Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Efficient Codes for Inverse Dynamics During Walking

Leif Johnson and **Dana H. Ballard**
Computer Science Department
The University of Texas at Austin
{leif,dana}@cs.utexas.edu

Abstract

Efficient codes have been used effectively in both computer science and neuroscience to better understand the information processing in visual and auditory encoding and discrimination tasks. In this paper, we explore the use of efficient codes for representing information relevant to human movements during locomotion. Specifically, we apply motion capture data to a physical model of the human skeleton to compute joint angles (inverse kinematics) and joint torques (inverse dynamics); then, by treating the resulting paired dataset as a supervised regression problem, we investigate the effect of sparsity in mapping from angles to torques. The results of our investigation suggest that sparse codes can indeed represent salient features of both the kinematic and dynamic views of human locomotion movements. However, sparsity appears to be only one parameter in building a model of inverse dynamics; we also show that the "encoding" process benefits significantly by integrating with the "regression" process for this task. In addition, we show that, for this task, simple coding and decoding methods are not sufficient to model the extremely complex inverse dynamics mapping. Finally, we use our results to argue that representations of movement are critical to modeling and understanding these movements.

Introduction

Research in computational motor control has suggested that human movements might lie along a relatively low-dimensional manifold embedded in the space of all possible movements (Latash, Scholz, and Schöner 2002). Not only do different aspects of a movement appear to be controlled differently based on the demands of a particular task (Scholz and Schöner 1999), but also the movements that a human typically makes do not appear to occupy the space of all possible human movements with equal probability. If the information contained in a movement actually does occupy a low-dimensionality manifold within the space of all possible joint angle changes, then this non-uniformity should be helpful in performing common movement analysis tasks such as predicting the outcome or identifying the "components" of a movement. Furthermore, identifying subspaces of movement might be useful when performing more complex analysis or even synthesis techniques such as inverse dynamics.

In recent decades, research spanning computational neuroscience (Olshausen and Field 1996), statistics (Tibshirani 1996) and feature learning (Bengio 2013) has extended our understanding of standard dimensionality reduction techniques like Principal Component Analysis (PCA), placing dimensionality reduction in a broader context that includes more general techniques for learning the shape and orientation of data manifolds. One view that has emerged consistently from this work is the idea that sparsity, or redundancy reduction (Attneave 1954; Barlow 1961), appears to be an important principle for representing information sampled from complex, real-world datasets (Olshausen and Field 2004; Smith and Lewicki 2006; Lee et al. 2007; Lee, Ekanadham, and Ng 2008; Le et al. 2011; Coates, Lee, and Ng 2011). However, the lion's share of this work to date has focused on analyzing visual data, often for image classification purposes.

In this paper, we contribute a perspective on feature learning from an atypical domain (movement information) and for an atypical purpose (regression for inverse dynamics). We investigate whether efficient coding might be useful for (a) representing data gathered from human movements, and for (b) performing a computational regression task instead of classification. The question we seek to address is, if one thinks of the human brain as a machine for producing torques that move the body through a sequence of desired postures, then what sorts of information processing techniques might be useful for performing this mapping? After describing the data, computational models, and results, we discuss this question further to conclude the paper.

Motion Data Analysis

To gather movement data, we used a 16–camera Phasespace [1] motion-capture system in conjunction with a standard treadmill (Figure 1). Human subjects in the motion tracking area wore a full-body suit equipped with active-pulse LED motion tracking markers and were recorded as they walked and ran on the treadmill at a variety of speeds.

For this paper, we recorded the positions of $M = 48$ markers from one subject as he walked at speeds ranging from 0.22 to 2.68 m/s for a duration of approximately twenty minutes. We configured the Phasespace system to produce

[1] phasespace.com/impulse_motion_capture.html

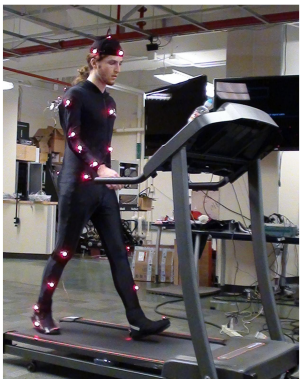

Figure 1: The motion capture environment consists of a full-body motion capture suit (black, with red LEDs), and a treadmill centered in the motion capture space.

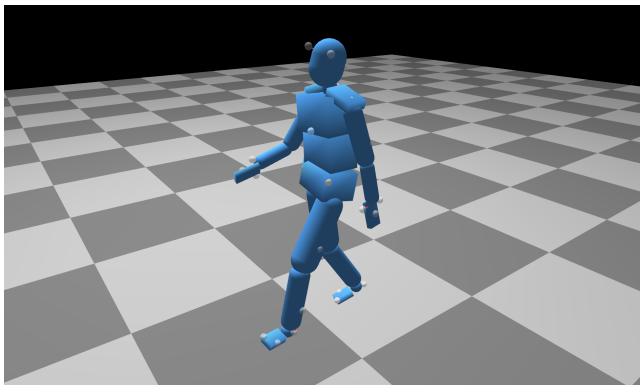

Figure 2: Our implementation of the articulated model described by (Cooper and Ballard 2012) constrains a skeleton composed of rigid bodies (blue) to move in a way that mimics the observed motion capture data (small white spheres).

frames of motion capture data at a rate of 120Hz, so the recording resulted in $N > 120,000$ frames of raw motion-capture data. Recorded frames were processed using an articulated forward model for computing inverse kinematics and dynamics (Cooper and Ballard 2012). Briefly, this model converts motion-capture marker data into kinematic (joint angle) and dynamic (joint torque) representations of the motion, by using an off-the-shelf physics simulator [2] to constrain an articulated skeleton model to follow the observed marker data as closely as possible. The articulated model that we used (Figure 2) consists of 21 rigid segments joined together with kinematic joints containing a total of 42 degrees of freedom. The skeleton model was configured to the motion data so that the model's range of motion captured the range of motion of the recorded human movements without visible jitter.

Motion-capture data were processed using this model in two passes. First, the articulated model was constrained to mimic the sequence of recorded poses in the motion-capture data as closely as possible, converting the raw motion-capture data into a series of frames of joint angles

$$\mathcal{A} = [\,\alpha(1) \ldots \alpha(N)\,]$$

where each $\alpha(t)$ is the vector of 42 real-valued angle measurements computed at frame t. In the second pass, the articulated model was constrained to follow the computed joint angles as closely as possible, resulting in a sequence of frames of joint torques

$$\mathcal{T} = [\tau(1) \ldots \tau(N)]$$

that the model was required to employ to match the target sequence of joint angles. The computed joint torques were "active" in the sense that the skeleton needed these torques in addition to the "passive" forces provided by gravity and the inertia of the articulated model itself.

Windowing

After processing the raw motion-capture data to obtain $\mathcal{A}$ and $\mathcal{T}$, we extracted 65k windows of motion from these datasets to train and test our models. We first chose a window length L—for these simulations we chose $L = 60$, spanning 500 milliseconds—and then extracted random subsequences of length L from the computed joint angles and torques. This resulted in a dataset of joint angles

$$\mathbf{A} = \left[\mathbf{a}^{(1)} \ldots \mathbf{a}^{(K)}\right]$$

where each of the vectors

$$\mathbf{a}^{(i)} = [\,\alpha(r) \ldots \alpha(r+L)\,]^\top$$

is a concatenation of the joint angles in each frame of the subsequence starting at frame r. The torque dataset

$$\mathbf{T} = \left[\mathbf{t}^{(1)} \ldots \mathbf{t}^{(K)}\right]$$

was constructed in the same way using consecutive windows from $\tau(r)$ to $\tau(r+L)$. Importantly, the torque windows were extracted from the same frame offsets as the angle windows, so that the angle changes captured in sample $\mathbf{a}^{(i)}$ occurred at the same point in time as the torque changes captured in sample $\mathbf{t}^{(i)}$. This dataset, constructed of joint angle values over time paired with torque values over time, could thus, in theory, be used to learn a model of the inverse dynamics of the skeleton model while walking like the originally-recorded human subject.

Whitening

We first whitened the extracted joint angles and joint torques to remove the global correlations present in each modality of data. We mean-centered and whitened each modality of data using PCA (Figures 3 and 4), retaining sufficiently many components to explain 99% of the variance of the data, and ensuring that the covariance of the whitened data was approximately equal to the identity matrix. Interestingly, the joint angle data were highly redundant and compressed from a raw size of 2520 dimensions down to the first 91 principal components (a compression of 96%); the first principal component in the joint angle dataset explained nearly 50% of

[2] http://www.ode.org

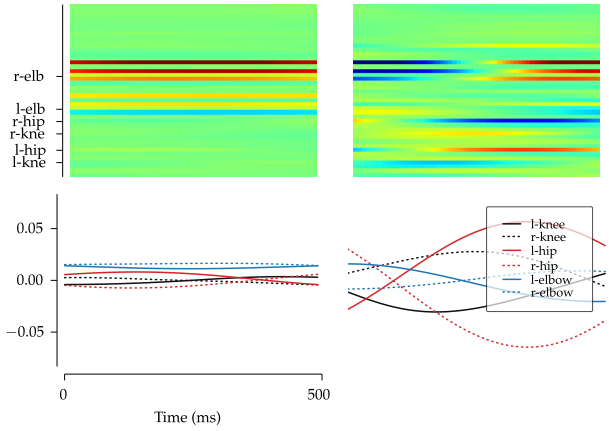

Figure 3: The second (left) and third (right) eigenvectors for joint angle data. The top row of plots shows each codebook element as a full spectrogram containing all 42 degrees of freedom for the 500ms time span of the windows in our dataset. The bottom row of plots shows six individual channels (x-rotation of the knee, hip and elbow on the left and right sides) from the spectrograms, meant to highlight patterns across multiple degrees of freedom in walking movements. Interestingly, the first and second principal components of variance primarily reflect joint angle configurations of the upper arm, but the third reflects the periodic linkage between knee, hip, and elbow.

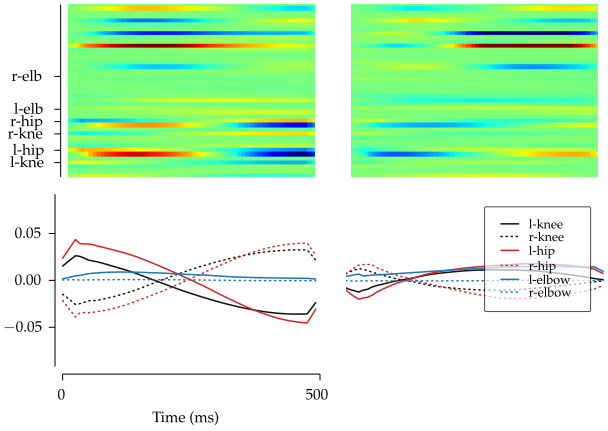

Figure 4: The first (left) and second (right) eigenvectors for joint torque data. See Figure 3 for a description of the plots.

the variance in the data! The joint torques also compressed significantly, down to the first 392 principal components (a compression of nearly 85%).

Finally, we set aside 10% of the whitened data to use for testing; the whitened windows of joint angles and torques in the testing dataset were only used to evaluate performance after training the models had been completed.

Computational Models

Mapping from joint angles to joint torques is not an easy task. To make some progress, we follow an existing decomposition of the problem (Johnson, Cooper, and Ballard 2013) into three stages: encoding of joint angles, regression from a coded set of joint angles to a coded set of joint torques, and finally decoding of the encoded torques. This architecture provides a mechanism for comparing different approaches to the inverse dynamics task, in addition to isolating sources of error in the process. Although similar in principle to canonical correlation analysis (Hotelling 1936), we see this breakdown of the problem as a variant of manifold or feature learning, in which manifolds learned in each data modality individually might be useful for performing regression tasks like inverse computations.

Isolated Coding

PCA is widely used for data preprocessing, but using PCA alone as an encoding technique often results in "dense" representations of data that can be difficult for humans to interpret. In contrast, sparse coding explicitly seeks a representation of data $\mathbf{Z}$ that uses as few elements of a given codebook $\mathbf{D}$ as possible to reconstruct the data $\mathbf{X}$. This goal can be formulated as an optimization problem:

$$\mathbf{Z} = \arg\min_{\mathbf{z}} \|\mathbf{D}\mathbf{z} - \mathbf{X}\|_2^2 + \lambda\|\mathbf{z}\|_1$$

Several extensions of this problem setting have been proposed to learn both the dictionary and the encoding using just a dataset $\mathbf{X}$ (Smith and Lewicki 2006; Mairal et al. 2009; Le et al. 2011); for this work we adopt a formulation of the problem that factors the dictionary into the product of a codebook matrix $\mathbf{W}$ with itself (Le et al. 2011):

$$\mathbf{W} = \arg\min_{\mathbf{w}} \|\mathbf{w}\mathbf{w}^\top\mathbf{X} - \mathbf{X}\|_2^2 + \lambda\|\mathbf{w}\mathbf{X}\|_1$$

In this formulation, which is relatively easy to optimize because of the linear coding and decoding process, the size of the codebook $\mathbf{W}$ and the degree of sparsity λ can be varied to evaluate the effectiveness of codes of various sizes. We used the `theanets` Python package [3] for defining and optimizing the appropriate losses.

Regression

Once codes have been computed for joint angles and joint torques, it remains to compute a map between the coded spaces. Ideally, the features identified by the isolated coding process would capture parts of the data that are invariant under some small transforms (scaling, translation, etc.), and thus regression between coded spaces should be easier than between the raw datasets. To test this hypothesis, we used ridge regression (Hoerl and Kennard 1970) to compute a weight matrix $\mathbf{R}$ that satisfies a constrained linear reconstruction loss between coded datasets $\mathbf{X}$ and $\mathbf{Y}$:

$$\mathbf{R} = \arg\min_{\mathbf{r}} \|\mathbf{X}\mathbf{r} - \mathbf{Y}\|_2^2 + \gamma\|\mathbf{r}\|_2^2$$

[3] http://github.com/lmjohns3/theano-nets

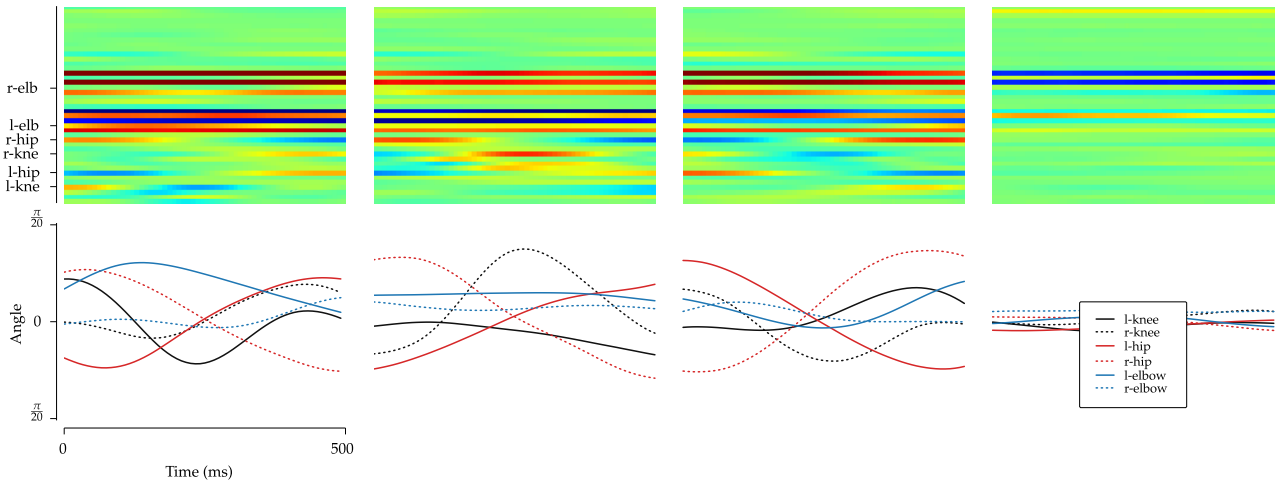

Figure 5: Four independent joint angle codebook elements learned using sparse reconstruction. (See Figure 3 for a description of the plots.) The first three examples show common stride patterns (e.g., left-right alternation, hip and knee phase coupling). In the example on the far right, the joint angles for hip, knee, and elbow are largely irrelevant since this codebook element focuses largely on the shoulder channels (dark blue stripes in the spectrogram).

We used implementations of these algorithms provided by `scikit-learn` (Pedregosa et al. 2011), a popular Python machine learning toolkit.

Effectively, the models described so far have all been linear, which could significantly hinder the performance of the inverse mapping problem at hand: a parametric map from joint angles to torques, should a tractable one exist, is surely not going to be linear! To test this hypothesis, we also implemented a nonlinear regression using a neural network containing a single hidden layer. In the form of a loss, the neural network is attempting to find parameter matrices $\mathbf{A}$ and $\mathbf{B}$ to minimize the prediction cost:

$$\mathbf{A}, \mathbf{B} = \arg\min_{\mathbf{a},\mathbf{b}} \|\mathbf{b}g(\mathbf{aX}) - \mathbf{Y}\|_2^2$$

Note here that twice as many parameters are available as in ridge regression, which led to significant overfitting in our tests. We once again used `theanets` with stochastic gradient descent and early stopping to train our model.

Unified Models

Rather than compute separate codebooks for angles and torques in isolation, and then use only the coded spaces in a regression setting, we also investigated whether performance could be improved by incorporating the three transforms (encoding, regression, and decoding) into a single, unified loss. Optimizing a unified loss would theoretically permit errors in each phase of data processing to be shared with the other stages, encouraging representations that are optimal for the inverse mapping task at hand, rather than for specific subtasks like reconstructing data from one modality.

$$\mathbf{E}, \mathbf{R}, \mathbf{D} = \arg\min_{\mathbf{e},\mathbf{r},\mathbf{d}} \|\mathbf{d}g(\mathbf{r}g(\mathbf{eA})) - \mathbf{T}\|_2^2 + \lambda\|g(\mathbf{eA})\|_1 + \lambda\|g(\mathbf{r}g(\mathbf{eA}))\|_1$$

In addition, unifying the loss for the entire model permits more natural introduction of nonlinearities; given that the inverse modeling problem appears to require nonlinear analysis, we used rectified linear activations (Nair and Hinton 2010; Glorot, Bordes, and Bengio 2011) on all hidden neurons in this model. We used the `theanets` package to define and optimize the appropriate losses for this model.

Results

Learned Features

Features learned from angle data (Figure 5) and torque data (Figure 6) show some interesting behavior that is distinct from the behavior of the corresponding principal components (Figures 3 and 4, respectively). In the angle data, the principal components either reflect significant single-DOF patterns or smooth, symmetric variations in angle change across a variety of degrees of freedom. The learned, sparse features, however, reveal more distinct patterns of variation in the relationships between different degrees of freedom. For example, the knee tends to exhibit a complex motion during walking that can be expressed as the sum of two sinusoids, but this complex motion shows up as one of the learned sparse features of the angle data. In addition, learned angle features tend to have repetitive forms (e.g., the motion of the hip joint) with varied phase relationships between different degrees of freedom. The sparse features are also able to isolate degrees of freedom that tend to be co-active from those that vary significantly on their own; for example, see the rightmost column in Figure 5.

In the torque data, the principal components follow a similar pattern, with the first principal component expressing low-frequency, simultaneous variation in torque across a small number of covarying channels. The learned features, however, exhibit much more ”ringing” and more complex

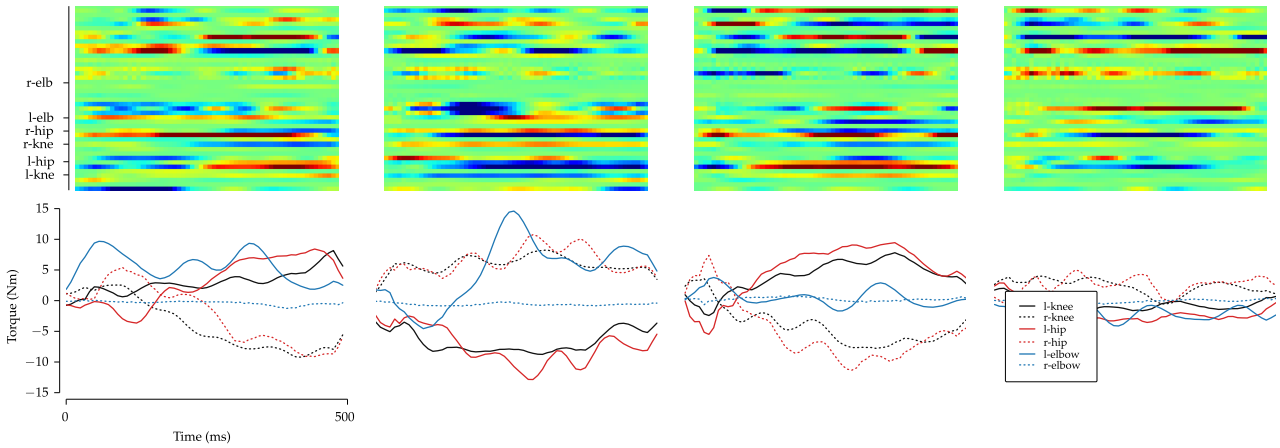

Figure 6: Four independent joint torque codebook elements learned using sparse reconstruction. (See Figure 5 for a description of the types of plots.) Torque features learned in this way exhibit some clear ringing effects.

phase relationships; these features, despite being learned in a sparse model, are more difficult to understand intuitively than the angle features, perhaps because, unlike joint angles, which directly describe the motion of different joints, it is difficult to visualize what a "torque feature" means for an articulated model. Nevertheless, it is clear just from inspection of Figure 6 that the sparsely learned torques exhibit visible ringing, which might play a role in the overall accumulation of error in the inverse dynamics mapping task.

Torque Error

Our torque prediction results, shown in Figure 7, show clearly how difficult it is to map from joint angles to joint torques. Even the best model, the unified loss with a nonlinearity, was only able to reduce torque error to slightly less than 60 Nm. (For comparison, a null model consisting of all zeros resulted in an RMS torque error of 100.5 Nm.) Given that our physics simulator—or any articulated body composed of rigid segments with low compliance—is extremely sensitive to perturbations in joint torques, this level of error would not permit a simulated skeleton to make anything resembling natural movements using the predicted torques.

Furthermore, the other models that were evaluated here performed even worse than the unified, nonlinear one. Using linear feature projections and a linear regression between codes, we would expect to do no better than straight linear regression between the whitened angles and whitened torques, and this is indeed what we found: baseline regression directly between whitened datasets incurred an RMS error of approximately 75.2 Nm, and the isolated models with linear regression asymptoted at this error level.

Finally, nonlinear regression between the linear features had erratic performance on this task. The nonlinearity appeared to help the model for a codebook with up to 512 elements, when compared with linear codebooks of the same size, but otherwise the nonlinearity actually hurt performance. This could have been due to lack of appropriate hyperparameter tuning—more important with larger codebooks—but the error behavior was consistent across several settings of hidden activation sparsity, and remained when trained using the same sparsity settings as all other model components discussed here. Regardless, linear features trained in isolation did not surpass the performance of a fully nonlinear regression model.

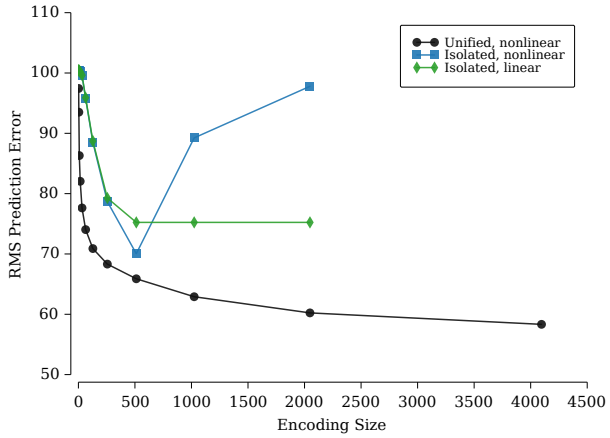

Figure 7: Mean RMS error for joint torques under different modeling conditions. The baseline, mapping directly from whitened angles to whitened torques, resulted in an RMS error of 75.20, which is equal to the asymptote visible in the graph for the "isolated, linear" series.

Discussion

The work described here sheds light on modeling human locomotion movements in several ways. Our results support existing evidence (Johnson, Cooper, and Ballard 2013) that computational models in single modalities, even mo-

tor modalities, can learn something useful about joint angles and joint torques, when these quantities are treated as "just data." However, even with a nonlinear, unified neural network model, the inverse dynamics regression problem of mapping joint angles to joint torques is extremely difficult to solve. The relatively large errors that we have reported here might be discouraging for those hoping to learn joint control as a feedforward mapping process, even over very short time intervals. In short, to return to the question posed at the beginning of the paper, if the brain is a machine for transforming sequences of target postures into sequences of movements that achieve those targets, then it must be performing this mapping task using a model that is more sophisticated than the ones examined here.

Despite this overall answer to the question, the results reported here also provide some small guidance for the general task of modeling human motion data. We have shown that, to a point, the vectors of joint angles and joint torques that we extract from our model can indeed be treated as "just data" and modeled in just the same way as, say, visual or audio data. Indeed, if the modeler is willing to devote enough resources to the problem, the inverse mapping task might be more tractable than it appears in our results, since the RMS error of the unified, nonlinear model shrinks by a constant amount for every doubling of the codebook size, over the range of codebook sizes that we tested. Put another way, the asymptote in model performance that one would expect to see eventually was not identified in the range of parameter settings that we explored here, and so, with a sufficiently large codebook, an inverse mapping might be possible.

However, this way of looking at the problem of modeling movements actually reveals a more fundamental flaw with the idea that joint torques can be modeled as feedforward computations using "just data." Instead of supposing that an enormous model is required to predict joint torques across large numbers of degrees of freedom and for many time steps, we see the result presented here as evidence against feedforward mappings for this task, in favor of using more explicit time-based representations of movement (Schaal, Ijspeert, and Billard 2003; Schaal et al. 2003; Taylor, Hinton, and Roweis 2007; Kulic, Takano, and Nakamura 2007). In particular, feedback during movement seems to be a critical aspect of the dynamical system that was not incorporated in the models investigated here.

Learning appropriate feature spaces in feedback processes—particularly in nonlinear systems—remains an important and largely unsolved problem, though there is much interesting work in this area. In particular, Gaussian Processes (Rasmussen 2006) have been used successfully to model walking movements in humans across a variety of contexts (Wang, Fleet, and Hertzmann 2008; Calandra et al. 2014), as well as a general-purpose model of dynamics in reinforcement learning settings (Deisenroth and Rasmussen 2011). Although Gaussian Processes and neural networks like the unified model are closely related (MacKay 1998), the Gaussian Process as a prior over smooth functions is an elegant solution to modeling movement information.

Along these lines, we think the models presented here have particular promise in two areas of modeling human movements. first, by coming up with a prediction of joint torques, even if the prediction is not extremely accurate, feedforward models might be usefully embedded inside a distal learning framework (Jordan and Rumelhart 1992) and then used to learn a more accurate model of the inverse problem. Second, the codebooks for joint angles and joint torques, while not solving the regression problem on their own, could be useful for segmenting motion data into chunks, both in time (akin to "options" (Sutton, Precup, and Singh 1999)) and across the available degrees of freedom (akin to joint "synergies" (Latash, Scholz, and Schöner 2002)). These chunks might then be modeled independently using more sophisticated techniques that explicitly incorporate dynamics using time or feedback.

Acknowledgements

We would like to acknowledge the anonymous reviewers' time and thoughtful comments, as their input helped to improve this paper.

References

Attneave, F. 1954. Some informational aspects of visual perception. *Psychological review* 61(3):183.

Barlow, H. 1961. Possible principles underlying the transformation of sensory messages. *Sensory Communication* 217–234.

Bengio, Y. 2013. Deep learning of representations: Looking forward. *arXiv preprint arXiv:1305.0445*.

Calandra, R.; Rasmussen, C.; Peters, J.; and Deisenroth, M. 2014. Manifold gaussian processes for regression. *arXiv preprint arXiv:1402.5876*.

Coates, A.; Lee, H.; and Ng, A. 2011. An analysis of single-layer networks in unsupervised feature learning. In *Proc. Intl. Conf. on Artificial Intelligence and Statistics*, volume 1001, 48109.

Cooper, J., and Ballard, D. 2012. Realtime, physics-based marker following. In *Proc. Motion in Games*, 350–361. Springer.

Deisenroth, M., and Rasmussen, C. 2011. PILCO: A model-based and data-efficient approach to policy search. In *Proc. 28th Intl. Conf. on Machine Learning*.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *Proc. 14th Intl. Conf. on Artificial Intelligence and Statistics*.

Hoerl, A., and Kennard, R. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* 12(1):55–67.

Hotelling, H. 1936. Relations between two sets of variates. *Biometrika* 28(3-4):321–377.

Johnson, L.; Cooper, J.; and Ballard, D. 2013. Unified loss functions for multi-modal pose regression. In *Proc. Intl. Joint Conf. on Neural Networks*.

Jordan, M., and Rumelhart, D. 1992. Forward models: Supervised learning with a distal teacher. *Cognitive Science* 16(3):307–354.

Kulic, D.; Takano, W.; and Nakamura, Y. 2007. Incremental on-line hierarchical clustering of whole body motion patterns. In *Proc. 16th IEEE Intl. Symposium on Robot and Human Interactive Communication*, 1016–1021. IEEE.

Latash, M.; Scholz, J.; and Schöner, G. 2002. Motor control strategies revealed in the structure of motor variability. *Exercise and Sport Sciences Reviews* 30(1):26–31.

Le, Q.; Karpenko, A.; Ngiam, J.; and Ng, A. 2011. ICA with reconstruction cost for efficient overcomplete feature learning. *Advances in Neural Information Processing Systems*.

Lee, H.; Battle, A.; Raina, R.; and Ng, A. 2007. Efficient sparse coding algorithms. *Advances in Neural Information Processing Systems* 19:801.

Lee, H.; Ekanadham, C.; and Ng, A. 2008. Sparse deep belief net model for visual area V2. *Advances in Neural Information Processing Systems* 20.

MacKay, D. 1998. Introduction to gaussian processes. *NATO ASI Series F Computer and Systems Sciences* 168:133–166.

Mairal, J.; Bach, F.; Ponce, J.; and Sapiro, G. 2009. Online dictionary learning for sparse coding. In *Proc. 26th Intl. Conf. on Machine Learning*, 689–696. ACM.

Nair, V., and Hinton, G. 2010. Rectified linear units improve restricted boltzmann machines. In *Proc. 27th Intl. Conf. on Machine Learning*, 807–814. Omnipress Madison, WI.

Olshausen, B., and Field, D. 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381(6583):607–609.

Olshausen, B., and Field, D. 2004. Sparse coding of sensory inputs. *Current Opinion in Neurobiology* 14(4):481–487.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, É. 2011. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research* 12:2825–2830.

Rasmussen, C. 2006. *Gaussian processes for machine learning*. Citeseer.

Schaal, S.; Peters, J.; Nakanishi, J.; and Ijspeert, A. 2003. Learning movement primitives. In *Proc. Intl. Symposium on Robotics Research*. Springer.

Schaal, S.; Ijspeert, A.; and Billard, A. 2003. Computational approaches to motor learning by imitation. *Philosophical Transactions: Biological Sciences* 358(1431):537–547.

Scholz, J., and Schöner, G. 1999. The uncontrolled manifold concept: identifying control variables for a functional task. *Experimental Brain Research* 126(3):289–306.

Smith, E., and Lewicki, M. 2006. Efficient auditory coding. *Nature* 439(7079):978–982.

Sutton, R.; Precup, D.; and Singh, S. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112(1):181–211.

Taylor, G.; Hinton, G.; and Roweis, S. 2007. Modeling human motion using binary latent variables. *Advances in Neural Information Processing Systems* 19:1345.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 267–288.

Wang, J.; Fleet, D.; and Hertzmann, A. 2008. Gaussian process dynamical models for human motion. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 30(2):283–298.