The Dissertation Committee for Leif Morgan Johnson
certifies that this is the approved version of the following dissertation:

Redundancy Reduction in Motor Control

Committee:

Dana Ballard, Supervisor

Risto Miikkulainen

Richard Neptune

Jan Peters

Peter Stone

Redundancy Reduction in Motor Control

by

Leif Morgan Johnson, BA Multidiscipl Stds; BS App Math; BS Comp Sci

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2015

To Lily and Rose.

Acknowledgments

Progressing through graduate school turns out to be surprisingly difficult, even taking into account the fact that progressing graduate school is surprisingly difficult. For me, getting a doctoral degree would quite literally not have been possible without a lot of help, from many people, in many forms.

The colleagues in my research group, variously called the Embodied Cognition or Virtual Reality Laboratories, have provided fascinating discussions, an inspiring breadth of expertise, and helpful feedback on posters, papers, and presentations. Their support, interest, and ideas have helped steer me toward productive areas of science—even if I didn't always follow their directions.

My supervisor, Dana Ballard, has done his best to help me focus my energies on a reasonably small swath of topics. It is a Sisyphean task, and he has succeeded more often than most. Throughout he has also given me freedom to explore and learn.

I am also grateful for financial support from the Department of Computer Science and the Center for Perceptual Systems.

Finally, I would most like to acknowledge the inexpressibly valuable support of my wife, Lily, and our daughter, Rose. They make my day.

Redundancy Reduction in Motor Control

Publication No. _____

Leif Morgan Johnson, Ph.D.
The University of Texas at Austin, 2015

Supervisor: Dana Ballard

Research in machine learning and neuroscience has made remarkable progress by investigating statistical redundancy in representations of natural environments, but to date much of this work has focused on sensory information like images and sounds. This dissertation explores the notions of redundancy and efficiency in the motor domain, where several different forms of independence exist. The dissertation begins by discussing redundancy at a conceptual level and presents relevant background material. Next, three main branches of original research are described. The first branch consists of a novel control framework for integrating low-bandwidth sensory updates with model uncertainty and action selection for navigating complex, multi-task environments. The second branch of research applies existing machine learning techniques to movement information and explores the mismatch between these methods for extracting independent components and the forms of redundancy that exist in the motor domain. The third branch of work analyzes full-body, goal-directed reaching movements gathered in a novel laboratory experiment,

using explicitly measured information about the goal of each movement to uncover patterns in the movement dynamics. Each branch of research explores redundancy reduction in movement from a different perspective, building up a sort of catalog of the types of information present in movements. Redundancy is discussed throughout as an an important aspect of movement in the natural world. The dissertation concludes by summarizing the contributions of these three branches of work, and discussing promising areas for future work spurred by these investigations. More detailed models of voluntary movements hold promise not only for better treatments, improved prosthetics, smoother animations, and more fluid robots, but also as an avenue for scientific insight into the very foundations of cognition.

Table of Contents

List of Figures

Part I

Foundations

Chapter 1

Introduction

Voluntary movement is one of the defining characteristics of nearly all animals, and yet a complete scientific understanding of voluntary movement remains elusive. At the same time, in sensory neuroscience, new answers have been proposed for an old but important question, which can be framed today as: *What are the independent components of natural scenes?* This question focuses on the interplay between statistical redundancies in the natural world and the representations that sensory neurons in the central nervous system appear to develop in response to those redundancies. Asking, refining, and answering questions related to "efficiency" and "redundancy reduction" has led not only to conceptual advances like models of the independent components of images (Bell and Sejnowski, 1997), but also to beautiful empirical results using images (Olshausen and Field, 1996; Lee et al., 2007) and sounds (Smith and Lewicki, 2006; Lee et al., 2010). These results have provided deep insights into the function and organization of sensory systems in both natural and artificial agents.

This dissertation is motivated by asking a similar question: *What are the independent components of natural movements?* The question appears

familiar on the surface, but movement is qualitatively different from sensory information. At a minimum, movements unfold in time in addition to space, but voluntary movements also tend to incorporate ongoing sensory feedback, as well as a mixture of dynamics. Thus, movement begs a different treatment than existing "passive" redundancy analyses for sensory information. The hypothesis pursued in this dissertation is that computational methods from machine learning and sensory neuroscience can be applied to movements to gain new understandings of the dynamics and organization of voluntary movements.

This document is organized into four parts: Foundations, Computational Work, Empirical Work, and Conclusion. Foundations establishes the theoretical basis for the work to be described. The first chapter traces the development of redundancy reduction as a research theme in sensory neuroscience and concludes with a sort of manifesto for exploring redundancy and sparsity in movement. Two subsequent chapters present relevant background information: Chapter 2 presents mathematics and notation that will be used in the rest of the document, and Chapter 3 presents a very brief summary of related research work in movement modeling.

Computational Work describes computational models that were created while developing the thesis. Chapter 4 describes a computational model of control that explicitly incorporates uncertainty into the controller, and Chapter 5 describes several small studies aimed at applying existing computational approaches to movement information.

Empirical Work describes a novel experiment conducted in the labo-

ratory to collect and analyze full-body, goal-directed movements. Chapter 6 describes a novel model of movement generation that is used to think about how and why movements are chosen. Chapter 7 describes an experiment conducted in our motion-capture laboratory that aimed to collect full-body, goal-directed movements from healthy adults. In this experiment, movement data were collected along with quantitative annotations of the moment-by-moment goals of the movements. Chapter 8 describes a few computational results derived from the resulting dataset, including analysis of the full-body Jacobian that forms the best linear control approximation to the observed movement. Analyzing the patterns and redundancy present in the Jacobian reveals several interesting aspects of movement that might be useful for spurring future development.

Finally, the dissertation concludes by discussing the contributions of the thesis and describing several challenges aimed at helping to steer future work. Throughout the document, the focus remains on movement and its decomposition into components using various views of independence. Before discussing the research work, then, it will be useful to first understand how redundancy reduction and related research concepts have played out in sensory domains, and how they are thought to play a role for many types of natural data, including movements.

1.1 Redundancy

Our world features repetition and overlap at nearly every scale, across space and time. But what exactly is redundancy? For the time being, we can think of redundancy simply as "repetition"—but a few examples will help to illustrate this phenomenon in more detail, and also to suggest ways in which redundancy can be measured and applied to computational modeling tasks.

- The state of the world, in general, tends to be redundant in time. In other words, as a general rule, things in the world tend to change smoothly, so that knowing the state of the world at one time can normally help predict the state of the world a short time in the future. Conversely, when things in the world change rapidly, there is often something interesting or unusual happening.

- The sensory phenomena we experience in the world are redundant. For example, measuring the brightness of a pixel in a photograph helps predict the values of the neighboring pixels (Kersten, 1987). Likewise, the harmonics of most naturally occurring sounds provide overlapping information about the source of the sound.

- Human language is redundant—almost surprisingly so. Words that are semantically related commonly appear together in documents, so, for example, seeing the word "senator" in a document makes it less surprising to subsequently see "vote" than it would be to see "invent." Words with related meanings tend to be used together (Harris, 1954).

- Repetition exists at several levels of information flow in human culture. Consider, for example, the dissemination of news. Different news outlets often distribute variations of the same stories whenever an event happens, so two stories about an event will tend to provide overlapping information. Even a single document will tend to be redundant within itself: reading a synopsis of a document, while not exactly the same as reading the entire thing, provides a large portion of the information that the full document would provide (Shannon, 1948).

- Finally, movements in animals are redundant in several different ways. An animal's body posture at one time is strongly correlated with its posture at a nearby time. In addition, the angles of the joints in an animal's body—for example, the legs of a cheetah wile running—tend to be strongly correlated (or anticorrelated) with each other. Computationally, such correlations have the result that the postures of an animal's body occupy a significantly smaller effective subspace than the "raw" degrees of freedom permitted by the body (e.g., d'Avella and Bizzi, 1998; Sanger, 2000; D'Souza et al., 2001; Perrier et al., 2007). Furthermore, the musculoskeletal structure of an animal tends to limit movements to small regions of joint-angle space, because tendons and ligaments often prevent full rotation around a given axis. These limits encourage normal postures to occupy a smaller set of states than might be possible if, say, a skeleton with the same number of degrees of freedom were constructed out of artificial parts.

Reflecting on these examples, redundancy seems to imply that knowing one thing about a phenomenon helps to predict other things about that phenomenon. A bit more precisely, if something is redundant and you make a measurement of it, then that observation will change your expectations of what you might see if you make a subsequent measurement. In this way, redundancy can be expressed more precisely as a manifestation of *mutual information*. Formally, for two random variables X and Y, the mutual information of these variables is given by

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{1.1}$$

where

$$H(X,Y) = -\sum_{x \in X} \sum_{y \in Y} p(x,y) \log p(x,y) \tag{1.2}$$

is the *joint entropy* of random variables X and Y and

$$H(X) = -\sum_{x \in X} p(x) \log p(x) \tag{1.3}$$

is the *marginal entropy* (or just *entropy*) of random variable X.

The joint entropy measures the amount of information needed to describe events x and y happening together, while the marginal entropy is the amount of information needed to describe an event x alone. Mutual information, then, is a measure of the amount of information that X and Y share—the amount of information overlap between X and Y. Note that this notion of mutual information could equally well apply to two random variables X_t and X_{t+1} that represent a single process sampled at different moments in time.

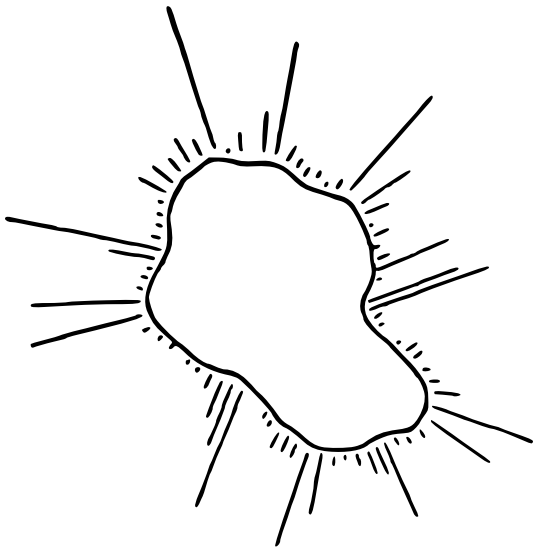


Figure 1.1: Subjects were asked to place 10 dots along the curve to capture its structure. The bars around the curve represent a histogram of dot locations chosen by the subjects. Image traced from Attneave (1954).

So redundant variables—neighboring pixels of an image, words in an article, and natural movements—all seem to offer large amounts of mutual information. For an animal living in this world, it would be nearly impossible—not to mention exhausting—to try to capture all of this redundant information. Somehow animals must take advantage of this redundancy, both to avoid storing or even processing unneeded information, but also to help make sense of the world at all.

Attneave (1954) was one of the first to propose that techniques from information theory (Shannon, 1948) might be useful to quantify our intuitions about the redundancy of natural scenes. Attneave proposed an application of Shannon's counting game, in which the pixels of a natural image (a photograph

or drawing of a scene) would be transmitted serially to a human observer. The observer would have to guess the color of the next pixel, given the pixels already seen. The number of incorrect guesses in this game would then serve as an estimate of the amount of information needed to reconstruct the scene. For an extremely basic scene containing a table and an ink pot, Attneave estimated the redundancy of the scene at greater than 99%. In a related experiment, Attneave asked subjects to place 10 dots along a continuous, closed curve such that the dots "represented" the curve as closely as possible. Subjects tended to place dots on areas of the figure with the largest curvature (see Figure 1.1).

These experiments demonstrated that, at the perceptual and behavioral levels, people are exquisitely sensitive to the information structure of their sensory environments. Attneave used these results to argue that perception appears to be strongly related to redundancy reduction, and proposed several cognitive rules to help understand perceptual processing.

1.2 Efficiency

Perceptual rules based on redundancy reduction made some headway toward Attneave's "perception as economical description," but still missing was some sort of cause for believing that perceptual rules had a basis in physiology. Barlow (1961) proposed a particularly elegant explanation of the role that sensory neurons might play in these higher-level perceptual processes, by arguing that sensory neurons attempted to recode the incoming information in an economical way.

Neurons are metabolically expensive to maintain and to use. When a neuron spikes, the resting potential across the cell membrane is lost, and an active process must work to reconstruct the ion imbalance across the membrane. This process—the "sodium-potassium pump" (Skou, 1957)—requires a relatively large amount of energy, making the nervous system one of the most metabolically expensive groups of organs in many animals (Kety, 1957; Laughlin et al., 1998).

By these lights, an animal's nervous system would use energy most efficiently if its neurons ceased to fire. However, the animal still needs to move and act to survive, so some neurons must fire sometimes. The question then becomes, how might neurons maximize the utility of their spikes? Barlow brought the vocabulary of information theory to bear on a simplified model of sensory neurons and proposed a model for efficiency in sensory neurons based on recoding: "The hypothesis is that sensory relays recode sensory messages so that their redundancy is reduced by comparatively little information is lost." Barlow argued that spikes in such a model ought to be reserved for unusual events, since so much of a sensory neuron's input would be predictable.

By tuning themselves to represent unusual events, sensory neurons would thus maximize the information content of each spike. Such a model works only because the redundancy of the natural world ensures that "unusual events" are relatively rare. As Attneave and others had discussed, these unusual events bear most of the information content in the incoming sensory data stream. Recoding incoming information to represent unusual events al-

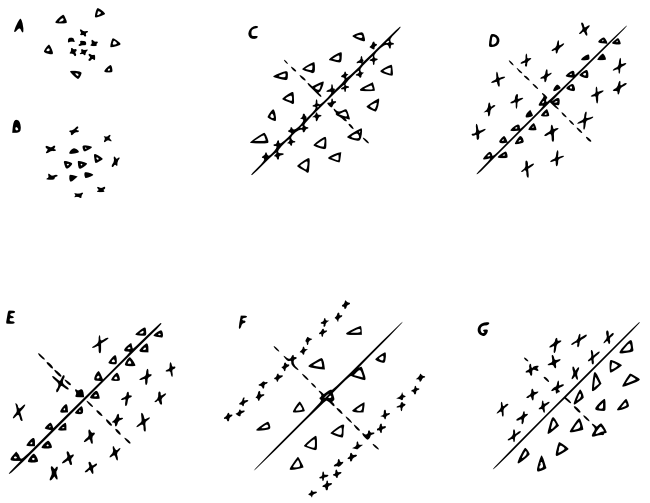


Figure 1.2: Receptive fields from the visual cortex of the cat. Fields (c) through (g) are from cells in primary visual cortex. Points marked with an $\times$ indicate excitatory responses; points marked with Δ indicate inhibitory responses. Traced from Hubel and Wiesel (1962).

lows sensory neurons to operate with maximal efficiency for the animal.

The assumption that sensory neurons attempt to maximize the information content of their spikes was, in some ways, a missing bridge between neurons and behavior. Using tools from information theory, sensory neurons can be regarded as being metabolically efficient by firing as infrequently as possible—assuming that "unusual" events are relatively rare—while still preserving the information content of their representations.

1.3 Sparsity

In animals, sensory neurons fire in response to stimuli from the external world. In the primate visual system, for example, a neuron in primary visual

area V1 fires when a specific pattern of light—the *preferred stimulus*—hits a specific area of the retina—the location of the cell's *receptive field* (Hubel and Wiesel, 1962). Rather than firing when any old pattern of light hits any old area of the retina, V1 neurons in mammals are somehow specialized to respond to specific features in the visual world; see Figure 1.2 for an example. The same phenomenon appears to hold in each of the primary sensory areas; for example, sensory neurons in primary auditory cortex area A1 have a *best frequency* and fire preferentially in response to this frequency (Reale and Imig, 1980).

For any given stimulus, only a few of the sensory neurons will fire; this phenomenon agrees in principle with proposals from Barlow and others regarding efficiency in sensory neurons. It took several decades, however, before Olshausen and Field (1996) built a new, important bridge from neuroscience to computation by demonstrating that a formal mathematical model of sensory images demonstrated produced "receptive fields" to those found in living animals. Later, a similar phenomenon was shown for auditory data and acoustic receptive fields (Smith and Lewicki, 2006).

These modeling results were significant because they showed that a mathematical formalism could be used to model the organization of sensory neurons in animals. Like Barlow's proposal, the model developed by Olshausen and Field relied on optimizing two objectives simultaneously: (a) the ability to represent input data, while simultaneously (b) using as few "spikes" as possible. This model, now widely known as the "sparse coding" model, is

described formally in Section 2.1. Its defining characteristic is *sparsity*, the use of as little information as possible to represent a given stimulus.

As Barlow also noted, a similar model to the recoding hypothesis was also proposed around the same time by MacKay (1956). In the model from MacKay, neurons attempt to match the statistics of the input as closely as possible, and any residual errors can be recoded, recursively. This model is now known as the *matching pursuit* (Mallat and Zhang, 1993) sparse coding algorithm; it was proposed around the same time as the model of Olshausen and Field and is widely used in practice for analyzing efficient codes.

1.3.1 Departure from Biology

Without much fanfare, the discussion has taken a subtle but important turn. Until now, the focus has been on biological systems: first the reality of redundancy in the world, then Attneave's rules for perception, and finally Barlow's (and MacKay's) proposals to model interesting signals without using too many spikes.

Sparsity, on the other hand, is a mathematical concept. Although it has many applications, including metabolic efficiency in neural systems, sparsity itself is a modeling tool that has its own characteristics, and modelers choose to use sparsity for reasons other than metabolic efficiency. For instance, Tibshirani (1996) notes that sparse models tend to yield features that are more easily interpreted by the humans who design and use computational models. In this and many other ways, sparsity goes beyond a simple metabolic

explanation, leading to a new world of representations and manifolds.

To help make the bridge from neural systems to mathematical sparsity, we will start talking about representing signals using a "basis" rather than a population of neurons. There are important differences between a mathematical basis and a neural population, but for the moment the two concepts are useful for building a bridge.

1.4 Overcompleteness

Sparsity alone can be difficult to reconcile with Barlow's proposed second goal, namely retaining information about stimuli. If a computational model is to represent some d–dimensional information using a basis with n elements, then forcing the model to use as few basis elements as possible is, at its most extreme, akin to requiring the model to memorize the most salient information in the environment, at the expense of other information that might be less prevalent. In the extreme case where a sparsity requirement is so strong that only one basis element can be used to represent each stimulus, then the best basis will contain a distribution of elements that match the distribution of data in the world as closely as possible (often in a squared-error sense)—such a constraint results in a Gaussian mixture model with hard cluster assignments. But if n is small compared with the variability of information that needs to be represented, such a basis will incur notable losses when trying to represent information that is far from any of the basis elements.

Sparsity alone, then, can be a handicap when modeling a complex set

of information from the environment. This is particularly so when the distribution of relevant information has complex local structure, or also when $n \ll d$. One way of overcoming this issue is to relax the required level of sparsity in the model. This can be problematic for a living organism due to the concomitant increase in metabolic cost, and it can be problematic for a mathematical model due to the resulting lack of interpretability or utility of the model. Another possibility is to add more basis elements to the model, so that representations can still be sparse, but more prototypes can be allocated to fit the data distribution more closely. When the size of the basis is increased so that $n > d$, there are more basis elements than dimensions in the stimulus. Such a basis is described as *overcomplete*.

Overcompleteness seems to be one of the solutions favored by some brain areas, including the primate primary visual cortex. This brain area contains far more cells (approximately 10^8 simple cells in the human area V1) than would be necessary to represent the information from each cell in the retina (approximately 10^6 ganglion cells in a human retina) (Wandell, 1995). Furthermore, many of the oriented Gabor filters that serve as preferred stimuli in V1 simple cells are replicated in different areas of the visual field. This repetition at first seems unnecessary: why duplicate the same preferred stimulus several different times in V1, particularly if it is so important to conserve energy? But from a mathematical perspective, the overcompleteness strategy is appealing, because the representation can remain sparse, while also retaining an enormous amount of information about the stimulus—in the case

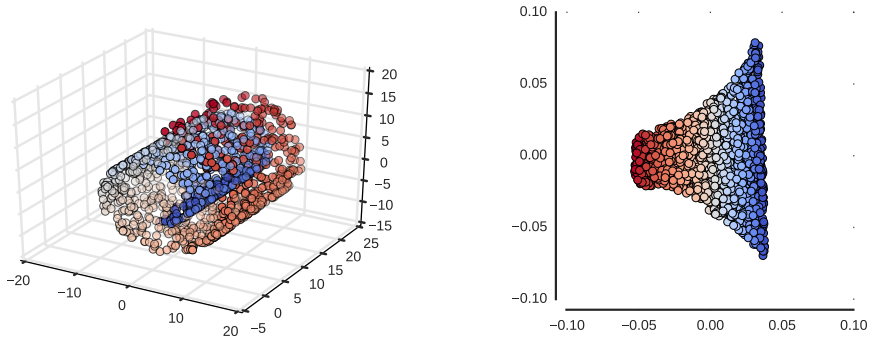


Figure 1.3: The "Swiss roll" is a widely used toy manifold. On the left is the manifold shown embedded in 3–dimensional space, and on the right is the manifold "unrolled" using Locally Linear Embedding (LLE; Roweis and Saul, 2000).

of V1 simple cells, a representation can capture both the type of edge that is present in a scene, as well as its location in the visual field.

1.5 Manifolds

Overcompleteness serves a sparse representation particularly effectively from the perspective of *manifolds*. A manifold is a set of data embedded in a high-dimensional vector space such that points on the manifold exhibit local structure with much lower dimensionality. Consider, for example, a sheet of paper (essentially a 2–dimensional surface) curled up in a spiral in 3–dimensional space. It is possible to represent any point in this manifold using three "world coordinates"—for example, the standard world Cartesian coordinate system—or, alternatively, using two coordinates that describe the displacement along the surface of the manifold from some origin.

Figure 1.3 shows a well-known "toy" manifold that exhibits this structure. The points that make up the Swiss roll are 3–dimensional, but the manifold can be "flattened" onto two dimensions to reveal its local structure. Importantly, simply projecting the roll onto two dimensions using a linear map cannot capture the structure of the manifold, because points that are topologically distant on the manifold might be near each other under a linear projection; nonlinear techniques must typically be used to model such a surface accurately.

Several nonparametric methods have been developed for capturing local, low-dimensional structure in high-dimensional spaces. Locally linear embedding (LLE; Roweis and Saul, 2000), ISOMAP (Tenenbaum et al., 2000), and Probabilistic PCA (Tipping and Bishop, 1999b,a) are some of the most popular. Such methods typically build many local, linear models of the manifold and use these local linearities to project points on the manifold to "flattened" spaces with lower dimensionality. Interestingly, the goal of many of these techniques is dimensionality *reduction*; that is, techniques like LLE and ISOMAP seek to provide a description of a dataset in a small-dimensional space. In the Swiss roll example, for instance, manifold learning techniques are applied to compute for each 3–dimensional data point the 2–dimensional coordinates of the point along the surface of the manifold. (Often, these low-dimensional representations are used to generate visualizations of datasets that can be printed on 2–dimensional sheets of paper.) But nonparametric decompositions are, in some sense, highly overcomplete: the number of repre-

sentations of the manifold grows with the size of the dataset, as opposed to the size of the dimensionality of the dataset.

In more recent years, sparse codes have been shown to be effective in learning the structure of manifolds from natural datasets. Sparse models are thought to be effective for representing sensory data that lie along low-dimensional manifolds because the basis vectors in the code can be applied efficiently (i.e., using just a few nonzero coefficients for a particular data point) to indicate, for a particular data point, its location along the manifold rather than its coordinates in the higher-dimensional space. Depending on the structure of the manifold, sparse representations work particularly well for this task when the basis is overcomplete. Overcomplete representations basically permit some elements of the basis to apply to one world-space region of the manifold, while other elements of the basis can apply to a different region. For a data point in one region, only the basis elements that "care about" that region will be used to represent the data point; the others will remain silent.

In machine learning, much of the recent work on these codes has focused on sensory data like images or sound, using classification tasks for evaluation (e.g., Ranzato et al., 2007; Dahl et al., 2010). Concurrent research in motor control, however, has suggested that human movements might also lie along a relatively low-dimensional manifold embedded in the space of all possible movements (Scholz and Schöner, 1999; Latash et al., 2002). Sparse codes might be useful, then, for representing information about movement and pose in humans, and such codes might be useful for regression tasks as well. One of

the many open questions in movement research is whether sparsity is indeed useful for this sort of modeling task, and, if so, what the relationship is between sparsity (and overcompleteness) and the redundancy of movement information.

1.6 Redundancy in Movement

The material in this chapter started by discussing redundancy as a historical aspect of research in sensory neuroscience, and from there spiraled outward to reach an almost purely mathematical perspective, one that might apply to many different forms of data in the natural world. But one might wonder at this point how redundancy applies specifically to movement, and how those relationships might help guide research in this area of science.

Bernstein (1967) is widely credited with formulating a key question that spans most aspects of research in movement, known as the "degrees of freedom problem." Broadly stated, an animal tends to have many more degrees of freedom (DoF) than is required to accomplish a specific task. This is a problem for those who seek to understand movement because it remains unclear how an animal chooses to limit its intrinsic DoFs, and whether there are any principles for doing so that might be employed across all animals or all natural movements.

Seen from the perspective of the discussion above, movements in animals are clearly redundant at multiple levels. Because of the low number of kinematic DoFs required to accomplish many tasks, animals can achieve a desired goal state in many ways—for example, a human can grasp an object

from a table using any of a large range of elbow and shoulder angles. However, redundancy in movement is not limited to the kinematic level, either. Animals making movements are also neurally redundant, since multiple motor neurons synapse onto a given muscle fiber, and yet the nervous system somehow decides how to signal the muscle fibers to accomplish a given task.

Indeed, in almost all aspects of behavior one sees redundancy reduction built-in to the neuro-bio-mechanical system. Modeling this process of redundancy reduction is one goal of this dissertation; for more detail in applying redundancy-reduction techniques to walking movements, see Chapter 5, and for analysis of whole-body movements during a reaching task see Chapters 7 and 8. Before discussing these experimental results, however, a brief discussion of the available computational models will help set the stage.

Chapter 2

Computation

Consider a vector $\mathbf{x} \in \mathbb{R}^d$. This vector might represent a measurement from some physical system, such as the light intensities on each pixel of a CCD camera sensor, the sound pressure levels in a microphone over a period of time, or the angles of the joints in a person's skeleton at a particular moment. Regardless of its interpretation in a particular context, the vector $\mathbf{x}$ *represents* something. However, because the world is redundant and noisy, the values in $\mathbf{x}$ are probably also redundant and noisy, and so the interpretation of what $\mathbf{x}$ represents can be difficult and prone to error. This chapter describes several common computational techniques for managing data vectors like $\mathbf{x}$ and interpreting their representations in different spaces.

Often we have access to several measured vectors at one time. In these cases, a data matrix $\mathbf{X} \in \mathbb{R}^{d \times m}$ represents a collection of m independent measurements. Each of the m columns $\mathbf{X}_j (1 < j \leq m)$ of $\mathbf{X}$ denotes a single d–dimensional sample of data. Samples are usually assumed to be drawn IID from some underlying distribution—often a manifold embedded in d–dimensional space.

2.1 Coding

Given the prevalence of redundancy in the natural world, it is quite common to seek a *representation* of $\mathbf{x}$ by projecting it into another space. Formally, the goal is to represent each of the m \$d\$–dimensional samples in another vector space of dimensionality k. Thus the corresponding encoding for $\mathbf{X}$ is given by $\mathbf{Z} \in \mathbb{R}^{k \times m}$, where each of the m columns $\mathbf{Z}_j$ in $\mathbf{Z}$ represents the coded form of the corresponding column in $\mathbf{X}$.

The simplest form of encoding process is linear. With a linear representation, a matrix of parameters $\mathbf{W} \in \mathbb{R}^{d \times k}$ represents a set of k *basis vectors*, one \$d\$–dimensional basis vector per column. This parameter matrix, variously called a *basis*, a *dictionary*, or a *codebook*, decodes $\mathbf{Z}$ to $\mathbf{X}$ by expressing the observed data $\mathbf{X}$ as a weighted sum of basis vectors. The encoding $\mathbf{Z}$ typically provides the weights for the projection so that $\mathbf{X}$ is "explained" as a weighted sum of basis vectors

$$\mathbf{X} = \mathbf{W}\mathbf{Z}. \tag{2.1}$$

Some representations are to be preferred to others. For example, we could use a degenerate coding process that represents any $\mathbf{x}$ as a vector of length k containing only ones. This representation is poor because it prevents us from differentiating $\mathbf{x}_1$ from $\mathbf{x}_2$, or from recovering $\mathbf{x}$ accurately given $\mathbf{z}$. One commonly used criterion for a good representation, then, is to require that the representation $\mathbf{z}$ allows us to recover the original data point $\mathbf{x}$, much

like Barlow's requirement that sensory relays preserve information about the stimulus.

This "fidelity requirement" is commonly accomplished formally by constructing a squared-error loss

$$\mathcal{L}(\mathbf{z}; \mathbf{x}, \mathbf{W}) = \frac{1}{2}\|\mathbf{W}\mathbf{z} - \mathbf{x}\|_2^2 \tag{2.2}$$

which can be interpreted probabilistically as assuming that $\mathbf{x}$ is Gaussian distributed, centered at $\mathbf{W}\mathbf{z}$ with some small additive spherical noise: $p(\mathbf{x}|\mathbf{z}) \sim \mathcal{N}(\mathbf{W}\mathbf{z}, \sigma^2\mathbf{I})$.

2.1.1 Regularized Linear Models

The most common encoding and dimensionality reduction technique, Principal Component Analysis (PCA; Pearson, 1901; Hotelling, 1933) can be expressed naturally as a linear encoding model as above. To discover the basis vectors $\mathbf{W}$ for PCA, the most common method is to compute the closed-form singular value decomposition (SVD) of the data matrix $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$; then $\mathbf{W} = \mathbf{U}_{:k}\mathbf{S}_{:k}^{-1}$ is the k largest singular vectors, weighted by the inverse of the corresponding singular values. However, one can also compute $\mathbf{W}$ by minimizing a loss function with respect to the codebook:

$$\mathcal{L}_{PCA}(\mathbf{W}) = \frac{1}{2m}\sum_{i=1}^{m}\|\mathbf{W}\mathbf{W}^\top\mathbf{X}_i - \mathbf{X}_i\|_2^2 \tag{2.3}$$

$$\text{s.t.c.} \qquad \mathbf{W}\mathbf{W}^\top = \mathbf{I}_k. \tag{2.4}$$

Once $\mathbf{W}$ has been computed, then $\mathbf{Z} = \mathbf{W}^\top\mathbf{X}$. Encoding and decoding $\mathbf{X}$ using a PCA codebook recovers the data matrix exactly when $k = \mathcal{R}(\mathbf{W}) =$

$\mathcal{R}(\mathbf{X})$ (here, $\mathcal{R}(\mathbf{A})$ is the rank of $\mathbf{A}$). When $k = \mathcal{R}(\mathbf{W}) < \mathcal{R}(\mathbf{X})$, then the optimization process described above recovers the subspace spanning the top k eigenvectors of the covariance matrix of $\mathbf{X}$, which corresponds to the subspace computed using the SVD.

By formulating PCA as an optimization, many interesting parallels become possible. For example, another commonly used coding technique, particularly for sensory data like images and sounds, is the Independent Component Analysis (ICA; Hyvärinen and Oja, 2000; Bell and Sejnowski, 1997; Jutten and Herault, 1991). Many formulations of computing an ICA codebook have been proposed, but there exists a formulation with a nice parallel to the formulation for PCA above. Again the decoding process is expressed as the linear operation, but in computing $\mathbf{W}$, instead of trying to minimize the sum of the squared reconstruction errors, ICA tries to minimize the coefficient values:

$$\mathcal{L}_{ICA}(\mathbf{W}) = \frac{1}{2m} \sum_{i=1}^{m} g(\mathbf{W}^\top \mathbf{X}_i) \tag{2.5}$$

$$\text{s.t.c.} \qquad \mathbf{W}\mathbf{W}^\top = \mathbf{I}_k \tag{2.6}$$

where typically $g(\cdot) = \log(\cosh(\cdot))$ is used as a continuous approximation to the L1 norm.

In the expression of ICA given in Equation 2.6, the basis $\mathbf{W}$ must still be orthonormal, which prevents it from being overcomplete: with an orthonormal basis, k must be less than or equal to d. To work around this restriction, Le et al. (2011) proposed RICA, a novel way to define ICA that

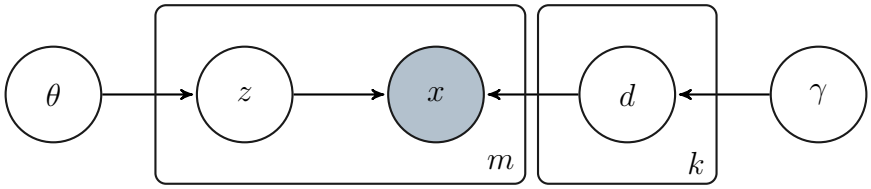


Figure 2.1: Generative structure of a simple coding model. Hyperparameters θ can be used to specify a prior on coefficients z, while γ can be used to specify a prior on codebook vectors d.

permits overcomplete codebooks:

$$\mathcal{L}_{RICA}(\mathbf{W}) = \frac{1}{2m}\sum_{i=1}^{m} \|\mathbf{W}\mathbf{W}^\top\mathbf{X}_i - \mathbf{X}_i\|_2^2 + \lambda\|\mathbf{W}^\top\mathbf{X}_i\|_1. \tag{2.7}$$

Note here the unconstrained optimization, which encourages the basis to be orthonormal but is not a hard constraint. ICA is expressed probabilistically by assuming the same Gaussian observation model as in PCA, but with an additional prior over the coefficients.

In this sense, the sparse coding technique developed by Olshausen and Field (1996) and Tibshirani (1996) simultaneously computes $\mathbf{W}$ and $\mathbf{Z}$ given $\mathbf{X}$:

$$\mathcal{L}_{SC}(\mathbf{W}, \mathbf{Z}) = \frac{1}{2m}\sum_{i=1}^{m} \|\mathbf{W}\mathbf{Z}_i - \mathbf{X}_i\|_2^2 + \lambda\|\mathbf{Z}_i\|_1 \tag{2.8}$$

$$\text{s.t.c.} \qquad \|\mathbf{W}_j\|_2 \leq 1 \text{ for } j = 1, \ldots, k. \tag{2.9}$$

Note that here the basis vectors are constrained to be (at most) unit-length, but are not constrained to be mutually orthogonal as in PCA or the original formulation of ICA.

Once a loss is formulated for a particular modeling problem, additional sparsity constraints can be added to the problem easily by adding additional

terms to the loss. In particular, to isolate specific elements in the codebook vectors, they can be made sparse:

$$\mathcal{L}(\mathbf{W}, \mathbf{Z}) = \frac{1}{2m} \sum_{i=1}^{m} \|\mathbf{W}\mathbf{Z}_i - \mathbf{X}_i\|_2^2 + \lambda \|\mathbf{Z}_i\|_1 + \gamma \|\mathbf{W}\|_1. \tag{2.10}$$

Note that this makes the optimization more difficult and has an additional hyperparameter to tune, but incorporating terms into losses like this provides modelers with a way of translating their intuitions about a problem into the mathematics that drive computer simulations.

2.1.2 Alternative Coding Models

The losses for many of the linear models share a common "factorization" viewpoint on the dataset being modeled. These models attempt to represent (recreate under a squared-error loss) a dataset $\mathbf{X}$ by factoring it into a basis matrix $\mathbf{W}$ and a coefficient matrix $\mathbf{Z}$ (see Equation 2.1). Often a regularization penalty is added to the coefficients to encourage sparsity.

This coding pattern has a closely related counterpart known as non-negative matrix factorization (NMF; Paatero and Tapper, 1994; Hoyer, 2004). In NMF, the squared error loss is optimized, subject to the constraint that all elements of $\mathbf{X}$, $\mathbf{W}$, and $\mathbf{Z}$ are non-negative. This is a natural condition for some domains (e.g., recommendation systems, word-count analysis in text documents, spectrogram modeling) and can easily be enforced for negative $\mathbf{X}$ by adding a constant to every element of the matrix.

In practice, NMF yields representations that, like sparse codes, are

often easy for human modelers to interpret. In addition, the NMF optimization process, while not closed-form, is easy to accomplish using stochastic gradient techniques.

Finally, another common technique for representing the distributions of large datasets is known as k-means (MacQueen, 1967). This approach 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, $\mathbf{z} = [\xi_1 \dots \xi_k]^\top$ such that

$$\xi_i = \begin{cases} 1 & \text{if } \|\mathbf{x} - \mathbf{W}_i\|_2 < \|\mathbf{x} - \mathbf{W}_j\|_2 \text{ for } j \neq i \\ 0 & \text{otherwise.} \end{cases} \tag{2.11}$$

The basis corresponding to this coding approach is typically learned from a training dataset by setting the $\mathbf{W}_i$ to random perturbations of elements from the training data, and then iteratively adjusting the columns of $\mathbf{W}$ so that the sum of the Euclidean distances from each data point to its closest codebook vector is minimized. This is equivalent to a Gaussian mixture model with fixed, spherical covariance and hard cluster assignments. Additionally, k-means is equivalent to an NMF model in which the representations are maximally mutually orthogonal.

2.1.3 Neural Network Models

Formulating a loss for a coding model brings within reach many interesting techniques for computing nonlinear codes. Nonlinear coding is particularly active in the domain of neural networks, which can be viewed as computation

graphs with a particular ℓ–partite structure.

Many types of neural networks are used to approximate an unknown map from some dataset to a target label for each data point. Classification and regression are two of the most basic tasks in this area. However, neural networks can also be applied to datasets without labels, in which case these "autoencoder" models look like an encoding-decoding pair whose parameters are jointly optimized.

Consider an autoencoder with one "hidden" layer and a nonlinear activation $\sigma(\cdot)$. The optimal parameters for this model when applied to dataset $\mathbf{X}$ are given by minimizing the loss

$$\mathcal{L}_A(\mathbf{W}, \mathbf{V}) = \frac{1}{2m} \sum_{i=1}^{m} \|\mathbf{V}\sigma(\mathbf{W}_i\mathbf{X}_i) - \mathbf{X}_i\|_2^2 \tag{2.12}$$

where $\mathbf{W}$ is the matrix of parameters for the encoding process, and $\mathbf{V}$ is the matrix of parameters for the decoding process. (Often there are also associated vectors of bias parameters, but these have been omitted for brevity.)

Clearly, if the weights in the encoder and decoder are "tied" and $\sigma(z) = z$, then the resulting model is equivalent to a PCA model. But neural networks permit the addition of nonlinearities such as the logistic or rectified linear activation functions, respectively:

$$\sigma(z) = \left(1 + e^{-z}\right)^{-1} \tag{2.13}$$

$$\sigma(z) = max(0, z). \tag{2.14}$$

These activation functions map the input data into a nonlinear space

and provide, in theory, the ability to model any function. However, in practice optimizing the loss for a neural network, particularly a "deep" one with many layers, proves difficult in practice. Much research work in the past decade has been devoted to devising new stochastic gradient algorithms that can tune parameters in these models effectively (e.g., Zeiler, 2012; Graves, 2013; Kingma and Ba, 2014). Similarly, research has produced semi-linear activations (Nair and Hinton, 2010; Glorot et al., 2011; Goodfellow et al., 2013) and novel regularization techniques (Vincent et al., 2008; Hinton et al., 2012) that help make the loss for network models more trainable in practice.

2.2 Sequence Models

While sparsity appears to be an important driver of feature learning in natural datasets, most research in this area has focused on "static" sensory data such as images and sounds. Movements, in comparison, unfold critically through time, requiring models that can capture some notion of temporal structure among variables.

State space models are a broad class of models that are used to infer latent state trajectories $z_1 \dots z_T$ from observations $x_1 \dots x_T$ according to the general recurrence:

$$z_{t+1} = f(z_t, u_t) + \epsilon_t^z \tag{2.15}$$

$$x_t = g(z_t) + \epsilon_t^x \tag{2.16}$$

where the observation emissions and state transitions are governed by $g(\cdot)$ and

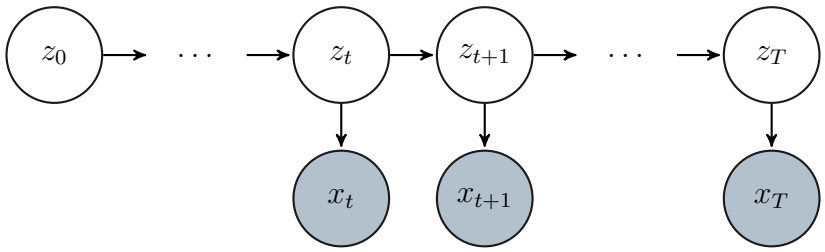


Figure 2.2: Generative structure of a simple state space model without control inputs.

$f(\cdot)$, u_t represents the control to the system (if any), and ϵ^z and ϵ^x represent noise in state transitions and observation emissions.

Several families of state space models are widely used to represent and understand temporal data. For example, Hidden Markov Models (HMMs; Rabiner and Juang, 1986) assume discrete, evenly-sampled time, discrete states and observations, and linear dynamics; inference is relatively easy thanks to dynamic programming methods, but their expressive range is somewhat limited. At another extreme, recurrent neural networks (RNNs; e.g., Elman, 1990; Jaeger and Haas, 2004; Sutskever, 2013) can represent nonlinear dynamics and observations but are quite difficult to train; the current state-of-the-art relies on sophisticated gradient-based methods.

Recent developments in feature learning and neural networks have started to change the way that high-dimensional temporal data are modeled. For example, HMMs have been combined with Restricted Boltzmann Machines (RBMs; Hinton, 2002; Hinton et al., 2006) to extract powerful features from speech data, surpassing the state-of-the-art by a wide margin (Mohamed

et al., 2009, 2010a,b; Mohamed and Hinton, 2010; Dahl et al., 2010; Jaitly and Hinton, 2011). To model human walking data, Taylor et al. (2007) developed the conditional RBM (cRBM), a variant of the RBM that computes a time-dependent bias for each hidden unit (see also Taylor, 2009). To model sequences of letters in text, Sutskever et al. (2011) proposed the multiplicative RNN, a recurrent neural network with a factored, input-dependent dynamics. Several variants of Long Short-Term Memory (LSTM; Hochreiter and Schmidhuber, 1997) and, more recently, the Gated Recurrent Unit (GRU; Chung et al., 2014) have established a new place for recurrent models as best-performers across a variety of remarkable modeling tasks such as providing text annotations for video (Venugopalan et al., 2014) or high-quality machine translations (Sutskever et al., 2014).

2.3 Multi-View Models

Even more difficult than modeling movement alone is the task of understanding how sensation and motion interact. Cognitively, the primate brain appears to have a sophisticated infrastructure for modeling the actions of the individual as well as conspecifics (e.g., Rizzolatti et al., 1996, 1997). In the domain of speech understanding, a long-standing proposal by Liberman et al. (1967) (updated in Liberman and Mattingly (1985) and reviewed more recently by Massaro and Chen (2008)) argues that understanding speech relies on an estimate of the talker's motor gestures, in addition to the acoustic information generated by the talker's vocal apparatus. This argument has recently been

supported by experimental evidence: Pulvermüller et al. (2006) showed that the lip and tongue areas of motor cortex activate preferentially in humans listening to labial and dental consonants, respectively; and D'Ausilio et al. (2009) showed that transcranial magnetic stimulation (TMS) of the same areas of motor cortex had significant effects on the accuracy and speed when recognizing corresponding syllables.

Clearly, humans have sophisticated capabilities for modeling sensory and movement information together. How exactly these capabilities are realized remains far from clear, yet a few pieces of research over the past two decades have attempted to model this process explicitly. The most common domain for this modeling comes, like the MSP, from speech generation and perception. Both Markey (1994) and Guenther (1994) developed explicit models of neural information processing integrating speech processing and synthesis; the model by Markey is cast in a reinforcement learning framework, while that of Guenther explicitly tries to capture the neural information processing that is known to take place in humans.

Multi-view models seem critically important for animals in developing appropriate control systems. In the famous "kitten carousel" experiment, Held and Hein (1963) raised pairs of kittens in the dark, except for limited time in a "carousel" that permitted one "active" kitten's movements to affect the position and orientation of both kittens in the world. The second, "passive," kitten was restrained inside a basket so that it could only move its head. The two kittens were linked so the actions of the active kitten also changed the

position and orientation of the passive kitten. In this way, both kittens received equivalent visual input during development. However, the active kitten developed normally, while the passive kitten, when released from the basket after six weeks of training, had difficulty moving and appeared not to have developed normal depth perception.

More general models have been developed to integrate sensory and motor information; in general, this task used to fall under the label of "semi-supervised learning" or "co-learning" (de Sa, 1994), and today the more appropriate label is "multi-view learning" (e.g., Memisevic, 2012). However, perhaps the first principled approach to modeling data in two modalities simultaneously is Canonical Correlation Analysis (CCA Hotelling, 1936). CCA is a generalization of PCA that incorporates two observed vector spaces simultaneously when considering what makes an optimal space for representation. Whereas in PCA the coded representation $\mathbf{Z}$ of a dataset $\mathbf{X}$ is guaranteed to preserve the most variance in $\mathbf{X}$, the coded representation $\mathbf{Z}$ in CCA is guaranteed to preserve the maximal variance in both of the observed spaces $\mathbf{X}_1$ and $\mathbf{X}_2$.

Many other multi-view models exist. Coen (2006) developed a general-purpose model to capture two simultaneously occurring modalities of data, and applied it to phoneme classification. Lange and Riedmiller (2010) and related work has focused more explicitly on coding sensory information so that it becomes useful for reinforcement learning approaches.

Chapter 3

Motor Control

Motor control is an immense area of scientific research, and much progress continues to be made in motor control research. However, there are two existing problems with models of motor control. On one hand, there are no good models of motor control: it is an immensely complex subject, and no single research group or team can safely claim to know everything about motor control. On the other hand, there are too many good models of motor control: each of the dozens of scientific disciplines that are concerned with movements have developed their own models of control, correspondingly emphasizing the aspects of movement or control that are most important in the respective discipline. This balkanization of scientific ideas is not unique to motor control, but it does make summarizing all of motor control nearly impossible. Nonetheless, this chapter attempts to describe several existing models for motor control. The goal is to give a sense of the breadth of existing work, rather than to provide a complete review of all different models. (See Wolpert (1997); Schaal (1999); Scott (2004); Turvey and Fonseca (2009); Franklin and Wolpert (2011); Haith and Krakauer (2013) for reviews from the perspectives of robotics, optimization, and neuroscience.)

To have some sort of framework for thinking about motor control, this chapter considers four broad groups of motor control models. Like most scientific partitioning schemes, this one falls apart quickly when inspected in detail; however, the distinctions remain important when thinking about motor control at a high level. The four approaches we will consider here are focused on force, position, optimization, and learning.

3.1 Force Control Models

Some models of motor control think of movement as forces. In these models, muscles—or, for robots, motors—provide some contractile or rotational force, and this force causes the skeleton to move. In a force-based control model, an agent continually sends force "instructions" to its motor elements, and the motor elements' forces cause continual posture changes to attain the movement goal of the agent.

In models based on forces, the agent's nervous system—or, for robots, some computational equivalent—is responsible for computing an *inverse transform* to change the agent's cognitive goals into the forces that will accomplish these goals. Once the forces have been computed, the agent's body, which is more generally known as the *plant*, uses these forces to generate movement, in a process known as the *forward dynamics*.

To accomplish goals, a cognitive agent must map the desired world state of its body somehow onto the forces that are required to move the body into this goal state. It is quite difficult to invert the forward dynamics of a general

plant, but one way of performing the inverse transform is to learn a simplified, invertible model of the forward dynamics and invert the forward model. This approach was discussed at length by Jordan and Rumelhart (1992), particularly in the context of using a neural network as both the forward and the inverse model; this technique has also been used to learn a controller for backing up trucks with long trailers (Nguyen and Widrow, 1989), and as a mechanism for producing speech using a forward speech synthesizer (Howard and Messum, 2007). A related approach, PILCO (Deisenroth and Rasmussen, 2011), learns a generic model of the dynamics of the world using a Gaussian process (MacKay, 1998; Rasmussen and Williams, 2005).

In humans, it is widely speculated that the cerebellum provides information to the motor cortex that is in some way analogous to the information that a mathematical forward model of the body's dynamics would provide. Wolpert et al. (1995) showed that force perturbations during planar arm movements resulted in movement dynamics that closely match those of a simulated system using a Kalman Filter (Kalman and Bucy, 1961) as a forward model (also see Kawato, 1999; Berkes et al., 2011).

At the muscle level, force-feedback loops have been shown to be important in stabilizing gait patterns (Desai and Geyer, 2013). Additionally, central pattern generators (CPGs; Ijspeert, 2008; Frigon, 2012) are thought to be primitive "factories" of movement dynamics. Embedded in the spinal cord, CPGs appear to be responsible for cyclic, closed-loop feedback-based movements.

3.2 Position Control Models

In contrast with force-based models, some models of motor control think of movements as changes in the positions of joints and body segments. The most notable of these approaches is known as the Equilibrium Point (EP) hypothesis (Asatryan and Feldman, 1965; Feldman, 1966; Bernstein, 1967), the λ–theory, or, more recently, the Threshold Control Theory (TCT; Feldman and Levin, 2009).

According the TCT, voluntary movements are made in animals by specifying the kinematic threshold at which neural activity starts to recruit motoneuron activity in muscles. To make a movement, the organism's central nervous system specifies a "threshold posture" where all muscles are just about to be activated. Any environmental effects that move the muscles past these positions—for example, gravity acting on body segments, the momentum of previous movements, the force of a load being exerted on the body—stretch some muscles past their threshold position, which causes motoneurons to be recruited in those muscles and results in a contraction. Proponents of the TCT emphasize that the interaction between organism and environment is critical for specifying movements; for example, humans find it cognitively effortless to speak while standing up or while lying down, and yet the muscles involved in moving the lungs appropriately during this action are almost completely different.

Feldman and Levin (2009) emphasize that the TCT, like other position-based control theories, provides a fundamentally different approach to move-

ment than force-based theories, stating that "although motor actions are described in terms of mechanics and EMG patterns, the question of how these actions are controlled cannot be answered in these terms." The TCT attempts to describe how voluntary movements are specified from a more cognitive perspective, while force-based approaches attempt to describe the dynamics of voluntary movements. Both approaches are important; but, as described at the outset of this chapter, movement is complex enough to have multiple levels of analysis, each of which prefers a different type of explanation.

Position-based control has garnered some recent support from studies of motor cortex in monkeys (Graziano et al., 2002, 2005). In these results, complex microstimulation of M1 neurons was observed to elicit coordinated, multijoint arm movements such that, regardless of initial state, the monkey's hand ended up in one of several distinct kinematic states. For example, one stimulation site caused the hand to be brought to the animal's mouth—again, regardless of initial state prior to stimulation. These studies appear to provide some basis for the idea that the motor cortex has some way of representing movements in terms of kinematic positions, rather than in terms of the forces required by the muscles to attain those positions.

Several computational approaches to movement modeling also adopt a kinematic-only perspective. Wang et al. (2008) proposed a widely used model based on Gaussian Processes (GPs; Rasmussen and Williams, 2005) that has primarily been used to model changes in joint angles over time. In this model, an unobserved latent space is hypothesized to generate observed

movement data; the latent variable is sampled from a GP to encourage smooth transitions in the latent space, and the movement data are sampled from a GP conditioned on the latent variable, again to encourage smoothness. Calandra et al. (2014) proposed an extension that used a neural network to learn the latent space, in order to handle moments where movements change suddenly (for example, a heel strike during walking). Taylor et al. (2007) developed a neural network model that, when tuned appropriately, could capture different styles of walking movements.

Although these approaches mostly focus on modeling the kinematics of a movement, it must be emphasized that not necessarily all of the approaches adopt a strictly position-based control model. Instead, many of the computational studies mentioned here are aimed only at capturing the patterns in a database of movement; often the modelers assume that, should an actual movement be required, the kinematic models can be used to specify dynamic sequences of set-points, and an off-the-shelf, low-level controller—like a PID controller—can be used to generate the appropriate forces or torques should a need arise.

A related line of work has pursued the idea that computing inverse quantities like joint torques can be accomplished by learning a direct mapping from joint angles, using some parametric model (Nguyen-Tuong and Peters, 2010; Johnson and Ballard, 2014b). Once a mapping is learned, the agent can plan movements by simulating their effects using the current state of the movement as input.

3.3 Optimal Feedback Control Theory

There appears to be a tension between strictly force-based models and strictly position-based ones. Force-based models attempt to describe movements primarily at a low level, while position-based models (especially the TCT) tend to describe movements at a high level. A third school of thought in motor control revolves around the idea that movements are the result of some optimization procedure that the brain computes in order to achieve cognitive goals: as long as the central nervous system is computing an inverse model, or mapping from positions to forces, we might be able to model this computation as an optimization process (Todorov and Jordan, 2002; Scott, 2004; Todorov, 2004; Diedrichsen et al., 2009).

Like other models of control, optimal feedback control captures something intuitively important about movement, notably the idea that movement is somehow optimized to accomplish some goal effectively. However, the approach is not without its critics (e.g., Friston, 2011), and, much like Bayesian approaches, it also faces some practical difficulties in the sense that the computations required to arrive at a solution are often prohibitively expensive.

In addition to practical considerations, an important theoretical issue lies in optimal control: it is nearly impossible to know which objective is being optimized (Nelson, 1983). While optimal control is an appealing approach because optimization is an intuitive high-level idea, and it can be applied so widely in theory, at some point it becomes difficult for an approach based

purely on optimization to reveal more about the process that an actual organism is using to make movements.

3.4 Learning To Control

Having a model of motor control is all well and good, but *learning* is critical for individual agents to use their movements effectively. In many instances, a model has parameters that require tuning to a particular body configuration or goal; learning is a generic term for the process of adjusting parameters to better accomplish some goal.

If models of motor control constitute a wide field of research, learning encompasses yet more scientific territory; for every theory of motor control there is at least one—but often many more—associated mechanism for learning and adaptation. So, just like the families of control models above, the methods listed here are only a brief survey of the existing research on motor learning.

3.4.1 Gradient-Free Methods

One way of tuning a motor controller is to guess the appropriate parameter settings and try it out. This process works less well for an individual organism than it does for an entire population; as such, methods that do not use gradient information to tune their control parameters often make use of some form of evolutionary computation. Broadly speaking, evolutionary approaches maintain a "population" of potential solutions to a control problem and iteratively improve the entire population by evaluating each solution's

"fitness" and then creating noisy copies of the fittest performers for the next "generation."

Sims (1994) appears to be one of the first—and probably the most widely recognized—examples of using evolution to create simulated creatures that can move to accomplish tasks (e.g., touching a light in the environment). Sims' work also highlighted one of the natural advantages of evolutionary approaches to learning controllers: not only can they search through the solution space for a control solution, they can just as easily search through the joint space of control solutions *and plants* (although this tends to make the search space larger).

Jointly optimizing both the controller and the body was used again nearly twenty years later to develop remarkably robust control mechanisms for animated, bipedal organisms with nonlinear actuator dynamics (Geijtenbeek et al., 2013). Instead of a "classic" genetic algorithm, this work used CMA-ES (Hansen et al., 2003) to model the fitness landscape; CMA-ES has also been used in developing optimal controllers for robots in simulated soccer environments (Urieli et al., 2011; Farchy et al., 2013).

While evolutionary methods have produced some impressive solutions to difficult control problems, they also tend to require large amounts of computation time, as the search strategy is basically "guided" brute-force through the entire solution space. On the other hand, algorithms in this family (e.g., NEAT and NERO; Stanley and Miikkulainen, 2002; Stanley et al., 2005) can easily incorporate dynamic changes in the fitness score (e.g., "shaping"), and

they have been used effectively in this area to evolve controllers for game agents.

3.4.2 Reinforcement Learning

Reinforcement Learning (RL; Sutton and Barto, 1998) is a widely used technique for optimizing control policies. Rather than requiring continuous supervision during a movement or a task, RL algorithms rely only on the presence of positive or negative feedback about task outcomes. RL can learn effective control policies even when the reward signals are sparse (i.e., most rewards are zero), and even in stochastic environments.

Markey (1994) and Guenther (1994) developed RL-based approaches to vocal learning. Markey's approach attempted to match predefined phonetic features extracted manually from target speech sounds. Guenther's model contained several dedicated neural network modules, including realistic neural delays in the interconnections among these modules. In addition, Guenther's model applied intermittent control (Milton, 2013) to keep the speech output within some predefined error tolerances.

Mnih et al. (2015) presented a deep neural network architecture that is capable of playing Atari video games. This approach used a multilayered, convolutional neural network to map the pixels of the game screen onto value estimates for each of the possible control actions permitted by the Atari simulator. This work was the first successful replication of the general approached used twenty years earlier to create professional-level backgammon players (Tesauro,

1995).

Gomez et al. (2006) compares several evolutionary algorithms to several RL algorithms in terms of learning speed and CPU requirements.

In reinforcement learning, many researchers have noticed that complex control problems can be factored in several ways to make the learning process tractable. One approach is to factor control problems in time using a hierarchy of controllers (Sutton et al., 1999). A related approach relies on a set of low-level parametric "motor primitives" (Schaal et al., 2000, 2003a,b; Ijspeert, 2008; Muelling et al., 2010), inspired in part by the "central pattern generators" seen in vertebrate spinal cords (Frigon, 2012). A third approach is to factor state spaces into parallel "modules" (Sprague and Ballard, 2003; Rothkopf and Ballard, 2013).

Many of these approaches break control problems down using prior knowledge of the problem domain, but techniques for learning control manifolds from data alone would be useful in identifying either (a) common temporal patterns of movement, or (b) common patterns spanning multiple degrees of freedom (e.g., "synergies"; Latash et al., 2002). The work described in this document fits most closely with existing work in this area. For example, Bocsi et al. (2013) identified kinematic subspaces using PCA in two robots independently, then learned a mapping between them. Kober et al. (2008) used a system built using motor primitives to learn co-occurring perceptual features in the environment. Berniker and Kording (2015) captured the variance in movement trajectories using a deep neural network model. Lange and Ried-

miller (2010) used a deep neural network to map from visual features to a value representation.

Part II

Computational Work

Chapter 4

Modular Control with Attention[1]

Humans and other animals routinely interact with complex, noisy, dynamic environments to accomplish tasks in the world. In the motor domain, for example, locomotion requires a sequence of coordinated, multi-joint movements arranged in time that simultaneously (a) place the feet in stable positions on the ground while also (b) maintaining balance and (c) making progress toward some higher-level goal. At a more cognitive level, while driving a car, a person navigates to a desired destination (e.g., grocery store) while paying attention to different types of objects in the environment (pedestrians, vehicles, etc.) and obeying traffic laws (speed limit, stop signs, etc.). In all of these situations, and many others, the subject must decide continuously which action to execute on the basis of noisy sensory information that flows into the brain through several relatively narrow channels. In the case of primates especially, the retinal fovea provides a relatively small visual region of extremely high acuity, which must be actively moved to different targets to obtain high-resolution imagery (often while simultaneously executing other motor plans)

[1]The work described in this chapter was published in a conference venue by Johnson et al. (2013c) and as a journal paper by Johnson et al. (2014). The human behavioral data were collected and analyzed by Sullivan et al. (2011). B. Sullivan and I developed and tested the computational model, with much help from D. Ballard and M. Hayhoe.

to gather new information from the environment.

Humans are particularly adept at balancing competing task demands while simultaneously gathering information from the world through this foveated visual system. During the deployment of overt attention—measured here using eye movements towards an object—humans are sensitive to bottom-up salience (color, motion, etc.) as well as top-down task priority and the rewards associated with a task (Knudsen, 2007; Wolfe et al., 2003). In particular when engaged in "natural" tasks, eye movements are largely directed towards task relevant objects (Hayhoe and Ballard, 2005; Land and Hayhoe, 2001). Typically in natural environments, there are multiple task relevant objects spread over space and time that require active visual strategies to properly gather information. While human vision research has often focused on models of visual saliency, i.e., a stimulus based controller of attention (Bruce and Tsotsos, 2009; Itti and Koch, 2001; Zhang et al., 2008), such models are inappropriate to address task-based behavior because they do not incorporate information about the state of the agent whose vision is being modeled. An alternative approach is to consider vision as part of a control process where information from the senses is used to guide motor behavior (Butko and Movellan, 2010; Nunez-Varela et al., 2012; Senders, 1980; Sprague and Ballard, 2003; Sullivan et al., 2011). Both stimulus and task-based approaches have led to a variety of formulations concerning how eye movements should be selected, e.g., using energy models, information theoretic measures, or measures of reward and uncertainty. In the present work, we focus on how selection of eye

movement targets may be controlled in part by task related uncertainty and reward.

We present a model of visual processing and control that simultaneously takes into account the reward and uncertainty in multiple tasks associated with a dynamic, noisy driving environment. The model successfully accounts for variations in gaze deployment seen in humans driving in a virtual reality driving environment. Additionally, we discuss future research allowed by inversion of the soft barrier model. Inversion allows human data to be mapped into parameters in the model space so that it can be understood and compared quantitatively within the model framework.

4.1 Model

The model described in this chapter follows the modular architecture of Sprague and Ballard (2003) by factoring complex behaviors like driving into a set of simple control modules that each focus on a well-defined task—for example, a module to follow the road and another to avoid oncoming cars. Intuitively, a module is an abstract black-box controller that can be used alone to guide an agent through a single task. More interestingly, modules can be used together dynamically to engage in multiple ongoing behaviors. While the human visual system is highly parallel, processing and attentional focus are largely biased towards the fovea, meaning humans typically get information in a serial fashion by foveating different objects over time. In our model, multiple task modules run concurrently; however, to incorporate the foveation

constraint, only one module at a time actively gains new perceptual information.

At a high level, modules are responsible for gathering and updating information about specific aspects of the state of the world, and for using that information to generate control signals for the agent. A central component of the model is that it requires a usable control policy even in the absence of updating its state information. Human short term memory decays with time, so to simulate this we allow the state information upon which the actions are computed to be corrupted by noise. We incorporate these into our model using simple scalar values for each module. Formally, we define a module as a tuple $M = (\mathcal{S}, \mathcal{A}, \pi, \mathbf{s}^*, \rho, \epsilon)$, where:

- $\mathcal{S} = \{s_1, \ldots, s_n\}$ is a set of the n state variables that are relevant to the module,
- $\mathcal{A} = \{a_1, \ldots, a_k\}$ is a set of the k action variables that are relevant to the module,
- $\pi : \mathbb{R}^n \rightarrow \mathbb{R}^k$ is a control policy that maps state values onto actions,
- $\mathbf{s}^* \in \mathbb{R}^n$ is a vector of target state values,
- ρ is a scalar uncertainty threshold value for the module, and
- ϵ is a scalar noise value for the module.

The first three elements of M are common to typical Markov decision process (MDP) scenarios. The state space, spanned by elements of $\mathcal{S}$, represents all possible combinations of world state that are relevant for the task. The action space, spanned by elements of $\mathcal{A}$, describes all possible actions that the agent can take. The control policy maps states to actions; an optimal policy maps states to the best action for each state. The fourth element of the tuple, $\mathbf{s}^*$, is a vector of target values for each state variable. These target values are used in place of the more traditional formulation of scalar reward; this is discussed in further detail below. Finally, each module incorporates explicit values for both task priority $1/\rho$ and task uncertainty ϵ, which are also explained below.

A learning agent is equipped with N individual modules $M^{(1)}, \ldots, M^{(N)}$ that each specialize in one task and can be used in conjunction to control behavior in the world. To simplify the control problem, in our model all modules share a common set of action variables. In the driving environment described in this chapter, there are two action variables: one represents changes in the vehicle's speed and another represents changes in the vehicle's heading.

4.1.1 State Estimates

Each module depends on a set of world-state variables that are relevant to the module's specific task. When driving, relevant state variables for a car-following task, for example, could include the agent-centric distance and angle to the leader car, the speed and heading relative to the leader car, etc. Relevant

state variables for a target-speed task might be as simple as monitoring the absolute speed of the agent.

In MDP scenarios, agents are assumed to have constant access to accurate state variable information. Humans, on the other hand, have a foveated visual system that often requires active serial collection of updated state information. We assume that this serial process requires that when one visual task is accessing new information all other tasks must rely on noisy memory estimates.

To incorporate this state uncertainty into the model, each module $M^{(i)}$ maintains an explicit *estimate* of the current value of each of its state variables, $\hat{\mathbf{s}}^{(i)}(t)$. (We will henceforth omit the module superscript except to resolve ambiguities.) This estimate could be designed to incorporate many sorts of prior information about the evolution of the world, but the model in its current state simply treats state estimates as samples drawn IID from a spherical normal distribution

$$\hat{\mathbf{s}}(t) \sim \mathcal{N}\left(\mu(t), \sigma^2(t)I\right) \tag{4.1}$$

where $\mu(t) = [\,\mu_1(t)\ \ldots\ \mu_n(t)]^\top$ is a vector of the most recently observed state values, and $\sigma(t)$ is the standard deviation for the state variable estimates in the module. Figure 4.1 shows the state updates over time for a simple, hypothetical system containing two modules, each tracking one state variable.

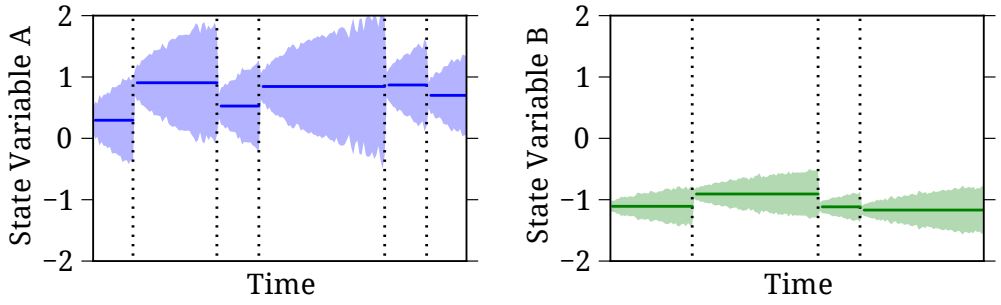


Figure 4.1: Evolution of state variable and uncertainty information for two single-variable modules. On the left, the solid blue lines represent the observed values of the state variable A over time, while the shaded blue regions represent the region in which the true value of the state variable is likely to occur. On the right, the observed values and uncertainty regions are shown in green for a different state variable, B. Vertical dashes in each plot indicate times where the state estimates are updated with a new observation of the true value; these updates also reduce the magnitudes of the uncertainties in each estimate towards zero. The ϵ parameter in this scenario is greater for the module tracking variable A than for the module tracking variable B.

4.1.2 Uncertainty Propagation

Uncertainty propagates over time within each module by maintaining a small set of J "uncertainty particles" $\mathcal{E} = \{\beta_1(t), \ldots, \beta_J(t)\}$. Each particle represents one potential path of deviation that the true state value might have taken from the last-observed state value. At every time step in the simulation, all uncertainty particles are displaced randomly by a step drawn from $\mathcal{N}(0, \epsilon)$, thus defining a random walk for each particle. The root-mean-square value of the uncertainty particles is then used to define the standard deviation

$$\sigma(t) = \sqrt{\frac{1}{J}\sum_{j=1}^{J}\beta_j^2(t)} \tag{4.2}$$

of state estimates for this module. Periodically, a module will be updated with accurate state information from the world (described below); when this happens, the magnitude of each uncertainty particle for the module is reduced according to $\beta_j(t+1) = (1-\alpha)\beta_j(t)$. After an informal parameter search, we set $\alpha = 0.7$ for all modules; with $\alpha = 1$, the model tends to produce many short updates because uncertainty is instantly reduced to 0, but with $\alpha < 1$ the uncertainty increase due to noise competes with the uncertainty reduction from the updated state information. Figure 4.2 shows the uncertainties over time for the hypothetical two-module system shown in Figure 4.1.

The state estimation approach described here can be seen as a sort of particle filter (e.g., Arulampalam et al., 2002), using an uninformed proposal distribution and equal weights for all particles. Interestingly, the behavior of the simulation was largely unaffected by the choice of J; for our simulation, we used $J = 10$.

4.1.3 Control Policy

Each module relies on a policy to determine which action to take when the world is in a particular state. There are multiple ways an MDP may be solved for a control policy, e.g. in reinforcement learning a Q–table can be learned, which explicitly represents the expected future reward for each possible state and action combination; the policy is then given by a simple maximum over available actions for each state.

For a task like driving, however, continuous variables are the most nat-

ural representation of state (distance to another car, current speed, etc.) and action (change in speed, change in steering) variables. Although MDP algorithms can converge on policies for tasks in continuous spaces, for many real-world tasks the resulting policies can be more easily modeled using a simple parametric function. In addition, many algorithms for solving MDPs require significant training time to arrive at these regularly-shaped policies. The model described here instead uses a continuous proportional-integral-derivative (PID) control strategy.

4.1.4 PID Controllers

A PID controller $C(e)$ is a feedback control functional that maps state errors $e(t)$ onto control signals $u(t)$. Formally,

$$C(e) = K_P\, e(t) + K_I \int_0^t e(v)\, dv + K_D\, \dot{e}(t) \tag{4.3}$$

where K_P, K_I, and K_D are parameters that affect the convergence speed and stability of the PID controller output when encountering a step change in error. In our model, these parameters are tuned manually for each module in isolation (O'Dwyer, 2006) to produce qualitatively appropriate driving behavior.

Each module in the model uses one PID controller for each state variable. Given estimates $\hat{\mathbf{s}}(t)$ of the current values of each variable and a vector of target state values $\mathbf{s}^*$, the control policy becomes

$$\pi(\hat{\mathbf{s}}(t)) = U\,[\; C_1(\hat{s}_1(t) - s_1^*)\; \ldots\; C_n(\hat{s}_n(t) - s_n^*)\;]^\top \tag{4.4}$$

where U is a $k \times n$ mixing matrix that combines control policy recommendations from each PID controller into a final control output for each action variable. Note that, in this model, the control policy π does not have access to the true state values $\mathbf{s}(t)$, but rather to the module's estimates of those state values $\hat{\mathbf{s}}(t)$.

The composition of U is determined by the needs of the modeling task. For the driving task, for example, each module generally has one state variable corresponding to a desired distance, and another corresponding to a desired heading. For this case, U is set to the 2×2 identity matrix, since the PID controller that is monitoring a distance variable provides a natural control signal for vehicle speed, and the PID controller that monitors an angle variable provides a control signal for the vehicle heading. The exception to this is the module focusing on maintaining a target speed; this module only monitors current speed in the world, so it always provides a zero-output control value for the change-of-heading action variable.

4.1.5 Priority

Modules can be prioritized by increasing their importance relative to one another, to allow modular agents to perform one task (for example, following a leader car) in preference to another (like achieving a target speed). In a traditional MDP scenario, this is modeled by controlling the ratio of reward values between two subsets of world states. In the present model, module priority is manipulated through the ρ parameter: as ρ increases, the module's

relative priority decreases.

This relative priority value is incorporated into the model as a soft bound on the diffusion of uncertainty for each module. The specifics of this integration of uncertainty and priority are described in more detail next, as part of the perceptual arbitration process.

4.2 Simulation

In simulation, an agent is placed in a two–dimensional virtual driving world. The world contains a single road with multiple lanes. Several non-agent cars are placed on the lanes at random locations, and one of the non-agent cars is designated as the leader car.

The basic simulation loop updates the state of the world at a fixed frequency f_s (set to 60Hz to match experimental conditions from (Sullivan et al., 2012)) according to an elementary physics simulation. At each time step, each car in the world moves ahead proportionally to its speed, in a direction given by its heading. For the non-agent cars, the simulator constrains these speed and heading values so that the cars always follow the lanes in the world.

After moving all vehicles in the world, the simulation additionally requests a control update from the learning agent, which changes the heading and speed of the agent before the next frame begins. Every time the simulator requests a control update for the learning agent, the modules are also updated by displacing their uncertainty particles according to each module's

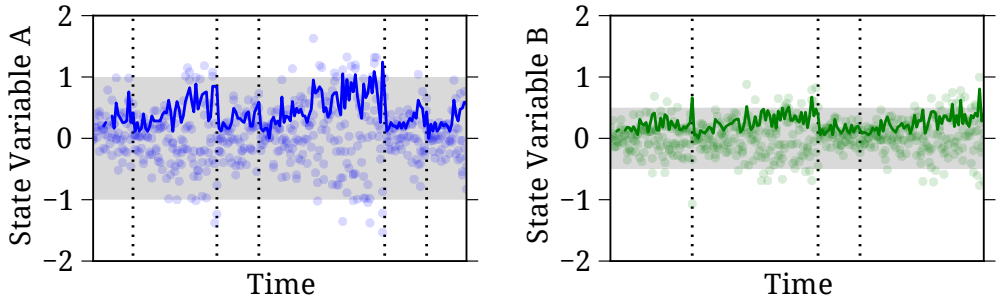


Figure 4.2: Example random walks for uncertainty particles in the two modules shown in Figure 1. The individual particles are shown as small dots in each plot; their RMS uncertainty value $\sigma(t)$ is shown with a solid blue (left) and green (right) line. Vertical dotted lines indicate time steps when each module received an update; these updates reduce the magnitudes of the uncertainty particles towards 0. The uncertainty threshold ρ for each module is indicated by the shaded gray region in the center of the plot; in this example, the module tracking state variable B (right) has higher priority (lower uncertainty threshold ρ) than the module tracking variable A. However, because the module on the left has a higher noise parameter ϵ, it receives more updates than the module on the right in the same duration of time.

noise parameter ϵ.

4.2.1 Perceptual Arbitration

If the simulation only performed the steps above, the agent's performance would become increasingly erratic over time, because the uncertainty particles would drift further away from zero. The resulting erroneous state value estimates would produce poor PID controller outputs, and the resulting actions chosen by the agent would further compound the uncertainty in the state estimates. In a human driver, this behavior would be analogous to taking one look at the world when getting into the car, and then driving with a

blindfold thereafter.

Clearly this is not what humans tend to do when driving. Instead, people continually and regularly reposition their gaze toward objects in the environment as the driving task progresses. The final step in our model is to incorpate a scheduler to arbitrate between task modules, such that updated sensory information is delivered to the PID controllers dependent on task uncertainty and priority. Like Sprague and Ballard (2003), we hypothesize that this repositioning serves to reduce uncertainty about the state of relevant variables in the world—distance to a leader car, current speed, etc. To capture this behavior, the simulator periodically selects a module for receiving updated state information through a perceptual arbitration mechanism. This selection process happens at a constant frequency f_p (set to 3Hz for the results reported here to approximate the frequency of human gaze behavior).

The perceptual arbitration process incorporates priority and uncertainty in the following way. We first define, for each module $M^{(i)}$, a weighted uncertainty at time t that incorporates both the RMS uncertainty and the scalar priority of the module:

$$\zeta^{(i)}(t) = \sigma^{(i)}(t) - \rho^{(i)}. \tag{4.5}$$

We also define a global variable $\phi(t)$ to represent the index of the module that gets updated at time t. Then the soft barrier model defines the probability that module $M^{(i)}$ is selected for update at time t as a Boltzmann distribution

over each of the priority-weighted module uncertainties:

$$p(\phi(t) = i|\zeta^{(1)}(t), \ldots, \zeta^{(N)}(t)) = \frac{\exp\left(\zeta^{(i)}(t)\right)}{\sum_{j=1}^{N} \exp\left(\zeta^{(j)}(t)\right)} \tag{4.6}$$

Intuitively, if the uncertainty in $M^{(i)}$ is currently above the threshold for that module—that is, if $\$\sigma^{(\mathrm{i})}(\mathrm{t}) > \rho^{(\mathrm{i})}\$$—then $M^{(i)}$ is much more likely to be selected for update than another module, especially if none of the other modules have uncertainties exceeding their thresholds. However, the softmax selection process allows for nondeterminism: even if $\zeta^{(i)}(t) > \zeta^{(j)}(t)$ for $j \neq i$, there is some nonzero probability that i will not be selected for update at time t. Finally, because module updates are always selected at frequency f_p by sampling from the above distribution at the appropriate time, a module might be selected for update even if none of the agent's task modules have exceeded their uncertainty boundary (i.e., if $\zeta^{(i)} < 0$ for all i).

Although inspired by diffusion models of decision making, this model contrasts somewhat with traditional models. Many diffusion models with "hard" bounds were developed for forced-choice, two-alternative tasks (e.g., Carpenter and Williams, 1995); our model, in comparison, is designed to incorporate a wider variety of tasks. The "soft" barrier, driven at a fixed frequency, can incorporate more than two choices into the model simultaneously, while accounting for biologically realistic delays in planning and executing saccades.

As described above, and illustrated in Figures 4.1 and 4.2, when a module is selected for update, it is provided with the true state of each world state variable in $\mathcal{S}^{(i)}$, and each of its uncertainty particles β_j is reduced towards

zero for every simulation frame until another module is selected for update.

4.3 Simulation Results

We implemented the model described above[2] and ran several simulations to assess its qualitative behavior. Our simulated driving environment was identical in layout to the virtual environment used by Sullivan et al. (2012), so that we could directly compare our results to human performance. Our implementation consisted of three modules: a "speed" module $M^{(s)}$ that attempted to drive at a particular target speed; a "follow" module $M^{(f)}$ that attempted to follow a lead car, and a "lane" module $M^{(l)}$ that attempted to steer so as to follow the nearest lane on the road. All cars in the simulation drove in a simulated 2–dimensional world, described above. Each time gaze was allocated to a new module, we recorded the module that received the gaze, as well as several behavioral measurements (e.g., distance to leader car, current speed, etc.) to verify that the agent was driving appropriately.

4.3.1 Categorizing Looks

The gaze selection process in our model is Markovian, meaning that each selected module is independent of the previously-selected modules; more formally, $p(\phi(t)|\phi(t-n), \cdot) = p(\phi(t)|\cdot)$ for all $n > 0$. Thus, it is possible that multiple consecutive module updates are directed at the same module, or $\phi(t) = \phi(t-n)$. Similar refixation behavior exists in human gaze during com-

[2] `http://github.com/lmjohns3/driving-simulator`

plex tasks; presumably observers use the visual information across multiple fixations for a continuous control signal for a single task. To make analysis simpler and more consistent between simulation and human results, we grouped multiple consecutive updates for a given module into a single "look." For instance, in the example shown in Figures 4.1 and 4.2, updates are provided first to module A, then B, then A twice, then B twice. In this example, each module receives two "looks," with the second look for each module being twice as long as the first.

4.4 Comparison with Human Results

Sullivan et al. (2012) instructed subjects driving in a virtual environment to follow a leader car and maintain a certain speed, but the priority of which of the two tasks was most important was varied so that one was high and the other was low. Additionally, subjects drove in some conditions where noise was added to the speed of the car, with the intent of disrupting the maintenance of a constant speed. These manipulations resulted in four conditions where either following a leader or maintaining a constant speed was most important, and velocity noise was either present or absent. They found that task priority increased fixation behavior on task-related objects. Additionally, an interaction between priority and uncertainty was found, whereby uncertainty alone did not guarantee increased fixation behavior. Instead, only if a task related object had sufficiently high priority did the addition of uncertainty further increase fixation behavior. Look duration histograms for this

experiment are replicated in the top row of Figure 4.3.

We ran a set of simulations with our model attempting to replicate this behavior using parameters set to mimic the orginal human driver conditions. We used a simple grid search to locate these parameters. Because all of the parameters taken together can present a scaling ambiguity (e.g., if all $\epsilon^{(i)}$ and $\rho^{(i)}$ are multiplied by 2, then the same qualitative behavior will result) we fixed $\rho^{(f)} = 1$ and explored only settings for the other parameters.

Once we identified the parameter settings corresponding to the experimental conditions, we evaluated our model by running it in each of these conditions 10 times, with each simulation run for approximately 4000 steps. The sequence of module updates for each simulation run was stored and labeled as looks as described above, then normalized to form a probability distribution. These results were compared the distributions of look durations from the human data. The model was able to capture several important aspects of the human data, including a sensitivity to both noise and priority, but also a gating effect whereby noise in low-priority tasks had a smaller effect than noise in high-priority tasks. Our results, shown under the human data in Figure 4.3, are qualitatively similar to the human performance in a virtual driving environment.

In addition to our scheduling model, a baseline fixation scheduler was run in the simulation. This scheduler incorporated only the priority of each task in selecting modules for update, but uncertainty was not incorporated. The results from this baseline scheduler are shown in the bottom row of Figure

4.3. The probability distributions from our scheduler and the baseline compared against the human data via the Kullback-Leibler (KL) divergence. Over all the conditions, our model had an average KL divergence of 2.20, versus 4.43 for the baseline scheduler (lower numbers are better).

4.5 Discussion

This chapter described a modular, "soft" barrier approach for modeling eye movements in human drivers. The model includes explicit measurements of an agent's estimates of external world state, and uses a random walk to model the uncertainty in these estimates over time. Uncertainty, modulated by the priority of a task, is then used to arbitrate gaze allocations among competing modules. Our priority-plus-uncertainty model provides a better fit of a set of human driving data than a priority-only baseline fixation scheduling model. We are currently working on comparing this model to predictions from a standard salience model (Itti and Koch, 2001), a central bias model (Tatler and Vincent, 2009) and the original scheduling model that inspired our work (Sprague and Ballard, 2003). In addition, the softmax approach to selecting modules for update permits a clean inversion of the model; that is, given human eye fixation behavior, the model can be inverted to provide the most likely set of parameter settings to explain those data.

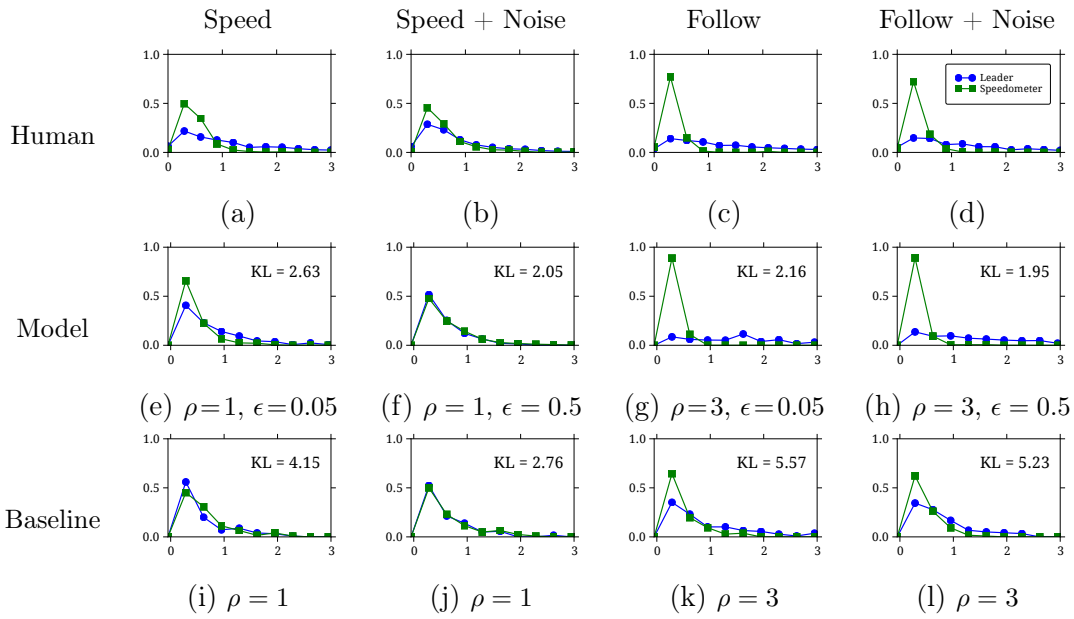


Figure 4.3: Distribution of look durations for human subjects (Sullivan et al., 2012, a-d; from), model predictions (e-h), and baseline predictions (i-l). In all plots, look duration (in seconds) is shown along the abscissa, with the proportion of looks indicated on the ordinate. Looks to the speedometer are plotted with green squares; looks to the leader car are plotted with blue circles. (a, b) In conditions where driving at a target speed was emphasized, human looks at the speedometer were approximately matched in duration to looks at the leader car. (c, d) In conditions where following a leader car was emphasized, human looks at the speedometer were brief. Noise added to the car's speed (b, d) affected human looks in the speed task more than looks in the following task. Similar results hold for our model (e-h), but not for a baseline model that incorporates task priority but ignores the effects of uncertainty (i-l).

Chapter 5

Efficient Coding for Walking Movements[1]

This chapter presents 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. Because it is computationally straightforward, the model allows us to compare and evaluate several possible approaches to this coding and regression task. It shows that, for the class of techniques captured by a particular encoding and decoding model, sparsity is useful for representing and manipulating pose data, but only inasmuch as sparse codes tend to encourage overcomplete codebooks. Even though the decoding error associated with sparse codes can be larger than the corresponding decoding error for dense codes, the information captured by each coefficient in a sparse code is larger than for dense codes. The work described here then combines the coding, regression, and decoding stages of the model into a single neural network–style loss function and shows that substantially lower prediction errors can be achieved using an integrated system than by learning codes in isolation.

[1]The work described in this chapter was published by Johnson et al. (2013a), Johnson et al. (2013b), and Johnson and Ballard (2014b). Human movement data were collected in our laboratory by J. Cooper, and I developed and tested the computational models.

5.1 Problem Setting

Humans are extremely competent at mapping movement information from one modality into another. For example, to mimic a conspecific's actions, a person needs to, at some level, transform visual information into dynamic muscle controls. However, studying human movements is difficult: the parameters describing movements are extremely high-dimensional, and, in addition, most of the quantities that are relevant for describing the control of movement are invisible to an outside observer. Even though we cannot measure all of the joint torques or angles during a complex, multi-joint human movement, we can use motion capture (Figure 5.2) to measure the external aspects of movement, and convert these observations into angles and forces that would have been required for a simplified model of the human skeleton to effect the same movements (Cooper and Ballard, 2012). Such computed angles and forces—while still a coarse proxy for the information that might be used by the central nervous system in, say, mimicking an action—serve as the data for this paper.

For a regression task, 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 likely to be inefficient for at least three 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 regarding human movements, then each modality of the raw data will have statistical redundan-

cies that would need to be captured by the regression process, in addition to mapping between domains. Third, linear transformations of movement data are unlikely to capture important variations in these information modalities. 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 and Salakhutdinov, 2012), possibly using some nonlinear transform. The questions addressed by this paper are, which types of codes are most efficient for processing information about movement, and which loss functions generate the best codes for computing regression between modalities?

5.2 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

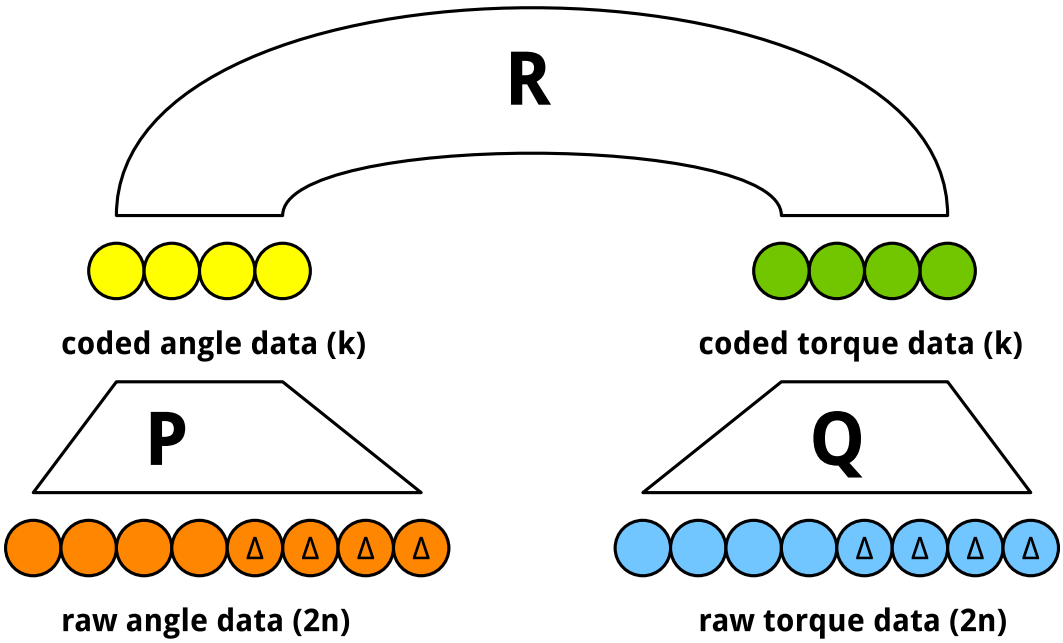


Figure 5.1: Information processing architecture for multi-modal coding and regression. Information from a frame of one modality of pose data, such as angles (orange), is to be mapped onto information from another modality, such as torques (blue). This mapping is accomplished by coding a frame of angle data, augmented with its derivative Δ, using parameters $\mathbf{P}$; the coded angles (yellow) are then mapped to coded torques (green) using a regression with parameters $\mathbf{R}$; finally, the coded torques are then decoded into a frame of raw torque values (augmented with its derivative Δ).

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

$$\mathbf{A} = \begin{bmatrix} \mathbf{B} \\ \Delta\mathbf{B} \end{bmatrix} \tag{5.1}$$

$$\mathbf{V} = \begin{bmatrix} \mathbf{U} \\ \Delta\mathbf{U} \end{bmatrix}, \tag{5.2}$$

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 (Figure 5.1), 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 et al.,

2010). Regression parameters $\mathbf{R}$ can then be learned using the encoded data from each modality. Given our parametric framework, the coding approaches considered here all assume finite codebooks $\mathbf{D} \in \mathbb{R}^{2n \times k}$ whose columns $\mathbf{d}_i$ each represent a basis vector that is used in some way to encode data. Although this paper does not focus specifically on learning the codebook, it does describe a few approaches to codebook learning below.

Separating the model into isolated coding and regression stages brings three advantages to the problem at hand, but also comes at a cost. The first advantage is that this approach 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. The second advantage of separate modules for coding in each modality, and regression between codes, is that this separation 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.

On the other hand, separating the model into distinct coding, regression, and decoding phases comes at a fairly high cost. Most importantly, this

separation isolates sources of error in the model from one another. This isolation prevents information about, for example, decoding errors from flowing into the regression or encoding parameters. Isolating these sources of information from one another could prevent parameters from adapting together to find a solution with low overall error.

5.2.1 Coding Algorithms

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

$$\mathcal{L}(D) = \|g(\mathbf{D}, \mathbf{z}) - \mathbf{x}\|_2^2 + \lambda R(\mathbf{z}) \tag{5.3}$$

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. We evaluate several approaches to coding, each of which is described in Chapter 2.

5.2.2 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 and 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 $\mathbf{z}_\beta$ as optimizing the cost function

$$\mathcal{L}(R) = \|\mathbf{R}\mathbf{z}_\alpha - \mathbf{z}_\beta\|_2^2 + \lambda\|\mathbf{R}\|_F^2 \tag{5.4}$$

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.

5.3 Unified Loss Function

The framework presented above consists of a set of parameters for each of the three tasks in mapping from one modality of data to another: $\mathbf{P}$ and $\mathbf{Q}$ contain the parameters for encoding the source and target data vectors, respectively, and $\mathbf{R}$ contains the parameters for mapping from the source code to the target code.

In the architecture described so far, these parameters are trained separately and only combined at test time to evaluate their efficacy on the entire data mapping pipeline. While advantageous for the reasons discussed above—separation of tasks for straightforward analysis, representation of data using hypothesized underlying manifolds—this isolated coding strategy presents a fundamental learning difficulty, namely, that errors introduced into any of the stages of the model cannot be accounted for by other stages of the model. This is particularly problematic when a specific task, such as the multimodal

regression task considered in this paper, might benefit from a slightly different coded representation than the manifold (or other sparse code) representation. Consider, for example, a code for joint angles that maps each joint–angle vector onto a coded representation that is optimal for reconstruction.[2] This coding approach, while powerful for many tasks, probably captures information about the "principal axes" of the manifold on which the data lies. However, these principal directions might not be relevant or important for a specific task like computing a regression to another coded information modality.

We can remedy this separation of errors by taking advantage of the large body of knowledge from the neural networks community. We define a unified loss function that retains the general architecture presented earlier—and in particular has the same parameterization—but allows errors in each stage to flow across parameter boundaries:

$$\mathcal{L} = \|\mathbf{Q}g\left(\mathbf{R}g\left(\mathbf{P}\mathbf{x}\right)\right) - \mathbf{y}\|_2^2$$

where $\mathbf{x}$ and $\mathbf{y}$ are input and target vectors, respectively, and $g(\cdot)$ is a nonlinear activation function that prevents the model from collapsing to a single linear operation. This unified loss function can be used to train all parameters in the network simultaneously, which permits errors introduced by one parameter to be compensated for by other parameters.

[2] As shown above, reconstruction is often the criterion for obtaining coefficients under such coding regimes.

5.3.1 Nonlinearities

The only issue introduced by the unified loss is which nonlinearity to choose. In this paper we evaluate two methods, one traditional and one recently popularized by work in the feature learning community.

5.3.1.1 Logistic

Traditionally, g is set to be a sigmoid function like the logistic $g(z) = 1/(1+e^{-z})$, because such functions are bounded and often have an easily computed, continuous derivative. However, logistic activations suffer from the problem of vanishing gradients (Bengio et al., 1994), in which networks with many layers—even, in practice, more than two layers—erroneously appear to converge during training because first-order gradient computations reveal negligible slope, while in fact the parameter settings might be located on a broad saddle point of the loss. This problem has been addressed recently by Martens (2010) and Martens and Sutskever (2011), who developed a practical, second-order Hessian-free training method for neural network models. This learning method is capable of dramatically improving parameter settings for neural network models that appear to have converged using first-order gradient descent, as described in more detail below.

5.3.1.2 Rectified Linear

A parallel line of work in the deep learning community (Glorot et al., 2011) has recently emphasized the performance advantages of a different kind

of nonlinearity, the relu or rectified linear unit $g(z) = \max(0, z)$. This activation function addresses the vanishing gradient problem by having a constant derivative for $z \neq 0$. As a side benefit, networks of rectified linear units tend to be naturally sparse, because $g(z) = 0$ (a true zero, not just a small nonzero value) for $z \leq 0$. In practice, rectified linear networks can be trained using simple first-order learning techniques, and they tend to perform equivalently to large networks of logistic sigmoid units that require significantly more effort to train.

Using a rectified activation function, the unified loss becomes surprisingly simple:

$$\mathcal{L} = \|\mathbf{Q}\left[\mathbf{R}\left[\mathbf{P}\mathbf{x}\right]_+\right]_+ - \mathbf{y}\|_2^2$$

where $[\cdot]_+$ denotes the rectification nonlinearity. For some input $\mathbf{x}$, some of the units in the first hidden layer of the network will have negative pre-activations and will be set to zero by the nonlinearity, while the remainder will simply pass their activations along to the next layer of the network. This process repeats for all hidden layers, effectively using the nonlinearity as a switch to select a subset of network nodes to encode, regress, and decode each input. Using $\mathbf{z}_+$ to denote only the nonzero elements of $\mathbf{z}$, the network output for $\mathbf{x}$ becomes a linear operation $\hat{\mathbf{y}} = \mathbf{Q}_+\mathbf{R}_+\mathbf{P}_+\mathbf{x}$. This switching behavior retains the speed and simplicity advantages of linear codes, while simultaneously working to combine exponentially many such linear codes into one model (Nair and Hinton, 2010).

5.3.2 Training

As mentioned above, parameters in neural networks are commonly optimized using one of two learning algorithms. For this work, all network parameters were first adjusted using first-order stochastic gradient descent (SGD), which computes parameter updates as the mean of the gradient over a mini-batch of r training examples

$$\Delta\theta_b = \frac{1}{r}\sum_{i=1}^{r}\frac{\partial\mathcal{L}}{\partial\theta}\left(\mathbf{x}^{(br+i)}\right)$$

and then updates parameters using a small learning rate α such that $\theta_{b+1} = \theta_b + \alpha\Delta\theta_b$. Usually, α is reduced over time; for our experiments, we started with $\alpha = 0.1$ and reduced this value by 1% after every mini-batch.

For rectified linear networks, training stopped when SGD converged to a region of zero gradient, because these networks' second derivatives are zero and do not benefit from a second-order training strategy. For the logistic networks, however, second-order information can be used to compute further parameter updates once an apparent minimum has been reached. After convergence of first-order SGD, then, we further updated parameters in these networks using a Hessian-free algorithm (Martens, 2010; Martens and Sutskever, 2011). Briefly, this algorithm uses conjugate gradient (CG) to compute a parameter update

$$\mathbf{u}_b = \arg\min_{\mathbf{v}} \mathbf{H}\mathbf{v} + \lambda\mathbf{v} - \Delta\theta_b$$

where $\mathbf{H}\mathbf{v}$ is an implicit computation of the full Hessian in a specific direction $\mathbf{v}$. Because $\mathbf{u}_b$ is a solution to the second-order approximation of $\mathcal{L}$ at θ_b,

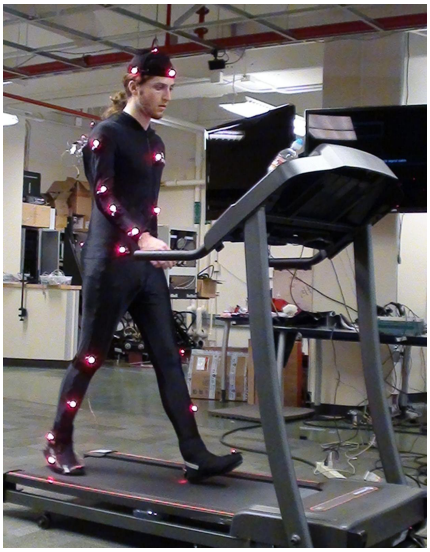


Figure 5.2: The motion capture environment consists of a full-body motion capture suit (black, with red LEDs), and a treadmill centered in the motion capture space.

parameters are then updated without a learning rate, such that $\theta_{b+1} = \theta_b + \mathbf{u}_b$. This training regimen is able to reduce error significantly in "converged" sigmoid networks, often by more than 50% (Figure 5.5).

5.4 Experiments

We used a 16–camera Phasespace [3] motion capture system in conjunction with a standard treadmill (Figure 5.2) to capture human movements. 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.

[3] `http://phasespace.com/impulse_motion_capture.html`

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.

5.4.1 Preprocessing

Here, 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

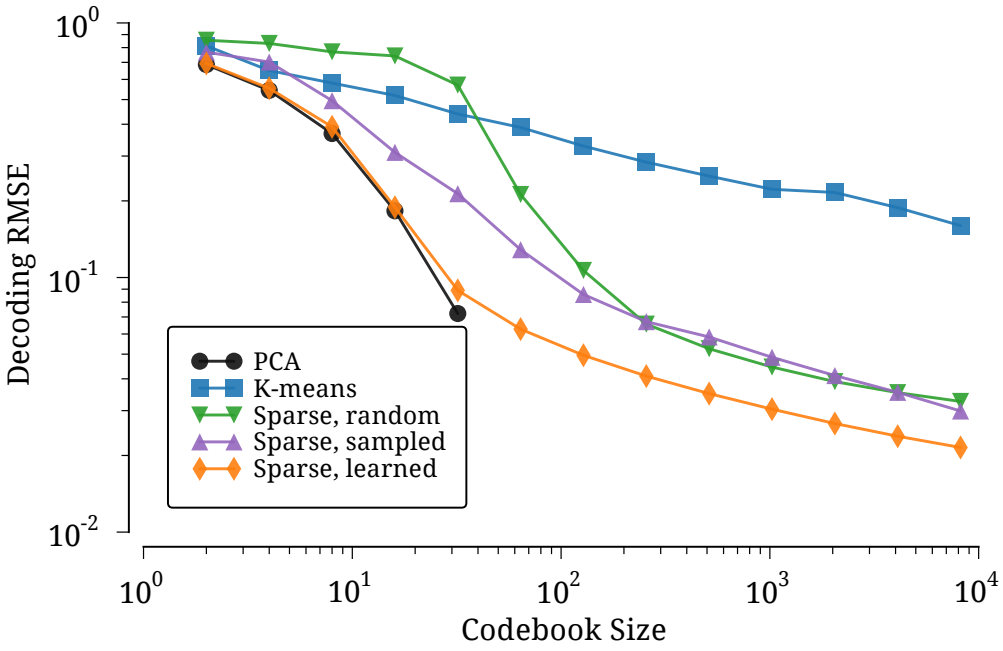


Figure 5.3: Mean RMS decoding error for joint torques, measured with respect to the size of the codebook. Larger codebooks result in codes that capture more of the variance in the data, even when the codebook is created using IID standard normal samples. A log scale has been used on both axes to reveal trends more clearly.

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.[4]

[4]The first frame was dropped from each dataset to match the number of frames of data

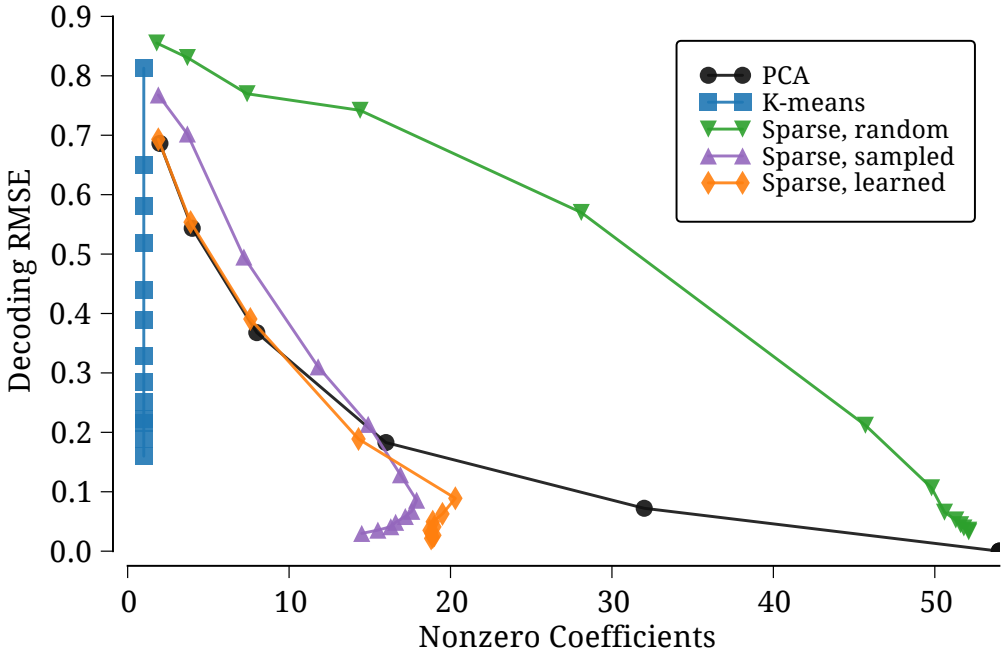


Figure 5.4: Mean RMS decoding error for joint torques under isolated coding strategies, measured per nonzero coefficient in the encoding. Sparse codes like lasso regression were more effective, per coefficient, than dense codes like PCA, but only when the codebook was tuned to the dataset. The markers on each data series in this plot correspond to codebook sizes, just as in Figure 5.3.

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

with the number of frames of derivative.

10\each segment were used for validation, 10\remainder were available for training.

5.4.2 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:

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

Figure 5.3 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 5.4: 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.

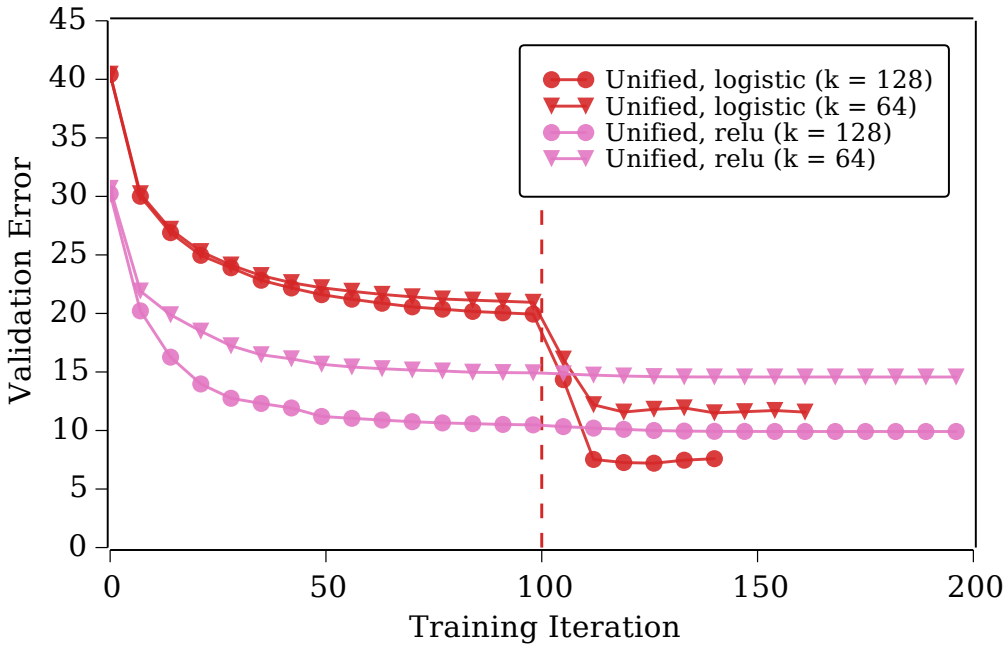


Figure 5.5: Learning curves for two example neural networks using the unified loss. All networks were trained first using first-order stochastic gradient descent, while the logistic networks were additionally trained with a Hessian-free second-order method after iteration 100.

5.4.3 Unified Loss Performance

The two nonlinearities that we evaluated in the unified loss showed different behavior during both training and testing. The rectified linear activation was much faster to train because it achieved low error using a simple first-order gradient method, while the logistic activation function required enormous amounts of time to compute the second-order Hessian-free learning method, which was started after the first-order method appeared to plateau. The logistic network training process required so much time to train, in fact,

that it did not complete for $k = 2048$.

Sample learning curves for these networks are shown in Figure 5.5; these curves reveal two trends that occurred at nearly all network sizes during learning. First, the first-order stochastic gradient training method appears to plateau for logistic networks at a higher error rate than for rectified linear networks of the same topology. Second, applying a second-order Hessian-free learning method to these trained networks resulted in large performance improvements for logistic networks, but no improvement for rectified linear ones.

5.4.4 Feature Correlations

Because the processing architecture described here shares a common parametric formulation, we can analyze the parameter groups $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{R}$ across learning algorithms. For instance, across models we can treat $\mathbf{P}$ as being responsible for "encoding" the input angle vector, and examine the elements of this matrix in that light, regardless of whether the matrix in question was learned using the unified or the isolated losses. To get some idea of how the features in each algorithm self-organize, we mapped each of our learned angle codebooks onto the first two principal components of the angle data (Figure 5.6). Codebooks differed noticeably across algorithms; for instance, random features tended not to be strongly correlated with any principal component of the data, since random vectors in high-dimensional spaces tend to be orthogonal to each other. In comparison, a learned dictionary like the one

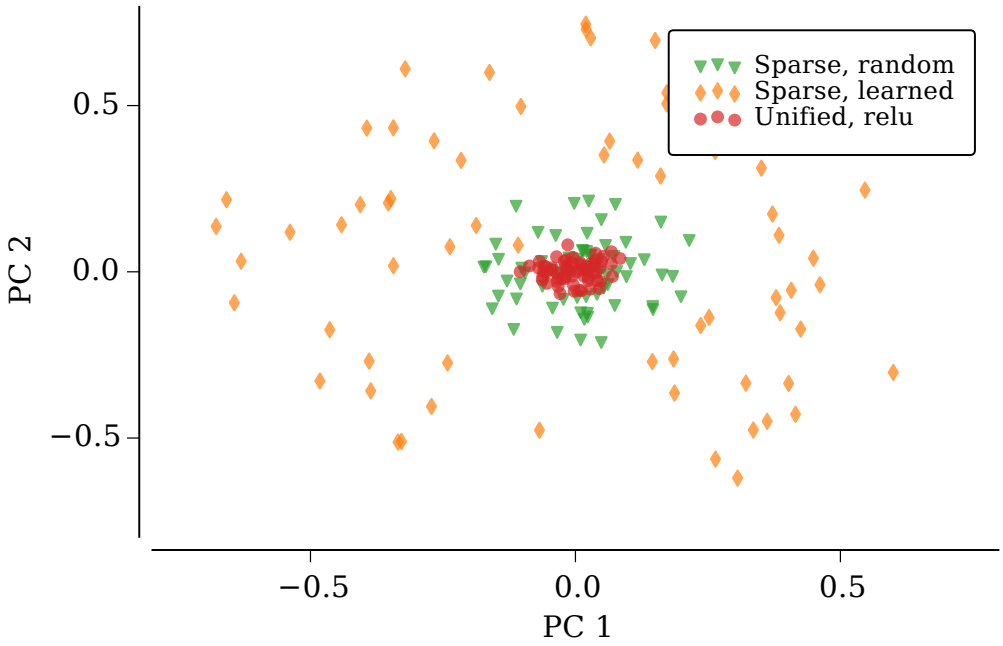


Figure 5.6: Learned angle codebooks, projected onto the first two principle components of the angle data. Values near the origin indicate codebook vectors that do not correlate strongly with either of the first two principal components of the angle data.

obtained from sparse coding tended to model more strongly the space defined by the principal components. Finally, the differences in learned features between the two neural network models was striking: networks of rectified linear units have the same correlation with the principal components as a random set of vectors, while networks with logistic activations tended to have features that were nearly always orthogonal to the principal components.

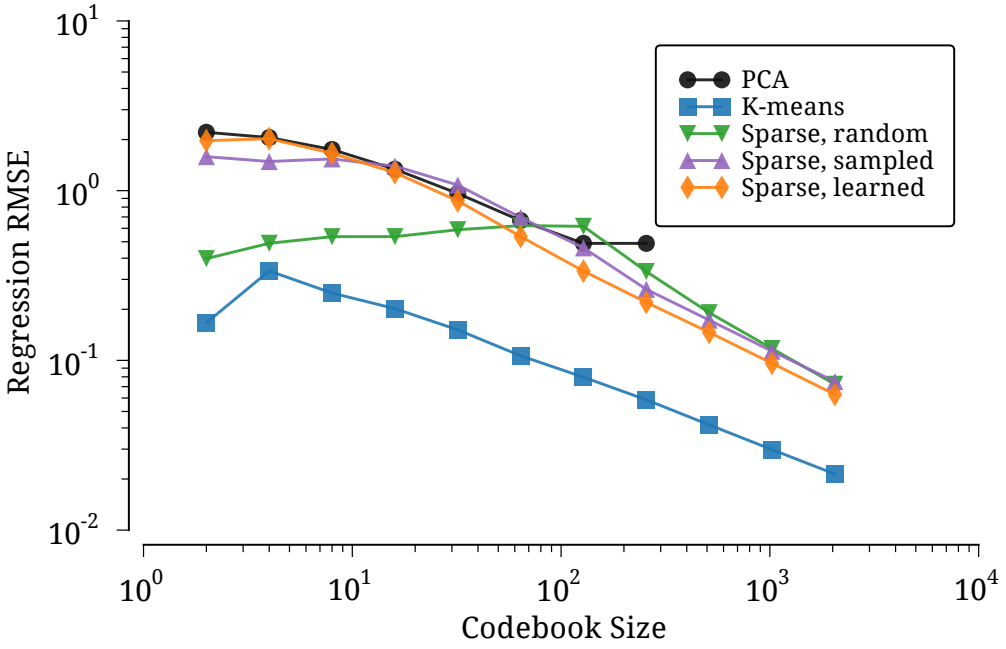


Figure 5.7: RMS regression error for isolated coding strategies, measured with respect to encoded torque values. A log scale has been used on both axes to reveal patterns more clearly.

5.4.5 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. From one perspective, this task could be seen as a coarse approximation for a control task: given a target kinematic pose, what are the torques that would be associated with that pose?

Because the analysis framework proposed in this paper breaks this task

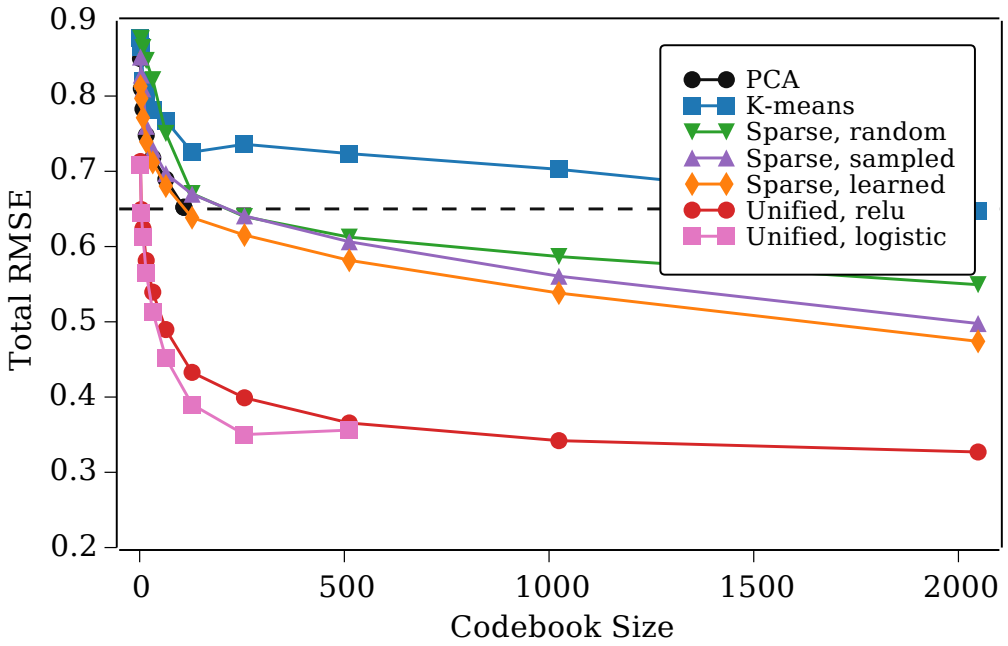


Figure 5.8: Mean overall RMS error, measured with respect to raw torque values for each frame. The dashed black line shows the baseline RMS error: a ridge regression model is computed directly from raw angles to raw torques, and then that model is used to compute RMS error when mapping test data directly from angles to torques. Logistic neural networks for $k > 1000$ consistently failed to converge during training.

into three separate stages—encoding, regression, and decoding—we could analyze, for the isolated coding strategies, the regression component of the task separately from the other components. In general, we found that RMS error for the regression task alone (Figure 5.7) 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 the regression task between two codes, since K-means yielded the lowest regression errors, while

PCA yielded relatively large errors. Intuitively, sparse codes should be easier to perform regression between than dense codes, since there are more zeros in sparse source and target vectors.

Bringing together all stages of the framework, we compared the overall torque regression error for the entire coding and regression pipeline, as measured by comparing the outputs from our processing model with the true torques measured during the experiment (Figure 5.8). 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 below baseline for undercomplete codebooks, and at baseline for complete codebooks, which is unsurprising since PCA simply rescales the original data. However, some of the sparse coding approaches were able to 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 isolated coding approaches examined here.

Finally, we compared the performance of models trained using a unified loss with the isolated coding strategies. The unified loss function dramatically outperformed even the best sparse coding strategy, suggesting that coding alone (or, more specifically, coding based on optimizing a reconstruction loss) is insufficient or inappropriate for the specific regression task that we used in this paper. This is intuitively clear from the top group of results in Figure 5.8: while large codebooks and sparse codes are able to outperform the baseline approach, the absolute RMS error for isolated approaches remains relatively

high. In contrast, a unified loss that can correct for errors across separate coding, regression, and decoding stages is better able to model the task at hand. Interestingly, one exception to this claim seems to be the performance of large networks of sigmoidal units—such networks on our task were not only extremely slow to train, they failed to converge to error rates below baseline, suggesting that the loss landscape for these models might have many poor local optima that are difficult to avoid even with a second-order learning approach.

5.5 Discussion

This paper presented a 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 computing regressions from one modality to another. 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. Combining all data processing parameters into a single, unified loss function significantly increases the performance of the processing pipeline.

The results shown here complement much of the recent work from the feature learning community (e.g., Hinton and Salakhutdinov, 2006), in which features that are learned during a purely unsupervised pre-training phase tend

to be useful for many tasks, but are often fine-tuned using a second, supervised learning step that is specific to the demands of the task at hand. If some neurons in the brain learn receptive fields in an unsupervised manner, then the results shown here suggest that the unification of many of these neurons into one cognitive whole could outperform a pool of isolated, low-level sensory or motor neurons for a regression task.

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.

Part III

Experimental Work

Chapter 6

Marionette Movement Model

Voluntary movements are the end product of a large number of non-linear dynamical systems embedded within an animal. Many of these systems are understood incompletely, even in isolation. Additionally, as discussed in Chapter 3, models of motor control tend to be both good and bad. As a result, detailed scientific understanding of animal movements is typically limited to small subsets of the overall process. For example, existing research work has focused on the contractile force of an isolated piece of muscle tissue (e.g., Gordon et al., 1966), the activations of a small set of spinal neurons (e.g., Frigon, 2012), activity in the motor cortex during a limb movement (e.g., Churchland et al., 2012), and so forth.

Given that detailed knowledge of the entire movement process is unavailable, efforts to model movement computationally must make simplifications and assumptions. Unlike in Chapter 3, this chapter organizes computational efforts to model movement along a continuum of complexity. At one extreme, detailed biophysical models like OpenSIM (Delp et al., 2007) try to incorporate as many details about the movement system as possible. These models include 3–dimensional bone geometry, sophisticated inertial models of

limbs, muscles with nonlinear activation dynamics and multiple attachment points, and elastic tendons and ligaments. Such detailed models tend to require significant computation time to determine the dynamics of a movement. The complexity, and particularly the nonlinearity, of these models prevent efficient inversion; determining neural activations for a given movement can require weeks or months of computation time (Neptune et al., 2009).

At the other extreme, simplified models attempt to represent a movement using one or two ideal spring or pendulum oscillators (Raibert, 1986; Raibert and Hodgins, 1991; Collins et al., 2005; Matthis and Fajen, 2013). These models have the advantage of being easy to compute, easy to understand, and easy to invert. However, their simplicity prevents them from being applied to many types of movement at once. In fact, one simple model is typically constructed manually for each type of movement being modeled, such as walking or hopping. The passive walker described by Collins et al. (2005) was meticulously hand-tuned for the exact slope of its walking surface.

This chapter describes a novel "Marionette Movement Model" (MMM) for understanding movement that falls between the two extrema. Like the simplified models, the MMM uses linear dynamics, which can be inverted in real-time. Like the complex models, the MMM retains the representational power to capture a wide class of human movements. The model is properly viewed as a computational reformulation of the TCT (Feldman and Levin, 2009), so the presentation will begin with a re-examination of what movement looks like through the lens of the TCT.

6.1 Muscle Contractions and Threshold Control

As discussed at the beginning of this chapter, voluntary movements in animals are made thanks to a cascade of numerous systems. First, neurons in the motor cortex fire in response to a wide array of inputs from neurons in the cortex and midbrain. Next, these signals travel to the cerebellum and down the spinal cord. Neurons in the spinal cord are thought to embody a dynamical system that starts oscillating (Ijspeert, 2008; Frigon, 2012), which ultimately sends neural signals to motoneurons embedded in the muscle tissue. In response to stimulation from the motoneurons, spring-like proteins in the muscles then contract, pulling on the tendons that attach the muscle to one or more bones. This pulling action applies a dynamically changing pattern of forces to the bones, and the bones then move to change the posture of the skeleton.

Purely from the point of view of physics, however, a movement happens when a skeletal muscle contracts, forcing two bones to rotate around a joint—the neural processes that underlie the contraction are not required for the system to operate. A brain does, however, produce the observed patterns of muscle contractions that cause animals to move. What does this process look like from the brain's perspective? In vertebrates, the brain is "trapped" inside the skull, perched at one end of the spinal cord, inundated by sensory and proprioceptive information about the world. The brain must somehow compute the desired movement it wishes the body to make, and then generate nerve impulses so that the causal chain of events progresses downward and

outward, through the spinal cord and into the motoneurons, resulting in some set of muscle contractions that will achieve the movement goal.

As described above, the details of this process remain somewhat poorly understood. The TCT attempts to explain the patterns of observed muscle activity, then, by postulating an unobserved *threshold length* λ_i for each voluntary muscle in the skeleton. Muscles whose actual length x_i is smaller than λ_i will not be recruited to exert forces on the skeleton, while those for which $x_i > \lambda_i$ will be recruited. It is important to note that, in the view of the TCT, the x_i are the result of the body's interaction with the environment, so that the actual amount of force produced by the muscles depends critically on the organism as well as its environment.

In this view, the brain produces movements by changing the λ_i over time, such that the pattern of muscle lengths interact with the threshold lengths to produce the necessary forces to effect a movement. The TCT explicitly views the forces produced by the organism's muscles as "emergent" (Feldman and Levin, 2009) and makes no attempt to explain them mathematically.

From a cognitive perspective, the TCT presents an appealing view of motor control: the central nervous system is allowed to think in terms of threshold postures, and some lower-level mechanism is responsible for converting the target postures into forces. The TCT begins to fall apart, however, when looking at continuous, full-body movements. For example, the TCT does not make any claims about the evolution of the threshold postures that the

central nervous system is hypothesized to generate! And while the threshold postures are simple enough to measure for single-muscle, single-joint systems, it is difficult to measure the hypothesized thresholds in full-body, continuous movements; the thresholds are truly latent in the neurobiomechanical system. The MMM provides a way to address this lack of measurability, but first, full-body movements need to be made somewhat more concrete by measuring them.

6.2 Measuring Movements

Movements are commonly studied using a motion-capture ("mocap") tool. In studies using a mocap tool, a participant enters a laboratory and has anywhere from 10 to 100 "markers" mounted onto their body. Markers are attached in fixed, often anatomically prominent locations such as the sternum (chest), iliac crest (hips), or patella (kneecap). Markers are normally affixed to the subject using either a piece of elastic or sometimes by gluing or taping the markers directly onto the skin.

The markers for most mocap systems are either reflective—i.e., they reflect light from other sources—or active—i.e., they generate light themselves. Either way, once the markers are illuminated, a system of cameras mounted at fixed positions in the laboratory observes the displacements of these markers over time. Suitably instrumented, the subject makes movements in a volume of laboratory space that can vary from the size of a tabletop—e.g., for studies of hand or arm movement—to the size of a large room—e.g., for walking or other

full-body movement. As the subject moves their limbs, the 3–dimensional locations of the attached markers are recorded by the cameras, resulting in a file containing movement data at the end of the trial or experiment. Movement, then, as recorded by a mocap tool, results in a raw array $\mathbf{P} \in \mathbb{R}^{T \times 3M}$ of motion-capture marker positions over time, containing T frames of data and M marker locations.

Many mocap systems also record a "confidence" value for each observed marker location; if insufficiently many cameras can see a marker at a given moment, then the system can indicate that it has low confidence in the measured location of the marker. If the confidence drops below some threshold, then the motion-capture system is said to have "dropped" that marker on that frame. Typically dropped frames require post-processing to reestablish a continuous stream of recorded data.

6.3 Kinematic and Dynamic Models

Once movements have been recorded from a subject, many recording tools map the temporal patterns of marker positions onto angular displacements of joints in a simplified model of the human skeleton. This process typically proceeds in two stages. First, the skeleton model is scaled to fit the recorded marker positions reasonably well; often this is done by having the subject stand in a canonical calibration pose at the start and end of the trial or experiment. Second, the positions of the limb segments at each frame are computed. This process permits easy recovery of an array of angles $\mathbf{A} \in \mathbb{R}^{T \times N}$

from each of the N angular degrees of freedom in the skeleton model (Veldpaus et al., 1988; Challis, 1995; Kirk et al., 2005; Delp et al., 2007).

This conversion process loses information captured during the recording: not only are there typically far fewer degrees of freedom in the model than there were markers during the recording (i.e., typically $N < M$), but the skeleton model is known to be an approximation of the human's skeleton. The skeleton fitting process is entirely kinematic: at each frame of mocap data, the positions of the skeleton bodies are computed that jointly minimize the squared error between recorded marker location and projected marker position on the simulated skeleton.

Once the joint angles are computed in the simulated skeleton, many motion-capture tools discard the recorded marker positions $\mathbf{P}$; the joint angles $\mathbf{A}$ describe the movement of the simulated skeleton in exactly as much detail as the marker data, assuming that the skeleton is a reasonably accurate model of the body that generated the recorded marker displacements.

Once joint angles are computed, many computational models focusing on movement kinematics attempt to work entirely in joint-angle space. For instance, Taylor et al. (2007) modeled different styles of walking data by predicting a frame of joint angles given the preceding frames. Many similar studies treat movement likewise, as a change of kinematic joint angles over time. But movement is a dynamic process; mass and inertia play important roles in the pattern of joint angles that are metabolically efficient for a given skeleton to use. The modeling question then becomes, how does one compute the forces

needed to move the (simulated) body? If the physical model has realistic, nonlinear muscle activations, the only known method is brute force: guess a set of neural activations, apply them to the simulation, and check whether the resulting movement matches some target (e.g., Neptune et al., 2009).

Another approach, proposed by Cooper and Ballard (2012), uses an off-the-shelf physics simulator[1] to drive the skeleton model through the series of observed joint angles, incorporating limb mass in the process. This process proceeds in two phases: inverse kinematics, and then inverse dynamics.

6.3.1 Inverse Kinematics

First, the observed mocap marker locations are translated into the physics simulator as infinite point masses, not subject to gravity or collisions. The simulated skeleton is attached to these point masses using virtual springs. The physics simulator uses the springs to compute the positions of the rigid bodies in the skeleton that collectively minimize the spring forces across the entire skeleton and marker system. This minimization process is extremely fast, because the dynamics are linear, and the physics simulator incorporates a very fast approximate linear solver.

At each frame t, the mocap marker masses are moved to the recorded marker position $\mathbf{p}_t$, and the velocity of each marker mass is set to the observed secant-line approximation of the marker's velocity. The physics simulator is

[1] `http://ode.org`

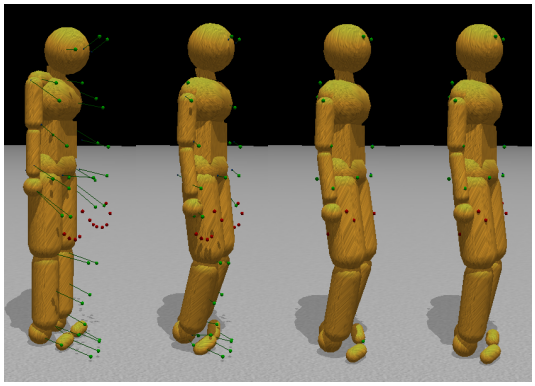


Figure 6.1: Mocap markers, represented as infinite point masses, are attached to the skeleton model using virtual springs.

stepped forward by one frame, and the resulting angles in the skeleton model $\mathbf{a}_t$ are returned.

6.3.2 Inverse Dynamics

Having computed the joint angles $\mathbf{A}$ for the entire mocap recording, the state of the world is reset to the beginning of the recording, i.e., $t = 0$. The springs attaching the marker masses to the skeleton are made several orders of magnitude less stiff. A PID controller is installed at each angular degree of freedom (DoF) in the skeleton.

Then, for each frame t, the PID controller is used to set a target velocity for its DoF, using $\mathbf{a}_t$ as the target angle for that frame. Typically the k_p constant is set such that the proportional term from the PID controller will restore between 90 and 100 percent of the angular error in one frame. The physics simulator again processes all of the frames of data in the recording,

but in this second pass the joint torque required to attain the given change in angle is recorded, resulting in the set of torques $\mathbf{T} \in \mathbb{R}^{T \times N}$ for the entire recording.

Like any approach based purely on mocap recordings, the "Cooper" approach ignores elasticity and any muscle stiffness that does not result in movement—these quantities must be measured using an out-of-band instrument such as an electromyograph (EMG) device. Nonetheless, the model is useful for converting a recording of kinematic movement information $\mathbf{P}$ into force information $\mathbf{T}$.

6.4 Movement as Telekinesis

The "Cooper" model largely assumes that movements are made in a similar way to the muscular system. Instead of linear-pull muscle actuators, the Cooper skeleton uses rotational torque motors, but the movement model remains mostly the same: to generate a movement, a desired change in joint angles is computed, and then force is sent to the actuators, resulting in a posture change. Instead of assuming that the recorded motion data describe a series of joint angles over time, the Marionette Movement Model conceives of movement in a completely different way, but uses the same machinery as the Cooper model.

Consider again the perspective of a brain, "trapped" inside the skull, needing to move the body's posture to attain some goal in the external world. In reality, as described above, the brain accomplishes this task by perform-

ing some nonlinear optimization that ultimately sends neural signals to the muscles, causing forces that change the body's posture. But, as also discussed above, this process is difficult to measure using existing motion-capture tools, since mocap tools only record the observed displacements of the limbs over time. Simplifying assumptions are then required to map these displacements onto a model of the skeleton, to compute the angles of the joints in this skeletal model, and then additionally to use these angle changes to compute torques that move the skeleton.

Instead of requiring these assumptions about the body, the MMM imagines that the brain controls the movement of the body by displacing the infinite-mass mocap markers that are attached to the body, as though these marker locations were the ends of some invisible marionette strings. The displacements of the mocap masses cause forces to be applied to the body due to the springs that attach the masses to the skeleton segments. The resulting pattern of forces then produces a posture change that moves the body through world coordinates.

On the face of it, the MMM is a bit crazy: who would imagine that movement inside the body takes place because invisible springs are pulling the body into some position from the outside? And what mechanism would permit a person to use telekinesis to move these infinite point masses? Consider once more the TCT, however, which does not make any claims about how the central nervous system sets its muscle threshold lengths λ_i over time. The TCT also remains blissfully agnostic about the mechanism that produces forces in

muscles in response to muscles that exceed their threshold length, $x_i > \lambda_i$.

In some sense, the MMM adopts these agnostic positions as well: telekinesis is hardly a physically realistic model of movement! However, the MMM fills in the TCT's agnostic positions with a specific, spring-based mathematical model. In doing so, the MMM moves the threshold concept outside of the body, which makes it easier to test and easier to tie to the recorded movement information—which, once more, is limited to the instantaneous positions of the mocap markers, $\mathbf{P}$. In addition, by hypothesizing actual spring dynamics between the infinite marker point masses and the simulated skeleton, an off-the-shelf physics simulator suffices as the "model" that translates threshold positions into forces inside the body.

So in the Marionette Movement Model, a movement is made because an agent willfully displaces an imaginary point mass in the world. This point mass is attached to the agent's body in a particular location using a spring, so the displacement of the point mass exerts some force on the body, causing its components to move about and exert other forces according to the kinematic makeup of the skeleton. During a recorded movement, the positions of these imaginary point masses are just the same as the positions of the mocap markers being recorded. However, by using spring dynamics and point masses, the MMM permits "threshold" positions that are far outside the kinematic range of motion of the skeleton itself. For example, a point mass attached to the foot might be imagined to move to the bottom of a swimming pool during a diving movement, so that the spring attachment provides some "anchoring" force

that might be difficult to describe otherwise. Even though the body cannot physically satisfy all of the individual "constraints" provided by the positions of the point masses, the body-spring-mass system as a whole can still settle into an optimal configuration that jointly minimizes the forces acting on the system by the environment.

The MMM risks degeneracy in the sense that mocap markers are recorded in world coordinates, and so a brain that knows the world coordinates of its mocap markers could just move these marker locations to a final goal location and let the skeleton "settle" into a stable state. Such a movement strategy would not result in realistic limb trajectories, for instance, in the case of needing to take several steps to get the body from one location to another. To help prevent such degenerate solutions, the central nervous system in the MMM maintains a body-relative estimate of the position of each mocap marker over time, and all control of the point marker masses takes place in this body-relative reference frame. The mapping between body-relative marker position and world-space marker position is not known *a priori*, so some motor learning might be required. Indeed, the MMM permits investigation of several different types of motor learning precisely because of the separation into body-relative and goal-relative (or world) reference frames:

- The Jacobian can be thought of as a state-dependent map between the two reference frames that would need to be learned. This could be thought of as learning an internal model of the dynamics of the plant.

- The desired trajectories of the point masses themselves would need to be tuned for a particular movement goal. This process could be thought of as learning a controller that maps desired states onto actions (changes in point mass location, or perhaps changes in stiffness for each point mass spring).

At its core, the MMM thinks of movements as changes in position, but it proposes using a well-defined (and computationally convenient) mechanism for mapping desired changes in position into forces. I think this model will have many uses in motor modeling, but in this document we will only look at a small example of its application to recorded movements.

Chapter 7

Cube Experiment

Humans regularly make a wide variety of movements to accomplish everyday tasks such as walking, speaking, or picking up an object. However, many of these tasks have not yet been the object of movement modeling approaches that consider goals explicitly; most often, movements are modeled kinematically as a sequence of changes in joint angle, without any analysis of the goal state for the movement. To make a first step toward addressing this gap, this chapter describes a laboratory experiment designed to collect natural, full-body movement data from normal adults engaged in a realistic task.

Everyday movements are typically directed toward the achievement of some sort of task-specific goal—in the examples above, these might be situations like walking to get to the store, speaking to ask a question, or picking up a glass to drink some water. Cognitive goals can sometimes be quite difficult to formulate symbolically or descriptively; however, as in dancing or singing, movements are themselves often elegant solutions to their corresponding tasks. While some movement goals might be relatively obvious—placing food in the mouth or pushing a button during an experiment, for example—even move-

ments that have complex or subtle goals can be productively characterized as being conceived and executed with respect to one or more consecutive goal states.

Unrestricted movements can be difficult to measure precisely or manipulate experimentally, and the goals of such movements are even more difficult to quantify. Studies of movement conducted in laboratory settings are often made tractable by restricting the experimental task to a small set of degrees of freedom, and studies of movement without restrictions on the degrees of freedom can only use a limited set of measurement techniques. The extent to which such limitations skew the applicability of laboratory results to other situations remains unknown. Similarly, existing databases of full-body movement tend to lack detailed annotations that might make possible a quantitative analysis of both movement and goal; currently, movement databases tend to be used primarily for classification (Johnson and Ballard, 2014a), even though there do exist several models that focus on movement generation Taylor et al. (e.g., 2007); Calandra et al. (e.g., 2014).

This chapter describes a novel experimental paradigm for recording full-body human movements along with annotations describing the frame-by-frame goals of those movements. After describing the experimental setup and data collection, some basic summary analyses are presented, including a description of the preprocessing that was required to convert the raw motion-capture data into a usable dataset for analysis. After a basic analysis, the next chapter contains descriptions of qualitative and quantitative efforts to model

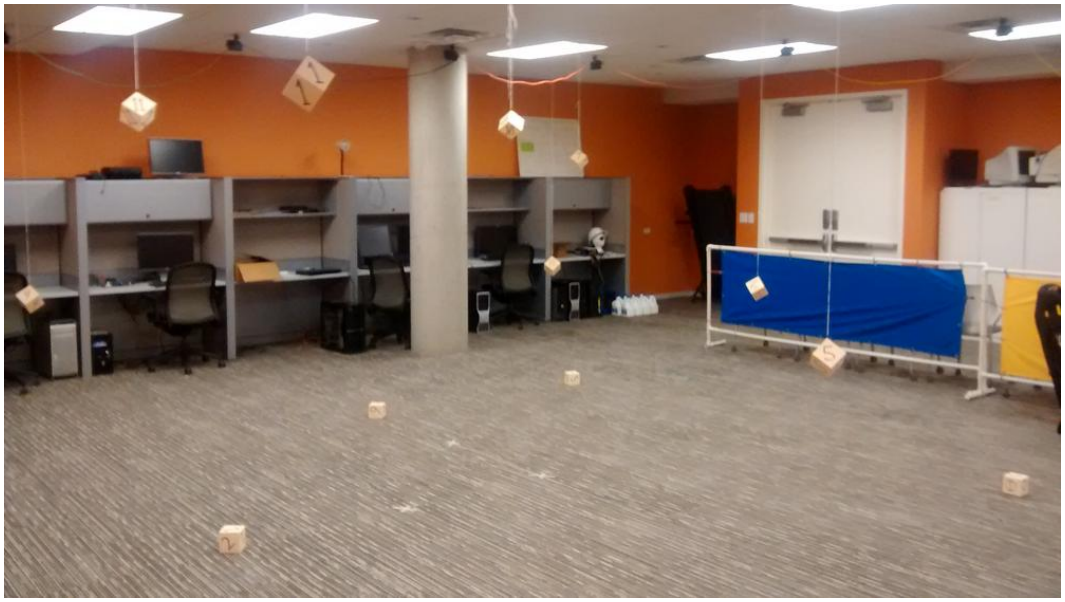
the Jacobian that can be computed using the data recorded in this experiment.

7.1 Reaching Task

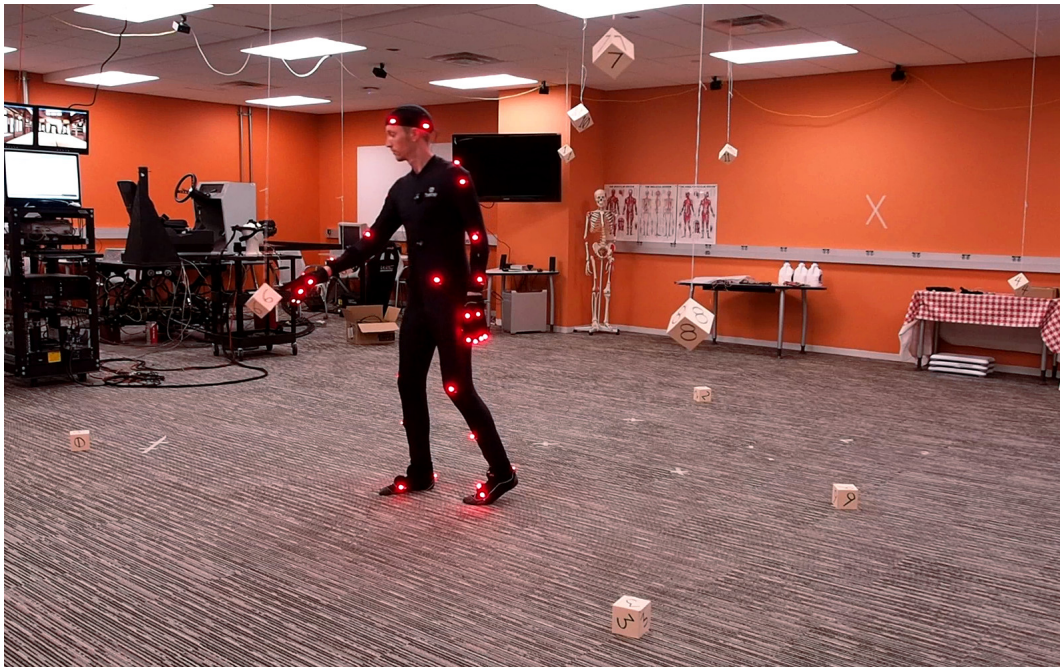
To analyze goal-directed movements, we first need a set of data that includes both measurements of movement and the associated, quantitative, goal states. Unlike studies of fronto-parallel planar arm reaching, an ideal task would involve minimal experimental instruction and elicit a variety of natural movements in the course of completing the task. With this in mind, I designed and conducted an experiment to elicit and measure natural, full-body movements from people engaged in a natural reaching task. As participants completed each trial, their movements were measured using a motion-capture (mocap) system; additionally, the positions of the "source" and "goal" were added to each frame of mocap data as it was recorded.

The experiment was designed as a full-body reach-to-target task: participants in the experiment were required to touch, using the index finger on their right hand, a series of targets stationed at fixed locations in the laboratory. In this variant of the reach-to-target task, however, the targets were spread out in a volume of space approximately 4m $\times$ 4m $\times$ 2m, large enough that participants needed to plan and execute sequences of full-body movements—walking, crouching, and reaching—to complete the task.

The targets for the task were 12 paper cubes measuring 10cm on a side. Each cube was labeled on all faces with its target number, an integer between 0 and 11 (inclusive). Target cubes were placed at a subset of the 27 points



(a) Cube placement in the motion-capture space.



(b) Subject doing the task.

Figure 7.1: A participant engaged in the reaching task, wearing the full-body mocap suit. The 3–dimensional position of each suit LED was recorded during the task. The 12 targets are visible as white cubes in the mocap area.

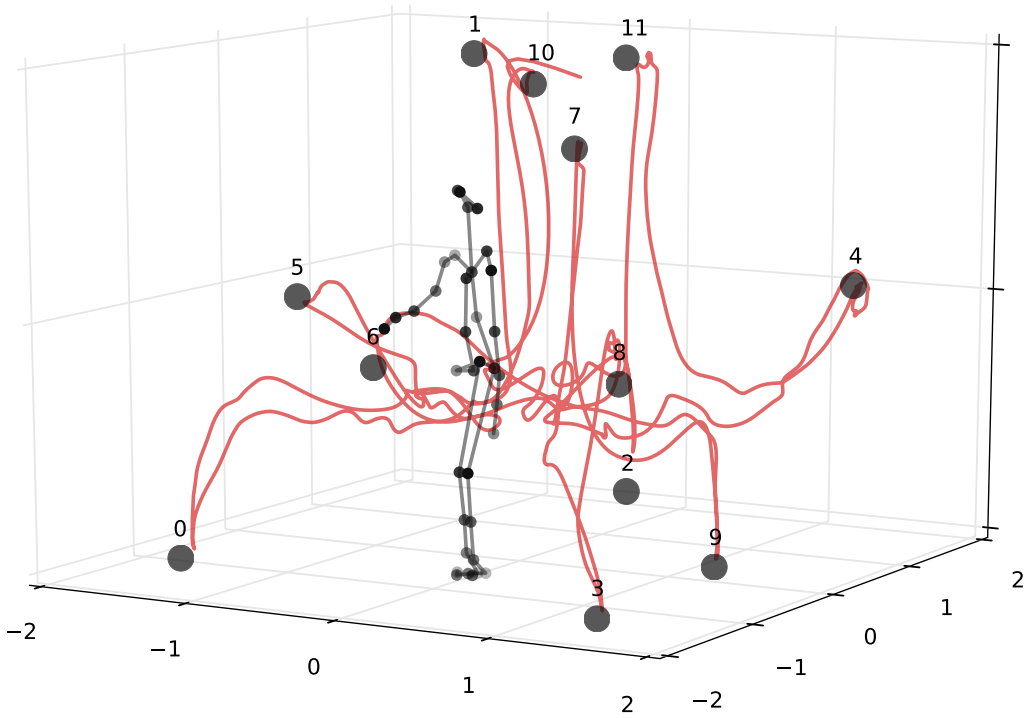


Figure 7.2: Example motion-capture marker data from the experiment. Numbered circles are reach targets, black dots are mocap markers, and the red line is a trace of the location of the right index finger for this trial. The skeleton is rendered for clarity but is not part of the mocap recording.

defined by a virtual grid in the mocap space—three xy planes at different z–elevations, three yz planes at different x–offsets from the origin, and three xz planes at different y–offsets from the origin—such that targets were distributed approximately evenly among the three planes in each orientation (see Figures 7.1 and 7.2). For example, four of the targets were placed on the xy plane on the floor, four were on the xy plane at mid–height ($\approx$ 1m from the floor), and four were on the xy plane above head level ($\approx$ 2m from the floor). Once the targets were placed in the mocap area, their precise quantitative locations were registered in the experiment software using a calibration script.

The experiment was organized in a series of blocks, each of which consisted of six trials. In each trial, participants touched each of the 12 targets exactly once, in a predefined but randomly generated order. A trial proceeded as follows: the experiment software played a synthesized audio clip pronouncing the number of one of the 12 targets (one of the words "zero" through "eleven"). The participant then started the trial by touching the corresponding target with the index finger of their right hand. As soon as a target was touched, the experiment software would name a new target number, and the participant would have to locate and touch the corresponding target. A trial ended when the participant touched the twelfth target in the sequence.

Targets in a trial were ordered according to one of six permutations generated before the experiment began. The six trial permutations were created so that the number of distinct target pairs (for example, starting at target 1 and moving toward target 2) covered by the experiment was maximized. Each

block in the experiment consisted of all six trial permutations, presented in the same order across blocks and subjects.

Eleven healthy undergraduate and graduate students were recruited from The University of Texas at Austin and were paid for their participation after giving informed consent. All participants completed at least four blocks (24 trials) of the task, but most participants completed five blocks; each trial normally took about 1 minute to complete, so that a single subject needed about 45 minutes total to complete the experiment.

7.2 Motion Capture

During the experiment, participants wore a motion-capture suit equipped with 48 active-pulse LED markers (see Figure 7.1). Between the start and end of each trial (i.e., the moments of contact with the first and last targets in the trial, respectively), a 16–camera Phasespace[1] motion-capture system measured the 3–dimensional location of each of these markers at a frequency of 100Hz and with an absolute error of approximately 1mm. Positions of the markers were recorded by the experiment software at intervals of approximately 10ms, resulting in approximately 5000 frames of mocap data per trial. For each frame, the 3–dimensional locations of the current "goal" target as well as the previous "source" target were also recorded.

The software for running the experiment and recording the motion cap-

[1] `http://phasespace.com`

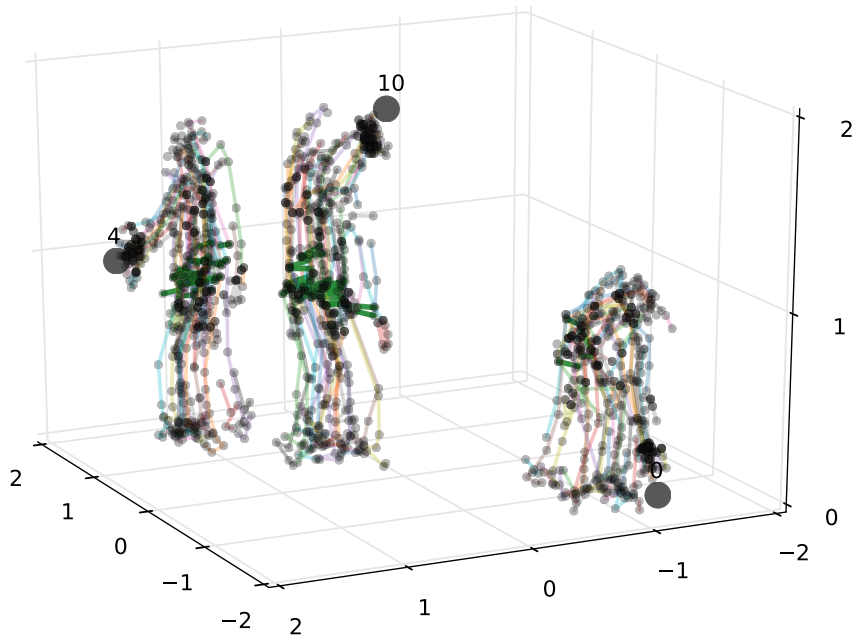


Figure 7.3: In the cube experiment, subjects were required to touch target objects using their right index finger. Even across subjects, postures were highly regular.

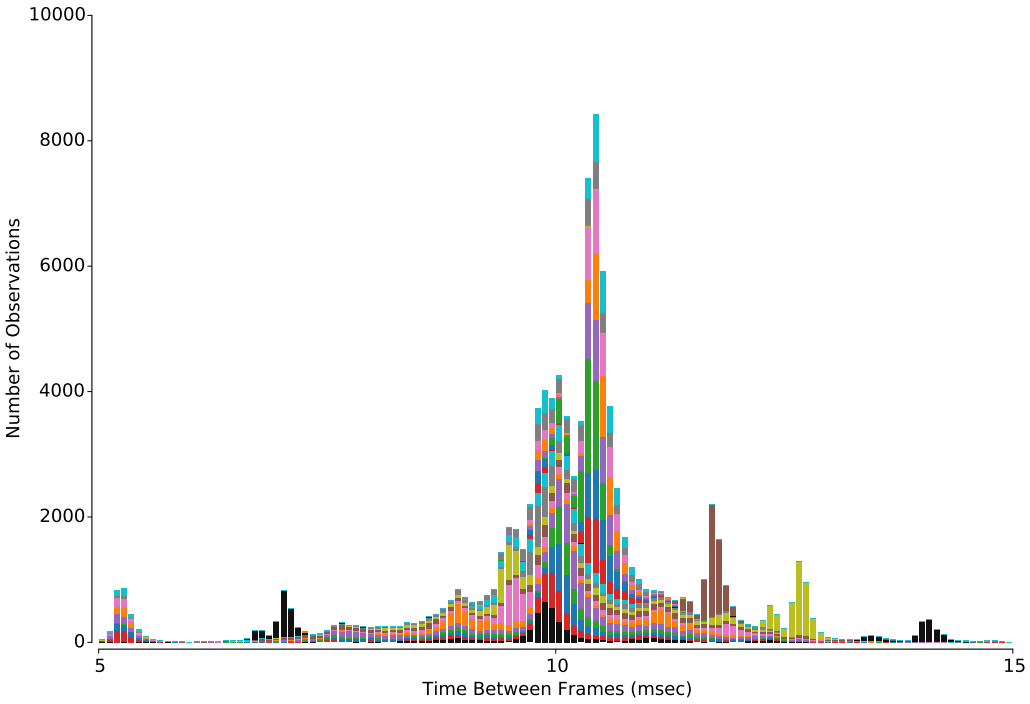


Figure 7.4: Motion-capture data were recorded at a nominal frequency of 100Hz during the cube experiment, but due to inaccuracies in scheduling the recording callback, the actual time interval between successive frames varied somewhat. These inaccuracies were remedied after recording by interpolating observed data points, as described in the text.

ture data was implemented using version 4 of the WorldViz Vizard software package[2] and is freely available online.[3]

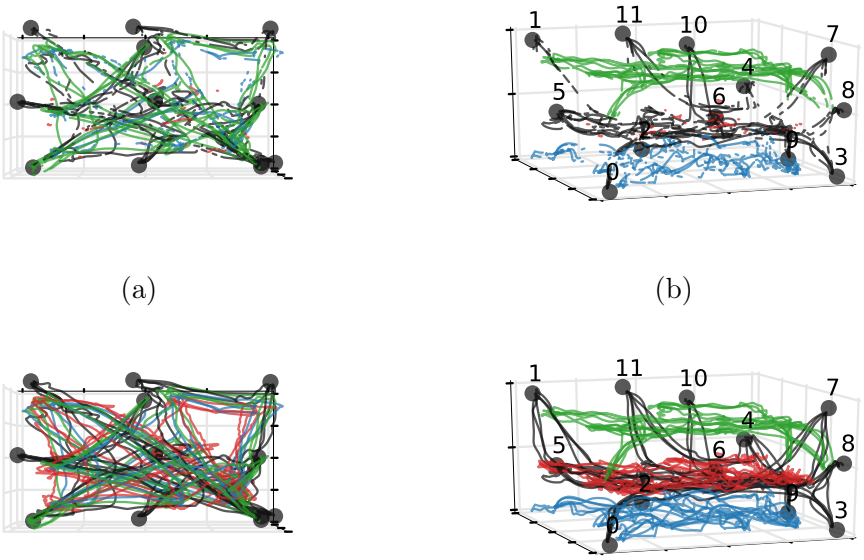


Figure 7.5: Raw motion-capture data recordings in (a) and (b) contained noticeable segments of "dropouts" where the mocap system could not compute the location of a marker. After preprocessing using SVT (Cai et al., 2010), filled data in (c) and (d) reconstructed the dropouts while preserving the statistical structure of the movements.

7.3 Preprocessing

Although the mocap system and experiment software were running at a fixed nominal frequency of 100Hz, the actual time interval between recorded frames varied, sometimes by as much as 50% (see Figure 7.4). To ensure that further analysis was based on regularly sampled data, mocap recordings were first downsampled to 50Hz, using the closest available observation within 100ms to do the resampling.

As mentioned in the previous chapter, Raw mocap data contain "dropouts," or frames where the mocap system is unable to determine the 3–dimensional location of a marker. Ideally, dropouts would consist of just a few intermittent frames, distributed randomly among the mocap markers on the body suit. In practice, however, dropouts are not randomly distributed—for instance, some subjects hold their hands upside-down while walking, preventing the mocap markers on the backs of the gloves from being seen by most of the ceiling-mounted cameras. In addition, dropouts in practice often occur in large chunks, say because a marker is occluded by a body part or goes outside the calibrated recording range of the mocap system.

These facts about the distribution of dropouts make linear interpolation a poor choice for reconstructing the missing marker data: if a foot marker drops out for an entire stride during walking, for example, a linear interpolation

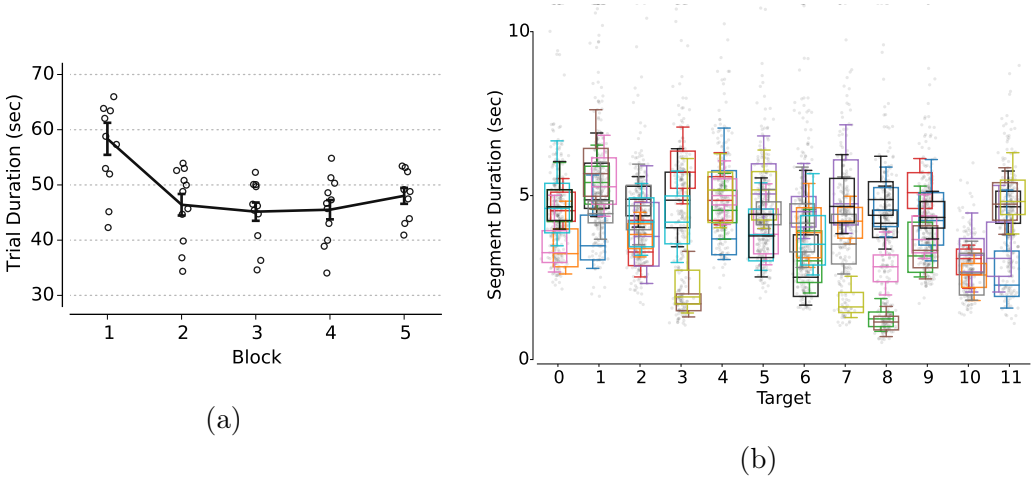
[2] `http://worldviz.com/products/vizard`
[3] `http://github.com/lmjohns3/cube-experiment`

would just "connect the dots" and miss the structure present in the different phases of a stride during walking.

To reconstruct dropped data with a minimal impact on the statistical structure of the data, trajectory-based singular value thresholding (Tan et al., 2013; Cai et al., 2010) resulted in good performance, although it was quite resource-intensive. Trajectories were reconstructed using a window of 10 frames of data, with an absolute reconstruction tolerance of 10^{-3} for visible data, approximately equal to the known precision of the Phasespace mocap system. The data for each subject was interpolated using a separate matrix completion model.

7.4 Basic Behavioral Analysis

Several facts are immediately clear when looking at the gross behavioral patterns in the cube experiment movement database. The time required to complete a trial decreases sharply in the first block, after fewer than ten exposures to each target location. Furthermore, the structure of the variance for the observed movements tends to corroborate existing observations about goal-directed movement: variance decreases with time-to-contact, but noticeably more so for the task-relevant end effector (here, the right index finger) than for task-irrelevant ones (e.g., the left index finger).

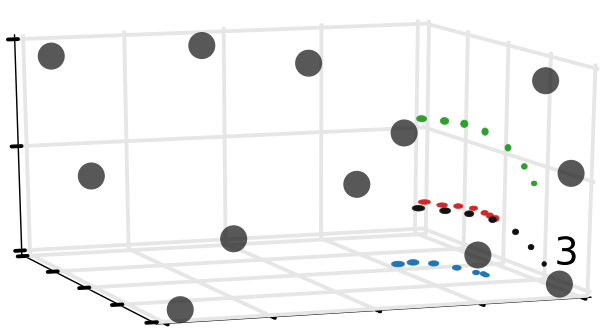


(a) (b)

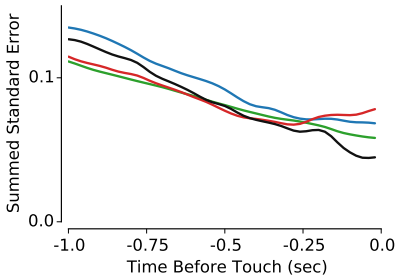
Figure 7.6: (a) Subjects performed trials faster as the experiment proceeded, indicating that they were learning where the targets were in the space. (b) Segments of each trial varied in difficulty depending on the source and goal targets.

7.4.1 Target Location Learning

As shown in Figure 7.6a, subjects quickly learn—even within the space of one block, or six exposures to each cube—the locations of the targets in the experiment space. Anecdotally, subjects perform each trial in a staccato or punctate manner. For each target, a subject first spends some time moving mostly their head and eyes, until they have located the target in the mocap space. Having located the target, they then start to walk towards it, and when their body comes within reaching distance of the target, they move their index finger, either upward toward the target from waist level, or by bending at the waist for targets on the floor. As each trial exposes the subject to each target once, subjects rather quickly learn the locations of the targets, spending less time at the beginning of each segment visually locating the target.



(a)



(b)

Figure 7.7: Subjects demonstrated many classic features of task-oriented movements in the cube experiment (Scholz and Schöner, 1999). The variance of different mocap markers, shown in (a) using ellipsoids of radius one standard deviation and in (b) as the summed standard deviation, decreases as subjects approach a target, but less so for markers attached to task-irrelevant markers (e.g., the left index finger). Here, variance is measured across all subjects.

Time to achieve target contact also varies dramatically for each segment of a trial, as shown in Figure 7.6b. This phenomenon appears to be strongly correlated with the distances between the source and goal targets in each segment.

7.4.2 Task-Relevant Variance Structure

Many movement researchers have noticed a structured decrease in end effector variability as subjects approach the kinematic goal state of a movement (Scholz and Schöner, 1999). The cube experiment also displayed such a structure, with variability of each mocap marker showing a noticeable de-

crease as a subject approached the moment of target contact (see Figure 7.7). Variance—even when measured across subjects—tended to decrease basically monotonically for the task-relevant end effector, but variance for an irrelevant marker such as the left index finger was observed to "plateau" at some elevated threshold near the moment of target contact.

In some ways, the kinematic structure of the skeleton itself can be seen as explaining such a variance structure: to make contact with the target, the right index finger must be placed in a particular location in world coordinates, so at the moment of contact the variance of the end effector mocap marker must de facto be minimal. Markers attached to the right hand would be expected to show small variance, as there are only so many ways for the hand to move away from the index finger without compromising task performance. A similar, but less strong, argument holds for a marker at a place like the right shoulder, since a relatively simple kinematic linkage connects the shoulder to the index finger.

However, the variance structure observed in the cube experiment goes beyond what might be expected of a simple kinematic linkage. For instance, consider Figure 7.8, which shows the locations of the foot markers, across subjects, for a particular segment of the experiment, a transition from the "5" target to the "7" target. The foot markers show clear placement patterns on the floor, even across subjects who have different skeleton sizes. The amount of overlap in different foot placements leading up to the target contact is remarkable.

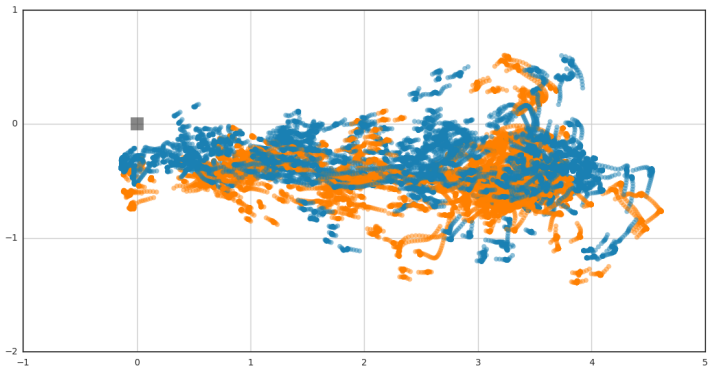


Figure 7.8: Subjects placed their feet in similar x–z positions over the course of a given movement segment in the experiment, despite differences in skeleton kinematics and task performance. Shown in orange are the markers on the left foot, and in blue are markers on the right foot, for all subjects on a particular segment of the experiment. In this case, the segment was the transition from the "5" target (not shown, located at approximately (4, 0)) to the "7" target (gray square at the origin). Dots are colored so that fast-moving frames are mostly transparent, while frames with lower speed are mostly opaque.

7.5 Discussion

This chapter has described a novel laboratory experiment designed to elicit and record full-body, goal-oriented movements in a natural reaching and pointing task. The data, collected in our motion-capture laboratory, demonstrate several classic features of task-oriented movements. In particular, variance of task-relevant mocap markers was observed to decrease more for task-relevant markers than for markers that were not relevant to the task. Furthermore, even without constraints on the types of movements permitted, subjects tended to adopt remarkably similar movement patterns to accomplish the task.

These basic results imply not only that movements themselves are redundant; they also indicate some consistency in the pattern of movements that different subjects choose over time. The next chapter describes an effort to model a first-order approximation to this temporal movement process.

Chapter 8

Coding Jacobians

In addition to the phenomena directly related to a movement—changes in posture, joint torques, flexions of muscles, and so on—voluntary movement is also a cognitive phenomenon, in the sense that voluntary movement is the result of some sort of control or planning process initiated by the mover. (This can be contrasted with involuntary movement, which is "hard-coded" by the cells in the nervous system.) In fact, particularly when viewed through the lens of the MMM (Chapter 6), all voluntary movements can rightly be conceived as having some sort of goal state. The goal state of a movement can, under the MMM, be expressed most generally as a temporal sequence of positions and spring stiffness values for the point masses that are assumed to drag the skeleton around. A given movement might or might not achieve the goal state—for example, "pure" vocal tract configurations are rarely achieved during fluent speech—but nonetheless there is a goal for a given voluntary movement.

As such, all voluntary movements involve at least two coordinate systems: an "internal" one that the volitional agent controls, and an "external" one that defines the goal and the agent's relationship to the goal. If they are observed simultaneously, these two spaces can be used to compute a Ja-

cobian, which is the best linear approximation to the instantaneous solution of the goal at any point in time. Empirically, like so many other facets of movement, Jacobian values also appear to live on a low-dimensional manifold, further reinforcing the notion that efficiency will be an important component of representing movement.

There seems to be some debate about the number and nature of reference frames involved in control; there is some evidence that more than two coordinate frames are involved in motor learning and planning (Berniker et al., 2014). However, for the present purposes, we will consider movement as though it has a fixed, external coordinate frame, and a controllable, internal one. These two frames are fairly easy to describe with respect to the measurements made in the cube experiment.

8.1 Body and Goal Reference Frames

As described in Chapter 6, the brain is essentially "trapped" inside the skull, at one end of the spinal cord. It must decide how to issue control commands to the muscle fibers in the body, so that the body can attain whatever external movement goals the brain has in mind.

8.1.1 Internal Frame

To a roboticist, the controls available to the central processor are often thought of as the torques capable of being generated by the motors in the robot. To a biological brain, the control signals can be thought of as the

nervous inputs to the muscle fibers. However, biological motor control is often abstracted into an "internal" space of joint angles, much like in the robot. While the actuators in a biological body are often quite different in nature from those in a robot, the basic physics of needing to use torques to create rotations remain the same.

Often, then, the internal reference frame for an analysis of movement is expressed as a set of joint angles. Here, however, instead of assuming that the internal control mechanism relies on moving muscles or rotational motors, we will adopt the perspective of the MMM. In this view, movements are made by changing the locations of the point masses surrounding the body. (Movements might also be made by changing the stiffness values of the springs attaching those masses to the body; however, this will not be addressed here.)

Any motion-capture dataset can be converted easily from world coordinates into an "internal" reference frame suitable for control under the MMM. In the case of the cube experiment described in the previous chapter, the body-relative reference frame was obtained from the mocap marker data alone by performing three linear transformations. First, marker data was centered on the location of the hip markers at each frame. Then, the centered markers were rotated so that the hips faced in the positive x direction at every frame. Finally, the scaled, rotated marker values were converted to z-scores by subtracting each marker's mean and dividing by its standard deviation. In addition to providing a subject-specific rescaling of the marker displacements so that they were no longer in world-compatible units, converting to z-scores

also normalizes somewhat for skeleton size, and for usage patterns in how the skeleton is employed to make movements. These three operations were performed separately for each trial.

Formally, converting the recorded data $\mathbf{P}$ into a body-specific reference frame $\mathbf{B}$ was achieved by

$$\hat{\mathbf{B}} = \mathbf{R}_t \left(\mathbf{P} - \mathbf{C}_t \right) \tag{8.1}$$

$$\mathbf{B} = \frac{\hat{\mathbf{B}} - \mathbb{E}[\hat{\mathbf{B}}]}{\mathbb{E}[\hat{\mathbf{B}}^2]} \tag{8.2}$$

where $\mathbf{C}_t$ is a matrix containing the 3–dimensional location of the hip markers at each frame t, $\mathbf{R}_t$ is the two-dimensional rotation matrix orienting the skeleton along the positive x axis at each frame t, and the $\hat{\mathbf{B}}$ are normalized to z-scores over all frames.

In the resulting body-specific reference frame, the three coordinate axes are:

- x: points in the direction the subject is facing.
- y: normal to the ground plane of the mocap area.
- z: obtained from the cross product $x \times y$.

8.1.2 External Frame

Every voluntary movement can be thought of as having a goal. Furthermore, as with many everyday movements like walking or picking up an object,

this goal will often be expressible in some way related to the world coordinate frame. In the cube experiment, the goal is quite simple to formulate in terms of world coordinates: on each segment of each trial in the experiment, the goal has been achieved when the index finger of the right hand coincides with the location of the target for that movement segment. Movement goals expressed in world coordinates might also include some component of velocity, as with many goals in sports such as a baseball swing or soccer kick. These will not be addressed explicitly here, although the principles remain the same.

Converting data collected in the cube experiment to a goal-relative reference frame was straightforward and involved two linear transforms. During the course of each trial in the experiment, the mocap marker data stream was recorded along with the locations of the relevant targets in the laboratory. To convert the data to a goal-centered reference frame, then, the location of the target was subtracted from each frame of the data, so that participants always appeared to approach the origin as they neared a target. Second, the centered marker data was rotated by the angle between the previously-touched target and the current goal, so that the direct path between the two targets was always from the positive x axis toward the origin.

Formally, converting the recorded data $\mathbf{P}$ into a goal-specific reference frame $\mathbf{G}$ was achieved by

$$\mathbf{G} = \mathbf{R}_t \left(\mathbf{P} - \mathbf{T}_t \right) \tag{8.3}$$

where $\mathbf{T}_t$ is a matrix containing the 3–dimensional positions of the target at

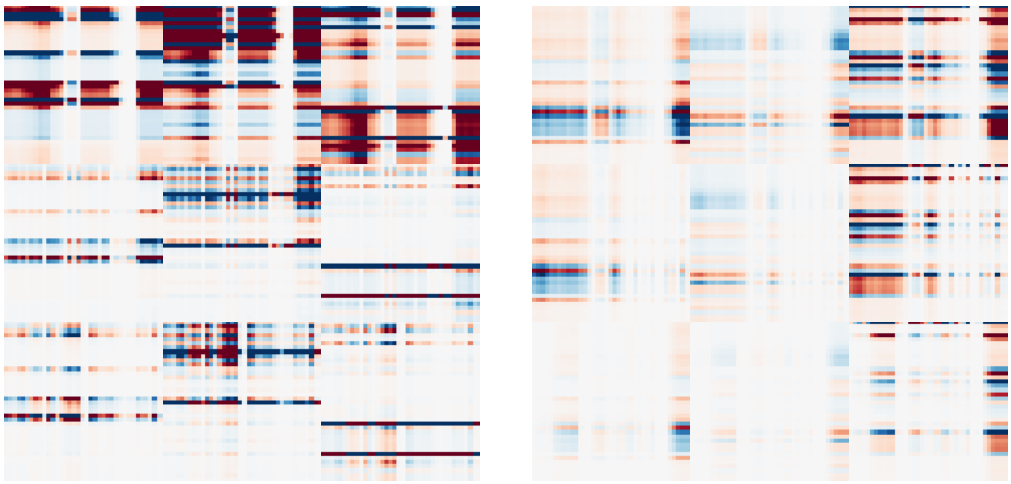


Figure 8.1: A Jacobian matrix for a single frame maps a change in each mocap marker's body-relative position to the corresponding change in its position in world coordinates. Shown here are example Jacobian matrices from two different moments in time during one trial. Qualitatively, the two moments show remarkably different dynamics.

each frame t, and $\mathbf{R}_t$ is the two-dimensional rotation matrix orienting the position data along a line from the source to the target at each frame.

In the resulting goal-specific reference frame, the three axes are:

- x: points from the goal target at each frame to the "source" target.
- y: normal to the ground plane of the mocap area.
- z: obtained from the cross product $x \times y$.

8.2 Empirical Jacobians

Having defined an internal and a corresponding external reference frame for the cube experiment, computing the empirical Jacobian for a trial was

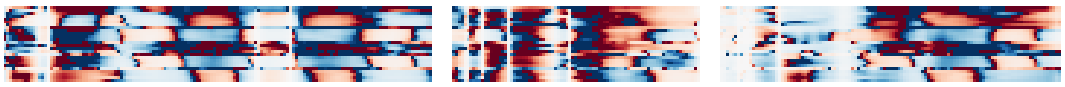


Figure 8.2: This image shows a consistent slice through Jacobian matrices over the course of a single trial. The slice shown here relates the movement of the right index finger in the x direction in world coordinates (i.e., toward the goal) to movement in the x direction in body coordinates (i.e., forward from the center of mass of the subject).

accomplished easily using basic division. Formally, the Jacobian is the matrix of partial derivatives of a vector–valued function f of a vector input:

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial x_m} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix}. \tag{8.4}$$

In the case of the cube experiment, the function f is the map from the body-centric coordinate frame to the goal-centric one, i.e., $f(\mathbf{b}_t) = \mathbf{g}_t$. We can approximate this Jacobian at frame t using

$$J_f^t = \begin{bmatrix} \frac{\Delta \mathbf{g}_1^t}{\Delta \mathbf{b}_1^t} & \cdots & \frac{\Delta \mathbf{g}_{3M}^t}{\Delta \mathbf{b}_1^t} \\ \vdots & \ddots & \vdots \\ \frac{\Delta \mathbf{g}_1^t}{\Delta \mathbf{b}_{3M}^t} & \cdots & \frac{\Delta \mathbf{g}_{3M}^t}{\Delta \mathbf{b}_{3M}^t} \end{bmatrix} \tag{8.5}$$

where $\Delta \mathbf{x}_i^t$ represents the difference between $\mathbf{x}_i$ at time t and $\mathbf{x}_i$ at time $t + \Delta t$.

For the cube experiment, the resulting Jacobian was a matrix of size 126×126 for each time frame in a trial. To encourage stability, Δt was set to 100ms, i.e., a distance of 5 frames in the preprocessed data. Example Jacobians for single frames are shown in Figure 8.1; the coordinates in the matrices are grouped together so that all x coordinates appear first, followed by all y and then z coordinates.

In addition to having structure within each frame of a movement, the Jacobian also evolves over time. A particularly interesting subset of the Jacobian data, shown in Figure 8.2, concerns the relationship between the right index finger's movement in the x direction (i.e., toward the goal) when the markers in the body change in the forward x direction (i.e., frontwards with respect to the center of the skeleton). This slice, shown for every frame of a subset of one trial, reveals striking visual patterns of movement. Each row of pixels in the figure corresponds to a different mocap marker's movement in body coordinates, and how that movement relates to movement of the index finger toward (or away from) the goal. Mocap markers are attached to the body in roughly topological order, so that the head markers are at the top, then the left and right arms, then the torso and pelvis, followed by the left and right legs. Clearly visible in the figure are moments of walking, characterized by bands of bilaterally alternating movements in the arms and legs. Also visible are moments of relatively little movement, when a subject has either just started a new movement segment and might be looking around for the target, or when a subject has approached a target and is reaching for it using only their arm.

8.3 Encoding the Jacobian

Computing the Jacobian is the first step towards a representation of movement that takes into account the observed dynamics that subjects choose to employ during the full-body reaching tasks of the cube experiment. Once

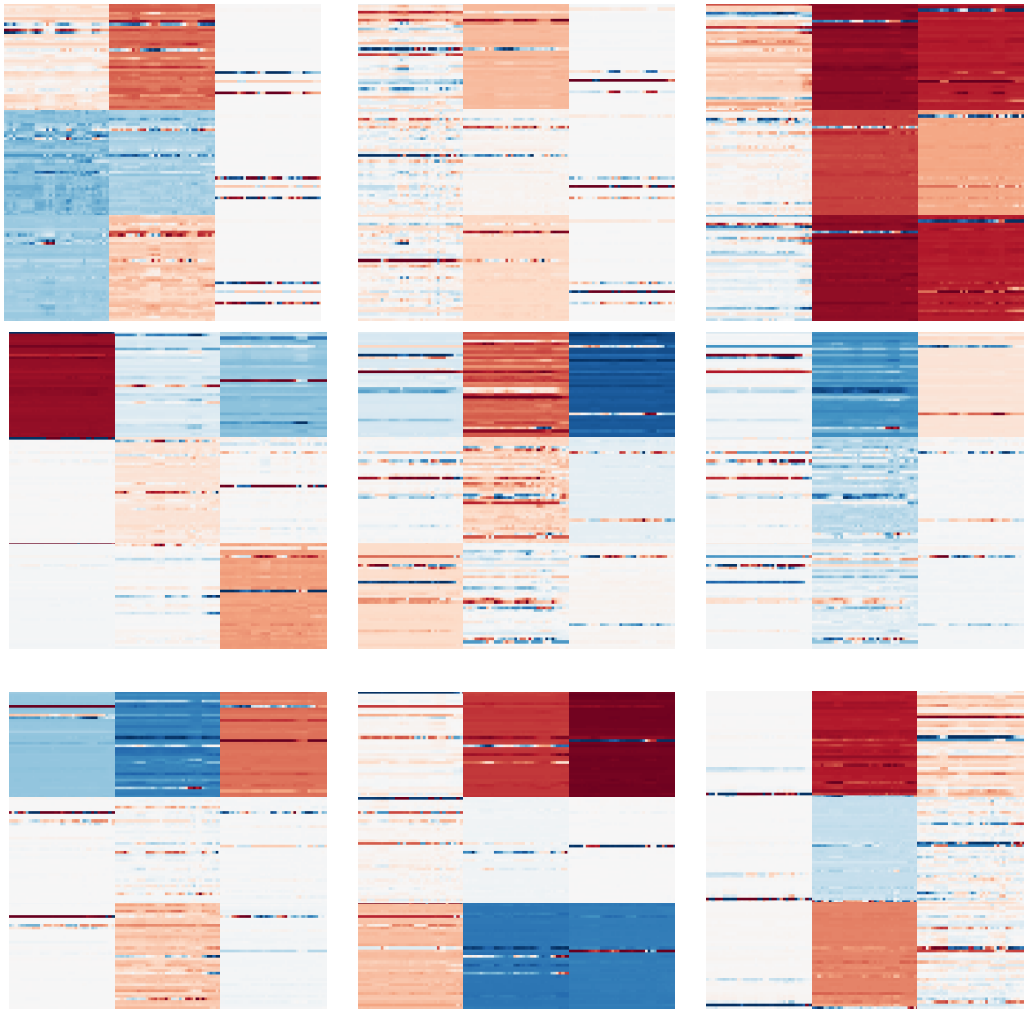


Figure 8.3: The first nine PCA basis elements for representing Jacobians capture approximately 97 percent of the total variance in the empirical Jacobian dataset.

Jacobians have been computed, we wish to learn more about the manifold that contains these patterns of movement.

The first tool often employed to model a low-dimensional, ellipsoidal data manifold is Principal Component Analysis (PCA; Hotelling, 1933). As described in Chapter 2, PCA computes the linear projection that maximizes the variance of the projected data. PCA is extremely widely used and fast to compute. Figure 8.3 shows the first nine basis elements—or "eigen-Jacobians"—of the PCA projection for the observed Jacobians in the cube experiment. The details of the basis elements are not extremely important; for the moment, it suffices to observe that the eigen-Jacobians capture both overall directional correlations (large blocks of color) as well as local interactions between specific markers in the goal space and the body space (small stripes within a block).

Although PCA is computationally convenient and widely used, it cannot capture any nonlinearities in the structure of the Jacobian manifold. To uncover more details of the Jacobian manifold, then, a regularized PCA approach permits a tradeoff between representing a dataset accurately (in a squared-error reconstruction sense) and permitting the learned basis to be non-orthogonal and overcomplete.

To evaluate the qualitative performance of a nonlinear encoding approach with the Jacobian dataset, we applied "sparse PCA"[1] and observed the resulting basis, shown in Figure 8.4. This basis, like the PCA basis, also shows

[1]Implemented in the Python `scikit-learn` package.

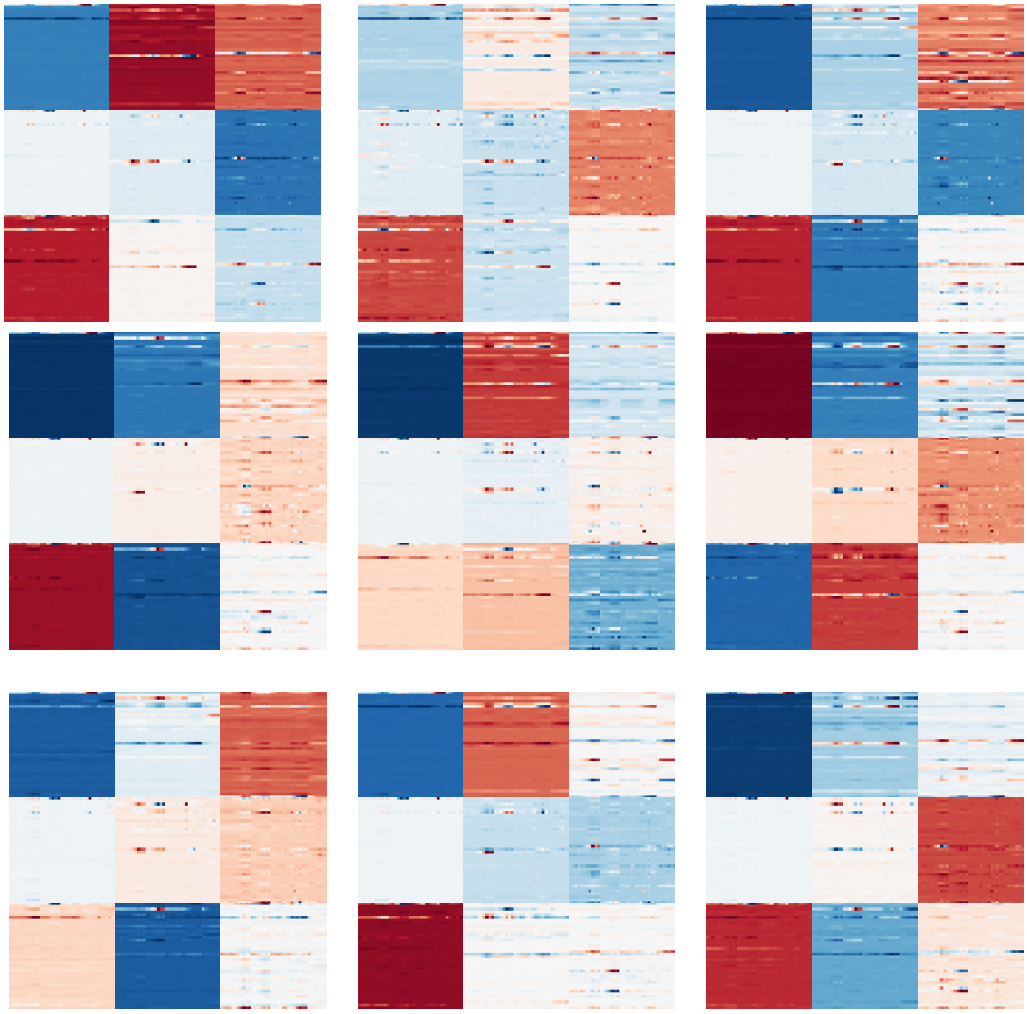
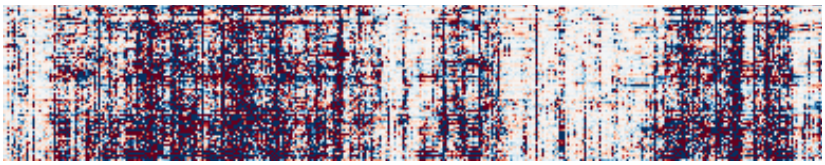


Figure 8.4: Nine basis elements for representing Jacobians using L1-regularized PCA. This basis is qualitatively different from the unregularized PCA basis.

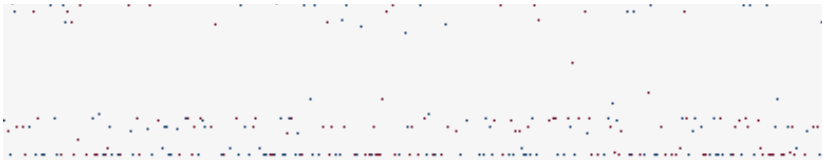
some block-level structure relating changes in one direction in world space to changes in another direction in body space, across all mocap markers simultaneously. Also like the PCA basis, the sparse basis relates changes in specific mocap markers in the two spaces; these changes appear as stripes or dots in the basis matrices.

Upon further analysis, the sparse PCA basis appears indeed to be balancing the requirements of sparsity and effective representation. For example, consider the first PCA basis in Figure 8.3, at the top left. This eigen-Jacobian shows a zero relationship between changes in the goal space and almost all of the changes in the z coordinate of the body space (the third vertical column in this basis element). Changes in the z coordinate of the body space, however, do seem to have an effect on a couple of the markers in the goal space, as shown in the horizontal blue and red stripes within this column. However, given that the first PCA basis element accounts for 88 percent of the total variance in the Jacobians, it seems unlikely that changes in the the z coordinate of the body space would have such a strongly zero relationship with changes in the goal space.

Another striking difference between PCA and an encoding with a sparsity constraint occurs when examining the coded versions of a series of Jacobian values across different coding algorithms. Figure 8.5 shows encoded Jacobians using a "dense" projection onto the PCA basis (shown in Figure 8.3), and a "sparse" projection onto the regularized basis (shown in Figure 8.4). The sparse representation of Jacobian information employs far more zero coeffi-



(a) Densely encoded.



(b) Sparsely encoded.

Figure 8.5: Jacobians encoded (a) using PCA exhibit small, dense structure, while those encoded (b) using a sparse algorithm show significantly fewer nonzero coefficients to achieve the same level of error.

cients than the dense representation. The sparse codebook might, then, be learning to represent the "important" or information-dense segments of the Jacobian manifold, much like Barlow's original proposal describes.

8.4 Predicting Jacobians from Postures

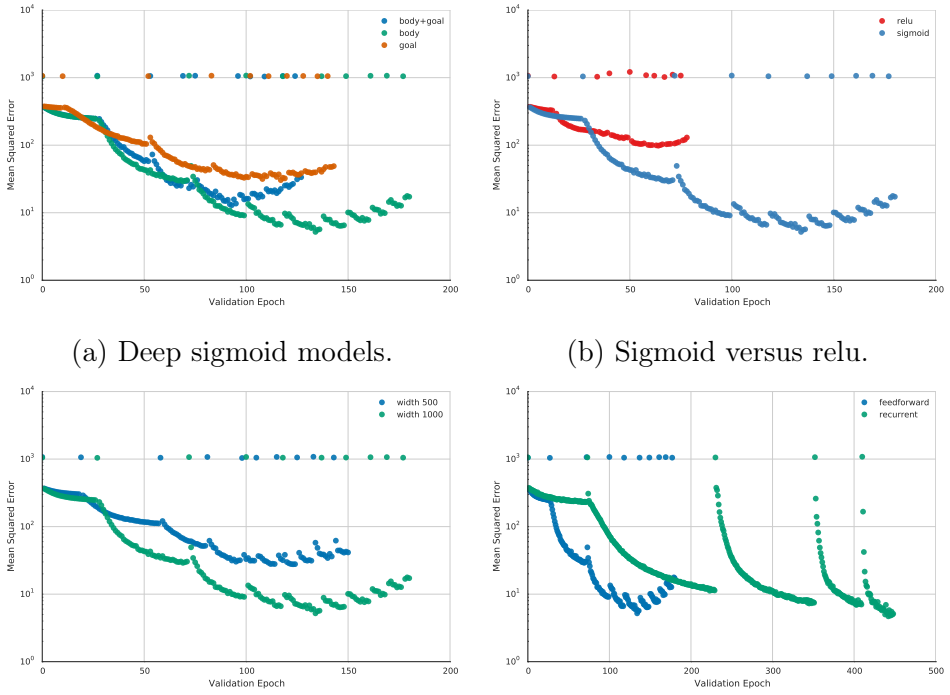
Consider again the brain, trapped inside a skull at one end of the spinal cord. Under the MMM, when the brain wants to effect a movement, it decides how to make a change in the body-relative positions of the point masses that drive the spring-skeleton system. In this chapter we have considered the Jacobian as the simplest approximation to a control system that maps changes in body state to their resulting changes in goal state. The brain could be making its movement decisions by obtaining the Jacobian for its current body state,

and then using the Jacobian to make a movement that reduces the task error in the extrinsic goal space.

The Jacobian is dependent on the current state of the body, so it is not sufficient to have a single estimate of the Jacobian. Instead, a brain must have a way of mapping the current body state onto an estimate of the Jacobian. Similar to the investigation described in Chapter 5, linear models are insufficient for this task, and even single-layer neural networks are too difficult to optimize to reduce the error in predicting Jacobians.

However, for the dataset collected during the cube experiment, large, multilayer neural networks appear to be fairly well-suited for the Jacobian prediction task. Training a neural network with many layers is well-known to be quite difficult, partly due to the vanishing gradient problem (Bengio et al., 1994). Although some progress has been observed in training entire deep models using rectified linear unit activations (Glorot et al., 2011), these results were not observed for modeling Jacobians; the rectified linear activation appears to "chop up" the input space, while for modeling Jacobians a smooth representation appears to be important (see Figure 8.6b).

Modeling the Jacobians was handled best by a deep, wide network of sigmoid layers, but such networks are not possible to train directly. We trained the deep models in a greedy, layer-wise fashion similar to the `TrainGreedySupervisedDeepNet` method described by Bengio et al. (2007). In this method, a one-layer regression network is trained on the Jacobian dataset. Once the training has converged, a second layer is inserted between



(a) Deep sigmoid models.

(b) Sigmoid versus relu.

(c) Sigmoid model widths.

(d) Model including GRU.

Figure 8.6: Neural networks trained to output Jacobians given some input state. (a) Mapping Jacobians worked best when the input was the body state; including goal state information degraded the quality of the model, probably because the goal state is not position-independent like the body state. (b) Deep regression networks using ReLU activations performed worse than those using sigmoid activations. (c) Wider sigmoid models outperformed narrower models. (d) A network containing a recurrent layer was able to match performance of the best feedforward model.

the hidden and output layers of the trained model, and the model is again trained to convergence. Training proceeds in this manner for as many layers as needed to minimize the objective.

The learning curves shown in Figure 8.6 contain the process of training these multilayer models over the entire training regimen. Each "J" curve is the training of an additional new layer in the model. In general, adding more layers to a regression model tends to improve performance (note the log scale on the vertical axis), but for this dataset about five layers seemed to capture the most variance in the data, while adding additional layers degraded performance. This degradation could be due to difficulties optimizing the model rather than an inherent limitation on multilayer models; however, it is clear that the marginal gains tend to decrease with each additional layer.

Finally, the Jacobians computed from movements in the cube experiment are properly a temporal sequence of Jacobians. The Jacobian values evolve over the course of a trial as the subject makes the entire movement in the trial. A recurrent neural network could be a powerful, nonlinear state space model for capturing this sequence of Jacobian values, but, again, optimizing such models is quite difficult. Anecdotally, recurrent network models with just one layer captured about the same amount of variance as a single-layer feedforward model. However, embedding a Gated Recurrent Unit (GRU; Chung et al., 2014) between large, sigmoid feedforward layers did succeed in capturing about as much variance as a feedforward-only model (see Figure 8.6d). Notably, the GRU in this model contained only 100 neurons, while the

feedforward layers in this and the feedforward-only networks contained 1000 neurons. Some efficient compression of the hidden state is being achieved, but further investigation will be required to determine what that state represents.

8.5 Discussion

Because the data collected in the cube experiment is annotated with goal locations, two coordinate frames can be computed from the movement data, and these two frames, when combined with the MMM as a theoretical basis for how the movement is accomplished, can be used to compute empirical Jacobians for every moment in the experiment. The high-dimensional Jacobian matrices exhibit dramatically low-dimensional structure that can be captured using PCA, but is even more amenable to representation using a sparse coding algorithm.

Neural networks are effective nonlinear regression tools for modeling a map from current body state to goal-oriented Jacobian. These neural network models are difficult to optimize, but using a greedy, supervised, layer-wise approach resulted in a model with an acceptable level of error. Recurrent networks, when embedded within a coding-decoding regression architecture, seem to be capable of representing a hidden state effectively, but that state's purpose remains to be investigated.

Part IV

Conclusions

Chapter 9

Summary

The work described in this document attempts to provide an answer for the broad question: *What are the independent components of movement?* While acknowledging that the question itself is too broad to be answerable—particularly in only one dissertation—the motivation provided by the question provides a means of understanding the work contained herein.

The document opened with a description of the basic notions of redundancy, sparsity, and efficiency, and how those notions were related to encodings of several different classes. A brief review of several approaches that have already been developed in motor control built a basis for describing the three branches of completed research work.

One branch of completed work presented a model for continuous control in the face of uncertainty, noise, and intermittent state updates. This work looked at redundancy in control as a way of splitting apart a task state space into independent modules. The second branch of completed work applied existing sparse coding algorithms to movement data recorded in a motion-capture laboratory, with the goal of either classifying movements or computing movement-related control signals through regression. The third branch of work

presented a novel reformulation of the Threshold Control Theory (TCT; Feldman and Levin, 2009) motivated by several computational tools. This model of movement was applied to full-body, goal-oriented reaching data collected in a novel laboratory experiment. Recording goals along with movement information permits the computation of empirical Jacobian matrices, and ideas of independence were applied to the space of these Jacobians to extract their hidden structure.

9.1 Contributions of This Work

The work described in this document makes several contributions to the scientific understanding of movement and movement modeling. Contributions range from traditional academic papers to code and outreach. The work I am interested in can be difficult to slot cleanly into a traditional discipline, but contributions spanning discipline boundaries can be a fruitful source of new ideas and progress for many academics.

9.1.1 Academic Contributions

This dissertation, through its analysis of movements and its consideration of redundancy reduction, has made several contributions to our knowledge of movement, control, and planning.

- **Analysis of coding techniques for movement data.** In applying existing coding techniques to kinematic movement data, a deeper un-

derstanding is provided of the conditions under which efficient codes enhance specific computational tasks like classification and regression.

- **Database of goal-directed reaching movements.** A novel human subjects experiment resulted in a high-quality database of movement annotated with real-time goal information. This experimental data will be made public in a journal paper.

- **Integrated coding and state space model.** A novel extension to recurrent neural networks was tested to analyze the effect of coding in a powerful state-space model. Although recurrent models remain difficult to optimize, there are some positive possibilities for training effective models that have come from this work.

- **Public dissemination of research results.** Both completed and remaining work have been and will be made public to the research community in the form of conference and journal publications. The completed work described in this document has been made available through five well-received conference publications and two journal articles. Work that remains to be published—detailing the MMM and the cube experiment database—is anticipated to be made available through a journal article.

9.1.2 Teaching, Outreach, and Code

In addition to contributing directly to the research community, the work described in this dissertation has also resulted in tools and knowledge

made useful for broad audiences beyond academia.

- **Instructing others in computer science.** In my time at UT Austin, I have been a teaching assistant on seven separate occasions. My instructor evaluations for introductory Java students were overwhelmingly positive; I truly enjoy giving others the knowledge to apply computer science themselves. I have also had the good fortune to teach an introductory Java class for middle-school students.

- **Broad dissemination of acquired knowledge.** Knowledge of coding techniques has enabled me to give presentations for larger audiences who are less familiar with academic research. Example presentations include graduate machine learning lectures, Python tutorials, and a talk at the SciPy 2013 conference which has subsequently been viewed more than 3000 times.[1]

- **Python code for computational modeling.** As a graduate student, I have written and made public[2] more than 20000 lines of quality, open-source Python code to achieve the work described in this document. Four of my machine learning tools have more than ten stars on Github; one of these,[3] a toolkit for defining and optimizing neural network architectures,

[1] `https://www.youtube.com/watch?v=nT4SGi-aaMU`
[2] `https://github.com/lmjohns3`
[3] `https://github.com/lmjohns3/theanets`

has over 200 stars! Several of my software packages include detailed documentation suitable for use by a broad audience.[4, 5]

These contributions, while not directly related to citation counts or other, traditional markers of academic achievement, have already had, and will continue to have, a positive impact on the growing world that surrounds machine learning research. As industry and academia work more closely together, openly available code and related tools—plus the knowledge of how to use them—will enable many new applications of machine learning research.

9.2 Future directions

In many ways, movement modeling appears to be just taking off as a research discipline. Powerful RL, graphical and neural network models are all starting to make possible a detailed representation of movement and its dynamics, while researchers in neuroscience are simultaneously advancing studies of multi-cell electrode arrays as well as fMRI data during movement. The research described in this document will contribute to the growing mountain of knowledge, but almost endless research remains to be done.

9.2.1 Cube Experiment Extensions

Additional modeling in the cube experiment could also be performed to map the observed mocap marker displacements onto joint angles, allowing

[4] `http://theanets.rtfd.org`
[5] `http://downhill.rtfd.org`

the modeling work to proceed in a more traditional angle-based setting.

One specific area of research that appears quite promising is to apply the data collected in the cube experiment to a more formal, probabilistic model of movement. A model that gives probabilistic likelihood values could be used to compute formal measures of complexity like entropy and information gain, which has the potential to expose this set of unrestricted, full-body, goal-directed movements to a deeper quantitative breakdown of its dynamics.

In particular, information is a natural medium in which to analyze why particular movements are made over others. If an information-theoretic model of movements—perhaps a distribution over sequences of Jacobian values—could be formulated effectively, it might help decipher the appropriate moment-by-moment coordinate frames for a particular sequence of movements, or it might help explain how it is that a movement toward a distant reaching goal is broken down into a sequence of particular foot placements over time.

Finally, the cube pointing task described in this document is more than "just" a pointing task requiring movement. In addition to motor planning and execution, participants must also locate targets visually before formulating a motor plan. This location process is an example of visual search and is, in its own way, an interesting problem to investigate. Augmenting the mocap measurements with a portable eye tracker could further help bind the state of the world with the participants' cognitive state (Land and McLeod, 2000; Hayhoe et al., 2003). After eye movements are collected, a host of analyses could be completed relating the timing of the movement of the body to the

timing of the movement of the eyes in full-body reaching tasks. A model like the modular RL approach described in Chapter 4 could be applied to understand why eye movements are directed at their particular targets at the moments when they are observed.

Clearly there is much progress that could be derived from the cube experiment!

9.2.2 Other Applications

Overall, integration across information sources is one of the most promising areas for future work based on the research in this proposal. After building and evaluating both PID and recurrent models of movement, the resulting coded movements can be used as alternate feature spaces for learning control strategies. In particular, integrating RNN-based controllers into RL task modules holds great promise for better understanding the ongoing sensory-motor loop during movement. Better optimization techniques will be critical in applying these large neural networks to problems that involve movement and control; this is particularly true for networks that contain recurrent layers.

Speech and language are particularly rich sources of movements in humans that will likely benefit from more sophisticated coding and sequence modeling. While speech has been studied for centuries as a linguistic phenomenon, the ideas behind the motor theory of speech perception (Liberman et al., 1967; Liberman and Mattingly, 1985; Massaro and Chen, 2008), while

having received some recent neuroscience support (Pulvermüller et al., 2006; D'Ausilio et al., 2009), might be ripe for study using computational models of movement. Speech is a canonical example of using "chained posture targets" (vocal tract configurations for specific vowels and consonants) for generating a desired motor outcome. Bringing rich models of movement into speech processing could improve the state of the art in speech recognition as well as generation, but much work remains in this area before fruitful results might be found.

Humans and other animals **interact** with the world. By better understanding how these interactions take place, we can improve our scientific understanding of the mechanisms by which people understand and think about the way this world works.

Bibliography

M. S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, D. Sci, T. Organ, and S. A. Adelaide. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2): 174–188, 2002.

D. G. Asatryan and A. G. Feldman. Functional tuning of the nervous system with control of movement or maintenance of a steady posture: I. mechanographic analysis of the work of the joint or execution of a postural task. *Biophysics*, 10(5):925–934, 1965.

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

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

A. J. Bell and T. J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2): 157–166, 1994.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, 19:153, 2007.

Pietro Berkes, Gergő Orbán, Máté Lengyel, and József Fiser. Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science*, 331(6013):83–87, 2011.

M. Berniker and K. P. Kording. Deep networks for motor control functions. *Frontiers in computational neuroscience*, 9, 2015.

M. Berniker, D. W. Franklin, J. R. Flanagan, D. M. Wolpert, and K. P. Kording. Motor learning of novel dynamics is not represented in a single global coordinate system: evaluation of mixed coordinate representations and local learning. *Journal of neurophysiology*, 111(6):1165–1182, 2014.

N. Bernstein. The co-ordination and regulation of movements. 1967.

B. Bocsi, L. Csato, and J. Peters. Alignment-based transfer learning for robot models. In *Proc. IEEE Intl. Joint Conf. on Neural Networks*, 2013.

N. D. B. Bruce and J. K. Tsotsos. Saliency, attention, and visual search: An information theoretic approach. *Journal of Vision*, 9(3), 2009.

N. J. Butko and J. R. Movellan. Detecting contingencies: An infomax approach. *Neural Networks*, 23(8):973–984, 2010.

J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

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

R. H. S. Carpenter and M. L. L. Williams. Neural computation of log likelihood in control of saccadic eye movements. *Nature*, 377(6544):59–62, 1995.

J. H. Challis. A procedure for determining rigid body transformation parameters. *Journal of Biomechanics*, 28(6):733–737, 1995.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

M. M. Churchland, J. P. Cunningham, M. T. Kaufman, J. D. Foster, P. Nuyujukian, S. I. Ryu, and K. V. Shenoy. Neural population dynamics during reaching. *Nature*, 2012.

M. H. Coen. *Multimodal Dynamics: Self-Supervised Learning in Perceptual and Motor Systems*. PhD thesis, Massachusetts Institute of Technology, 2006.

S. Collins, A. Ruina, R. Tedrake, and M. Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

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

G. E. Dahl, M. Ranzato, A. Mohamed, and G. E. Hinton. Phone recognition with the mean-covariance restricted boltzmann machine. *Advances in Neural Information Processing Systems (NIPS)*, 23:469–477, 2010.

A. D'Ausilio, F. Pulvermüller, P. Salmas, I. Bufalari, C. Begliomini, and L. Fadiga. The motor somatotopy of speech perception. *Current Biology*, 19(5):381–385, 2009. ISSN 0960-9822.

V. R. de Sa. *Unsupervised classification learning from cross-modal environmental structure*. PhD thesis, University of Rochester, 1994.

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

S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen. Opensim: open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, 54(11):1940–1950, 2007.

R. Desai and H. Geyer. Muscle-reflex control of robust swing leg placement. In *IEEE Intl Conf on Robotics and Automation (ICRA)*, pages 2169–2174. IEEE, 2013.

J. Diedrichsen, R. Shadmehr, and R. B. Ivry. The coordination of movement: optimal feedback control and beyond. *Trends in Cognitive Sciences*, 14: 31–39, 2009.

A. d'Avella and E. Bizzi. Low dimensionality of supraspinally induced force fields. *Proceedings of the National Academy of Sciences*, 95(13):7711–7714, 1998.

A. D'Souza, S. Vijayakumar, and S. Schaal. Are internal models of the entire body learnable. In *Society for Neuroscience Abstracts*, volume 27, 2001.

J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

A. Farchy, S. Barrett, P. MacAlpine, and P. Stone. Humanoid robots learning to walk faster: From the real world to simulation and back. In *Proc. Intl. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 39–46. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

A. G. Feldman. Functional tuning of nervous system with control of movement or maintenance of a steady posture. 2. controllable parameters of muscles. *BIOPHYSICS-USSR*, 11(3):565, 1966.

A. G. Feldman and M. F. Levin. The equilibrium-point hypothesis–past, present and future. In *Progress in Motor Control*, pages 699–726. Springer, 2009.

D. W. Franklin and D. M. Wolpert. Computational mechanisms of sensorimotor control. *Neuron*, 72(3):425–442, 2011.

A. Frigon. Central pattern generators of the mammalian spinal cord. *The Neuroscientist*, 18(1):56–69, 2012.

Karl Friston. What is optimal about motor control? *Neuron*, 72(3):488–498, 2011.

T. Geijtenbeek, M. van de Panne, and A. F. van der Stappen. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics*, 32 (6), 2013.

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

F. Gomez, J. Schmidhuber, and R. Miikkulainen. Efficient non-linear control through neuroevolution. In *Proc. European Conf. on Machine Learning (ECML)*, pages 654–662. Springer, 2006.

I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

A. M. Gordon, A. F. Huxley, and F. J. Julian. The variation in isometric tension with sarcomere length in vertebrate muscle fibres. *The Journal of physiology*, 184(1):170–192, 1966.

A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

M. S. A. Graziano, C. S. R. Taylor, and T. Moore. Complex movements evoked by microstimulation of precentral cortex. *Neuron*, 34(5):841–851, 2002.

M. S. A. Graziano, T. N. Aflalo, and D. F. Cooke. Arm movements evoked by electrical stimulation in the motor cortex of monkeys. *Journal of Neurophysiology*, 94(6):4209, 2005.

F. H. Guenther. A neural network model of speech acquisition and motor equivalent speech production. *Biological Cybernetics*, 72(1):43–53, 1994. ISSN 0340-1200.

A. M. Haith and J. W. Krakauer. Model-based and model-free mechanisms of human motor learning. In *Progress in Motor Control*, pages 1–21. Springer, 2013.

N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

Z. S. Harris. Distributional structure. *Word*, 1954.

M. M. Hayhoe and D. H. Ballard. Eye movements in natural behavior. *Trends in Cognitive Sciences*, 9(4):188–194, 2005.

M. M. Hayhoe, A. Shrivastava, R. Mruczek, and J. B. Pelz. Visual memory and motor planning in a natural task. *Journal of Vision*, 3(1):6, 2003.

R. Held and A. Hein. Movement-produced stimulation in the development of visually guided behavior. *Journal of comparative and physiological psychology*, 56(5):872, 1963.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002. ISSN 0899-7667.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504, 2006.

G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. ISSN 0899-7667.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

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

I. S. Howard and P. Messum. A computational model of infant speech development. In *Proc. XII Intl. Conf. on Speech and Computers*, pages 756–765, 2007.

P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004.

D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.

A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653, 2008.

L. Itti and C. Koch. Computational modeling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, 2001.

H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

N. Jaitly and G. E. Hinton. Learning a better representation of speech soundwaves using restricted boltzmann machines. In *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, pages 5884–5887. IEEE, 2011.

L. Johnson and D. H. Ballard. Classifying movements using efficient kinematic codes. In *Proc. 36th Annual Mtg. of the Cognitive Science Society*, 2014a.

L. Johnson and D. H. Ballard. Efficient codes for inverse dynamics during walking. In *Proc. Assoc. Advancement for Artificial Intelligence*, 2014b.

L. Johnson, J. Cooper, and D. H. Ballard. Efficient codes for multi-modal pose regression. In *Proc. 35th Annual Mtg. of the Cognitive Science Society*, 2013a.

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

L. Johnson, B. T. Sullivan, and D. H. Ballard. A soft barrier model for modeling human visuomotor behavior in a virtual driving task. In *Proc. 35th Annual Mtg. of the Cognitive Science Society*, 2013c.

L. Johnson, B. T. Sullivan, M. M. Hayhoe, and D. H. Ballard. Predicting human visuomotor behaviour in a driving task. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1636):20130044, 2014.

M. I. Jordan and D. E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16(3):307–354, 1992. ISSN 0364-0213.

C. Jutten and J. Herault. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal processing*, 24(1): 1–10, 1991.

Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory. *Journal of Fluids Engineering*, 83(1):95–108, 1961.

M. Kawato. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9(6):718–727, 1999.

D. Kersten. Predictability and redundancy of natural images. *Journal of the Optical Society of America A*, 4(12):2395–2400, 1987.

S. S. Kety. The general metabolism of the brain in vivo. *Metabolism of the nervous system*, pages 221–237, 1957.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A. G. Kirk, J. F. O. Brien, and D. Forsyth. Skeletal parameter estimation from optical motion capture data. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 782–788. IEEE, 2005.

E. I. Knudsen. Fundamental components of attention. *Annu. Rev. Neurosci.*, 30:57–78, 2007.

J. Kober, B. Mohler, and J. Peters. Learning perceptual coupling for motor primitives. In *Proc. IEEE Intl. Conf. on Intelligent Robots and Systems*, pages 834–839. IEEE, 2008.

M. F. Land and M. M. Hayhoe. In what ways do eye movements contribute to everyday activities? *Vision Research*, 41(25):3559–3565, 2001.

M. F. Land and P. McLeod. From eye movements to actions: how batsmen hit the ball. *Nature neuroscience*, 3(12):1340–1345, 2000.

S. Lange and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *Proc. IEEE Intl. Joint Conf. on Neural Networks*, pages 1–8. IEEE, 2010.

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

S. B. Laughlin, R. R. de R. van Steveninck, and J. C. Anderson. The metabolic cost of neural information. *Nature neuroscience*, 1(1):36–41, 1998.

Q. V. Le, A. Karpenko, J. Ngiam, and A. Y. Ng. ICA with reconstruction cost for efficient overcomplete feature learning. In *Advances in Neural Information Processing Systems*, volume 23, 2011.

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

H. Lee, Y. Largman, P. Pham, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in Neural Information Processing Systems*, 22:1096–1104, 2010.

A. M. Liberman and I. G. Mattingly. The motor theory of speech perception revised. *Cognition*, 21(1):1–36, 1985.

A. M. Liberman, F. S. Cooper, D. P. Shankweiler, and M. Studdert-Kennedy. Perception of the speech code. *Psychological Review*, 74(6):431, 1967. ISSN 1939-1471.

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

D. M. MacKay. Towards an information-flow model of human behaviour. *British Journal of Psychology*, 47(1):30–43, 1956.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, page 14. California, USA, 1967.

S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

K. L. Markey. *The sensorimotor foundations of phonology: a computational model of early childhood articulatory and phonetic development*. PhD thesis, University of Colorado at Boulder, 1994.

J. Martens. Deep learning via hessian-free optimization. In *Proc. 27th Intl. Conf. on Machine Learning (ICML)*, volume 951, 2010.

J. Martens and I. Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proc. 28th Int. Conf. on Machine Learning (ICML)*, 2011.

D. W. Massaro and T. H. Chen. The motor theory of speech perception revisited. *Psychonomic Bulletin & Review*, 15(2):453, 2008.

J. S. Matthis and B. R. Fajen. Humans exploit the biomechanics of bipedal gait during visually guided walking over complex terrain. *Proceedings of the Royal Society of London B: Biological Sciences*, 280(1762):20130700, 2013.

R. Memisevic. On multi-view feature learning. In *Proc. Intl. Conf. on Machine Learning*, 2012.

John G Milton. Intermittent motor control: The "drift-and-act" hypothesis. In *Progress in motor control*, pages 169–193. Springer, 2013.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

A. Mohamed, G. E. Dahl, and G. E. Hinton. Deep belief networks for phone recognition. In *Proc. NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

A. Mohamed, G. E. Dahl, and G. E. Hinton. Acoustic modeling using deep belief networks. *IEEE Trans. on Audio, Speech, and Language Processing*, (99):1–1, 2010a.

A. Mohamed, D. Yu, and L. Deng. Investigation of full-sequence training of deep belief networks for speech recognition. In *Proc. 11th Annual Conf. of the Intl. Speech Communication Association*, 2010b.

A. R. Mohamed and G. E. Hinton. Phone recognition using restricted boltzmann machines. In *Proc. IEEE Intl. Conf. on Acoustics Speech and Signal Processing*, pages 4354–4357. IEEE, 2010.

K. Muelling, J. Kober, and J. Peters. Learning table tennis with a mixture of motor primitives. In *Proc. IEEE-RAS Intl. Conf. on Humanoid Robots*, 2010.

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

W. L. Nelson. Physical principles for economies of skilled movements. *Biological cybernetics*, 46(2):135–147, 1983.

R. R. Neptune, D. J. Clark, and S. A. Kautz. Modular control of human walking: a simulation study. *Journal of biomechanics*, 42(9):1282–1287, 2009.

D. Nguyen and B. Widrow. The truck backer-upper: An example of self-learning in neural networks. In *Proc. IEEE Intl. Joint Conf. on Neural Networks*, pages 357–363. IEEE, 1989.

D. Nguyen-Tuong and J. R. Peters. Incremental sparsification for real-time online model learning. In *International Conference on Artificial Intelligence and Statistics*, pages 557–564, 2010.

J. Nunez-Varela, B. Ravindran, and J. Wyatt. Gaze allocation analysis for a visually guided manipulation task. *Proc. From Animals to Animats*, pages 44–53, 2012.

A. O'Dwyer. *Handbook of PI and PID controller tuning rules*, volume 2. World Scientific, 2006.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583): 607–609, 1996. ISSN 0028-0836.

P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

P. Perrier, J. Perkell, Y. Payan, M. Zandipour, F. Guenther, and A. Khalighi. Degrees of freedom of tongue movements in speech may be constrained by biomechanics. *arXiv preprint arXiv:0709.1405*, 2007.

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

F. Pulvermüller, M. Huss, F. Kherif, F. Moscoso del Prado Martin, O. Hauk, and Y. Shtyrov. Motor cortex maps articulatory features of speech sounds. *Proc. National Academy of Sciences*, 103(20):7865–7870, 2006. doi: 10.1073/pnas.0509989103.

L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

Marc H Raibert. *Legged robots that balance*. MIT press, 1986.

Marc H Raibert and Jessica K Hodgins. Animation of dynamic legged locomotion. In *ACM SIGGRAPH Computer Graphics*, volume 25, pages 349–358. ACM, 1991.

M. A. Ranzato, Y. L. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. *Advances in Neural Information Processing Systems (NIPS)*, 20:1185–1192, 2007.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005. ISBN 026218253X.

R. A. Reale and T. J. Imig. Tonotopic organization in auditory cortex of the cat. *Journal of Comparative Neurology*, 192(2):265–291, 1980.

G. Rizzolatti, L. Fadiga, V. Gallese, and L. Fogassi. Premotor cortex and the recognition of motor actions. *Cognitive brain research*, 3(2):131–141, 1996.

G. Rizzolatti, L. Fadiga, L. Fogassi, and V. Gallese. The space around us. *Science*, 277(5323):190–191, 1997.

C. A. Rothkopf and D. H. Ballard. Modular inverse reinforcement learning for visuomotor behavior. *Biological Cybernetics*, 107(4):477–490, 2013.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323, 2000.

T. D. Sanger. Human arm movements described by a low-dimensional superposition of principal components. *The Journal of Neuroscience*, 20(3): 1066–1072, 2000.

S. Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

S. Schaal, S. Kotosaka, and D. Sternad. Nonlinear dynamical systems as movement primitives. In *IEEE International Conference on Humanoid Robotics*, pages 1–11, 2000.

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

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

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

S. H. Scott. Optimal feedback control and the neural basis of volitional motor control. *Nature Reviews Neuroscience*, 5(7):532–546, 2004.

J. W. Senders. *Visual scanning processes*. Drukkerij Neo Print, Soest, Netherlands, 1980.

C. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(7):379–423, 1948.

K. Sims. Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372, 1994.

J. C. Skou. The influence of some cations on an adenosine triphosphatase from peripheral nerves. *Biochimica et biophysica acta*, 23:394–401, 1957.

E. C. Smith and M. S. Lewicki. Efficient auditory coding. *Nature*, 439(7079): 978–982, 2006. ISSN 0028-0836.

N. Sprague and D. H. Ballard. Eye movements for reward maximization. *Advances in Neural Information Processing Systems*, 16, 2003.

N. Srivastava and R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2231–2239, 2012.

K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Real-time neuroevolution in the nero video game. *IEEE Trans. on Evolutionary Computation*, 9(6): 653–668, 2005.

B. T. Sullivan, L. Johnson, D. H. Ballard, and M. M. Hayhoe. A modular reinforcement learning model for human visuomotor behavior in a driving task. In *Proc. AISB Symposium*, pages 33–40. Proceedings of the AISB 2011 Symposium, 2011.

B. T. Sullivan, L. Johnson, C. A. Rothkopf, D. H. Ballard, and M. M. Hayhoe. The role of uncertainty and reward on eye movements in a virtual driving task. *Journal of Vision*, 12(13), 2012.

I. Sutskever. *Training Recurrent Neural Networks*. PhD thesis, University of Toronto, 2013.

I. Sutskever, J. Martens, and G. E. Hinton. Generating text with recurrent neural networks. In *Proc. 28th Intl. Conf. on Machine Learning*, pages 1017–1024, 2011.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

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

C.-H. Tan, J.-H. Hou, and L.-P. Chau. Human motion capture data recovery using trajectory-based matrix completion. *Electronics Letters*, 49(12):752–754, 2013.

B. W. Tatler and B. T. Vincent. The prominence of behavioural biases in eye guidance. *Visual Cognition*, 17(6-7):1029–1054, 2009.

G. W. Taylor. *Composable, distributed-state models for high-dimensional time series*. PhD thesis, University of Toronto, 2009.

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

J. B. Tenenbaum, V. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319, 2000.

G. Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

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

M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2):443–482, 1999a.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999b.

E. Todorov. Optimality principles in sensorimotor control. *Nature neuroscience*, 7(9):907, 2004.

E. Todorov and M. I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature neuroscience*, 5(11):1226–1235, 2002.

M. T. Turvey and S Fonseca. Nature of motor control: Perspectives and issues. In *Progress in Motor Control*, pages 93–123. Springer, 2009.

D. Urieli, P. MacAlpine, S. Kalyanakrishnan, Y. Bentor, and P. Stone. On optimizing interdependent skills: A case study in simulated 3d humanoid robot soccer. In *Proc. Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 769–776. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

F. E. Veldpaus, H. J. Woltring, and L. J. M. G. Dortmans. A least-squares algorithm for the equiform transformation from spatial marker co-ordinates. *Journal of Biomechanics*, 21(1):45–54, 1988.

S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

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

B. A. Wandell. *Foundations of Vision*. Sinauer Associates, 1995.

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

J. M. Wolfe, S. J. Butcher, C. Lee, and M. Hyle. Changing your mind: on the contributions of top-down and bottom-up guidance in visual search for

feature singletons. *Journal of Experimental Psychology: Human Perception and Performance*, 29(2):483, 2003.

D. M. Wolpert. Computational approaches to motor control. *Trends in cognitive sciences*, 1(6):209–216, 1997.

D. M. Wolpert, Z. Ghahramani, and M. I. Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

M. D. Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. W. Cottrell. SUN: A bayesian framework for saliency using natural statistics. *Journal of Vision*, 8(7), 2008.

Vita

Leif Morgan Johnson was born in Coeur d'Alene, Idaho and graduated from the North Carolina School of Science and Mathematics in 1997. He received the Bachelor of Science degree with honors in Computer Science, the Bachelor of Arts degree with honors in Multidisciplinary Studies, and the Bachelor of Science degree in Applied Mathematics from North Carolina State University in 2002. He worked as a software engineer in Graz, Austria, relocated to San Francisco in 2003, and completed a solo cross-country bicycle tour—Cape Hatteras, North Carolina to Cape Flattery, Washington—in the summer of 2004. He then joined Google as a software engineer and worked until March 2008, when he left to spend a few months working with Ruby on Rails at a small music-sharing startup. In August of 2008 he returned to academia to pursue the PhD degree at The University of Texas at Austin.

Permanent address: lmjohns3@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.