, which changes at the 10sec marks (from 1 to 0 to -1). The different constants elicit different rhytmic behaviors.

tion $P(\mathbf{u}_{1\ldots T}, \mathbf{x}, \mathbf{y}_{1\ldots T})$. At time $T-n$ we have access to $\mathbf{y}_{1\ldots T-n}$ and $\mathbf{u}_{1\ldots T-n}$. The high-level controller clamps some of the factors $\mathbf{x}_{i_{1\ldots k}}$ to desired values. Then the low-level controller computes $E[\mathbf{u}_{T-n+1}|\mathbf{u}_{1\ldots T-n}, \mathbf{y}_{1\ldots T-n}, \mathbf{x}_{i_{1\ldots k}}]$ and outputs that control signal. Whether this feedback transformation actually simplifies the job of the high-level controller remains to be seen (i.e. we plan to compare reinforcement learning with and without the transformation).

In the meantime, we studied the effects of the proposed transformation by simply clamping one of the factors to a constant and observing the resulting behavior. All the factors we clamped produced interesting-looking repetitive motions, and the system made a transition from one dynamic attractor to another (Fig 5) as the value of the factor was changed.

References

[1] Sanger, T. 1995. Optimal movement primitives. Advances in Neural Information Processing 7.

[2] Todorov, E. and Ghahramani, Z. 2000. Degrees of freedom and hand synergies in manipulation tasks. Neural Control of Movement 5.

[3] Crawford, L. 1998. Learning control for complex skills. UC Berkeley Ph.D. Thesis.

[4] Precup, D. and Sutton, R. 1998. Multi-time models for temporally abstract planning. Advances in Neural Information Processing 10.

[5] Parr, R. and Russell, S. 1998. Reinforcement learning with hierarchies of machines. Advances in Neural Information Processing 10.

[6] Olshausen, B. and Field, D. 1996. Emergence of simple-cell receptive-field properties by learning a sparse code for natural images. Nature, 381:607.

[7] Lewicki, M. and Sejnowki, T. 1998. Learning nonlinear overcomplete representations for efficient coding. In Advances in Neural Information Processing 10.