Efficient codes for multi-modal pose regression

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

Abstract

Redundancy reduction, or sparsity, appears to be an important information-theoretic principle for encoding natural sensory data. While sparse codes have been the subject of much recent research, they have primarily been evaluated using readily available datasets of natural images and sounds. In comparison, relatively little work has investigated the use of sparse codes for representing information about human movements and poses. This paper proposes a basic architecture for evaluating the impact of sparsity when coding human poses, and tests the performance of several coding methods within this framework for the task of mapping from a kinematic (joint angle) modality to a dynamic (joint torque) one. We show that sparse codes are indeed useful for effective mappings between modalities and examine in detail the sources of error for each stage in the model.

Overview

Recent work from machine learning (Ranzato, Boureau, & LeCun, 2007; Lee, Battle, Raina, & Ng, 2007) and neuroscience (Olshausen & Field, 1996; Smith & Lewicki, 2006) has emphasized the role that sparsity, or redundancy reduction (Barlow, 1961), appears to play when coding sensory data. Sparse codes are well suited to represent natural sensory data because the space of all possible inputs (e.g., all possible 1000×1000 images) is not uniformly covered by the samples (e.g., photos or retinal inputs) that we tend to encounter in the natural world. In fact, many types of natural data are theorized to lie along a low-dimensional manifold embedded in the larger space (Olshausen & Field, 2004). Sparse codes are effective for representing data along such low-dimensional manifolds because the basis vectors in the code can be used efficiently (i.e., using just a few nonzero coefficients) to indicate, for a particular data point, its location along the manifold rather than its coordinates in the higher-dimensional space.

Many of the results in this area of research have focused on codes for sensory information like images and sounds. Concurrently, research in control theory has suggested that human movements might also lie along a relatively low-dimensional manifold embedded in the space of all possible movements (Scholz & Schöner, 1999; Latash, Scholz, & Schöner, 2002). Sparse codes might be useful, then, for representing information about movement and pose in humans. To our knowledge, however, such codes have not been evaluated extensively on natural movement or pose data.

This paper proposes a basic architecture for testing the effectiveness of a broad class of coding techniques when mapping from kinematic (joint angle) to dynamic (torque) data in human poses. While computationally straightforward, the model allows us to compare and evaluate several possible approaches to this coding and regression task. We show that, for the class of techniques captured by our model, sparsity is indeed useful for representing and manipulating pose data. In fact, even though the absolute decoding error associated with sparse codes can be larger than the corresponding absolute error for dense codes, the information captured by each coefficient in a sparse code is larger than for dense codes. In addition, sparse codes appear to facilitate the task of mapping or regressing from one information modality to another, making these codes particularly interesting from the perspective of a whole organism, which must integrate information from many different sources of information to make effective survival decisions.

Problem setting

The human body is marvelously complex, with over 630 muscles and, by some estimates, more than 240 degrees of freedom (Winter, 2009; Zatsiorsky & Prilutsky, 2012). Despite this complexity, humans are skilled at controlling their bodies to make movements that accomplish a wide variety of tasks in the world. Humans also seem to be skilled at transforming information about movement between different modalities. For example, a person can normally mimic the posture of another person without much conscious effort, even though this task requires some sort of conversion from the visual configuration of the conspecific's body (possibly expressed in world or visual coordinates) to the kinematic configuration of their own. Along these lines, it is conceivable that the tasks of computing potential movements, evaluating proposed movements, and selecting and executing a particular movement all require separate ways of looking at the movements.

Studying human movements is difficult: the parameters describing movements are high-dimensional and time-varying, and, in addition, most of the quantities that are relevant for describing the control of movement are invisible to an outside observer. Although we do not have a practical way to observe the control signals or even accurately measure all of the joint torques or angles during a complex, multijoint human movement, we can use technologies like motion capture (Figure 2) to measure the external aspects of movement with high accuracy. Given motion capture data, Cooper and Ballard (2012) proposed a technique to compute the angles

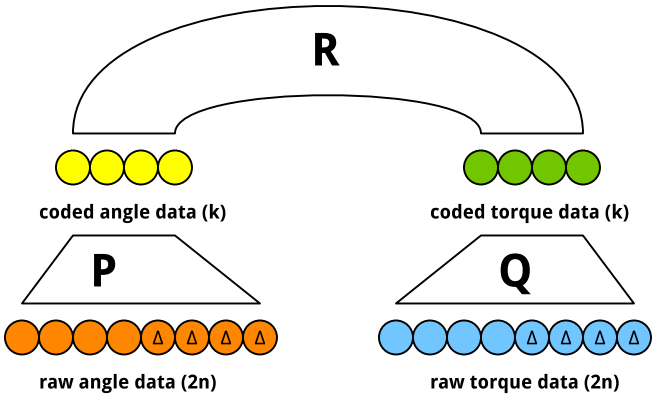

Figure 1: Information processing architecture for multimodal coding and regression. Information from a frame of one modality of pose data, such as angles (orange), is to be mapped onto information from another modality, such as torques (blue). This mapping is accomplished by coding a frame of angle data, augmented with its derivative Δ, using parameters $\mathbf{P}$; likewise, torques augmented with derivatives Δ are encoded using parameters $\mathbf{Q}$. Finally, a parametric regression $\mathbf{R}$ is computed between the codes (yellow and green).

and forces that would have been required for a simplified model of the human skeleton to effect the same movements. These computed angles and forces, while still a coarse proxy for some of the information that might be used by the central nervous system, constitute the data for this paper.

Theoretically, one could transform information from one modality into another by amassing a large quantity of corresponding data from these two modalities and computing regression coefficients directly. However, this is inefficient for at least two reasons. First, computing a regression between two datasets becomes increasingly problematic as the dimensionality of the data increases; this difficulty is compounded when there is noise in the data. Second, if the manifold hypothesis is accurate, then each modality of the raw data will have statistical redundancies that would need to be captured by the regression process. Rather than working in the space of raw measurements, then, we hypothesize that manipulating or combining movement information is more efficient in a space defined by codes that somehow represent the raw signals (Srivastava & Salakhutdinov, 2012). The question addressed by this paper is, which types of codes are most efficient for processing information about movement?

Pose coding and regression

We assume that we have a set of data that represents kinematic and dynamic views of human motion, modeled using an articulated body with n degrees of freedom, and measured over a consecutive sequence of m discrete time steps. Formally, we represent a sequence of raw joint angles as a matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$, where each column $\mathbf{b}^{(t)}$ represents a single frame of angle data. Similarly, we represent a sequence of raw joint torques as a matrix $\mathbf{U} \in \mathbb{R}^{n \times m}$ whose columns $\mathbf{u}^{(t)}$ each contain a frame of torque data. We define these matrices as complementary views of a single motion trajectory, so that for any frame t, the joint angles $\mathbf{b}^{(t)}$ correspond to the torques $\mathbf{u}^{(t)}$.

As mentioned above, movement is complex to model because it is high-dimensional (n is often large) and varies over time (m is often large). Rather than attempt to tackle both of these challenges at once, we simplify the modeling task here by considering the task of mapping between these two modalities for single poses (frames). Such a simplification makes the modeling task obviously difficult, since a single frame of kinematic pose data, for instance, does not indicate the direction in which the joint angles will be changing in subsequent frames. To address this issue, we make use of a common technique from speech recognition (Picone, 1993) and augment each of the raw data frames in our system with its first derivative. This provides information about the rates at which the angles and torques are changing, which could be useful when trying to compute torques on the basis of angles. The augmented data matrices $\mathbf{A}, \mathbf{V} \in \mathbb{R}^{2n \times m}$ are then defined as

$$\mathbf{A} = \begin{bmatrix} \mathbf{B} \\ \Delta\mathbf{B} \end{bmatrix} \quad \text{and} \quad \mathbf{V} = \begin{bmatrix} \mathbf{U} \\ \Delta\mathbf{U} \end{bmatrix},$$

where $\Delta\mathbf{X}^{(t)} = (\mathbf{x}^{(t+1)} - \mathbf{x}^{(t-1)})/2$ represents a secant approximation to the derivative at each frame.

Having created a set of kinematic and matched dynamic data describing sequences of human poses, we propose an information processing architecture for computing regressions from angles to torques. In this framework (see Figure 1), a single frame of n input angles, augmented with its derivative, is encoded first into k angle-code coefficients using a coding model characterized by parameters $\mathbf{P} \in \mathbb{R}^{k \times 2n}$. Then a regression model characterized by parameters $\mathbf{R} \in \mathbb{R}^{k \times k}$ transforms the k angle-code coefficients into a k torque-code coefficients. Finally, this torque encoding is converted back into a frame of n torques, augmented with its first derivative, by inverting the torque coding model characterized by parameters $\mathbf{Q} \in \mathbb{R}^{2n \times k}$. If the manifold hypothesis holds for human pose data, then code parameters $\mathbf{P}$ and $\mathbf{Q}$ can be learned independently, because these parameters will describe the structure of the manifold for each modality of pose data; codes for each manifold should then be useful for a wide variety of other information processing tasks (Vincent, Larochelle, Lajoie, Bengio, & Manzagol, 2010). Regression parameters $\mathbf{R}$ can then be learned using the encoded data from each modality.

Given our parametric framework for coding and regression, the coding approaches considered here all as-

sume a finite "codebook" $\mathbf{D} \in \mathbb{R}^{2n \times k}$ whose columns $\mathbf{d}_i$ each represent a "basis vector" that is used in some way to encode data. This paper does not focus specifically on learning the codebook, though it does mention a few approaches to codebook learning below.

Separating the model into coding and regression stages brings three advantages to the problem at hand. First, it allows us to manipulate the number of parameters in the model in a controlled way. The multistage model contains $k^2 + 4kn$ parameters, while direct regression requires $4n^2$ parameters. When $k < 2n$, the multistage model has fewer parameters than direct regression, but when k exceeds the dimensionality of the data, the multistage model has more parameters. Models with more parameters tend to be more accurate, but they might overfit the data and capture more noise than desired. Second, separate modules for coding in each modality, and regression between codes, allows for in-depth analysis of the performance of each module: codes for one modality that provide for low decoding error could also be ones that do not permit easy regression, for example. Finally, defining distinct coding modules permit an analysis of the degree to which coding, in isolation, provides an efficient representation of the data.

Coding algorithms

Mathematically, this paper treats coding as a general term for transforming a vector of raw data $\mathbf{x} \in \mathbb{R}^{2n}$ into another vector of coefficients $\mathbf{z} \in \mathbb{R}^k$, such that $\mathbf{z}$ contains sufficient information to recover $\mathbf{x}$ with some tolerated level of error. More formally, coding is often defined in terms of minimizing a cost function

$$\|g(\mathbf{D}, \mathbf{z}) - \mathbf{x}\|_2^2 + \lambda R(\mathbf{z})$$

where $g(\mathbf{D}, \mathbf{z})$ refers to a decoding operation that converts coefficients $\mathbf{z}$ into an estimate of the raw data $\hat{\mathbf{x}}$, and R is a regularizer that can be chosen to prevent overfitting, promote sparsity in the code, etc. For this paper, we evaluate several approaches to coding, each described briefly below.

PCA Principal component analysis (PCA) is widely used as a data preprocessing technique, but here we use PCA to refer to an encoding that simply computes the inner product of a data point $\mathbf{x}$ with each of the codebook vectors $\mathbf{d}_i$: that is, $\mathbf{z} = \mathbf{D}^T\mathbf{x}$.

The PCA codebook consists of the eigenvectors of the covariance matrix of the data, sorted in decreasing order of magnitude of the corresponding eigenvalue. By retaining only the first k eigenvectors in the codebook, PCA retains the maximally varying components of the data first, and progressively refines the error by retaining smaller components.

Used in this way, PCA implicitly models the underlying data as a multivariate normal distribution. Because the codebook for PCA is composed of eigenvectors (which are orthogonal to each other), there can be at most as many codebook vectors as dimensions in the input data.

K-means K-means (MacQueen et al., 1967) can be seen as an extremely sparse coding technique that represents a data point $\mathbf{x}$ using only the closest basis vector in the codebook: for this approach, $\mathbf{z} = [\xi_1 \dots \xi_k]^T$ such that

$$\xi_i = \begin{cases} 1 & \text{if } \|\mathbf{x} - \mathbf{d}_i\|_2 < \|\mathbf{x} - \mathbf{d}_j\|_2 \text{ for } j \neq i \\ 0 & \text{otherwise.} \end{cases}$$

The codebook corresponding to this coding approach is learned from the data by setting the $\mathbf{d}_i$ to random elements from the training data, and then iteratively adjusting the columns of $\mathbf{D}$ so that the sum of the Euclidean distances from each data point to its closest codebook vector is minimized.

Sparse coding Sparse coding (Tibshirani, 1996), also called lasso regression, computes the coefficients $\mathbf{z}$ for data point $\mathbf{x}$ by minimizing a least-squares cost function with a regularization penalty on the magnitude of the code coefficients:

$$\mathbf{z} = \arg\min_{\zeta} \|\mathbf{D}\zeta - \mathbf{x}\|_2^2 + \lambda \|\zeta\|_1 .$$

λ is a parameter that controls the tradeoff between accurate representation and sparsity; following results from the literature, we set $\lambda \propto 1/\sqrt{n}$.

Coates and Ng (2011) reported that sparse coding worked well across many types of codebooks for their tasks (image classification). For this coding algorithm, then, we tested several different codebooks: random, sampled, and learned. The random codebook consisted of IID vectors drawn from the standard normal distribution, normalized to unit length. The sampled codebook contained samples drawn uniformly from the training data, also normalized to unit length. The learned codebook used a fast, online algorithm developed by Mairal, Bach, Ponce, and Sapiro (2009) to tune the codebook to the data. Briefly, the general algorithm is a variation of coordinate descent, where sparse encoding computations are alternated with codebook updates. The codebook learning process attempts to minimize the lasso cost function above, both with respect to the codes $\mathbf{z}$ and also with respect to the codebook $\mathbf{D}$.

Regression

Once codes have been computed for the source and target datasets, the next task is to compute a regression matrix $\mathbf{R}$ that will convert coefficients from one modality into coefficients from another. We used ridge regression (Hoerl & Kennard, 1970) to compute the best parameters for inferring coefficients across coded modalities. We can express the regression task between codes $\mathbf{z}_\alpha$ and $\mathbf{z}_\beta$

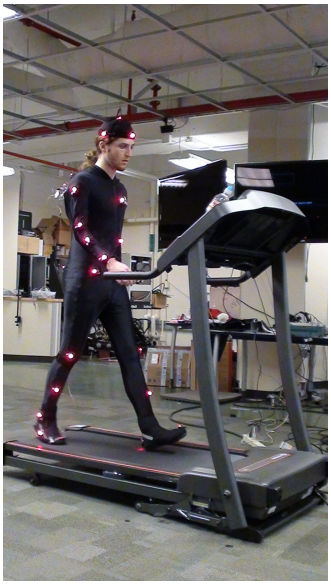

Figure 2: The motion capture environment consists of a full-body motion capture suit (black, with red LEDs), and a treadmill centered in the motion capture space.

as optimizing the cost function

$$\|\mathbf{R}\mathbf{z}_\alpha - \mathbf{z}_\beta\|_2^2 + \lambda\|\mathbf{R}\|_F^2$$

where λ captures the degree to which the modeler is willing to tolerate large values in $\mathbf{R}$ when explaining the observed data. Essentially, ridge regression is the same as linear regression, but it adds a penalty on large values of the coefficients that are used to describe the data. In our experiments, the value of λ was set empirically by cross-validation on the training set.

Experiments

To measure and collect human movement data, we used a 16–camera Phasespace[1] motion capture system in conjunction with a standard treadmill (Figure 2). 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 the results reported here, we recorded the positions of $L = 48$ markers from one subject as he walked at speeds ranging from 0.22 to 2.68 m/s. The recording lasted twenty minutes. The Phasespace system produces frames of motion capture data at a rate of 120Hz, so this recording resulted in more than 120,000 frames of raw motion-capture data. These frames were processed using the articulated forward model proposed by Cooper and Ballard (2012), resulting in three sequences of measurements for the observed motion: the sequence of interpolated marker positions $\mathbf{X} = \left[\mathbf{x}^{(1)} \dots \mathbf{x}^{(N)}\right]$ representing the positions of the segments of the articulated model over time; the sequence of observed angles $\mathbf{A} = \left[\mathbf{a}^{(1)} \dots \mathbf{a}^{(N)}\right]$ for each of the 54 degrees of freedom in the model; and the corresponding torques $\mathbf{V} = \left[\mathbf{v}^{(1)} \dots \mathbf{v}^{(N)}\right]$ that were necessary to cause those angles to move through the observed dynamic trajectory of the model.

[1] `phasespace.com/impulse_motion_capture.html`

Preprocessing

For this paper, we were concerned with mapping angles to torques, so we discarded the marker data $\mathbf{X}$. To obtain datasets for training and testing the coding and regression models, we needed to perform some preprocessing to obtain matched sets of frames that would permit a fair comparison.

First, the sequences obtained from the model were smoothed by convolving each channel in each modality with a 5-sample (42 millisecond) rectangular window over time. After smoothing, each channel of the data was normalized by subtracting out the mean value and dividing by the standard deviation. These steps ensured that the data did not contain residual noise due to marker dropouts, and also that the data values were all approximately the same scale.

Each frame of data was then augmented with an approximation of its first derivative by calculating the secant approximation of these quantities using the neighboring two frames.[2]

Next, the smoothed, normalized, derivative-augmented frames were segmented into three distinct regions, each containing 24000 frames (200 seconds) of data: the first (segment A) consisted of slow walks, the second (segment B) consisted of fast walks, and the third (segment C) consisted of running movements. To evaluate the coding and regression models, each segment was further partitioned into disjoint training, validation, and test sets such that 10% of frames from each segment were used for validation, 10% were used only for testing, and the remainder were available for training.

Coding efficiency

We first analyzed the performance of the different coding techniques discussed above when reconstructing the raw torque data using the torque codes. Formally, after training the dictionaries as needed, we computed $\mathbf{z}$ for each frame of augmented torque data $\mathbf{v}$ in the test set, and then computed the decoding operation to obtain an estimate $\hat{\mathbf{v}}$. The decoding error $e_{\mathbf{v}}$ was then defined as the RMS value of the residual:

$$e_{\mathbf{v}} = \sqrt{\frac{1}{n}\|\hat{\mathbf{v}} - \mathbf{v}\|_2^2}.$$

[2] The first frame was dropped from each dataset to match the number of frames of data with the number of frames of derivative.

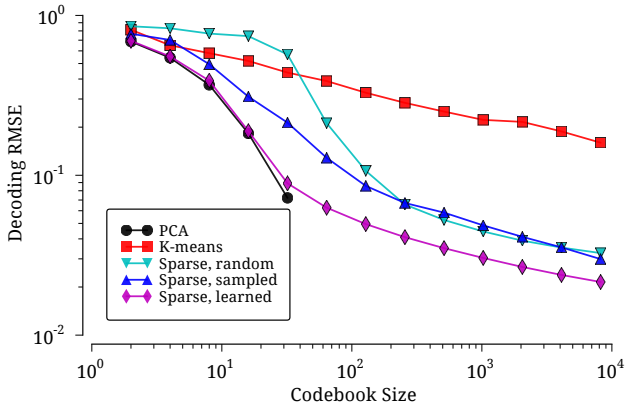

Figure 3: Mean RMS decoding error for joint torques, measured with respect to the size of the codebook. Larger codebooks result in codes that capture more of the variance in the data, even when the codebook is created using IID standard normal samples. A log scale has been used on both axes to reveal trends more clearly.

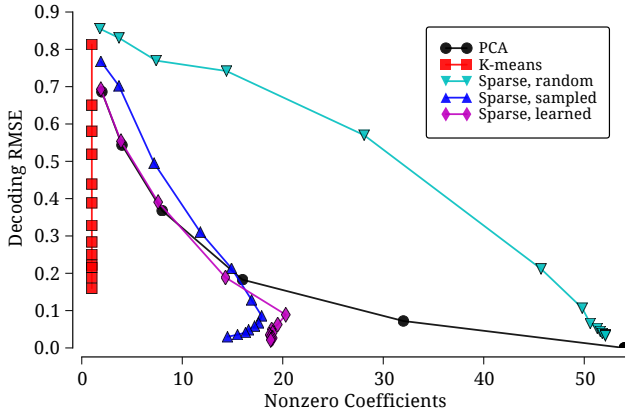

Figure 4: Mean RMS decoding error for joint torques, measured per nonzero coefficient in the encoding. Sparse codes like lasso regression were more effective, per coefficient, than dense codes like PCA, but only when the codebook was tuned to the dataset.

Figure 3 shows the mean RMSE for each coding approach, measured with various sizes of codebooks, and applied solely to the torque data. (Results for the angle data were similar.) Unsurprisingly, larger codebooks were able to capture more of the variance in the data than smaller codebooks, regardless of the coding method. Perhaps more interesting, however, was the finding shown in Figure 4: when measured by the number of nonzero coefficients used in the code, sparse codes produced more accurate reconstructions than dense codes. This was somewhat vacuously true of K-means, since it only uses 1 coefficient for each $\mathbf{z}$; in comparison, however, this was not true for sparse coding combined with the random codebook.

Predicting torques from angles

In addition to comparing the effectiveness of different coding schemes for torque data, we also used our framework to compare the encoding methods in a larger context, namely predicting torque values on the basis of angle values. This task could be seen as a coarse approximation for a control task: given a target kinematic pose, what might be the torques that would be associated with that pose?

Because the analysis framework proposed in this paper breaks down this task into three separate stages—encoding, regression, and decoding—we can analyze the regression component of the task separately from the other components. In general, RMS error for the regression task alone (Figure 5) followed the same pattern as errors for the encoding and decoding components: larger codebooks tended to yield lower errors. However, sparsity played a critical role in this task, since K-means yielded the lowest regression errors, while PCA yielded relatively large errors.

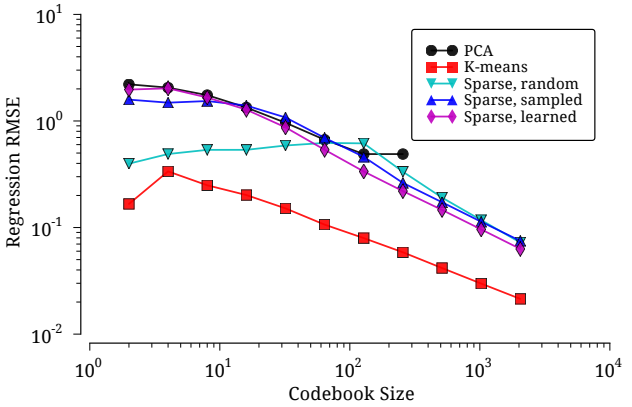

Figure 5: RMS regression error, measured with respect to encoded torque values.

Finally, we compared the overall torque regression error for all components of the model together, as measured by comparing the outputs from our processing model with the true torques measured during the experiment (Figure 6). As a baseline, we computed a direct regression from angle to torque data: this resulted in an RMS error of 0.65 on the test set. PCA performed at baseline for complete codebooks, which is unsurprising since PCA simply rescales the data. However, some of the sparse coding approaches did outperform PCA by a large margin (up to 30% reduction in error). In particular, using lasso coding combined with a large, learned dictionary produced lower RMS errors than any of the other approaches examined here.

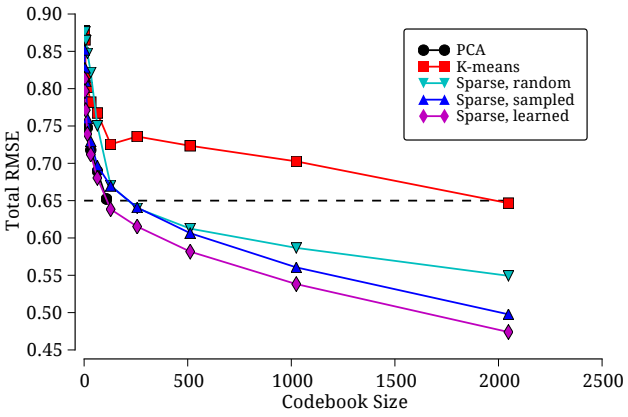

Figure 6: Mean RMS reconstruction error, measured with respect to true torque values for each frame. Note that the RMS reconstruction error for direct regression between true angles and true torques is 0.652, which is approximately at the asymptote shown for PCA.

Discussion

This paper presented an efficient coding and regression model for human pose information, and used this model to examine the performance of several coding algorithms on human pose information. The model allowed us to examine separately the errors in coding information about poses and in regressing from one modality to another. We learned that even though some approaches produce extremely low coding and decoding errors, and other approaches were conducive to learning regressions between codes, in order to perform well on the task of predicting information across information modalities, a coding approach must have extremely low error on both tasks.

In several ways this paper is just a first look at this sort of modeling on human pose information. In particular, we limited our examination of human pose information to snapshots of single moments in time. Movement, however, is fundamentally dynamic, so we plan to expand the techniques presented here to temporal sequences of poses, by learning codes for entire movements.

References

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

Coates, A., & Ng, A. (2011). The importance of encoding versus training with sparse coding and vector quantization. In *Proc. 28th Intl. Conf. on Machine Learning* (Vol. 8, p. 10).

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

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

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

Lee, H., Battle, A., Raina, R., & Ng, A. (2007). Efficient sparse coding algorithms. *Advances in neural information processing systems*, *19*, 801.

MacQueen, J., et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 1, p. 14).

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

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

Picone, J. (1993). Signal modeling techniques in speech recognition. *Proc. IEEE*, *81*(9), 1215–1247.

Ranzato, M., Boureau, Y., & LeCun, Y. (2007). Sparse feature learning for deep belief networks. *Advances in neural information processing systems*, *20*, 1185–1192.

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

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

Srivastava, N., & Salakhutdinov, R. (2012). Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems 25* (pp. 2231–2239).

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

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, *11*, 3371–3408.

Winter, D. (2009). *Biomechanics and Motor Control of Human Movement*. Wiley.

Zatsiorsky, V., & Prilutsky, B. (2012). *Biomechanics of Skeletal Muscles*. Human Kinetics.