Learning and Inferring Representations of Data in Neural Networks

by

Jesse A. Livezey

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Physics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Michael R. DeWeese, Chair
Professor Professor Marjorie D. Shapiro
Professor Bruno A. Olshausen
Kristofer E. Bouchard, Ph.D.

Spring 2017

Learning and Inferring Representations of Data in Neural Networks

Abstract

Learning and Inferring Representations of Data in Neural Networks

by

Jesse A. Livezey

Doctor of Philosophy in Physics

University of California, Berkeley

Professor Michael R. DeWeese, Chair

Finding useful representations of data in order to facilitate scientific knowledge generation is a ubiquitous concept across disciplines. Until the development of machine learning and statistical methods with hidden or latent representations, useful representations of data were generated "by hand" through scientific modeling or simple measurement observations. Scientific models often make explicit the underlying structure of a system which generates the data we observe and measure. To test a model, inferences must be made about the free parameters and the distributions of latent or unmeasured variables in the model conditioned on the data collected. At this time, many scientific disciplines such as astronomy, particle physics, wildlife conservation, and neuroscience have been moving towards collecting datasets that are large and complex enough so that no human will ever look at and analyze all measurements by hand. Datasets of this scale present an interesting scientific opportunity: to be able to derive insight into the structure of natural systems by creating models which can adapt themselves to the latent structure of large amounts of data, often called data-driven hypothesis testing. The three topics of this work fall under this umbrella, but are largely independent research directions. First, we show how deep learning can be used to infer representations of neural data which can be used to find the limits of information content in sparsely sampled neural activity and applied to improving the performance of brain-computer interfaces. Second, we derive a circuit model for a network neurons which implements approximate inference in a probabilistic model given the biological constraint of neuron-local computations. Finally, we provide a theoretical and empirical analysis of a family of methods for learning linear representations which have low coherence (cosine-similarity) and show that linear methods have limited applicability as compared to nonlinear, recurrent models which solve the same problem. Together, these results provide insight into how scientists and the brain can learn useful representations of data in deep and single layer networks.

This work is dedicated to people in the Bay Area and around the world who are fighting against injustice and fighting for their communities, bodies, and self-determination.

Contents

List of Figures

List of Tables

Acknowledgments

This work was largely completed in Berkeley, California. This area was first settled by the Ohlone people and was later violently stolen from them [1, 2]. The Ohlone people continue the struggle against colonialism today.

My parents and extended family are the main reason that I have been able to make it through multiple decades of education and scientific training. I am privileged to have parents who have always encouraged me to explore the things I am interested, often math and science. They have been both a safety net and a source of gentle encouragement. I also have a brother and sister who have been great siblings and role models and who are following in their older brother's footsteps in graduate programs as I write this. Beyond my interests in science, my family have also always taught me to value empathy, love, and care for others. This has had a large influence both on how I perceive my own role in science and how I allocate my time outside of "research" hours.

I also received an enormous amount of support from my partner, Sarah Maslov, over the course of my six years at Berkeley. This has included emotional support, sitting around doing taxes together, getting me out of the apartment to explore the Bay Area, and making sure our apartment stays habitable. I am grateful for the time and effort Sarah has put into our relationship.

The Berkeley Compass Project has been a space for me to explore the intersection of my interests in physics, education, and the more "human" parts of science. My fellow Compasseers have become my friends, extended family, mentors, and accomplices. From them I have learned to be a better scientist, teacher, mentor, and bureaucrat. I have also found a ton of support from members of Compass during times when I have struggled in grad school. I would like to thank everyone who has been a part of Compass and who continues to do the work.

I have been a member of Mike DeWeese's and Kris Bouchard's labs and have benefited from the diversity of views and mentoring styles from both labs. Having space to transform from a physicist to the strange multi-armed scientist I am now has been invaluable. I've had the opportunity to learn about neuroscience, information theory, parallel computing, and machine learning and I am grateful for those opportunities. Mike has been very supportive in allowing me to explore a variety of ideas and interests. Kris has helped me focus and refine a more narrow line of research. I have been very lucky to have a combination of these styles during my tenure at the Redwood Center.

I have been fortunate to be part of the Redwood Center in addition to the DeWeese and Bouchard labs. The Redwood PIs: Bruno Olshausen, Fritz Sommer, Tony Bell, Pentti Kanerva, and Mike DeWeese have create and a space where people from a diverse set of fields can come together and do interesting work. I would also like to thank all of the PIs, postdocs, research associates, and grad and undergrad students at the Redwood Center and in neighboring labs, especially Allie Fletcher; I've learned a lot from you all. I've also been fortunate to be able to work with a number of very hard working and creative undergrads. Our interactions and research have been enjoyable and productive.

There are collaborations with folks at UCSF and LBNL of which I have been a part. These collaboration have been a great opportunity for me to grow my perspective as a scientist.

Graduate and undergraduate students, staff, administrators, and faculty at Berkeley have put an immense amount of effort into equity and inclusion work. I am grateful for the work they have done and continue to do. There has been real leadership from my fellow physics and neuroscience students and other students at Berkeley.

In the physics department and on campus, there are a number of wonderful administrators and staff. In particular, Anne Takizama, Donna Sakima, Kathy Lee, Brian Underwood, and Claudia Trujillo do an enormous amount of work support graduate students. James Little, Roberto Barrueto-Cabello, Terry Buehler, Amin Jazaeri, and Ben Spike have also been very supportive of my and other's work in Compass. There are also more people behind the scenes that have made it possible for me to be successful, and I'm sure I have not appreciated them enough.

Finally, I would like to thank my thesis committee: Mike DeWeese, Kris Bouchard, Bruno Olshauen, and Marjorie Shapiro. Your feedback brought clarity to this document (and my thoughts).

Chapter 1

Introduction

The core process for generating knowledge in science is the iteration of creating and testing scientific models. Current experimental evidence is combined with theoretical assumptions to create models. These models make quantitative predictions about the statistics of future measurements. When those measurements become available, we can then compare our models with the data and iterate on this process when there are discrepancies.

Typically, scientists choose models which assume that the complexity observed in the data is actually the results of a much simpler but stochastic process which may have a number of parameters and latent variables. The free parameters of the models could be fundamental constants of the universe, population-level decision making probabilities in economics, or properties of ion channels on neuron membranes. They are assumed to have the same value across data measurements. A process for generating data may also have latent variables. These are quantities that we expect to vary stochastically from measurement to measurement in some unobserved way like the type of particles initially produced in a proton-proton collision at the LHC, the specific preferences of an individual in a decision making study, or the state of neighboring neurons in the brain. These parameters and latent variables determine the hypothesis for how data is generated in the model. Once data is collected, the values and distributions of these free parameters and variables needs to be inferred. This process of inference is what allows us to reason about how well our model describes the structure of the data.

In some scientific domains, the generative process of the data can be specified in detail. For instance, in particle physics experiments, although it may not be known which theory best represents the physics of the universe, for any given theory, detailed simulations can be made for what signals to expect in a particle detector. In many biological domains, such as neuroscience, the theories are not detailed or constrained enough to simulate and generate realistic data. It is therefore often the case that models with a pre-specified broad hypothesis must be able to adapt to the details of the data being fit. For instance, dynamical systems models can be fit to neural data measured during the production of motor movements, but it might not be known *a priori* which parts of cortex are controlling which articulators and this needs to be inferred from the data.

The brain is a complex system which learns how to better condition behaviors on sensory stimuli and past experiences. There are a hierarchy of recurrently interconnected regions which exhibit both local and global structure. As additional complication, different brain regions in humans evolved at different times and can have significantly different structure and function. When neuroscientists record neural data, they are subsampling in space and time the complex nonlinear electrical and chemical activity in the brain. Data analysis methods and models for neural data need to be able to uncover the underlying structure from the subsampled data. This structure could be spatial organization across the surface of the brain, organization in time-frequency space, activity which corresponds to the representation of sensory input, or spatially coordinated temporal dynamics which generate motor movements.

In an analogous process, there is evidence that the brain is learning from the structure of the sensory input it receives; the brain is not completely genetically hardwired and so the brain must be learning from and adapting to the stimuli it receives. There is a body of work in theoretical neuroscience which compares the structures learned by probabilistic models of natural sensory data and neural representations of the same data. These hypotheses look for connections between learning techniques that we successfully apply to data in machine learning contexts and strategies the brain may be using. Typically, connections can be made to the learned representations of data themselves in models and cortex, the learning rules which lead to those representations, or both. Often times, the machine learning models that we compare neural processing to require calculations, storage, or information transmission that could not conceivably be happen in neural circuitry. Some examples are when learning rules require neurons to have access to all synaptic strengths in the entire network, when feedforward connections and feedback connections need to be precisely tuned with respect to each other, or when a long history of activity needs to be stored for learning [3, 4, 5]. To make details hypotheses about learning and inference happening in the brain, these models must be adapted or approximated to fit the constraints of biology. This often requires a more fundamental understanding of the original limitations of machine learning methods.

If we want to understand the mechanism for how this relationship between machine learning and learning in the brain might be implemented, we must better understand the limitations of the algorithms we believe the brain may be (approximately) using. Overcomplete linear representations appear in many parts of machine learning models. Deep networks, linear models, Kalman filters, and HMM-GMMs all have linear representations as components of more complex representations. When a representation is overcomplete, i.e. it has higher dimensionality than the dimensionality of its input, it can often become degenerate or coherent, where different elements of the representation code for nearly identical patterns. Collapsing into identical representations is not beneficial for a model, but it may also not be optimal to have completely orthogonal representations with no redundancy in noisy systems (like the brain may be). This trade off can be implicit in the structure of the model and it can also be controlled through additional model terms. Understanding the limitations and trade offs of machine learning models and potential solutions will allow better algorithms to be developed and also allow us to understand what algorithms the brain may be implementing.

The remainder of this chapter will introduce the content of the three main chapters of

this work (chapters 4, 5, and 6). All three chapters are centered around learning or inferring latent representations of data, although they do not build sequentially on each other. The underlying theme is that through (probabilistic) modeling of data, the structure of the process that generated the data can be better understood. Specifically, we would like to understand the computations the brain is doing. In neural data, we would like to extract complex, potentially nonlinear relationships between the neural observables and sensory processing or motor coordination. The structure inferred from data can then be compared with existing theories or can be used to generate new theories of the computations happening in the brain.

1.1 Summary of results

Deep learning for neural data analysis

In chapter 4, the latent representations will be used in a supervised deep network which will be used to classify speech from electrical signals recorded from the surface of human cortex. These representation are learned by training a deep network to learn a non-linear mapping between neural data and speech tokens. We will rely on the probabilistic interpretation of these networks to learn about the representations of speech in sensorimotor cortex.

Vocal articulation of speech requires the coordinated and continuous orchestration of several parts of the vocal tract. This coordinated articulatory movement should be reflected in the cortical dynamics which generate these movements and therefore the neural data measured during speech production.

We apply supervised deep networks to this dataset in order to understand, first, whether deep networks are well suited to the task of classifying speech tokens from neural data, i.e. do they have better classification accuracy than traditional methods, and second, once these networks are trained on neural data whether the learned representations can be used to understand the representations of speech in sensorimotor cortex. In other domains, deep learning is the state of the art method. Neural data is subsampled in space and time which means that the underlying structure is also subsampled. Given this subsampling, developing methods which can best extract structure from neural data is important for developing power models of neural data. This work is also motivated by the larger trend in neuroscience and science in general of increasingly large and complex datasets. Deep networks have the potential to leverage these huge datasets, but their current application in scientific domains is relatively limited.

We find that deep networks have improved classification accuracy as compared to traditional methods such as linear classifiers. This improvement seems to be largest when the linear classifiers initially have good performance. At the same time, the estimated improvement if more data was to be collected is higher for deep networks. These higher classification accuracies have potential clinical applications in brain-computer interfaces for people who have lost the ability to use their speech articulators. Higher classification accuracy implies a higher information transfer rate and higher words-per-minute for a brain-computer interface.

Finally, we find that the representations learned for classifying consonants and vowels also reveal the structure of motor coordination in ventral sensorimotor cortex. Phonetic features are represented hierarchically on the basis of articulatory features with major articulators at the top of the hierarchy, place of constriction in the vocal tract in the middle of the hierarchy, and manner of construction and vowel features at the bottom of the hierarchy. These results indicate that deep networks can not only provide state-of-the-art accuracy in classification from neural signals, but that they also can be used to understand the latent structure of neural data from single-trial noisy data.

Inference using a network of leaky integrator neurons with strictly neuron-local computation

Chapters 5 and 6 are both focused on specific implementations of linear generative probabilistic models where the data is assumed to be generated from a set of latent sources, S, which are mixed through a linear transformation, M, to form the data, X, as:

$$X_j^{(i)} = \sum_k M_{jk} S_k^{(i)}, \tag{1.1}$$

where the superscripts indicate elements of a batch and subscripts indicate data or latent dimensions.

Chapter 5 focuses on how a network of leaky integrate-and-fire neurons can perform *maximum a posteriori* inference in a sparse coding model. Many probabilistic models have neural interpretations. These models and interpretations often have different levels of biological plausibility. In sparse coding models, inference is typically not "neuron local" meaning that the computations that are happening inside of a neuron depend on quantities that an individual neuron would presumable not have access to: the synpatic strengths of the other neurons in the network. Previous work [3] has created a network of spiking neurons which can approximate the learning rules of sparse coding with neuron local learning rules. This work is extended by modifying the inference circuit so that it will follow the gradient of the energy function, a condition that could not be guaranteed with the previous network.

We first derive a modified inference circuit from the gradient of the SAILnet objective function and show that it performs more stable inference. Deriving the inference from the objective function ensures that the dynamics of the model will actually perform inference over the latent variables of the model. This new circuit has the same level of biological plausibility as the original SAILnet circuit, namely neuron-local computation. This derivation also predicts that the total conductance of a neuron should scale with the total strength of its dendritic synaptic strengths. These analyses lead to a new rate-coding network which can be simulated and whose inference can be compared to analytic results.

Overcomplete independent components analysis

Chapter 6 focuses on overcomplete Independent Components Analysis (ICA), which is a linear generative model of data where it is assumed that the latent sources can be recovered through a linear transformation. When the sources have higher dimensionality than the data, the models are called overcomplete and require methods to prevent multiple basis elements from becoming co-aligned.

In overcomplete representations, the higher density of bases in the space can lead to high-coherence solutions, which is quantified by the maximum absolute value of the off-diagonal elements of the Gram matrix of a dictionary [6], W:

$$\text{coherence} \equiv \max_{i \neq j} |\sum_k W_{ik} W_{jk}| = \max_{i \neq j} |\cos \theta_{ij}|. \tag{1.2}$$

This is a generally unwanted property for a representation learning algorithm since it means that multiple bases are coding for the same feature which means that the representation may be inefficient and possibly overfit to the distribution of features in the training data.

Several methods have been previously proposed in order to control the coherence of a learned representation [7, 8, 9, 10]. Our first result is a proof that the L_2 cost does not provide coherence control as it was intended to. We provide some insight and intuition as to why this method seems reasonable on its face, but actually has internal symmetries which prevent it from operating as coherence control.

Based on these observations, we suggest several new methods for coherence control and a modification of one existing method. Our suggested methods achieve high coherence on learned representations.

We test these methods on a task to recover the known structure in a toy dataset. We find that the new methods perform comparably and that several previously suggested methods to not perform well in as broad of a range of conditions. We find that in general, methods with linear inference, i.e. ICA, do not perform as well as sparse coding which has nonlinear, recurrent inference. This may suggest a computational necessity for the local, recurrent activity found in sensory areas in cortex.

Chapter 2

Methods: Machine learning and deep learning as models of data

2.1 Introduction

Supervised and unsupervised learning are two main topics in machine learning. In both cases, distributions generated from a model are compared with the empirical data distribution. Typically, the parameters of the model are learned by trying to minimize the divergence between the model distribution and the empirical data distribution with respect to the parameters of the model. Maximum-likelihood (ML) or approximate ML methods are commonly used [11, 12].

Once a model is learned or trained, it is often used to reason about the properties of new data [13, 14]. In supervised models, predictions on new unlabeled data can be made. These predictions could be used to tag or sort data in a preprocessing pipelines, communicate language through a brain-computer interface, or translate an acoustic signal into text [15, 16]. Unsupervised models can be used to make inferences about the latent causes for any given datapoint, generate new samples from the approximated data distribution, or group data into clusters based on the inferred structure [17, 18].

This chapter will provide background on the relevant tools used in machine learning and theoretical neuroscience for later chapters. Chapter 3 provides more detailed background for chapters 5 and 6.

Notation and naming conventions for variables and parameters in distributions and energy functions

Since this work will generally not take a Bayesian perspective on parameter estimation, we will differentiate between model variables which have statistical distributions and model parameters which will only have point-estimates by placing them on the left or right side of

semicolons respectively in joint or conditional distributions, e.g.

$$\hat{P}(x;\theta),\ \hat{P}(x|y;\theta) \tag{2.1}$$

where x and y are model variables and θ are model parameters. In a Bayesian model [19], the model parameters would also have prior and posterior distributions for their estimates rather than just point-estimates as in ML learning. Occasionally, the model parameters will be left implicit for brevity. Prior distributions are sometimes added as model regularization, but are commonly still used to make a point-*maximum a posteriori* estimate [12].

For this chapter it will be important to distinguish (empirical) data and model distributions. Distributions with hats will be model distributions and distributions without hats will be data distributions.

Depending on the context, including the background of the person you are talking to, the terms objective (function), energy function, error function, loss (function), cost (function), and negative log-likelihood can be used interchangeably. The term negative log-likelihood will be used when referring to training objectives for probabilistic models and objective function where an explicit probabilistic formulation is not needed.

2.2 Maximum-likelihood learning

In ML learning or estimation, the model probability of measuring a data point, x, is maximized. In practice, it is often a monotonic transformation of the likelihood, the log-likehood, which is maximized. Alternatively, the negative log-likelihood (*nll*) can be minimized. Minimizing the *nll* will be the convention used in this work. For an entire dataset of independent measurements, $x_i\ i \in \{1, \ldots, N\}$, the total likelihood is the product of the likelihoods of each individual measurement (the joint probability of independent variables is the product of their marginal probabilities). The average *nll* is also commonly used which is just the total *nll* divided by the dataset size.

Relationship between maximum-likelihood learning and the KL divergence

For fitting parameterized models to data, there is a well known relationship between the *nll* and the KL divergence. This relationship will be useful for understanding the relationship between cost functions used in machine learning and deep networks and maximum likelihood learning. The empirical data distribution for a dataset, $x_i\ i \in \{1, \ldots, N\}$, is a mean of delta function, where each data point is represented as a delta function, and can be written as

$$P(x) = \frac{1}{N}\sum_{i}^{N}\delta(x - x_i). \tag{2.2}$$

Formally, the average *nll* of the data under the model can be written as

$$\langle nll \rangle = -\frac{1}{N}\sum_{i}^{N} \log \hat{P}(x_i;\theta) \tag{2.3}$$

and the KL divergence between the empirical data distribution and model distribution is

$$\begin{aligned} D_{\mathrm{KL}}(P(x)||\hat{P}(x;\theta)) &= \int dx\ P(x)\log\left(\frac{P(x)}{\hat{P}(x;\theta)}\right) \\ &= \int dx\ P(x)\log(P(x)) - P(x)\log(\hat{P}(x;\theta)) \\ &= -\int dx\ \frac{1}{N}\sum_{i}^{N}\delta(x-x_i)\log(\hat{P}(x;\theta)) + \int dx\ P(x)\log(P(x)) \\ &= -\frac{1}{N}\sum_{i}^{N}\log\hat{P}(x_i;\theta) + \int dx\ P(x)\log(P(x)). \end{aligned} \tag{2.4}$$

The second term in the KL divergence does not depend on the parameters of the model and so it will have no effect on parameter estimation.

Maximum-likelihood learning for a Gaussian distribution

For example, if you have one dimensional data, historical temperature measurements for May 1st for instance, and want to model the distribution of measurements over the years as a Gaussian distribution, $\mathcal{N}(\mu,\sigma)$, the likelihood and *nll* for one data point, x, are respectively

$$\hat{P}(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(\mu-x)^2}{2\sigma^2}), \tag{2.5}$$

$$-\log\hat{P}(x;\mu,\sigma) = \frac{(\mu-x)^2}{2\sigma^2} + \frac{1}{2}\log(2\pi\sigma^2). \tag{2.6}$$

and so the total *nll* for the dataset is the sum of the *nll*s for the individual data points

$$nll(\mu,\sigma) = \sum_{i}^{N} -\log P(x_i;\mu,\sigma). \tag{2.7}$$

To estimate the parameters μ and σ that make the measured data most probable under the model, the *nll* can be minimzed with respect to μ and σ

$$\frac{\partial}{\partial\mu}nll(\mu,\sigma) = \sum_{i}^{N}\frac{\mu-x_i}{\sigma^2} = 0 \implies \hat{\mu} = \frac{1}{N}\sum_{i}^{N}x_i, \tag{2.8}$$

$$\frac{\partial}{\partial\sigma}nll(\mu,\sigma) = -\sum_{i}^{N}\frac{(\mu-x_i)^2}{\sigma^3} + \frac{1}{\sigma} = 0 \implies \hat{\sigma} = \sqrt{\frac{1}{N}\sum_{i}^{N}(\hat{\mu}-x_i)^2}. \tag{2.9}$$

In the case of a Gaussian distribution model of the data, the parameters of the model were able to be estimated in closed form. In many cases, it is not possible to explicitly solve for the optimal parameters of the model. In these cases, it is common to find the local minima of the function through iterative gradient descent on the parameter vector, θ

$$\theta_{t+1} = \theta_t - \epsilon \frac{\partial}{\partial \theta_t} nll(\theta). \tag{2.10}$$

Fig 2.1 shows the *nll* for a Gaussian model fit to one dimensional data drawn from $\mathcal{N}(0, .5)$. For three different σ values, the *nll* is plotted as a function of μ. The *nll* is smallest near $x = 0$ for all models and the optimal σ has the lowest minimum. The convexity of the *nll*s is related to the fact that a closed-form solution is available. Even if this was not available, the model parameters could be updated by performing gradient descent on the *nll* (walking down the *nll* curves). For more complex models, the *nll* may not be convex and may have a number of local minima, saddle points, or areas where the gradients are very small.

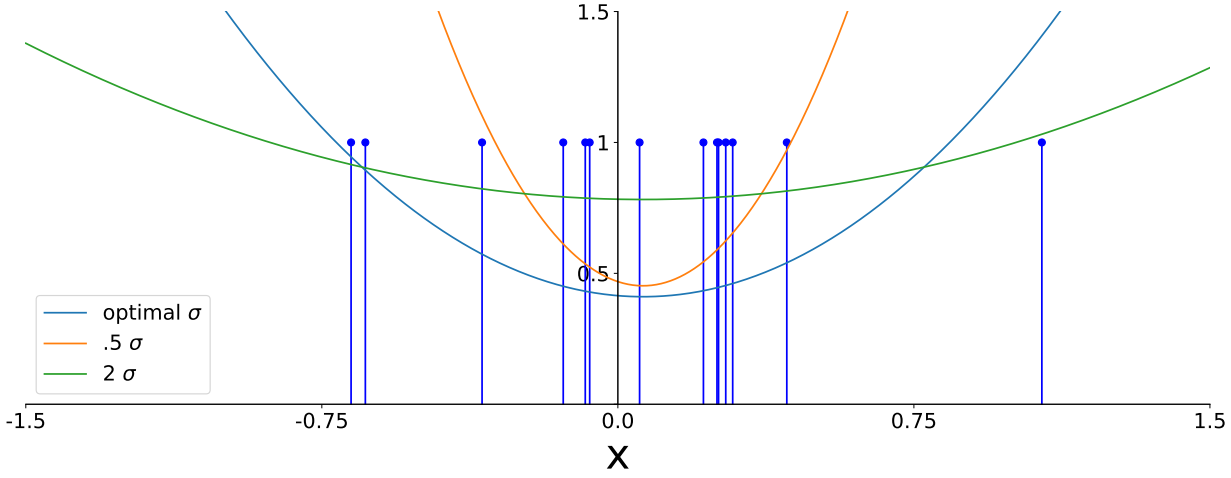

Figure 2.1: Plot of the *nll* of a Gaussian model as a function of μ, for three different σ parameters (equal to, smaller than, and larger than the optimal value). Blue pins are measurements of a one-dimensional variable, x, and the *nll*s have been consistently rescaled for clarity.

2.3 Supervised learning

The distinction between unsupervised and supervised learning is not always clear, but commonly the term supervised learning is used to describe the process of modeling conditional distributions of labels or tags with which the data has been annotated

$$\hat{P}(y|x;\theta) \sim P(y|x), \tag{2.11}$$

where y are commonly a set of labels, tags, or regression values associated with the data, x. Typically, supervised datasets need much more human curation than unsupervised datasets, e.g. labeling objects in images versus just collection images. The process of training supervised models is is the same as unsupervised models, the model's conditional distribution is matched to the data's empirical distribution using maximum likelihood learning (the KL divergence cost).

If the regression values, y, are real-valued vectors, a Gaussian loss is often used where the *nll* of the model is

$$nll(y_i, x_i, \theta) = \frac{(\mu(x_i, \theta) - y_i)^2}{2\sigma(x_i, \theta)^2} + \frac{1}{2}\log(2\pi\sigma(x_i, \theta)^2). \tag{2.12}$$

$\sigma(\cdot)$ is often assumed to be a constant function of the inputs and parameters which leads to the commonly used squared-error loss. If the regression values are labels or a class, the *nll* comes from the Bernoulli or Multinomial distributions

$$nll(y_i, x_i, \theta) = -\sum_i y_i \log(\hat{P}(y_i = 1|x_i; \theta)). \tag{2.13}$$

In these models, it is helpful to distinguish between the implicit quantities that are normally referred to as model parameters, μ and σ in a Gaussian model, the feature or class probabilities in a Bernoulli or Multinomial model, and the explicit parameters, θ, which are part of the network that generates the implicit parameters. In deep networks (or hierarchical probabilistic models), the implicit model parameters are often generated through a nonlinear process which is a (stochastic) function of other, often explicit, model parameters.

Fig 2.2 shows the *nll* for a Bernoulli model. A Bernoulli model and loss is often used to describe the ML model for a binary feature or label. Each feature can individually be present or not present in a element of the dataset. Each feature has one implicit parameter which is the probability that the datapoint contains the feature. If the feature is present, $y = 1$, then the *nll* peaks at when the feature is predicted with probability zero and decreases until y is 0. The opposite happens if the feature is not present in the data, $y = 0$.

For a classification (Multinomial) problem, the loss is similar, although the labels are constrained to be a *one-hot* binary vector where only one bit can be equal to 1 for any data point. The predicted class probabilities must be positive and sum to 1 and so a *softmax* nonlinearity is often used to satisfy these constraints

$$\text{softmax}(h_i) = \frac{\exp(h_i)}{\sum_j \exp(h_j)} \tag{2.14}$$

where h_i are proportional to the predicted log-probabilities up to a shared additive constant.

In the context of using classifying speech production, the data, x, is neural data. In this work, we will learn mappings between electrocortographic (ECoG) data and produced speech phonemes. The ECoG data will be timeseries data recorded from a number of channels. The data was recorded during the utterances of short speech tokens. Traditionally, this mapping

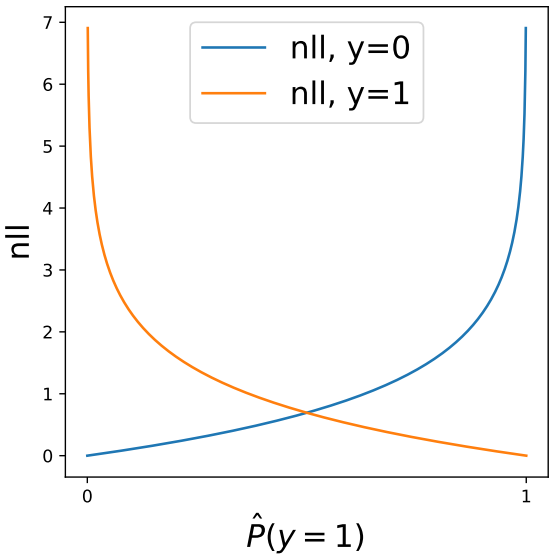

Figure 2.2: Plot of the *nll* of a Bernoulli model as a function of $\hat{P}(y=1)$. For any given instance of a feature and label, only one of these two *nll*s would be contributing to the loss and so the predictions are always pushed to be closer to 0 or 1.

has been done through training linear regression models like multinomial linear regression or linear discriminant analysis [16]. In these methods, the parameters of the implicit models are generated through a linear or affine transformation of the raw data. In this work, the affine transforms will be broadened to learned nonlinear transformations created by training supervised deep networks.

2.4 Supervised deep networks

Deep networks are a broad class of models which are commonly applied to problems where there are large datasets available, and hand-engineered features have not yet saturated performance. They can be used in unsupervised and supervised setting. This work will focus on supervised deep networks.

Supervised deep networks are highly parameterized models which learn mappings between their input data, x, and a set of regression features or labels, y. Input can be unstructured features vectors, data with topological organization like 2-dimensional images, or timeseries data like sound or neural signals. The mappings are constrained to be differentiable to allow the network parameters to be trained through gradient descent. The computations in a deep network are typically organized in a series of layers, where each layer does a small set of computations.

The structure of a layer

Most deep networks are built using a layered structure. A typical layer performs a chained affine-nonlinear transformation on its inputs, h^i, to create the outputs, h^{i+1}

$$h_J^{i+1} = f(\sum_k w_{jk} h_k^i + b_j) \tag{2.15}$$

where w is a matrix of weights, b is a vector of offsets, and $f(\cdot)$ is a nonlinearity. Common nonlinearities are rectification (ReLU), $\tanh(x)$, or sigmoid ($\frac{1}{1-e^{-x}}$) for internal layers of the network. The final layer nonlinearity is chosen to match the required output probability distribution: linear for a Gaussian mean, sigmoid for a Bernoulli feature probability, softmax for class probabilities.

For large images or timeseries, an affine convolution with a set of filters can be applied rather than a simple affine transformation [20, 21, 15, 22]. This will lower the computational complexity of the model (if the input and output dimensionality are kept fixed) since fewer multiplication and adds need to be done and will also result in parameters being shared spatially or temporally. In many datasets, e.g. images or recordings of speech, the statistics of the data are assumed to be stationary along the axes of convolution. In an image, any given pattern could appear at any location with equal probability and likewise in an audio recording any given sound could appear at any point in time.

Convolution can be used even when the statistics are not expected to be exactly stationary. This will decrease computational complexity, but may not provide the same amount of parameter sharing, since the filter activations may also not be stationary after training. In neural datasets, the subject is often performing a specific task. Many tasks to not have stationarity in their structure, and so the neural data may not be expected to have stationary statistics. Even with these non-stationary caveats, convolutions can often be an effective approach to data with topological structure.

To decrease dimensionality in convolutional networks, the convolutions can be strided, which will roughly decrease the output dimensionality by a factor of the stride, or a pooling layer can be applied. Striding a convolution is equivalent to downsampling the output by an integer fraction. In a pooling layer, a spatially local set of filter activations are compared and some 1-dimensional statistic is computed. Max pooling [23] is common where the largest activation. Mean pooling is also sometimes used [23].

Cost functions for deep classification

The cost functions used to train deep networks are typically the same as the ones used to train single-layer machine learning models. Deep networks can have squared-error or classification losses as in eqs. 2.12 and 2.13. Simple Bayesian priors are often used to regularize the parameters in the model, although estimation is generally a point-estimate, not a posterior distribution for parameters. These priors can be thought of as additional to the cost function of the model. This method of regularization and others will be covered later in section 2.4.

Training using backpropagation of errors

Deep networks typically have many parameters, often growing into the tens of millions or more. The dimensionality of the parameters places constraints on the types of algorithms which can be used to train them. For instance, memory requirements cannot be quadratic in the number of parameters, which means that most methods that require calculating the Hessian of the cost explicitly will not be able to scale to large problems.

These parameters are typically contained in stacks of convolutional filters of shape `(n_filters, n_channels, n_x, n_y)` with total size equal to the product of those dimensions, weight matrices of shape `(n_in, n_out)`, and biases are typically vectors of shape `(n_features)`. The cost function of a deep network will be a differentiable function of the layer parameters, w^i and b^i, and layer activations, h^i. We will need to compute the derivative with respect to the parameters in layer i, θ^i

$$\frac{\partial C(\theta^i)}{\partial \theta^i_j} = \sum_k \frac{\partial C(\theta^i, h^{i+1})}{\partial h^{i+1}_k} \frac{\partial h^{i+1}_k}{\partial \theta^i_j}, \tag{2.16}$$

$$\frac{\partial C(\theta^i)}{\partial h^{i+1}_j} = \sum_k \frac{\partial C(\theta^i, h^{i+1}, h^{i+2})}{\partial h^{i+2}_k} \frac{\partial h^{i+2}_k}{\partial h^{i+1}_j} \tag{2.17}$$

We can write out the derivatives with respect to any given layer's input activations or parameters using the chain rule as a function of the derivatives with respect to the activations of the layer. Intuitively the errors are being passes backward through the network which lead to the name "backpropagation" [24, 25, 26]. For an extensive history of the backpropagation algorithm in mathematics, physics, and machine learning, see [27] and citations within.

In this way, the derivative can be taken with respect to any of the layer activations or layer parameters. The deep network computation (forward-pass) and derivatives (backward-pass) are often represented as a directed computational graph where nodes represent computation and edges represent the flow of data. Fig 2.3 shows an example computational graph for the forward and backward passes of a deep network. This representation is very high level in that the entire computation that a layer does is represented as one node. In practice, there are subgraphs represented within each layer which represent matrix multiplication/convolution, bias addition, and nonlinearities as separate operations.

A naïve method for updating parameters from the gradients is to use basic gradient descent as in eq. 2.10. Basic gradient descent is generally very slow in that it requires many gradient steps to train the model to good performance and can get stuck near saddle points which are attractors [28]. It does not take into account that information in the history of gradients and update steps can be used to approximate second order gradient information. A number of methods, often inspired by techniques that have performance guarantees for convex problems, have been proposed and are commonly used. These include momentum [29] where the parameter state is treated as a massive particle with linear, velocity dependent damping with the cost function as a potential energy. These methods typically double or triple the memory requirements but often converge much faster.

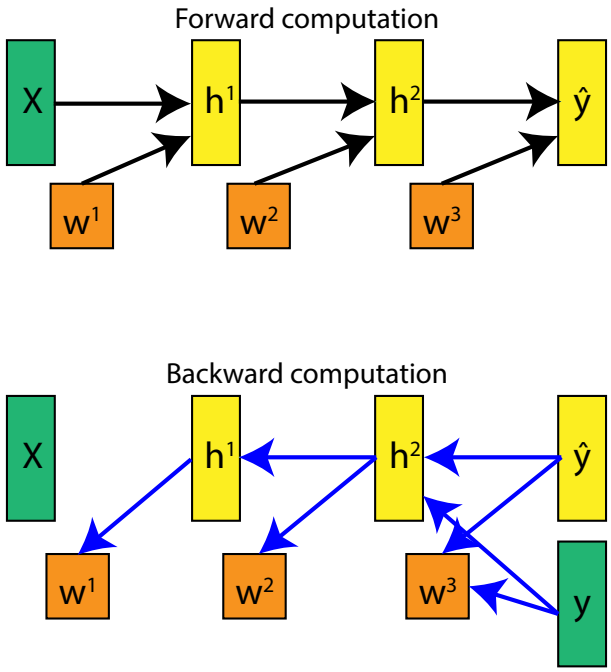

Figure 2.3: Computational graph representation for a layered deep network. Top panel shows the forward computation from left to right for computing the network's predictions. The bottom panel shows errors being passed backwards starting from a comparison of the network predictions, $\hat{y}$, and the true labels, y. Green nodes represents data entering the graph, orange nodes are network parameters, and yellow nodes represent the computations performed by the layers in the graph. Black and blue arrows represent the output of computations being passed between computational nodes.

Model regularization

The goal of training deep networks and machine learning models in general, is to create a model that can perform inference well on new, unseen data. When a model is trained on a fixed-size dataset, if the model has enough representational capacity (which is very easy to achieve with deep networks), it will eventually have better performance on the training data than it does on held-out validation data.

Fig 2.4 shows the dynamics of training with and without regularization. The first panel shows the training dynamics through the misclassifcation rate (lower is better) on the training, validation, and testing sets on a the MNIST dataset [30] where only a small subset ($n = 100$) of the training data are used to exaggerate the overfitting. The misclassification rate on the training set always approaches zero. With either L_2 regularization of the norm of the weight matrices or dropout the validation and testing misclassification rates are lower than an unregularized model.

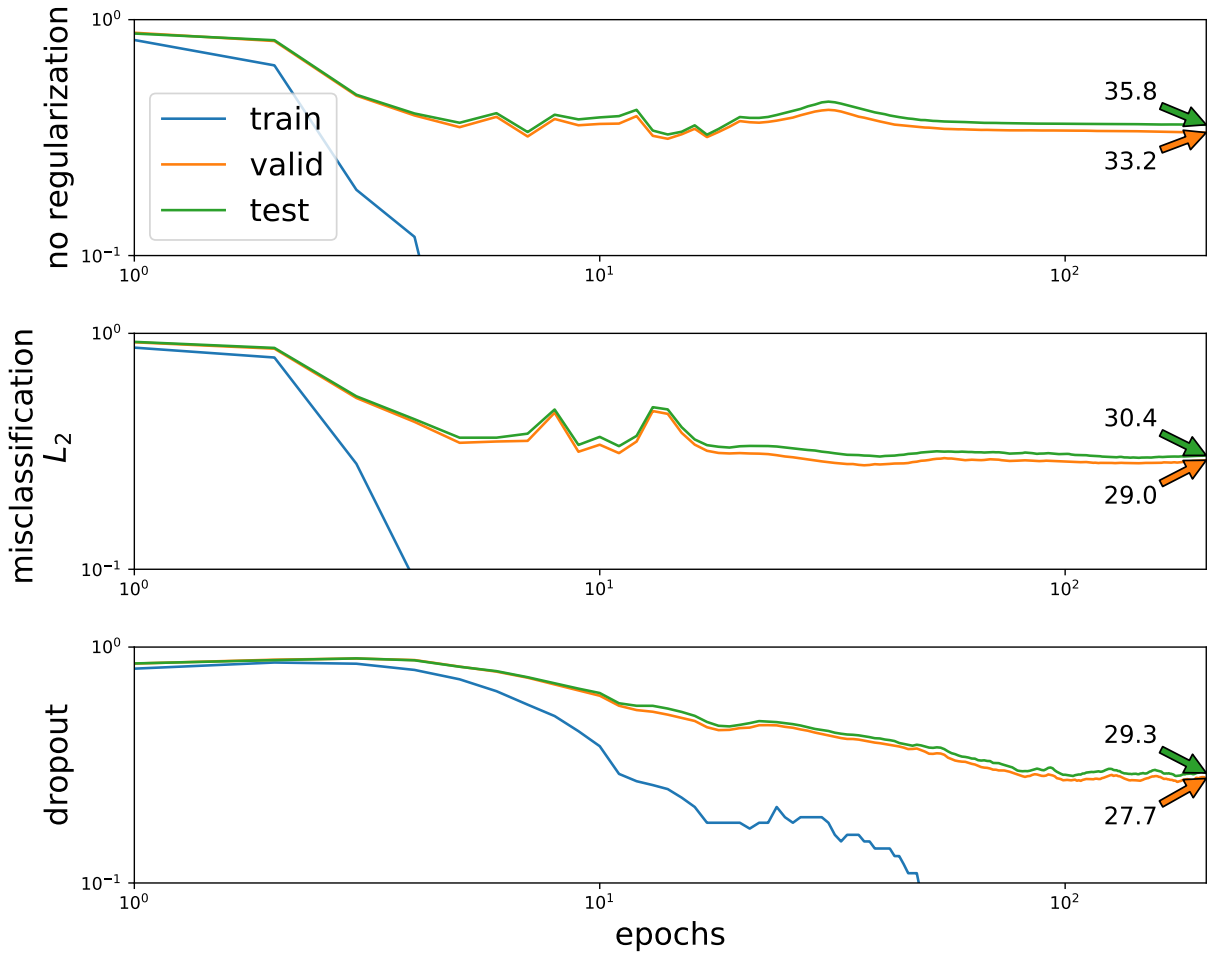

Figure 2.4: Plot of accuracy on the training, validation, and testing accuracy during model training on the MNIST dataset. Top panel shows training dynamics with no regularization, the middle panel shows training dynamics with L_2 norm regularization, and the bottom panel shows training dynamics with dropout.

L_2 regularization is implemented as an additional term in the training loss

$$\begin{aligned} C_{L_2}(W) &= \lambda \sum_{ij} W_{ij}^2 \\ P(W_{ij}) &\sim \exp(-\lambda W_{ij}^2) \end{aligned} \tag{2.18}$$

where λ sets the strength of the regularization. It can also be interpreted as a Gaussian prior on the elements of the weight matrix in a Bayesian framework. Dropout [31] is a regularization technique which is meant to combat one way in which a model might overfit: different parts of the model may learn to be dependent on other parts of the model co-activating. Dropout

randomly drops out (zero-masks) subsets of the inputs and model activation and rescales the remaining inputs and activations. This means that not every unit in a layer will be used for each training example and not every parameter will received a gradient. As can be seen by comparing the top and bottom panels of fig 2.4, dropout often makes training slower as it adds stochasticity to the training procedure.

For human neural datasets, the size of the dataset will almost always be a limiting factor is training models. Compounding this is the fact that neural datasets are typically high dimensional. For both of these reasons, model regularization will be a crucial part of training deep networks on neural data.

Hyperparameter selection methods

Deep networks, as described, have a number of parameters that need to be chosen to instantiate a model. A non-exhaustive list of potential parameters in a fully-connected network is tabulated in table 2.1.

Table 2.1: Network hyperparameters

Parameter	**Options**
Model hyperparameters	
number of layers	integer
dimensionality of each layer's output	integer
layer parameter initialization scheme	many options, e.g. [32, 33]
layer nonlinearity	ReLU, sigmoid, tanh, etc.
Training hyperparameters	
training algorithm	many options, e.g. [29, 34, 35]
training algorithm parameters	e.g. learning rate, decay rates
Regularization hyperparameters	
weight-norm regularization	positive coefficients
dropout parameters	dropout rate and rescale value

Cross validation is used to chose the optimal set of hyperparameters. Many models are trained with suggestions coming from either a random selection of hyperparameters [36] or a model-based hyperparameter estimation method [37, 38]. After many models are trained, the model with the best validation score is selected. The model scores on the test set are reported. This process is shown in a toy example in fig 2.5. In the left column, three models are fit to a one-dimensinoal dataset (blue pins represent a data point). Row A is a simple Gaussian model, row B is a complex, highly parameterized model, and row C shows a "parsimonious" model. Looking at the training data (left column) it seems that the simple model does not model the data well (lots of probability near zero with no data there), and the complex model

is a good fit to the data. When the models are evaluated on held-out data (right column). it becomes more clear that the parsimonious model captures the structure of the data best and the complex model has overfit to the details of the training data.

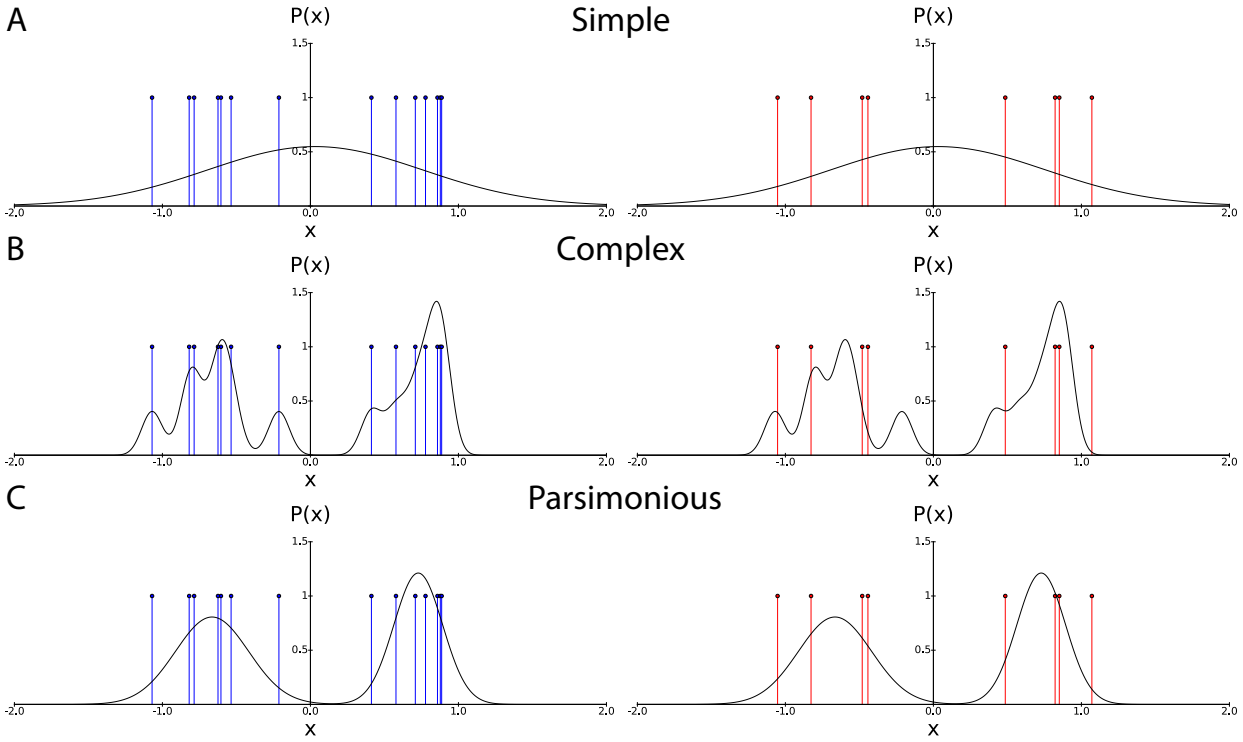

Figure 2.5: The cross validation procedure for model selection is schemetized here. The left column shows three different models fit to a one-dimensional dataset. The model pdf is shown as a line and the training data are the blue pins. **A** A Gaussian model of the data. **B** A seven-Gaussian mixture model. **C** A two-Gaussian mixture model. The right columns shows the model pdfs against held-out validation data.

2.5 Unsupervised learning

The goal of unsupervised learning is to understand the distribution of data in a dataset. Often this means trying to design models, $\hat{P}(X;\theta)$, which can model the distribution of data, $P(X)$ as closely as possible:

$$\hat{P}(x;\theta) \sim P(x). \tag{2.19}$$

Often the structure of the latent distributions are constrained in these models in order to have interpretable latent or unobserved variables. In models with latent (or hidden) variables, h, we define a larger model with latent and observed variables which is related to the model

distribution of the data through marginalization

$$\hat{P}(x;\theta) = \sum_h \hat{P}(x,h;\theta). \tag{2.20}$$

To train this model, the derivative of the *nll* must be taken, but usually it is not possible to analytically marginalize out h. We are left with

$$\begin{aligned}
\frac{\partial}{\partial\theta} nll =& \frac{\partial}{\partial\theta} - \log \sum_h \hat{P}(x,h;\theta) \\
=& - \frac{\sum_h \frac{\partial}{\partial\theta}\hat{P}(x,h;\theta)}{\sum_h \hat{P}(x,h;\theta)} \\
=& - \frac{\sum_h \frac{\partial}{\partial\theta}\exp(\log(\hat{P}(x,h;\theta)))}{\hat{P}(x;\theta)} \\
=& - \frac{\sum_h \hat{P}(x,h;\theta)\frac{\partial \log(\hat{P}(x,h;\theta))}{\partial\theta}}{\hat{P}(x;\theta)} \\
=& - \sum_h \hat{P}(h|x;\theta)\frac{\partial \log(\hat{P}(x,h;\theta))}{\partial\theta},
\end{aligned} \tag{2.21}$$

which is the expectation over the posterior distribution of the latent variables, $\hat{P}(h|x;\theta)$, of the derivative with respect to θ of the log-probability of the model. Computing the posterior distribution of the latent variables is often intractable given that it requires the marginalization of the joint distribution.

Many models of this form are created by specifying a prior distribution on the latent variables and a conditional distribution of data conditioned on the latent variables.

$$\hat{P}(x,h;\theta) = \hat{P}(x|h;\theta)\hat{P}(h;\theta). \tag{2.22}$$

In this case, the posterior is

$$\hat{P}(h|x;\theta) = \frac{\hat{P}(x,h;\theta)}{\sum_h \hat{P}(x,h;\theta)}, \tag{2.23}$$

and the denominator cannot be computed.

Many of the distributions used in practice are chosen because they can be written in a form which is equivalent to a Boltzmann distribution

$$\hat{P}(x;\theta) = \frac{\exp(-E(x;\theta))}{Z(\theta)} \tag{2.24}$$

which makes evaluating the derivative in the last line of eq. 2.21 straightforward.

This integral can be estimated or approximated in several ways. Sampling from the posterior distribution [39] can be used to estimate the value of the integral. There are efficient

Hamiltonian Monte Carlo samplers that can be used if the probabilities can be written in the form of eq. 2.24 [40]. Alternatively, Expectation Maximization can be used wherein the *nll* can be approximated with an estimate which is a upper bound to the *nll* and so minimizing this bound will minimize the *nll*. Neither of these techniques will be used in this work, but there is extensive literature available on both [41, 42].

Maximum a posteriori estimation

If the posterior distribution is dominated by modes, the final expectation in eq. 2.21 can be approximated by a point-estimate of the integral at the mode of the distribution. This technique is called *maximum a posteriori* or MAP estimation. To find the mode, posterior can be maximized or the negative log-posterior can be minimized with respect to the value of h. The location of the minimum of the negative log-posterior is

$$\begin{aligned} h^* &= \arg\min_h -\log \hat{P}(h|x;\theta) \\ &= \arg\min_h -\log(\hat{P}(x,h;\theta)) - \log(\sum_h \hat{P}(x,h;\theta)), \end{aligned} \tag{2.25}$$

which can be used to approximate

$$-\sum_h \hat{P}(h|x;\theta)\frac{\partial \log(\hat{P}(x,h;\theta))}{\partial\theta} \sim \frac{\partial \log(\hat{P}(x,h^*;\theta))}{\partial\theta}. \tag{2.26}$$

Since the second term does not depend on h, it can be dropped from the optimization in practice. The minima of this function are often found through numerical optimization.

This minimization can also be written to look like a dynamical circuit model [43], which has implications for how inference and possibly MAP inference could happen in a network of biological neurons.

Deep unsupervised models

As alternatives to structured models, many of the models which can currently generate the most convincing samples are deep networks which have latent variables which are generally hard to interpret such as generative adversarial networks, variational auto-encoders, and diffusion networks [44, 45, 18]. They will not be covered in this work.

2.6 Other types of machine learning

There are other types of machine learning models which are not covered under the umbrella of unsupervised and supervised learning. They will not be covered here. One example is reinforcement learning [46] where a model of an agent's interactions with the world is needed in addition to a model of the sensory input.

There are also machine learning models which do not use continuous gradient descent as a learning objective such as the original formulation of Hopfield networks or hyper-dimensional computing [47, 48].

Chapter 3

Methods: Linear generative models

3.1 Introduction

This chapter will extend the background methods presented in chapter 2 to a specific set of unsupervised probabilistic models: linear generative models. This will be the basis for chapters 5 and 6.

Many of the simplest models of sensory processing in theoretical neuroscience can be cast as linear generative probabilistic models. We follow the motivation and derivation of sparse coding and independent components analysis (ICA) from Olshausen [49]. In these models, the sensory stimuli, X, come from a set of hidden sources, S, which are mixed linearly. The sensory system may also have measurement noise, η added to the signal. This can be written as

$$X_i = \sum_j A_{ij} S_j + \varepsilon_i, \tag{3.1}$$

where A is a mixing matrix. To completely specify the model, it is necessary to specify a prior distribution on S and a distribution of the noise ε. In these types of models, it is conventional to call the latent variables sources S (activity, a, is another common convention) rather than h, but it is just a change in notation.

3.2 Biological implementations of MAP inference in sparse coding models

Sparse coding puts a sparse prior (independent, heavy-tailed) on the sources and a isotropic Gaussian distribution on the noise. One common formulation of sparse coding puts a Laplacian prior on the sources and a Gaussian distribution for the noise. We can write down the model

distribution as

$$\begin{aligned} P(X; A, \sigma, \lambda) &= \int dS P(X|S; A, \sigma) P(S; \lambda) \\ &= \frac{1}{Z(A, \sigma, \lambda)} \prod_i \int dS \exp\left(-\frac{(X_i - \sum_j A_{ij} S_j)^2}{2\sigma^2} \right) \prod_k \exp\left(-\lambda |S_k|\right). \end{aligned} \tag{3.2}$$

The posterior distribution of S is often required to do inference or learning. Since this is in general hard to compute, we approximate inference with MAP inference. So, rather than using $P(S|X)$, we use $\arg\min_S P(S|X)$. This is often done on the negative log posterior rather than on the posterior directly

$$\begin{aligned} \arg\min_S - \log P(S|X; A, \sigma, \lambda) &= - \log P(X, S; A, \sigma, \lambda) + \log P(X; A, \sigma, \lambda) \\ &= \sum_i \frac{(X_i - A_{ij} S_j)^2}{2\sigma^2} + \sum_k \lambda |S_k| \\ &\quad + \log(Z_{\text{joint}}(A, \sigma, \lambda)) + \log P(X; A, \sigma, \lambda). \end{aligned} \tag{3.3}$$

The process of inferring a set of latent variables from a stimulus, is a process that both machine learning models and biological systems need to do [50]. As described in section 2.5, machine learning systems have many options available to them for running inference. Biological systems are constrained to use (or, more accurately, have evolved to use) the neural substrate of the sensory system. If we take the idea that neural systems are performing probabilistic inference seriously, abstract computations must be mapped onto computations that a network of neurons could perform.

Luckily for the modeler, it is not always clear what the unit, timescale, or representation of computation is in the brain [51, 52]. This means that there is a large space of hypotheses to explore. One set of related hypotheses is that networks of sensory neurons in the brain are performing MAP inference on a model of the world in response to sensory stimuli. This computation can then be written in a form which could be implemented in a neural circuit and inspected for computation that may not be "biologically plausible". This will be the topic of chapter 5.

3.3 Constrained versus unconstrained optimization problems

Chapter 5 will compare two cost function: an unconstrained cost function and an approximation to this cost function with constraints. Here we will briefly describe the methods that will be used to minimize both cost functions.

An unconstrained objective function is simple to minimize through gradient descent, which was describes in chapter 2. As long as derivatives of the objective can be taken with

respect to the parameters of the model, the parameters can be updated to minimize the objective.

If the objective has an additional set of constraints, often written as $C_i(\hat{P}(X)) = 0$, then these terms can be added to the objective multiplied by their Lagrange multipliers, λ_i

$$\text{Constrained Objective} = \text{Unconstrained Objective} - \sum_i \lambda_i C_i(\hat{P}(X)). \tag{3.4}$$

We now want to minimize the unconstrained objective subject to the additional constraints. This can be done by updating both the model parameters through the gradient of the constrained objective with respect to the parameters and then updating the Lagrange multipliers through their gradients of the constrained objective with respect to themselves. This can be done in a "hard" way where at each step the gradients of the constrained objective with respect to the Lagrange multipliers are set exactly to zero, or through a "soft" gradient descent technique. The latter will be used in chapter 5.

3.4 (Overcomplete) Independent Components Analysis

Independent components analysis (ICA) models assume no measurement or reconstruction noise, i.e. they assume that the unobserved sources account for all of the variance in the measured data. For a model where the dimensionality of the data is the same as the dimensionality of the sources (complete), the linear generative process for ICA is

$$X_i = \sum_j A_{ij} S_j \tag{3.5}$$

which is the same as eq. 3.1 except for the lack of a noise term. A is assumed to be full rank and so the sources can be recovered from the data using the inverse of A

$$S_i = \sum_j W_{ij} X_j \tag{3.6}$$

with $W = A^{-1}$. This means that ICA has a very simple method for inferring the latent sources from data: a single linear transformation. It is often the case that we want to infer representations that have higher dimensionality than the data (overcomplete). In this case, the mixing matrix, A, is no longer square and cannot be inverted. Chapter 6 will cover existing overcomplete ICA models, provide a new theoretical understanding of aspects of the model, and suggestion new models and modifications to existing models to improve their properties. We will also show that the simple linear inference procedure of ICA is not as powerful as the iterative MAP estimate needed for sparse coding which provides insight into one reason why cortex, which is generally overcomplete in sensory regions, has recurrent connections.

Chapter 4

Deep learning for neural data analysis

Chapter coauthors: Kristofer E. Bouchard, Edward F. Chang

4.1 Introduction

Vocal articulation is a complex task requiring the coordinated and continuous orchestration of several parts of the vocal tract. To study the neural basis of speech requires monitoring cortical activity at high spatio-temporal resolution (on the order of tens of milliseconds) over large areas of sensorimotor cortex ($\sim$1300mm^2) [53]. To achieve the simultaneous high-resolution and broad coverage requirements in humans, electrocorticography (ECoG) is an ideal method for recording neural signals. Using such recordings, there has been a surge of recent efforts to understand the cortical basis of speech production [53, 54, 55, 56, 57]. For example, analyzing mean activity, Bouchard et. al. [53] demonstrated, much in the spirit of Penfield's earlier work [58], that the ventral sensorimotor cortex (vSMC) has a spatial map of articulator representations (i.e. lips, jaw, tongue, and larynx) that are engaged during speech production. Additionally, it was found that spatial patterns of activity across the vSMC network extracted with principal components analysis at specific time points organized phonemes along phonetic features emphasizing the articulatory requirements of production. Building off of these results, Bouchard and Chang [54, 55] demonstrated high-performance, single-trial continuous prediction of produced vowel acoustics from these recordings.

Many studies have attempted to classify produced speech from ECoG activity. For example, whole word studies have achieved classification performance just under 50% on a ten word corpus (5x over chance, 10%) [59]. However, classifying smaller phonetic parts of speech may better match the internal representations [53] and, in the context of brain machine interfaces for speech prosthetics, would exploit the inherent combinatorial nature of language. Pei et al. [60] used consonant-vowel-consonant (CVC) syllables and classified vowels or consonant pairs achieving 40.7 $\pm$ 2.7% for vowels and 40.6 $\pm$ 8.3% for consonant pairs (both 1.5x over chance, 25%) with four classes in each task. Herff et al. [61] use techniques from speech recognition and achieve phone accuracies above 50% (11x over chance, 4.7%),

but have incorporated a language model into their classifier unlike other studies. To our knowledge Mugler et al. [56] have achieved the best speech classification performance purely from ECoG recordings, with 36% accuracy on 24 consonants (5x over chance, 7.4%) and 24% on 15 vowels in a single subject (2x over chance, 12.9%).

The linear methods used in these works may not be able to fully extract the information present in the neural signals. This motivates the use of deep networks as decoders as they have surpassed or become competitive with previous state-of-the-art models in a number of fields including computer vision, text translation, and speech recognition [62, 63, 64, 65]. Deep neural networks have recently been applied as classifiers for other types of physiological data including electromyographic (EMG) and electroencephalographic (EEG) signals [66, 67, 68, 69] and on stimulus reconstruction in sensory regions using ECoG [70]. The goal of this work is to investigate deep networks as an alternative computing framework for brain machine interfaces/neural prosthetics, with a specific focus on the uniquely human capacity to produce spoken language, and to examine the structure of speech representations discovered by the networks towards understanding brain computations in general.

4.2 Materials and methods

Experimental data

The experimental protocol, collection, and processing of the data examined here have been described in detail previously [53, 54, 55]. Briefly, four native English speaking human subjects underwent chronic implantation of a subdural electrocortigraphic (ECoG) array over the left hemisphere as part of their clinical treatment of epilepsy. The subjects gave their written informed consent before the day of surgery. The subjects read aloud consonant-vowel syllables (CVs) composed of 19 consonants followed by one of three vowels (/a/, /i/ or /u/), for a total of 57 potential Consonant-Vowel (CV) syllables (not all subjects produced all CVs). The number of repetitions per CV varied across subjects and CVs from 10 to 50 and the total number of examples ranged from 1500 to 3500. All subjects had a small number of CVs which there was not enough data to do cross-validation on (< 10 examples) and these CVs were excluded per-subject.

Electrical field potentials were recorded directly from the cortical surface with a high-density (4mm pitch), 256-channel ECoG array and a multi-channel amplifier optically connected to a digital signal processor (Tucker-Davis Technologies [TDT], Alachua, FL). The time series from each channel was visually and quantitatively inspected for artifacts or excessive noise (typically 60 Hz line noise). These channels were excluded from all subsequent analysis and the raw recorded ECoG signal from the remaining channels were then common average referenced and used for spectro-temporal analysis. For each useable channel, the time-varying analytic amplitude was extracted from eight bandpass filters (Gaussian filters, logarithmically increasing center frequencies [70-150 Hz] and semi-logarithmically increasing band-widths) with the Hilbert transform. The high-gamma ($H\gamma$) activity was calculated by averaging the

analytic amplitude across these eight bands. This signal was down-sampled to 200 Hz and each channel was z-scored. Baseline is defined as a period of time in which the subjects were silent, the room was silent, and the subject was resting. Fig 4.1 shows z-scored data across trials (top) and the trial-average activity for one electrode (bottom). Based on the results described in [53, 54, 55], we focused on the electrodes in the ventral sensorimotor cortex (vSMC). The Hγ activity for each of the examples in our data set was aligned to the acoustic onset of the consonant-to-vowel transition. For each example, $\sim$ 1.3 seconds of data was extracted, giving 258 time points per example. The mean of the first and last ten time points was subtracted from the data per electrode and vocalization.

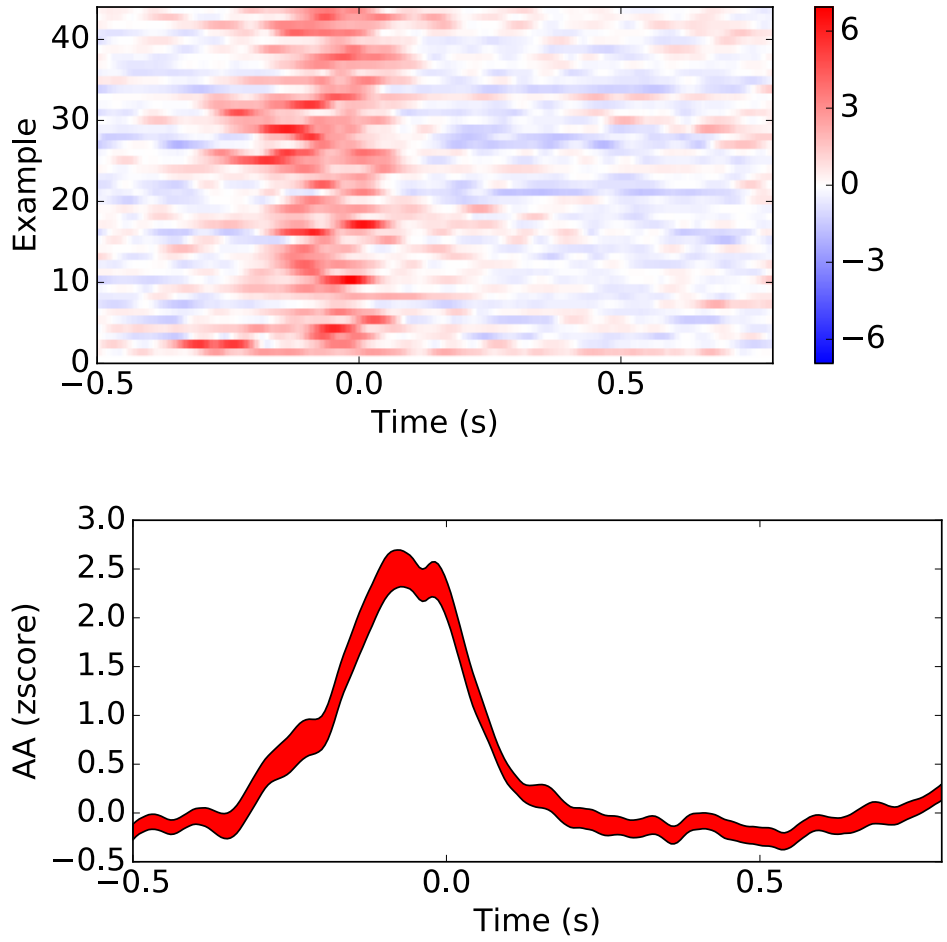

Figure 4.1: ECoG activity in vSMC during speech production. Top panel shows the z-scored, Hγ data across trials for one electrode during the production of /ga/ showing consistent response. Bottom panel is trial-averaged Hγ $\pm$ s.e.m.

Example trial-averaged Hγ ECoG data from Subject 1 for three CVs: /ba/, /da/ and /ga/, processed in the way described above, is shown in fig 4.2 from one subject. As described previously [53], different syllables are generated by distinct, but partially overlapping spatio-temporal patterns of activity. Furthermore, the Hγ activity of many electrodes and the consonant decoding accuracy (fig 4.2D and fig 4.3) begin rising several hundred milliseconds before acoustic onset, emphasizing the motor nature of the recordings.

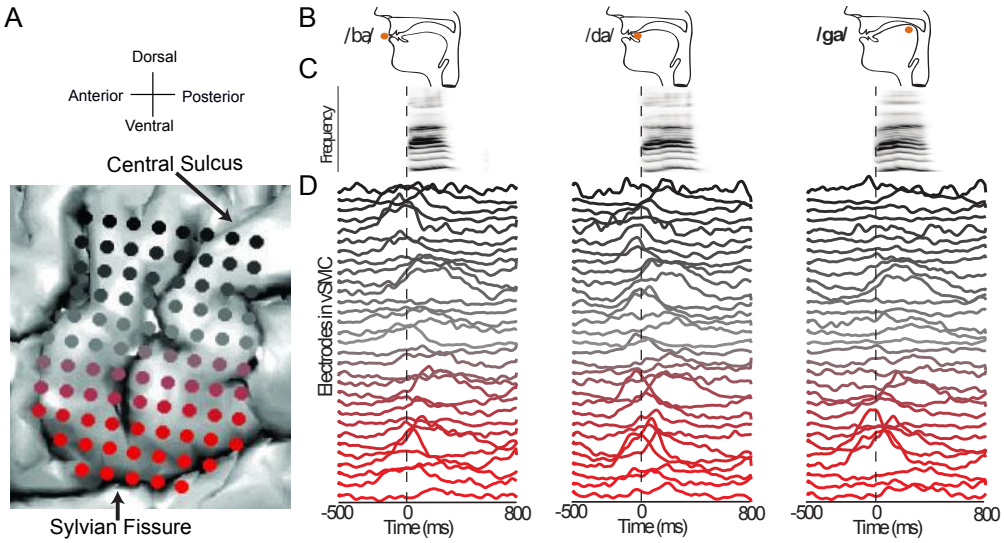

Figure 4.2: ECoG activity in vSMC during speech production. **A** Electrode layout overlaid on the vSMC. **B** Articulator position and point of constriction for the production of /ba/ (lips), /da/ (coronal tongue), and /ga/ (dorsal tongue). **C** The audio spectrogram aligned to CV acoustic transition (dotted line). **D** Mean high-gamma activity from electrodes in vSMC aligned to CV transition. Traces are colored red-to-black with increasing distance from the Sylvian fissure. The syllables /ba/, /da/, and /ga/ are generated by spatio-temporal patterns of activity across the vSMC.

A simple way to visualize the approximate time-courses for consonant and vowel representations is to train multinomial logistic regression models at each point in time to classify consonants or vowels. The results of this analysis is shown in fig 4.3. The light gray and red lines show the single subject consonant and vowel accuracies $\pm$ s.e.m. respectively for the four subjects. The black and red areas show the subject averaged results. The vertical dashed line show the time of the transition from consonant to vowel acoustics. Consonant classification accuracy rises about three hundred milliseconds prior to the vowel transition, before acoustics are produced for many consonants. Vowel classification accuracy rises after the transition. There is a period of time from just before the transition to about two hundred milliseconds after the transition where both the consonant and vowel can be classified, which is evidence of the coarticulation of the consonants and vowels which can be measured in the

acoustic and neural signals [54].

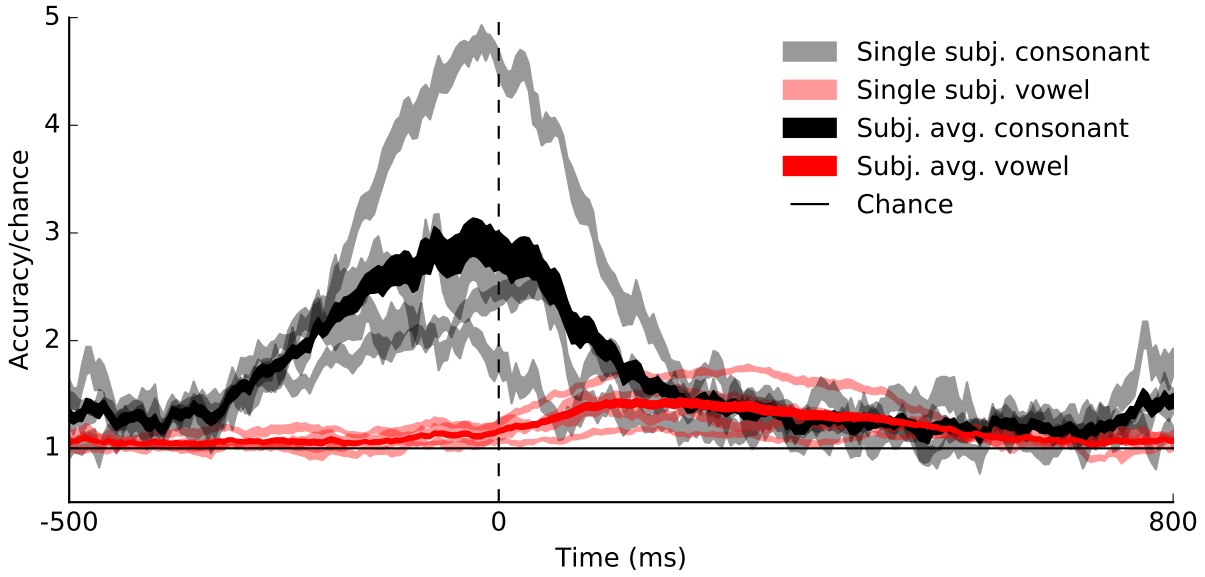

Figure 4.3: Accuracy of decoding consonants or vowels during production with multinomial logistic regression on time slices. Grey shaded areas are consonant accuracy ± s.e.m. for the 4 individual subjects. Light red shaded areas are vowel accuracy ± s.e.m. for the 4 individual subjects. Dark black and red areas are subject-averaged accuracies ± s.e.m.

Machine learning models

Deep networks

Deep networks have demonstrated state-of-the-art classification performance on a number of tasks [64, 71, 65]. They have also recently been used in neuroscience as state of the art encoding models of stimuli [72, 73, 74]. In this work, we evaluate deep networks as classifiers of produced speech from human ECoG. In addition to classifying CV syllables, we also consider classifying consonants and vowels as separate entities from the entire time series. Previous studies which have classified speech from ECoG have primarily used linear (single layer) models for classifying speech tokens. These models apply learned linear filters to the data, and are therefore limited in their ability to represent complex nonlinear relationships, which may be present in the neural data. Multilayer networks can combine features in nonlinear ways when making predictions. This gives them more expressive power in terms of the types of mappings they can learn at the cost of more model hyperparameters, more model parameters to train, and more difficult training dynamics.

Many machine learning methods can be cast as probabilistic models of data. Classification models can be interpreted as trying to find the model parameters, θ^*, which minimize the negative log-likelihood of the training data and labels, $\{x^{(i)}, y^{(i)}\}$, under a model which gives the conditional probability of the labels given the input data

$$\theta^* = \arg\min_{\theta} \; -\log P(Y|X;\theta), \; \{x^{(i)}, y^{(i)}\}. \tag{4.1}$$

Deep networks parametrize this conditional probability with a sequence of linear-nonlinear operations. Each layer in a fully connected network consists of an affine transform followed by a nonlinearity:

$$\begin{aligned} h_1 &= f(w_1 \cdot x + b_1), \\ h_i &= f(w_i \cdot h_{i-1} + b_i), \end{aligned} \tag{4.2}$$

where x is a batch of input vectors, w_i and b_i are trainable parameters (weights and biases, respectively) for the ith layer, h_i is the ith hidden representation, and $f(\cdot)$ is a nonlinearity. Linear networks are a special case of deep networks with no hidden representations. Convolutional networks were evaluated but did not provide better performance than full-connected networks.

For all network types, the final layer non-linearity is chosen to be the softmax function. This nonlinearity transforms a vector of real numbers into a vector which represents one draw from a multinomial distribution. It is the negative log-likelihood of this distribution under the training data which is minimized during training.

To train the networks, the data is randomly organized into 10 subsets (folds) with mutually exclusive test sets and 80-10-10% splits (training-validation-testing). Since the validation and testing sets may have fewer than 10 examples per class it was important to split each class proportionally so that all classes were equally distributed.

These networks have a number of hyper-parameters that govern network architecture and optimization, such as the number of layers, the layer nonlinearity, and the optimization parameters. A list of hyperparameters explored is given in table 4.1. For all results, hyper-parameters were selected by choosing the model with the best average validation classification accuracy across folds. Hyper-parameter search was done using the Spearmint library and random search [75, 36]. Deep networks perform best when large amounts of data are used for training. Since our corpus was relatively small for training deep networks, we regularized the models in three ways: dropout, weight decay, and filter norm-clipping in all layers of the model. The dropout rate, activation-rescaling factor, max filter norm, and weight decay coefficient were all optimized hyper-parameters. Pylearn2 [76] was used to train all models.

Linear models

As baseline models, we trained two linear classifiers on the data. One of the linear classifiers (softmax linear regression) was optimized using the same hyper-parameter selection method as

Table 4.1: Hyperparameters for fully connected networks

Name	Type	Range/Options
Init. Momentum	Float	.5
Terminate After No Improvement Epochs	Int	10
Num FC Layers	Int	1 to2
Layer dim.	Int	57 to 1000
Layer nonlinearity	Enum	Tanh, Sigmoid, ReLU
$\log_{10}$ Weight Init	Float	-5 to 0
$\log_{10}$ Learning Rate (LR)	Float	-3 to -1
$\log_{10}$ Min. LR	Float	-4 to -1
$\log_{10}$ One-minus LR Decay	Float	-5 to -1
$\log_{10}$ One-minus Final Momentum	Float	-4 to -.3
Momentum Saturation Epoch	Int	1 to 100
Batch Size	Int	100 to 512
Max Epochs	Int	10 to 100
One-minus Input Dropout Rate	Float	.1 to 1
Input Rescale	Float	.1 to 5
One-minus Default Dropout Rate	Float	.1 to 1
Default Rescale	Float	.1 to 5
$\log_{10}$ L2 Weight Decay	Float	-7 to 0
Max Filter Norm	Float	.1 to 5

the fully connected networks but was limited to having no hidden layers. The second was one-versus-all linear discriminant analysis (LDA) to compare deep networks with previous results. It is important to note that unsupervised dimensionality reduction is commonly applied before applying LDA. In our case, we have effectively applied 'dimensionality reduction' by choosing the Hγ band and the electrodes in the vSMC. Additionally, our goal was to evaluate methods with as little pre-processing as possible.

Performance metrics

Information transfer rate

To compare our results with previous speech classification studies, we report estimated information transfer rates (ITR) in addition to classification statistics. ITR is a unified way of calculating the effectiveness of different classifiers, which can have differing numbers of

classes, durations, and modalities. To calculate ITR, the symbol rate is multiplied by the information per symbol, defined as the channel capacity, C, between the ground truth class, Y, and predicted class, $\hat{Y}$:

$$C = \underset{P(Y)}{\text{supremum}}\, I(\hat{Y}; Y) = \underset{P(Y)}{\text{supremum}} \sum_{\hat{Y}_i} \sum_{Y_j} P(\hat{Y}_i|Y_j) P(Y_j) \log_2 \left(\frac{P(\hat{Y}_i|Y_j)}{P(\hat{Y}_i)} \right). \tag{4.3}$$

For previous work, we must approximate the channel capacity since we do not have access to the details of the classification performance, $P(\hat{Y}|Y)$. Wolpaw et. al. [77] suggest an approximation that assumes all classes have the same accuracy as the mean accuracy and all errors are distributed equally. To make a fair comparison, we compute this approximate value for our results (and so make fair comparisons) in addition to the exact value. We find that the approximation underestimates the true ITR.

Performance trend with training set size

We compared performance scaling of different models by training on variably sized fractions of the training set. For each fraction of the data, samples were randomly chosen so that the training set had roughly equal numbers of examples from each class. The validation and test splits were left the same size. Hyper-parameters were chosen independently for each fraction of the training data using an equal number of hyper-parameter optimization steps.

4.3 Classification results

For the models considered, we report overall classification accuracy and peak single phoneme classification accuracy for the best model. All models have classification accuracy significantly above chance ($p < 0.01$, Wilcoxon Signed Rank Test, 10 measurements from folds, chance calculated by training on data with shuffled labels). Additionally, we report consonant and vowel accuracy for multilayer networks. Furthermore, for increasing amounts of training data, we also show how performance scales for different models.

Deep networks achieve state-of-the-art classification accuracy

Categorical BMIs for speech rely on accurate classification of neural signals for robust and high-throughput communication. To this end, we study the ability of deep networks to classify ECoG signals as one of 57 CV syllables and phonemes (1 of 19 consonants or 1 of 3 vowels). We found that deep networks outperform a set of baseline models on all classification tasks examined here which is shown in fig 4.4 and tabulated in table 4.2. For the best subject, fully connected networks with one hidden layer gave the best performance, achieving 35.4 $\pm$ 2.5% accuracy (18x chance, 2.0%). The 4 subject average was 17.9 $\pm$ 10.3% accuracy (9x chance, 2.0%). For the best subject, classification accuracy on consonants and vowels, fully

connected networks achieved a prediction rate of 44.7 ± 2.1% (8x chance, 5.8%) and 70.0 ± 1.6% (2x chance, 32%) respectively. For the 4 subject average, classification accuracy on consonants and vowels was 27.3 ± 10.8% (5x chance, 5.8%) and 53.0 ± 10.9% (2x chance, 32%) respectively. The CV syllable with the best decoding performance is /ha/ with 72.0 ± 16.0% (7x chance, 11%) accuracy. Of the baseline models, softmax linear regression performs best, with 26.6 ± 2.0% (16x chance, 1.7%) accuracy on syllables. To make a close comparison to Mugler et. al. [56], we use LDA and perform consonant classification from the entire time series for the best subject. Using LDA, Mugler et. al. report 36.1% on a 24 consonant task (5x chance, 7.4%), while LDA on our data gives 26.9 ± 1.8% on a 19 consonant task (5x chance, 5.0%).

Table 4.2: Classification and ITR Results

Model	**Classification Accuracy**	**Folds/ Chance**	**Approx. ITR bps (Exact)**
Deep net, 57 CV, single subj.	35.4 ± 2.5%	18x	1.1 (3.0 exact)
Deep nets, 57 CV, subj. mean	17.9 ± 10.3%	9x	1.6 (.4 exact)
Linear Classifier, 57 CV, single subj.	24.8 ± 1.8%	12x	0.64 (2.6 exact)
Linear Classifier, 57 CV, subj. mean	13.6 ± 6.9%	7x	1.1 (3.0 exact)
LDA [56], 24 cons., single subj.	36.1%	5x	0.75
LDA [56], 24 cons., subj. mean	20.4 ± 9.8%	3x	0.25
LDA, 19 consonants	26.9 ± 1.8%	5x	0.4 (1.2 exact)
Deep net, Cons., 19 cons., single subj.	44.7 ± 2.1%	8x	0.9 (1.9 exact)
Deep nets, Cons., 19 cons., subj. mean	27.3 ± 10.8%	5x	0.4 (0.8 exact)
LDA [56], 15 vowels, single subj.	23.9%	2x	0.22
LDA [56], 15 vowels, subj. average	19.2 ± 3.7%	2x	0.12
Deep net, Vowel, 3 vowels, single subj.	70.0 ± 1.6%	2x	0.4 (0.4 exact)
Deep nets, Vowel, 3 vowels, subj. mean	53.0 ± 10.9%	2x	0.1 (0.1 exact)

Across subjects, performance varies likely due to the specific alignment of the grid of electrodes with relevant function areas and also. In general we find that higher performance with a linear classifier is correlated with larger improvements from deep networks. This is true for both the raw accuracy and the potential improvement with more data collection as we will show.

We additionally quantified performance by calculating the estimated information transmitted per symbol and per unit time. For the CV syllables, 35.4% classification accuracy would lead to a channel capable of transmitting 1.1 bits per syllable as reported in Table 4.2. Given typical syllable rates of 4 syllables per second [78], this could transmit up to 4.4 bits per second, which corresponds to 49 words per minute [79]. Together, these results demonstrate that both fully connected deep networks have state-of-the-art performance on

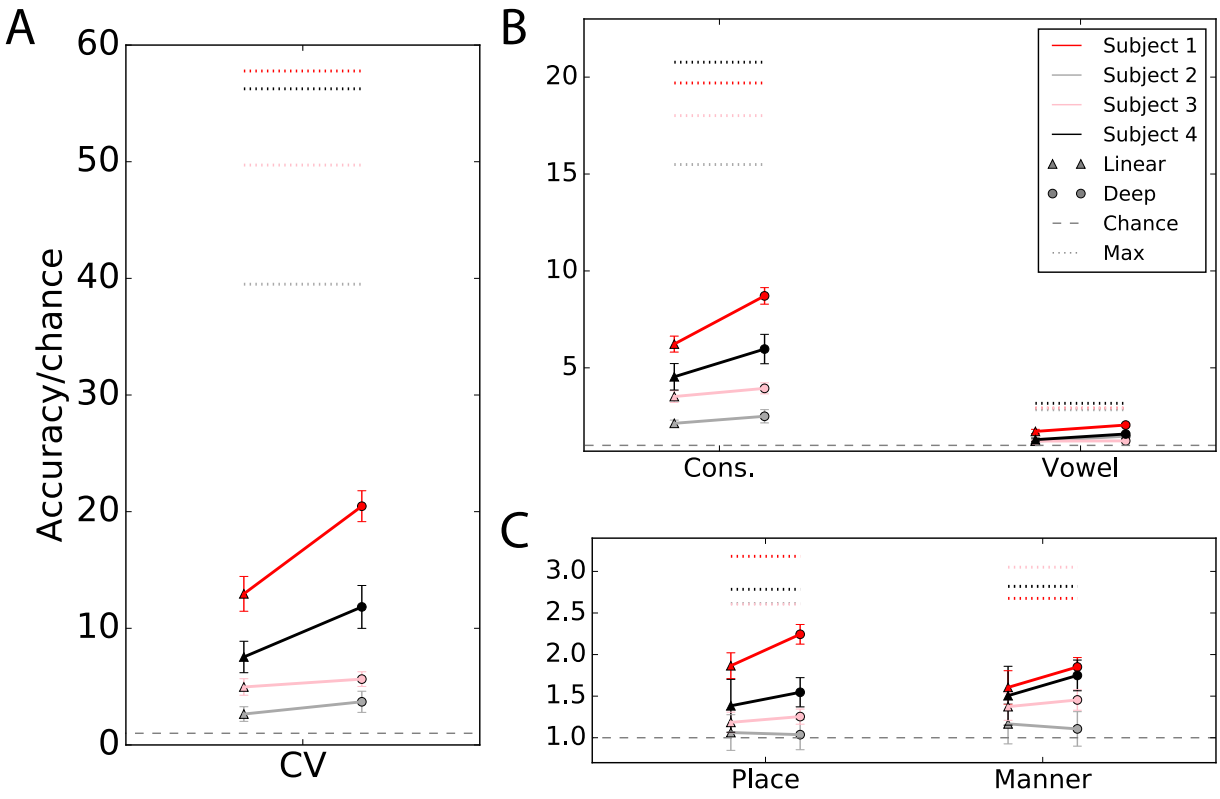

Figure 4.4: Linear versus deep accuracy comparison across tasks. Left column is accuracy of linear models and right column is the accuracy of deep models. Lines connect subjects from both categories. Accuracies are normalized to chance (chance at 1). Perfect classification for each subject is indicated by colored dotted lines. Differences in perfect performance are due to different distributions of CVs across subjects. **A** Linear versus deep accuracy comparison for the full CV task. **B** Linear versus deep accuracy comparison for consonant (19-way classification) and vowel (3-way classification) tasks. **C** Linear versus deep accuracy comparison for consonant place and consonant matter (both 3-way classification).

classifying produced speech from ECoG recordings. This strongly motivates the use of neural networks in BMIs to improve information transfer rates compared to previous methods.

Deep networks scale efficiently with limited data

Deep networks often out perform other classification methods when the amount of training data is large. In contrast to the data sets to which deep networks are commonly applied, which typically contain $\sim 10^7$ or more examples, most data sets in neuroscience are often limited to $\sim 10^2 - 10^3$ examples. In particular, since human ECoG data is only collected

in clinical situations with limited access to patients, knowing how these methods scale with increasing dataset size is important for future data collection, as well as their applicability to BMIs. To assess how classification performance of deep networks depended on the amount of available training data compared to other methods, we trained models on differently sized subsets of the data.

In Figure 4.5A, we plot the classification performance for three classifiers as a function of the number of samples included in the data set. While performance improves with increasing dataset size for all models, deep networks have the best scaling. The scaling of the fully connected networks: 11.0 $\pm$ 1.2% per thousand examples, is significantly better than linear models: 6.8 $\pm$ 1.0% per thousand examples ($p < 0.01$, t-test, 10 measurements from folds). Convolutional models did not have significantly better scaling with 7.4 $\pm$ 1.2% per thousand examples. This is summarized These results demonstrate that deep networks use limited data more efficiently than other methods, an important consideration when applying to neuroscience data.

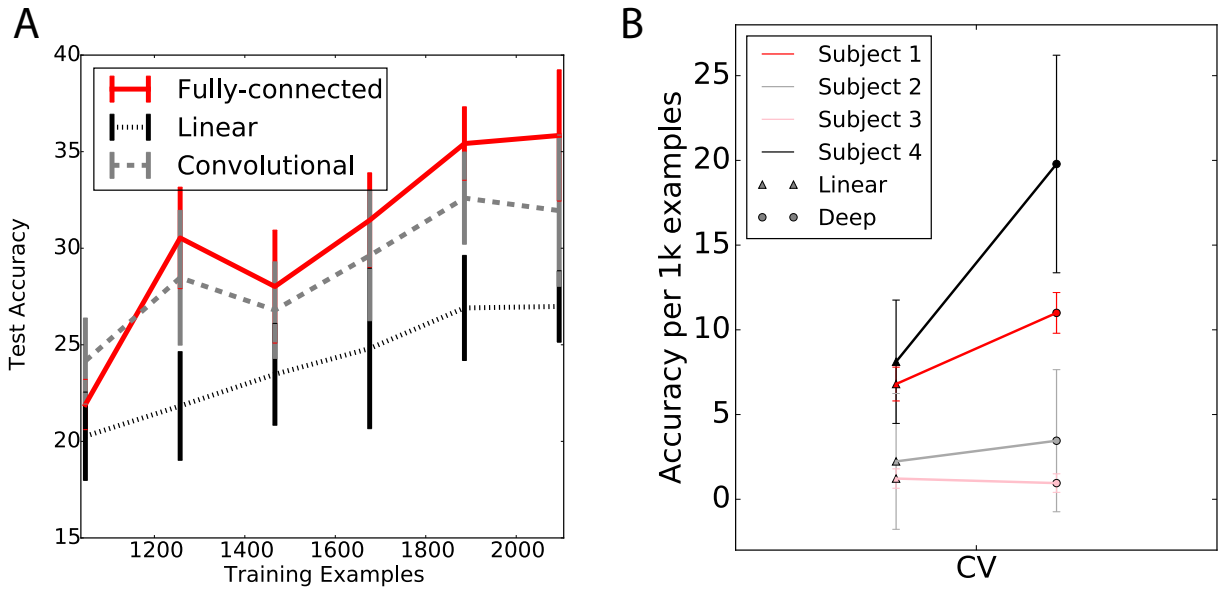

Figure 4.5: Accuracy as a function of dataset size. **A** For Subject 1, accuracy as a function of training dataset size. Performance of fully-connected, convolutional, and linear models are compared. **B** Comparison of slopes for linear versus deep models. Left column is slope of accuracy per one thousand data points for linear networks and the right columns is the slope for deep networks.

Fig 4.5B summarizes the estimated improvement with a larger dataset for 4 subjects. Again, we see that subjects with the best scaling from linear models have the best scaling

from deep networks. This correlation is useful for quickly gauging the likelihood of improved performance with deep networks from linear models which can be trained much faster.

4.4 Phonetic organization of deep network output

Brain computations are non-linear functions operating on spatio-temporal patterns of neural activity. However, most methods used to understand brain computation are linear, inherently limiting the capacity to extract structure from neural recordings. This is an issue not only for optimization of BCIs, but also for understanding brain computations. As deep networks are essentially adaptive bases function, non-linear function approximators, it is possible that they can extract structure from noisy, single-trial neural recordings that reveal important organization of representations.

We examined the structure of network output to more fully understand the organization of syllable representations in vSMC. In Figure 4.6, we show the average confusion matrix resulting from the output of the softmax layer of the fully connected network (i.e. before binary classification), with target syllables arranged along rows and predicted syllable across columns. The syllables are ordered according to the results of agglomerative hierarchical clustering using Ward's method. To the right is a bar-plot of the mean accuracy with which a specific syllable was correctly classified. Note that the syllable with worst accuracy is the one with the smallest number of examples in the dataset. At the highest level, syllables seem to be confused only within the articulator involved (lips, back tongue, or front tongue) in the syllable. This is followed by a characterization of the place of articulation within each articulator (bilabial, labio-dental, etc.). At the lowest level there seems to be a clustering across the vowel categories that capture the general shape of the vocal tract in producing the syllable. These results are in excellent agreement with the previous analysis of mean spatial patterns of activity at separate consonant and vowel time points [53]. In contrast, applying the same analysis to the predictions of a linear network only reproduces the top levels of the hierarchy. These results demonstrate the capacity of deep networks to reveal important structure in single-trial neural recordings that is not recoverable with other methods.

This hierarchy can be quantified by comparing this predicted deep network feature space with different phonetic/articulatory feature spaces. To do this we create two sets of pairwise distances and correlate them. The first is the pairwise Euclidean distances between each CV and every other CV in deep networks space. The second is the pairwise Hamming distances in phonetic feature space. These can be correlated across CV for each subject. The summary statistics are shown in fig 4.7. For consonant features, we see that the major articulator features are most correlated with the predicted network features, with consonant place and then manner as less correlated. Vowel features have small correlations which are similar in magnitude to manner. It is important to note that the deep network was not trained on phonetic labels, it only received consonant and vowel labels. Relationships between consonant structure was inferred from the data.

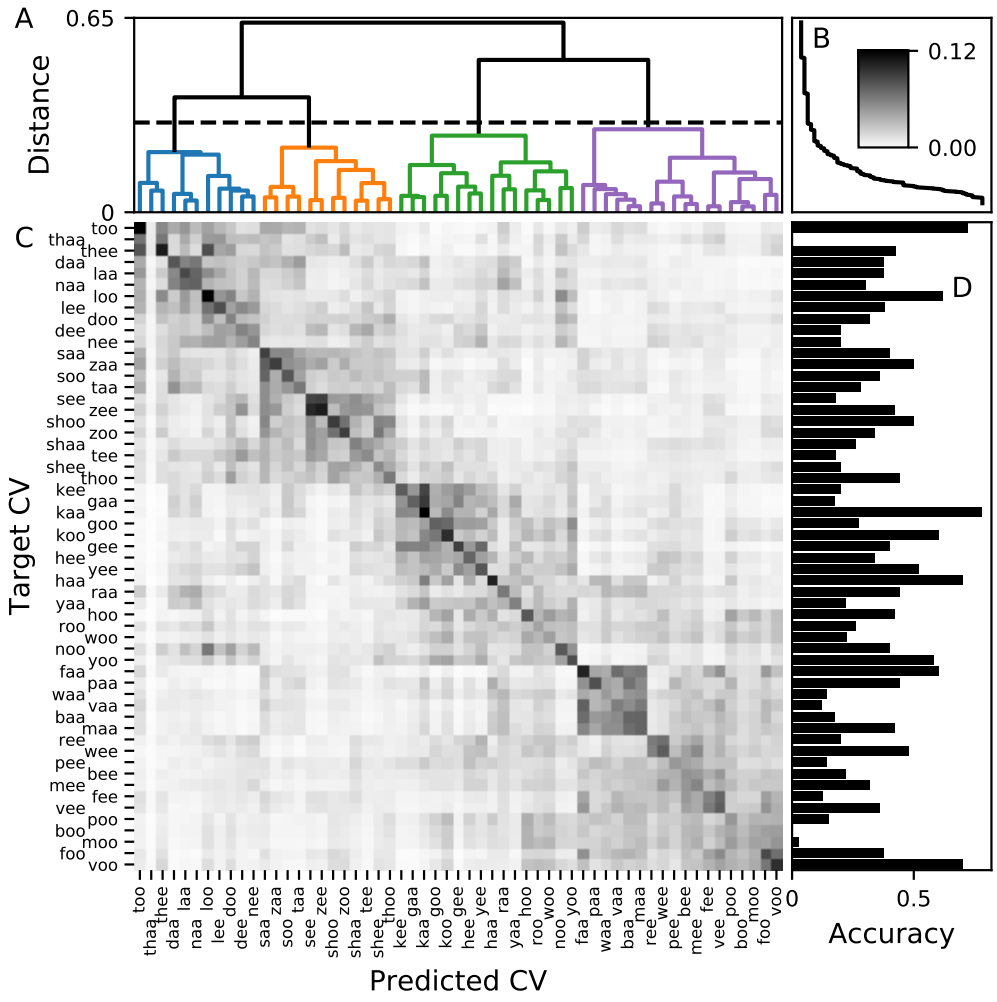

Figure 4.6: Analysis of the predictions of the deep network for the subject with the highest accuracy. **A** Dendrogram showing hierarchical clustering based on the softmax outputs. Blue nodes are largely coronal (front) tongue, orange nodes as sibilant, green dorsal (back) tongue, and purple are labial. **B** The trend of the number of discovered clusters as a function of minimum allowed distance between clusters. The dashed line represents the distance, 0.3, used for the clusters identified in panel **A**, which is in a relatively flat region of the number of clusters vs. threshold relationship. Inset is colorbar for **C**. **C** Trial-averaged prediction probabilities for all CVs. Network shows block-diagonal structure when ordered by clustering in **A**. **D** Mean accuracy per syllable across folds.

4.5 Discussion

This study is the first to use deep networks to classify produced speech from human sensorimotor cortex. Compared to other classification methods (LDA, logistic regression), we find

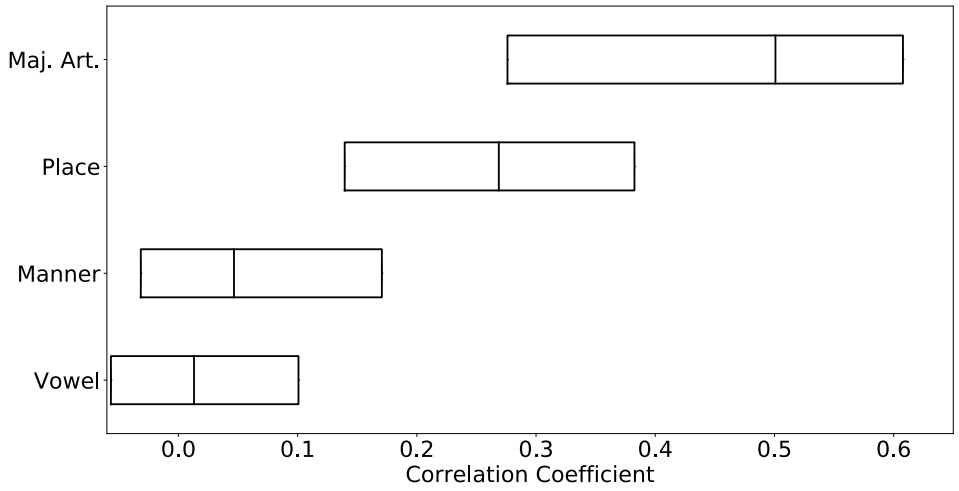

Figure 4.7: Comparison of the learned deep network feature space and different potential phonetic/articulatory feature spaces. For each category of features labeled on the left side of the axis (major articulator, consonant place, consonant manner, vowel) pairwise distances in the deep network space are correlated with pairwise distances in the feature space for each CV. The center line on each bar is the median correlation and the end bars are the 25th and 75th percentiles.

that deep networks classify syllables with state-of-the-art accuracy (∼35%) and push the boundary on information transfer rates for consonants, vowels and CV classification (49 words per minute). Fully connected architectures have the highest performance on this dataset, although convolutional networks also surpass previous methods. The success of these simple architectures encourages future work in applying recurrent networks to neural time series data. This could allow classification of variable length utterances which would not need to be hand-aligned. Beyond having the best current performance, deep networks exhibit the best performance scaling as a function of data set size, and are projected to keep improving with larger neural datasets. This is important in their practical application to most biological data sets, which have small sample sizes.

Outside of their utility as classifiers, deep networks can also provide insight into the latent structure of data. Since deep networks can learn continuous mappings to probabilities over classes, their outputs can contain implicit information about the relationship between data points. In fact, when hierarchical clustering was done on the CV confusion matrix from the softmax outputs of the network, we recover the phonetic organization of the spoken syllables, which is directly related to the articulators and shape of the vocal tract during speech production. This highlights the close relationship between neural signals in the vSMC and articulators that has also been seen in previous studies [53]. However, we note that

the previous study applied dimensionality reduction to trial-averaged data at hand-picked time points. Thus, the ability to reveal this structure from single-trials across all times is noteworthy, as trial-averaging is not possible in the BCI context. While the performance of deep networks on neural data is indeed encouraging, it falls short of the performance of similar networks on acoustic data (∼90% classification accuracy on the acoustic data from this dataset). This difference can be reduced by improved understanding and continued investigation into robust feature representations of ECoG data as related to speech production. For example, we found the highest (relative to chance) decoding performance was for the entire consonant-vowel syllable, which agrees with previous descriptions of context dependent phoneme representations (i.e. coarticulation) in vSMC activity [54]. Again, deep networks can enable this by revealing the latent structure of neural data as shown in this study (fig 4.6). The ability to extract meaningful structure from single-trial data is important for understanding the nature of brain computations and suggests that deep networks may be useful for this purpose in general.

Chapter 5

Inference using a network of leaky integrator neurons with strictly neuron-local computation

Chapter coauthor: Michael R. DeWeese

5.1 Introduction

Marr [80] famously divided descriptions of computation into three levels. They are: computation, algorithm, and implementation. The computational description is a high-level description of the problem that is being solved through computation, e.g. object detection, speech recognition, or navigation. The algorithmic description, is the abstract mathematical algorithm that will be used to solve the problem, e.g. deep networks, random guessing, or dynamical systems. Finally, the implementation level is the hardware details which are used to implement the algorithm such as a CPU, memristors, or neural substrate [81, 82].

A common hypothesis in theoretical neuroscience is that the brain is learning a structued probabilistic model of sensory stimuli. One implication of this hypothesis that that if a probabilistic model is trained on natural stimuli, i.e. natural images or sounds, the nodes in the model should have similar activity to functional units in the brain. Olshausen and Field [83], showed that a sparse coding model trained on natural images has response properties that are similar to those of simple cells in V1. Since then, there have been many results showing that responses in other parts of visual and auditory streams can be explained by learning models of natural scenes [84, 85, 86, 87, 88].

To make this hypothesis more exact, the computations required for learning and inference in these models must be mapped onto computations that functional units, typically a network of neurons, in the brain could carry out. Rozell et al. [43] proposed a set of locally-competitive algorithms to perform MAP inference, which have sub-membrane potential dynamics and thresholding behavior, similar to neurons in cortex. These dynamics have a clear neural

analog with was explored in a spiking version of this circuit [89].

Zylberberg et al. [3] proposed a network of spiking leaky integrate-and-fire neurons, SAILnet, which can perform approximate learning and inference in a sparse coding model. The parameter updates and inference circuit only require neuron-local computations, meaning that each computation in the model could be carried out by a single neuron given the information that a neuron in cortex likely has access to: its own membrane potential and recent spike rate, the value of its own synapses, and the spiking rate of neurons with which it has connections. It was shown how the local learning rules can be derived from an approximation to the full sparse coding objective function, but the leaky integrate-and-fire inference circuit was motivated from previous theoretical models of biological neurons and not derived from the objective function.

Note that to implement a network that can train itself and infer latent variables, the actual objective function never needs to be computed. The entire objective function will by definition not be neuron-local because all neurons have trainable parameters and so no neuron can compute a function of all trainable parameters. All that is needed is that the derivatives with respect to the network's parameters and latent variables be neuron-local.

In this chapter we derive a leaky integrator inference circuit, which takes a similar form to the one proposed in [3], and show numerically that it better follows the gradient of the objective function. Sections 5.2 and 5.3 summarize sparse coding and its implementation in SAILnet. Section 5.4 goes through the details of how to derive an inference circuit from the objective function.

5.2 Sparse coding

The original sparse coding objective [83] is a sum of reconstruction term and sparsity terms

$$E(X, S; A, \lambda) = \frac{1}{2}\sum_i (X_i - \sum_j A_{ij} S_j)^2 + \sum_j \lambda f(S_j), \tag{5.1}$$

where X_i is a data vector, A_{ij} is a dictionary of filters or feedforward weights, S_j are the sparse coefficients or latent variables, and λ is a parameter that controls the trade-off between reconstruction and sparsity. A common choice for $f(S)$ is the L_1 norm, which allows the objective to be easily interpreted as a simple probabilistic model with Gaussian errors on a linear reconstruction from latent variables which have a Laplacian prior.

Alternate sparse coding objective

Zylberberg et al. [3] gave an alternative objective function for sparse coding. Rather than putting a prior on the sparse coefficients, the problem was turned into a constrained optimization problem where the reconstruction term in the energy function was the same, but

the sparse prior was turned into homeostasis for the individual neuron's firing rates:

$$E(X, S; A, \theta, p) = \frac{1}{2}\sum_i (X_i - \sum_j A_{ij} S_j)^2 + \sum_i \theta_i (S_i - p) \tag{5.2}$$

where the second terms is a constraint implemented using the method of Lagrange multipliers.

The reconstruction term will lead to both non-local learning rules for A and a non-local inference circuit for infering S. So, it is approximated by an objective

$$E(X, S; A, W, \theta, p) = \frac{1}{2}\sum_{ij} (X_i - A_{ij} S_j)^2 + \sum_i \theta_i (S_i - p) + \sum_{ij} W_{ij}(S_i S_j - p^2) \tag{5.3}$$

which will lead to Oja's learning rule [90] and a local inference circuit. Notice that the sum on j has moved outside of the squaring operation in the first term. Zylberberg et al. [3] showed that gradient descent with respect to A, W, and θ gave local learning rules for the parameters of the model.

The change in the reconstruction term removes terms that implement explaining-away in the model during inference. The third term in the objective constrains the neurons activities to be decorrelated. W_{ij} is constrained to be non-negative and symmetric with $W_{ii} = 0$.

As shown in Fig 5.1, sparse coding can be represented as a graph wherein a set of latent variables, S, are explaining, through conditional probabilities, the data, X. In a graphical model, conditional dependencies are indicated with directed arrows and local statistical dependencies are indicated with lines. Graphical model representations of distributions are a powerful tool for reasoning about models. Wainwright and Jordan [11] present their use and properties. This work will not use the full machinery of graphical models, but will rely on the intuition gained from their graphical representation of statistical dependencies.

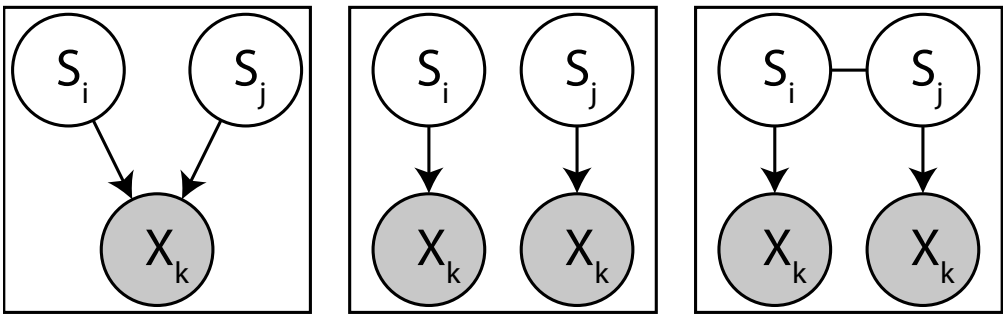

Figure 5.1: Graphical models for three sparse coding objectives. Latent variables S_i and S_j explain visible (data) variable X_k. The left panel shows the model for the original reconstruction objective. The middle panel shows the model for the local reconstruction objective before the decorrelating constraint is added. The right panel shows the local reconstruction model after the decorrelating connections have been added.

In the original objective represented in the left panel of fig 5.1, all latent variables are explaining each dimension of the data. This means that then the posterior of the distribution is computed, the latent variables will have an explaining-away effect on each other, i.e. $P(S_i|X_k) > P(S_i|X_k, S_j)$. This can be seen graphically because S_i and S_j have a connection through X_k.

In the approximate reconstruction objective, each neuron is independently trying to explain the data, which leads to a set of disconnected pairs of nodes as in the middle panel. Each latent variable in S in independently trying to explain all of the data dimensions X_k. With no other structure in the model, all neurons or latent variables will typically collapse into modeling the dimension with largest variance.

In the panel on the right, the decorrelation term adds a undirected dependence back between the latent variables which will approximate the explaining away effect from the left-most panel. This will prevent the neurons from learning a degenerate representation of the data.

When the objective function is made local by throwing away terms which are not neuron-local. This removes any explaining-away effects the latent variables have on each other during inference. This would lead to the neurons learning potentially degenerate solutions or representations. This motivates adding in auxillary synapses between units which may induce an approximate explaining-away effect. The decorrelation constraint which leads to inhibitory synapses is added to the network to approximate the explaining-away effect.

5.3 Comparing reconstruction objectives

Following the analysis of Zylberberg et al. [3], we can expand both the local and non-local objectives and compare terms. For the data averaged, non-local reconstruction objective we have

$$C_{NL}(X, S; A) = \frac{1}{2N} \sum_{ik} (X_i^{(k)} - \sum_j A_{ij} S_j^{(k)})^2, \tag{5.4}$$

where superscripts indicate dataset indices. For the local objective we have

$$C_L(X, S; A) = \frac{1}{2N} \sum_{ijk} (X_i^{(k)} - A_{ij} S_j^{(k)})^2, \tag{5.5}$$

where we are now including an average over a batch of size N with batch index denoted by superscripts.

Starting with the non-local objective, we get

$$\begin{aligned} C_{NL}(X,S;A) &= \frac{1}{2N}\sum_{ik}(X_i^{(k)} - \sum_j A_{ij}S_j^{(k)})^2 \\ &= \frac{1}{2N}\sum_{ik}\left((X_i^{(k)})^2 - 2X_i^{(k)}\sum_j A_{ij}S_j^{(k)} + (\sum_j A_{ij}S_j^{(k)})(\sum_j A_{ij}S_j^{(k)})\right) \\ &= \frac{1}{2}\sum_i \langle X_i^2\rangle - \sum_{ij} A_{ij}\langle X_i S_j\rangle + \frac{1}{2}\sum_{ijk} A_{ij}A_{ik}\langle S_j S_k\rangle \end{aligned} \tag{5.6}$$

where expectations are over the batch.

For the local objective, we get

$$\begin{aligned} C_L(X,S;A) &= \frac{1}{2N}\sum_{ijk}(X_i^{(k)} - A_{ij}S_j^{(k)})^2 \\ &= \frac{1}{2N}\sum_{ijk}\left((X_i^{(k)})^2 - 2X_i^{(k)}A_{ij}S_j^{(k)} + (A_{ij}S_j^{(k)})(A_{ij}S_j^{(k)})\right) \\ &= \frac{L}{2}\sum_i \langle X_i^2\rangle - \sum_{ij} A_{ij}\langle X_i S_j\rangle + \frac{1}{2}\sum_{ij} A_{ij}^2\langle S_j S_j\rangle \end{aligned} \tag{5.7}$$

where L is the number of neurons in the network.

There are two differences to note. The first is that the local objective has $L-1$ extra copies of the first term from moving the sum on neurons outside of the reconstruction. These terms have no parameter dependence so they will not affect inference or learning. The second important difference is in the final term where we are neglecting the off-diagonal terms from the sum over $A_{ij}A_{ik}\langle S_j S_k\rangle$). The decorrelation should keep $\langle S_j S_k\rangle$) small and Zylberberg et al. [3] shows that $A_{ij}A_{ik}$ and $\langle S_j S_k\rangle$) are correlated so the off diagonal terms should be small compared to the diagonal terms.

So we can relate the two objectives by

$$C_{NL}(X,S;A) = C_L(X,S;A) - \frac{M-1}{2}\sum_i \langle X_i^2\rangle + \frac{1}{2}\sum_{i,j\neq k} A_{ij}A_{ik}\langle S_j S_k\rangle \tag{5.8}$$

where the first term only depends on the data and the second term's function is being replaced by the inhibitory matrix W.

In the objective function, the explaining away is apparent when looking at the MAP inference differential equation. The neurons have an inhibiting effect on each other which is proportional to the inner product of their dictionary elements and the activity of the inhibiting neuron. This term is not neuron-local and could not reasonably be implemented in a networks of biological neurons.

5.4 A network circuit for inference

Before the parameters of the model can be learned, the sparse code for each data element needs to be inferred. For classic sparse coding, this is commonly done using MAP inference on the objective function with respect to the codes S. Zylberberg et al. [3] instantiated a leaky integrate-and-fire circuit (LIF) which had three terms in the integrator circuit

$$\begin{aligned} \tau \frac{du_i}{dt} &= -u_i + \sum_j X_j A_{ji} - \sum_{j \neq i} W_{ij} y_j \\ y_i &= 1 \text{ if } u_i > \theta_i \text{ else } 0 \end{aligned} \tag{5.9}$$

which are conductive leak, linear filtering driving, and inter-neuron inhibition. y_j is the instantaneous, binary spiking variable for neuron i. This circuit may not always descend the objective function, as we can show.

The negative gradient of the local objective function, eq. 5.3, is

$$-\frac{\partial E(S|X;A,W,\theta)}{\partial S_i} = \sum_j A_{ji} X_j - \sum_j A_{ji}^2 S_i - \theta_i - 2 \sum_{j \neq i} W_{ji} a_j. \tag{5.10}$$

The first term is the same linear filtering term as in SAILnet. The second is the leakiness term with an additional scaling by the length squared of the dictionary element. In sparse coding, the dictionary is commonly normalized to have length 1 to remove a degeneracy in the model, but since SAILnet has a homeostatic constraint on S, it is not be possible to also constraint the norm of A, have a homeostatic rate for S, and reconstruct the data, which has fixed norm. Empirically, the mean norm will be on the order of length 1, but can vary by a small integer factor and will have non-zero variance. It is an interesting prediction that the leakiness of the membrane of a neuron should scale with the overall strength of its synapses. $-\theta$ would be converted into a spike-threshold in a LIF version of this analog equation. Finally, the last term is twice the SAILnet value due to W being symmetric.

Given this analog optimization problem, there are a number of different ways to instantiate a leaky-integrator circuit to approximate inference. The negative gradient can be interpreted as the driving current in the circuit

$$\tau \frac{du_i}{dt} = -\frac{\partial E(S|X;A,W,\theta)}{\partial S_i}. \tag{5.11}$$

The remaining choices all have to do with how to interpret the analog S_i, in the spiking model. Some options are:

- u_i, the analog membrane potential - probably used for leakiness a_i,
- y_i, the instantaneous spike variable, - possible for inhibitory terms,
- $\bar{y}_i$, the current spike rate, - possible for inhibitory terms.

5.5 Numerical simulations for inference dynamics

Fig 5.2 shows the dynamics of inference when the network is first initialized, i.e. when the parameters have not been adapted to the data. The inhibitory weights and thresholds are initialized at zero and the reconstruction weights have been drawn randomly from a mean-zero unit normal distribution. This means that only the reconstruction objectives contribute to inference. The top panel shows the normal reconstruction error measures during 200 steps of inference. The bottom panel shows the independent reconstruction error during the same inference.

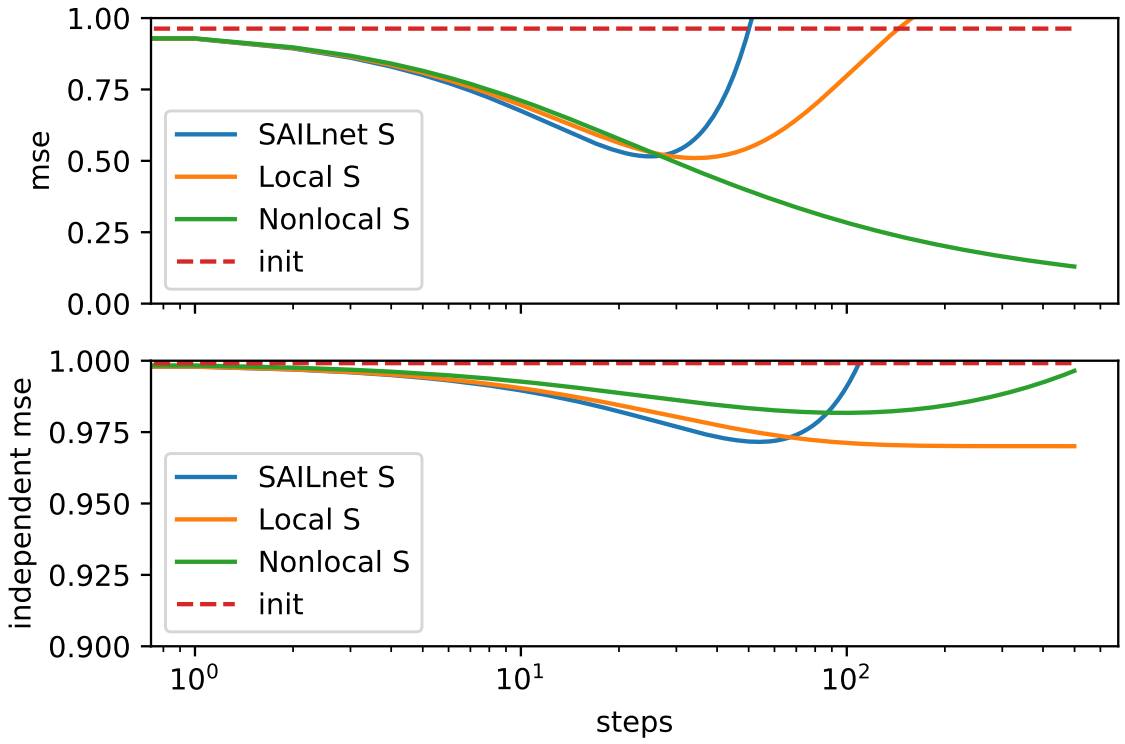

Figure 5.2: Inference dynamics at network initialization for a network running inference on the nonlocal objective function, local SAILnet objective function, and corrected local objective function derived here. The top panel shows the mean-squared error, which is equivalent to the nonlocal objective during the respective inference dynamics. The bottom panel shows the local error which is equivalent to the average of the each neuron's individual error.

Initially, all three inference circuits descend the objective function. The nonlocal circuit will continue to descend until the data is close to perfectly reconstructed. Initially, the minima that the local circuits are falling into is far away from the minima that the nonlocal circuit

has and so while they initially descend, at some point they move away from the nonlocal minimum and their reconstruction error rises. If the independent reconstruction error is measured, the local inference circuit will exactly desend this objective and we see that the nonlocal and SAILnet circuits

Inference after ten thousand steps of learning is shown in fig 5.3. Descent is qualitatively similar, but the local circuit more closely follows the nonlocal circuit as compared to the original SAILnet circuit. Interestingly the SAILnet circuit initially more quickly descends both objectives, but traverses the minimum in reconstruction error and climbs back to a solution with high error.

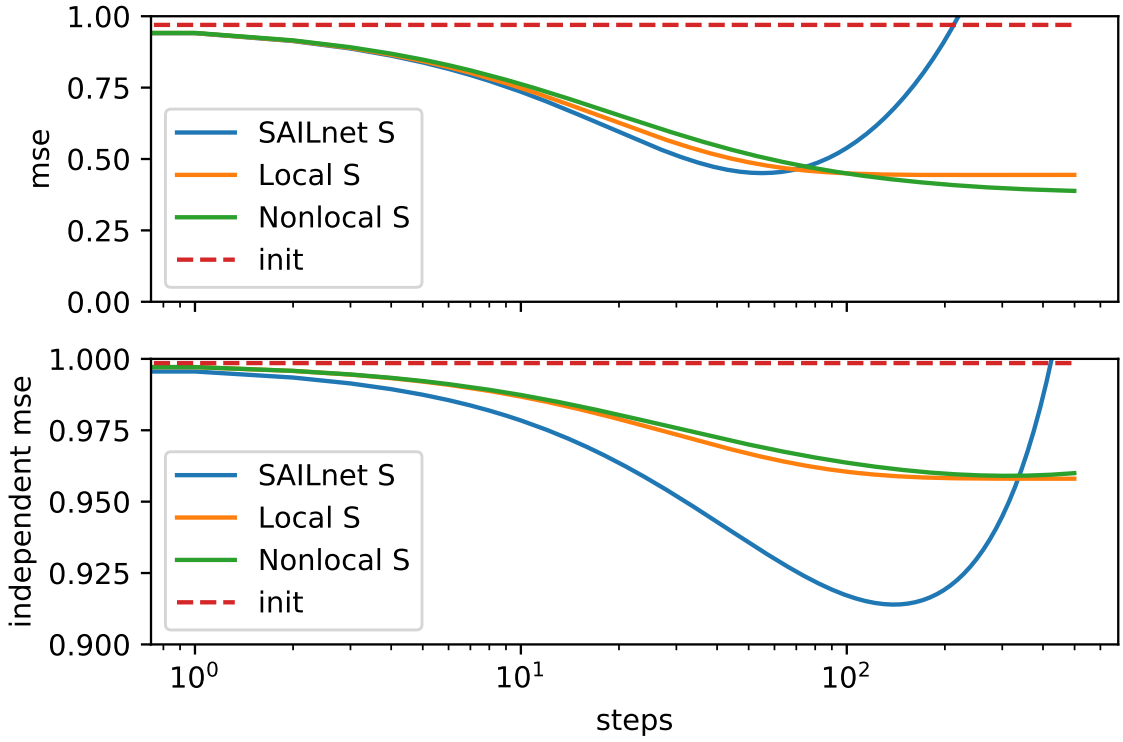

Figure 5.3: Same plots as fig 5.2 after ten thousand iterations of learning.

5.6 Discussion

We have developed a new dynamical neural circuit which implements constrained inference in an approximate sparse coding model. This circuit connects the second of Marr's level: algorithm to the third: a neural implementation. This connection requires approximation

given the neuron-local nature of information in a neural circuit model. Unlike previous work [3], we derive this circuit from the objective function of the model, which leads to a procedure which is interpretable as inference in a probabilistic model. In a rate-based neural implementation, we show that this new circuit is more stable and more closely follows the dynamics of the original nonlocal objective function of the model.

The insights gained from the translation of a probabilistic model, sparse coding, to an approximate neural implementation, SAILnet and modifications proposed here, can be applied to other classes of models. These types of modifications are tailored to the type of probabilistic model as opposed to general purpose translation of algorithms into spiking neural circuit as proposed in the Neural Engineering Framework [91].

Chapter 6

Overcomplete independent components analysis

Chapter coauthors: Alejandro F. Bujan, Friedrich T. Sommer

6.1 Introduction

Mining the statistical structure of data is a central topic of machine learning and an important tool for neuroscience. A prominent class of such algorithms is dictionary learning methods, which reveal structural primitives in the data, the dictionary, and a corresponding latent representation, often regularized by sparsity. Here we consider dictionary learning of the type first proposed under the name Independent Components Analysis (ICA) [92, 93], that are computationally tractable because the learned mappings between data and latent representation are both linear. Specifically, we focus on overcomplete dictionary learning [94, 95, 10], the case, when the dimension of the latent representation exceeds the dimension of the data and therefore the linear filters (dictionary) generating the data cannot be mutually orthogonal.

Dictionary learning has important applications in neuroscience as a computational model for understanding the formation of sensory representations in the brain [93, 96, 97, 98, 99, 100, 101] and as data mining tools [102, 103, 104]. Furthermore, overcomplete dictionary learning has been shown to learn more a more diverse set of features which more closely represent the diversity of receptive fields found in sensory cortex [99, 101, 105]. Overcompleteness may also be a strategy used by early sensory areas in cortex as it has been shown that they contain more neurons than afferent inputs [106, 107, 108, 109, 110, 111].

Unsupervised, overcomplete representation learning algorithms require either implicit or explicit mechanisms for preventing learned features from becoming coherent. The coherence of the dictionary is defined as the maximum absolute value of the off-diagonal elements of

the Gram matrix of a dictionary [6], W,

$$\text{coherence} \equiv \max_{i \neq j} |\sum_k W_{ik} W_{jk}| = \max_{i \neq j} |\cos \theta_{ij}|. \tag{6.1}$$

During inference, latent features in overcomplete sparse coding models [112] have an *explaining-away* effect on each other which prevent them from learning coherent solutions. In overcomplete independent components analysis (ICA) [95, 10], additional costs need to be added to the sparsity prior to prevent high-coherence solutions. Sparse coding methods which add additional coherence control costs have also been proposed [8, 9].

ICA is a technique for learning the underlying non-Gaussian and independent sources, S, in a dataset, X. ICA is formulated as a noiseless linear generative model:

$$X_i = \sum_j A_{ij} S_j, \tag{6.2}$$

where $A \in \mathbb{R}^{D \times L}$ is referred to as the *mixing matrix* wherein D is the dimensionality of the data, X, and L is the dimensionality of the sources, S. The goal of ICA is then to find the *unmixing matrix* $W \in \mathbb{R}^{L \times D}$ such that the sources can be recovered, $S_i = \sum_j W_{ij} X_j$ with $W = A^{-1}$. In the complete case the mixing matrix can be inverted. The unmixing matrix W can be obtained by minimizing the negative log-likelihood of the model:

$$\arg\min_W \quad \sum_{i=1}^{M} \sum_{j=1}^{L} g(\sum_k W_{jk} X_k^{(i)}) - \log(\det(W)) \tag{6.3}$$

where $g(\cdot)$ specifies the shape of the negative log-prior and is usually a smooth version of the L_1 norm such as the $\log(\cosh(\cdot))$, $X^{(i)}$ is the ith element of the dataset, X, which has M elements, and where the bases are constrained to be unit-norm. If the data has been whitened, the unconstrained optimization can be replaced by a constrained optimization where the second term in the cost function is replaced with the constraint $WW^T = I$ [113]. The log-determinant and the identity constraint will both prevent the bases from becoming coherent. In ICA, as an alternative to sparse coding, inference is a single linear transformation versus a *maximum a posteriori* (MAP) estimation or posterior estimation which requires computationally expensive iterative methods.

In overcomplete ICA models, neither the log-determinant cost nor the orthonormality constraint are viable and so the objective function can modified by adding a new cost, C, to the sparsity prior [7, 10]. The new unconstrained optimization problem becomes:

$$\arg\min_W \quad \lambda \sum_{i=1}^{M} \sum_{j=1}^{L} g(\sum_k W_{jk} X_k^{(i)}) + C(W). \tag{6.4}$$

This cost should prevent the co-alignment of the bases. Different methods for coherence control in overcomplete ICA have been proposed: a quasi-orthogonality constraint [114], a

reconstruction cost (equivalent to an L_2 Gram matrix cost) [10], and a Random Prior cost [7] (see section 6.6 for details). However, a systematic analysis of the properties of proposed ICA methods and a comparison with methods that extend more naturally to overcomplete, i.e. sparse coding, is still missing in the literature.

Here, we show that although the global minima of the L_2 cost has zero coherence for a complete basis, in the overcomplete case, it has global minima wherein pairs of basis vectors have coherence which can become one. We introduce a theoretical framework for evaluating different degeneracy control costs, and propose several new costs, which we compare with previously proposed methods. Our first novel approach is the L_4 cost on the difference between the identity matrix and the Gram matrix of the bases. The second method is a cost which we term the *Coulomb* cost because it is derived from the potential energy of a collection of charged particles bound to the surface of an n-sphere. We also propose a modification to existing cost function which allows them to learn less coherent dictionaries.

In addition to controlling coherence, we show that these costs will influence the distribution of an overcomplete set of bases in the high dimensional space. We investigate their effect on the distribution of pairwise angles between bases learned on a dataset with known structure and evaluate the diversity of bases learned on natural images using different degeneracy costs. Finally, we provide analytic and numerical methods for evaluating potential degeneracy controls.

6.2 Minima landscape of coherence control costs

The L_2 cost

Dictionary or representation learning methods often augment their cost functions with additional terms aimed at learning less coherent features or making learning through optimization more efficient. The L_2 cost [8, 10, 9], defined for a matrix, W, as:

$$C_{L_2} = \sum_{ij}(\delta_{ij} - \sum_k W_{ik}W_{jk})^2 = \sum_{ij}(\delta_{ij} - \cos\theta_{ij})^2, \tag{6.5}$$

is one of these costs and is motivated by compressed sensing [115, 6]. First, minimizing the L_2 cost is a necessary but not sufficient condition for finding *equiangular tight frames* (see section 6.6 for details), a certain class of minimum coherence solutions. Second, overcomplete representations are easier to recover when the mixing matrix has low coherence [116].

We prove that the L_2 cost has global minima with maximal coherence. This implies that the L_2 cost and its related costs are not providing coherence control in overcomplete dictionaries.

Pathological degeneracy in the L_2 cost

For the L_2 cost, it can be shown that in the overcomplete case, there exists a set of global minima in which the angle between pairs of bases is exactly zero. We prove the following

theorem:

Theorem 1. *Let W be an integral overcomplete unmixing matrix with data dimension D and latent dimension $L = n \times D$, with n an integer greater than 1. There exist global minima of the L_2 cost: W_0, with coherence $= 1$.*

This theorem also implies that an orthonormal basis is a global minimum in the complete case. We also show that there are transformations which move the degenerate solution (coherence $= 1$) into non-degenerate solutions (coherence < 1) to which the L_2 cost is invariant.

Theorem 2. *There exist continuous transformations: $\Phi = R \oplus I(L-D)$, $R \in O(D)$, $I(L-D)$ an $L-D$ dimensional identity transformation, on W_0 to which the L_2 cost is invariant. These transformed matrices $W_0\Phi$ have coherence < 1 and are global minima of the L_2 cost.*

Section 6.4 contains the proofs of these theorems.

These degenerate minima are illustrated with a two dimensional, two times overcomplete example in fig 6.1. It can be shown that there are pathological minima, fig 6.1A, which can be continuously rotated into other low coherence minima, fig 6.1B. These configurations are equivalent in terms of the value of the L_2 cost and lie on a connected global minimum.

In order to understand these minima, we evaluate the structure of the cost in the two dimensional example analytically. The global rotational symmetry of the L_2 cost allows us to parameterize all solutions with respect to one fixed basis element: $(1, 0)$, without loss of generality. The bases, shown in fig 6.1, are:

$$(1, 0), (\cos\theta_1, \sin\theta_1), (\cos\theta_2, \sin\theta_2), (\cos\theta_2 + \theta_3, \sin\theta_2 + \theta_3). \tag{6.6}$$

Setting θ_1 and θ_3 to $\pi/2$, i.e. creating a set of two orthonormal bases, forms a ring of minima as θ_2 is varied. This can be shown by computing the gradient and the eigenvalues of the Hessian of the cost at these points. The cost function, gradient, and Hessian are tabulated in section 6.3.

The gradient is zero everywhere along this path, which shows that it is critical point. One eigenvalue of the Hessian is 0 as shown in fig 6.1C. This corresponds changing θ_2 (see section 6.3 for details), which is equivalent to rotating the pairs of bases with respect to each other and has no effect on the cost. The second two eigenvalues will be positive as long as θ_2 is not $n\pi$ or $(n + \frac{1}{2})\pi$ for integer n, respectively. At the points where they are zero, the second derivatives vanish, but inspection of the cost along these axes show that they will be locally stable as the fourth derivatives are positive.

Numerical simulations, section 6.2, suggest that these pathological solutions are also present in the quasi-orthonormality constraint [114] which is an iterative update. In summary, we find that several proposed coherence control methods do not prevent dictionary from becoming coherent in the overcomplete regime indicating that there is a need for new forms of coherence control.

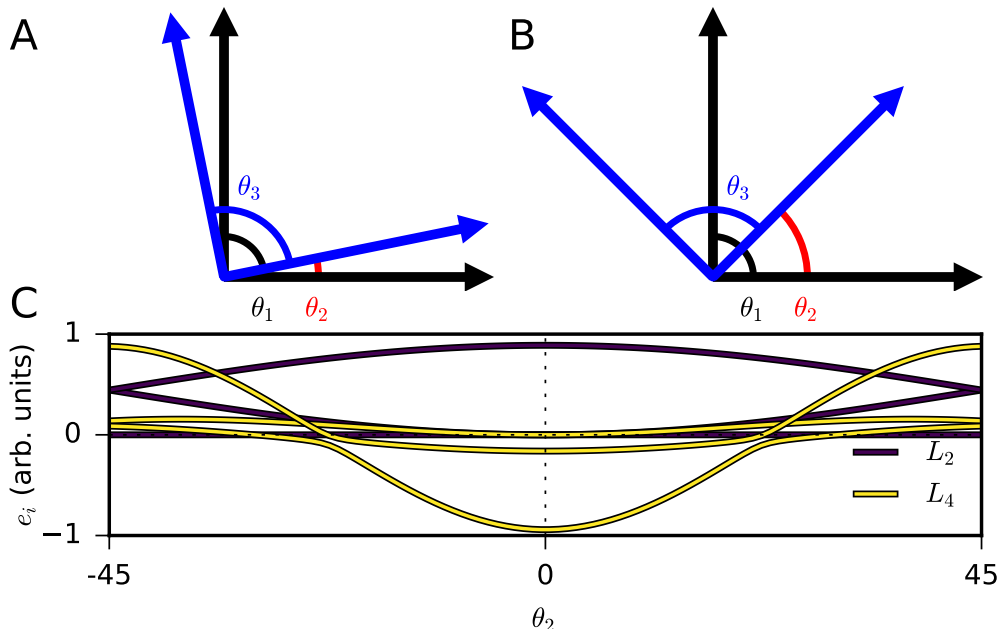

Figure 6.1: Stucture of the pathological global minimum in the L_2 cost which the L_4 cost corrects. For **A** and **B**, each arrow represents a basis element in a 2-times overcomplete basis in a 2-dimensional space. **A** Bases which have the same value of the cost as the bases in **B** for any θ_2 including the pathological solution $\theta_2 \to 0$ and **B** shows a potential good solution with low coherence. **C** The eigenvalues of the Hessian of the L_2 and L_4 costs at $\theta_1 = \theta_3 = \frac{\pi}{2}$ as a function of θ_2. Each purple line is one of the three eigenvalues of the L_2 cost as θ_2 is varied. Each yellow line is one of the three eigenvalues of the L_4 cost as θ_2 is varied.

Proposed coherence control costs

The L_4 cost

We propose a novel coherence control cost termed the L_4 cost, which transforms the degenerate minima of the L_2 cost into saddle points.

$$C_{L_4} = \sum_{ij}(\delta_{ij} - \sum_k W_{ik}W_{jk})^4 = \sum_{ij}(\delta_{ij} - \cos\theta_{ij})^4 \tag{6.7}$$

Following the same analysis as section 6.2, we show that the degenerate solutions are either reduced to saddle points at $\theta_2 = n\frac{\pi}{2}$ or local minima at $\theta_2 = (2n+1)\frac{\pi}{4}$ (fig 6.1C, which correspond to incoherent solutions. The cost function, gradient, and Hessian are tabulated in section 6.3.

The Coulomb cost

We also propose a second alternative cost, where the repulsion from high coherence is *Coulombic*: the Coulomb cost. Coherence control can then be related to the problem of

characterizing the minimum potential energy states of L charged particles on an n-sphere, an open problem in electrostatics [117]. The energy, E^{Coulomb}, of two charged point particles of the the same sign is proportional to the inverse of their distance, $\vec{r}_{ij}$:

$$E_{ij}^{\text{Coulomb}} \propto \frac{1}{|\vec{r}_{ij}|}. \tag{6.8}$$

To map this problem onto ICA, the cost should be made symmetric around $\theta = \pi/2$ rather than $\theta = \pi$, which can be accomplished by replacing θ with 2θ, i.e. $|r_{ij}| = \sqrt{1 - \cos^2(\theta_{ij}/2)} \rightarrow \sqrt{1 - \cos^2 \theta_{ij}}$. Therefore, the Coulomb cost[1] can be formulated as follows:

$$C_{\text{Coulomb}} = \sum_{i \neq j} \frac{1}{\sqrt{1 - \cos \theta_{ij}^2}}. \tag{6.9}$$

Coherence-based costs

The coherence of a dictionary [6] is defined as the maximum absolute value of the off-diagonal elements of the Gram matrix, as in equation 6.1, which can be used as a cost function during optimization. This cost is difficult to numerically optimize since the derivative through the max operation will only act on one pair of bases at each optimization step, although it should find solution with local minima of coherence. An easier to optimize, but heuristic, version of this cost is the sum over all off-diagonal elements whose squares are larger than the mean squared value

$$C_{\text{Soft Coherence}} = \sum_{i \neq j \text{ s.t. } \cos \theta_{ij}^2 > \cos \hat{\theta}^2} |\cos \theta_{ij}|, \text{ with } \cos \hat{\theta}^2 = \underset{i,j}{\text{mean}}(\cos \theta_{ij}^2). \tag{6.10}$$

We find that this cost does not work well for coherence control in ICA, but it can be used to generate mixing matrices for generating data with known structure as in section 6.5.

Flattened costs

The Random prior and Coulomb costs have quadratic terms that dominate their derivatives as $\cos \theta \rightarrow 0$. As will be described in section 6.2, these terms can have the undesired effect of pushing more pairs of bases close to orthogonal at the expense of having more bases with smaller pairwise angles. We also created versions of the Random prior and Coulomb costs where the quadratic terms have been removed, i.e.

$$C_{\text{Flat}}(\cos \theta_{ij}) = C(\cos \theta_{ij}) - \left. \frac{\partial^2 C(\cos \theta_{ij})}{\partial \cos \theta_{ij}^2} \right|_0 \cos^2 \theta_{ij}. \tag{6.11}$$

[1] We subtract off the value of the cost for perpendicular bases, 1, for each pair i, j to bring the cost into a better dynamic range.

Analytic and numerical investigations of coherence control

In order to understand the origin of the effects of the different coherence controls on the pairwise angle distributions, the coherence costs can be directly compared without the data dependent ICA sparsity prior. We use two different initializations of the bases and optimize the data-independent coherence costs. These initializations are: a noisy pathological initialization (as in section 6.2) and a random uniform initialization.

The noisy pathological initialization tiles an orthonormal, complete basis two times and adds Gaussian noise to every basis element to create W. As shown in fig 6.2A, most pairwise angles start close to either 90 or 0 degrees apart as shown in the two peaks in the initial distribution. The quasi-orthogonality update and the L_2 are unable to move away from this solution unlike the other costs, which generate comparable solutions for both initializations.

In the random uniform case, the elements of W are drawn independently from a uniform distribution on the unit hyper-sphere. After converging, the quasi-orthogonality update [114] produces a distribution which is more coherent than the initialization (fig 6.2A. All other costs decrease the coherence of the bases. The distribution of pairwise angles for the L_2 cost peaks at 90 degrees but also has a longer tail towards small angles. The other costs have shorter tails and have varying amounts of density near 90 degrees. Of all costs, the L_4 cost distributes the angles most evenly which is reflected by its distribution having the narrowest width and lowest coherence.

In order to gain more insight into the qualitative differences in the distributions of angles, we analyze the behavior of the costs around $\theta = 0$ and $\theta = 90$ (fig 6.2D-E respectively). The gradient of the cost close to $|\cos\theta| = 1$ is proportional to the force the angles feel to stay away from zero which will determine the tail of the angle distribution.

Taylor expanding all the costs near $\cos\theta = 0$ reveals that all cost functions have non-zero second order terms except for the L_4 cost which only has a fourth order term with linear and cubic terms in their gradients respectively as shown in fig 6.2E. Gradients which scale linearly will encourage pairs of basis vectors to be more orthogonal at the expense of skewing the angle distribution towards small values. This leads to distributions of pairwise angles which are less uniform over all pairs of elements of the basis. Fig 6.2C shows the effect of removing the quadratic terms from the Coulomb and Random Prior costs. In both cases, the costs have fewer pairwise angles close to 90 degrees and a smaller coherence.

This analysis shows that proposed coherence control methods prevent degeneracy to different degrees, and furthermore that the choice of coherence control, which is meant to affect the distribution of small pairwise angles, has an effect on the entire distribution of angles.

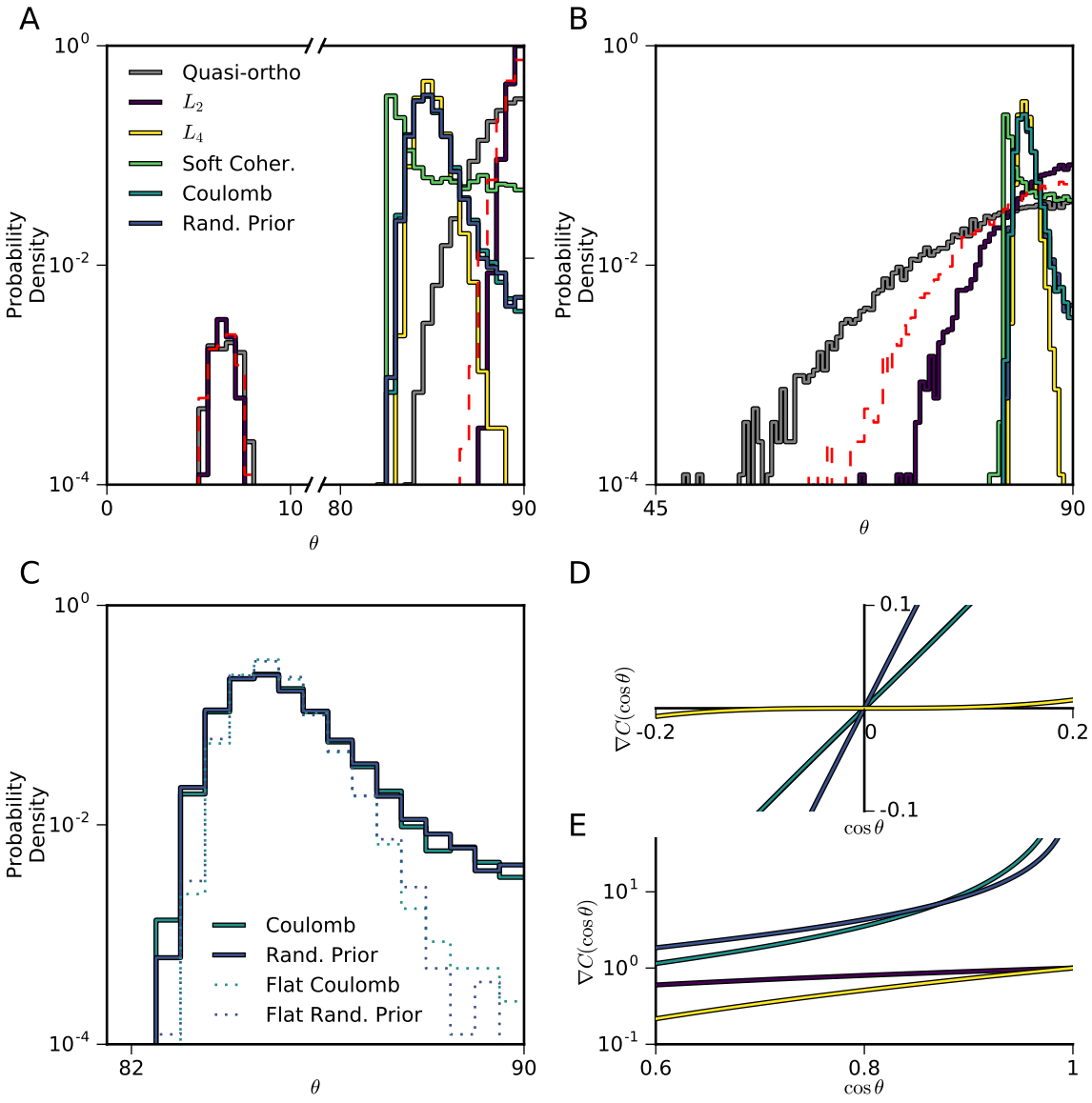

Figure 6.2: Analysis of different coherence control mechanisms. Color code is preserved across panels. **A** Angle distributions obtained by numerically minimizing different cost functions C for a random initialization of the bases. Red dotted line indicates the initial distribution of angles. **B** Angle distributions obtained (as in **A**) with a pathological initialization of the bases. **C** Comparison of standard and flattened Coulomb and Random Prior costs. **D** Gradient of the costs as a function of $\cos\theta$ near $\cos\theta \sim 1$. **E** Gradient of the costs as a function of $\cos\theta$ near $\cos\theta \sim 0$.

6.3 Minima analysis for the L_2 and L_4 costs for a 2-dimensional space

Here we tabulate the full Hessian matrices, eigenvalues, and eigenvectors for the analysis in section 6.2.

L_2 cost

$$
\begin{aligned}
C_{L_2}(\theta_1,\theta_2,\theta_3)|_{\theta_1,\theta_3=\frac{\pi}{2}} &= 4 \\
\frac{\partial C_{L_2}(\theta_1,\theta_2,\theta_3)}{\partial\vec{\theta}}|_{\theta_1,\theta_3=\frac{\pi}{2}} &= \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \\
H(C_{L_2})|_{\theta_1,\theta_3=\frac{\pi}{2}} &= \begin{pmatrix} 4 & 0 & 4\cos 2\theta_2 \\ 0 & 0 & 0 \\ 4\cos 2\theta_2 & 0 & 4 \end{pmatrix} \\
\text{Eig.}(H_{L_2})|_{\theta_1,\theta_3=\frac{\pi}{2}} &= \begin{pmatrix} 0 \\ 8\sin^2\theta_2 \\ 8\cos^2\theta_2 \end{pmatrix} \\
\text{EVec.}(H_{L_2})|_{\theta_1,\theta_3=\frac{\pi}{2}} &= \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}
\end{aligned}
\tag{6.12}
$$

L_4 cost

$$
\begin{aligned}
C_{L_4}(\theta_1,\theta_2,\theta_3)\big|_{\theta_1,\theta_3=\frac{\pi}{2}} &= 3+\cos 4\theta_2 \\
\frac{\partial C_{L_4}(\theta_1,\theta_2,\theta_3)}{\partial\vec{\theta}}\big|_{\theta_1,\theta_3=\frac{\pi}{2}} &= \begin{pmatrix} 2\sin 4\theta_2 & -4\sin 4\theta_2 & -2\sin 4\theta 2\end{pmatrix} \\
H(C_{L_4})\big|_{\theta_1,\theta_3=\frac{\pi}{2}} &= \begin{pmatrix} -8\cos 4\theta_2 & 8\cos 4\theta_2 & 4(\cos 2\theta_2+\cos 4\theta_2) \\ 8\cos 4\theta_2 & -16\cos 4\theta_2 & -8\cos 4\theta_2 \\ 4(\cos 2\theta_2+\cos 4\theta_2) & -8\cos 4\theta_2 & -8\cos 4\theta_2 \end{pmatrix} \\
\text{Eig.}(H_{L_4})\big|_{\theta_1,\theta_3=\frac{\pi}{2}} &= \begin{pmatrix} 4(\cos 2\theta_2-\cos 4\theta_2) \\ -2\cos 2\theta_2-14\cos 4\theta_2-\ldots \\ \ldots\sqrt{2}\sqrt{34-2\cos 2\theta_2+\cos 4\theta_2-2\cos 6\theta_2+33\cos 8\theta_2} \\ -2\cos 2\theta_2-14\cos 4\theta_2+\ldots \\ \ldots\sqrt{2}\sqrt{34-2\cos 2\theta_2+\cos 4\theta_2-2\cos 6\theta_2+33\cos 8\theta_2} \end{pmatrix} \\
\text{EVec.}(H_{L_4})\big|_{\theta_1,\theta_3=\frac{\pi}{2}} &= \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \\
&\quad \begin{pmatrix} -1 \\ (\frac{\sqrt{2}}{8}\sqrt{\begin{matrix} 2\cos 2\theta_2+\cos 4\theta_2-\ldots \\ \ldots 2\cos 6\theta_2+33\cos 8\theta_2+34 \end{matrix}}-\ldots \\ \ldots-2\cos 2\theta_2)\sec 4\theta_2+\frac{1}{4} \\ 1 \end{pmatrix}, \\
&\quad \begin{pmatrix} -1 \\ \frac{1}{4}-(2\cos\frac{1}{4}\theta_2+\ldots \\ \ldots\frac{\sqrt{2}}{8}\sqrt{\begin{matrix} -2\cos 2\theta_2+\cos 4\theta_2-2\cos 6\theta_2+\ldots \\ \ldots 33\cos 8\theta_2+34 \end{matrix}})\sec 4\theta_2 \\ 1 \end{pmatrix}
\end{aligned}
\tag{6.13}
$$

6.4 Minima analysis for high dimensions

This section contains the proof of Theorems 1 and 2.

L_2 cost minima and equiangular tight-frames: proof of Theorem 1

Here we prove Theorem 1 in two steps: first we can show the equivalence, up to an additive constant, of minimizing the L_2 cost and minimizing the L_2 norm in the error of equation 6.22. Then we show that the pathological solution (section 6.2) is at the global minimum of this cost.

Proof of Theorem 1. For a normalized ($\sum_k W_{ik}^2 = 1,\ \forall i$) matrix, W:

$$\begin{aligned} C_{L_2} &= \sum_{ij}(\sum_k W_{ik}W_{jk} - \delta_{ij})^2 \\ &= \sum_{ij}(\sum_k W_{ik}W_{jk} - \delta_{ij})(\sum_l W_{il}W_{jl} - \delta_{ij}) \\ &= \sum_{ijkl} W_{ik}W_{jk}W_{il}W_{jl} - 2\sum_{ijk} W_{ik}W_{jk}\delta_{ij} + \sum_{ij}\delta_{ij}^2 \\ &= \sum_{ijkl} W_{ik}W_{jk}W_{il}W_{jl} - 2\sum_{ik} W_{ik}^2 + \text{const.}(L) \\ &= \sum_{ijkl} W_{ik}W_{jk}W_{il}W_{jl} + \text{const.}(L) \end{aligned} \tag{6.14}$$

$$\begin{aligned} C_{\text{eq. 6.22}} &= \sum_{kl}(\sum_i W_{ik}W_{il} - \frac{L}{D}\delta_{kl})^2 \\ &= \sum_{kl}(\sum_i W_{ik}W_{il} - \frac{L}{D}\delta_{kl})(\sum_j W_{jk}W_{jl} - \frac{L}{D}\delta_{kl}) \\ &= \sum_{ijkl} W_{ik}W_{il}W_{jk}W_{jl} - 2\sum_{ikl}\frac{L}{D}W_{ik}W_{il}\delta_{kl} + \sum_{kl}(\frac{L}{D}\delta_{kl})^2 \\ &= \sum_{ijkl} W_{ik}W_{il}W_{jk}W_{jl} - 2\frac{L}{D}\sum_{ik} W_{ik}^2 + \text{const.}(L, D) \\ &= \sum_{ijkl} W_{ik}W_{il}W_{jk}W_{jl} + \text{const.}(L, D) \end{aligned} \tag{6.15}$$

where $\sum_k W_{ik}^2 = 1,\ \forall i$ is used extensively and the index letters were initially chosen to make the comparison of the final lines more clear. In Le et al. [10], the L_2 cost is also shown to be equivalent to the reconstruction cost with whitened data.

Now we can show that the same basis that was described in section 6.2: W_0, an integer overcomplete basis where each set of complete bases is an orthonormal basis, exactly satisfies equation 6.22 and so is a minimum of the L_2 cost. This solution is very far away from an ETF in the sense of equation 6.21. A basis of this form, $W \in \mathbb{R}^{L\times D}$, can be constructed as $W_{ij} = \delta_{(i \bmod D)j}$ with $L = n \times D,\ n \in \mathbb{N}$, i.e. a D dimensional identity matrix tiled N times.

This construction satisfies equation 6.22 and therefore has a value of 0 for $C_{\text{eq. 6.22}}$. Since $C_{\text{eq. 6.22}}$ is a sum of quadratic, and therefore non-negative, terms, this construction is a global

minimum of $C_{\text{eq. 6.22}}$ and the L_2 cost.

$$\begin{aligned}\sum_k W_{ki}W_{kj} &= \sum_k \delta_{(k \bmod D)i}\delta_{(k \bmod D)j} \\ &= n\delta_{ij} \\ &= \frac{L}{D}\delta_{ij} \\ &\Rightarrow C_{\text{eq. 6.22}} = 0\end{aligned} \tag{6.16}$$

as $k \bmod D = i$ a total of n times when $i = j$.

However, this construction has off-diagonal Gram matrix elements that are either 0 or 1:

$$\begin{aligned}\cos\theta_{ij} &= \sum_k W_{ik}W_{jk} \\ &= \sum_k \delta_{(i \bmod D)k}\delta_{(j \bmod D)k} \\ &= \delta_{(i \bmod D)(j \bmod D)},\end{aligned} \tag{6.17}$$

which is not equal or close to an equiangular solution: $\cos\theta_{ij} = \cos\alpha,\ \forall i \neq j$. □

Invariance to continuous transformations: proof of Theorem 2

Here we prove Theorem 2: the L_2 cost, initialized from the pathological solution, is invariant to transformations, $\Phi = R \oplus I,\ R \in \mathrm{O}(D)$, constructed as orthogonal rotations applied to any basis subset and an identity transformation on the remaining bases. This shows that low coherence and high coherence configurations are both global minima of the L_2 cost.

Proof of Theorem 2. For an D dimensional space with an $n \in \mathbb{Z}^+$ times overcomplete basis, the pathological case is n orthonormal subsets, each its own orthonormal basis. We label our bases into the sequential subsets of orthornormal subsets by label $W_1, \ldots, W_D, \ldots, W_{2D}, \ldots, W_{n\times D}$. So, bases W_1 through W_D form a full-rank orthonormal basis and this basis is tiled n times. Consider the following partition of the bases: partition $\mathcal{A}$ is the first orthonormal set, i.e. bases W_1 through W_D, and partition $\mathcal{B}$ the remainder of the bases, i.e. bases W_{D+1} through $W_{n\times D}$. If a rotation is applied to all elements of $\mathcal{A}$, only terms in the cost between elements of $\mathcal{A}$ and $\mathcal{B}$ will change. It is straightforward to show that the terms in the cost that have both elements within $\mathcal{A}$ or both within $\mathcal{B}$ are constant since the rotation does not alter the relative pairwise angles.

For $W_i \in \mathcal{B}$, we can write down the terms in the cost which contain itself and elements from $\mathcal{A}R$:

$$\begin{aligned} C_{W_i,\mathcal{A}R} &= \sum_{W_j \in \mathcal{A}} (R^T W_j^T W_i)^2 + (W_i^T W_j R)^2 \\ &= 2 \sum_{W_j \in \mathcal{A}} \text{Proj}_{W_j R}(W_i)^2 \\ &= 2|W_i|^2 \\ &= C_{W_i,\mathcal{A}}. \end{aligned} \tag{6.18}$$

Since the $W_j \in \mathcal{A}$ remain an orthonormal basis under a rotation, the sum of the projections-squared is the L_2 norm-squared of W_i which is constant. Since this is true for every $W_i \in \mathcal{B}$, the entire cost is constant under this transformation. This argument holds for any subset which forms an orthonormal basis and so all orthonormal subsets can rotate arbitrarily with respect to each other without changing the value of the L_2 cost, but the coherence of the matrix does depend on the transformation, Φ. □

6.5 Results on datasets

Mixing-matrix recovery

To investigate how these methods perform in overcomplete ICA, we compare different ICA cost functions and a sparse coding model on the task of recovering a known mixing-matrix from k-sparse data with a Laplacian prior. We compare three classes of overcomplete dictionary recovery methods. The first is a sparse coding baseline [96], the second are modified maximum-likelihood ICA models described in section 6.1, and the final is score matching [95], which is a non-maximum-likelihood method.

Two metrics are combined to compare the known and recovered matrices: ΔP which is a measure of how many basis elements were not recovered, and the median of $p_{\min}$, the distribution of angles between the original bases and nearest recovered bases (see section 6.6 for details). Both metrics are zero for a perfect recovery and we use their average normalized to be between zero and one as a single metric which varies between 0 and 1. For a 32-dimensional data space, we vary the k-sparseness and overcompleteness of the data. For each of these datasets, where dataset size was scaled with mixing matrix size, we fit all models to the data from 10 random initializations, for a range of sparsity weights: λ, if applicable, and then compare the recovery metrics across models.

Fig 6.3 shows the recovery metrics on a 32 dimensional dataset with a mixing-matrices created by minimizing the Soft Coherence cost. Fig 6.3A shows the the normalized average metric as a function of λ across models trained on a 12-sparse, 2-times overcomplete dataset (values where all methods perform well). All methods can recover the mixing matrix, but the L_2 and Score Matching costs perform slightly worse than the modified maximum-likelihood

ICA methods. All methods have a certain range of λ over which they recover the mixing matrix well and have differences in how they fail, for instance sparse coding has a very quick transition to poor recovery compared to ICA methods whose performance tends to decrease slowly as λ moves outside of the optimal range.

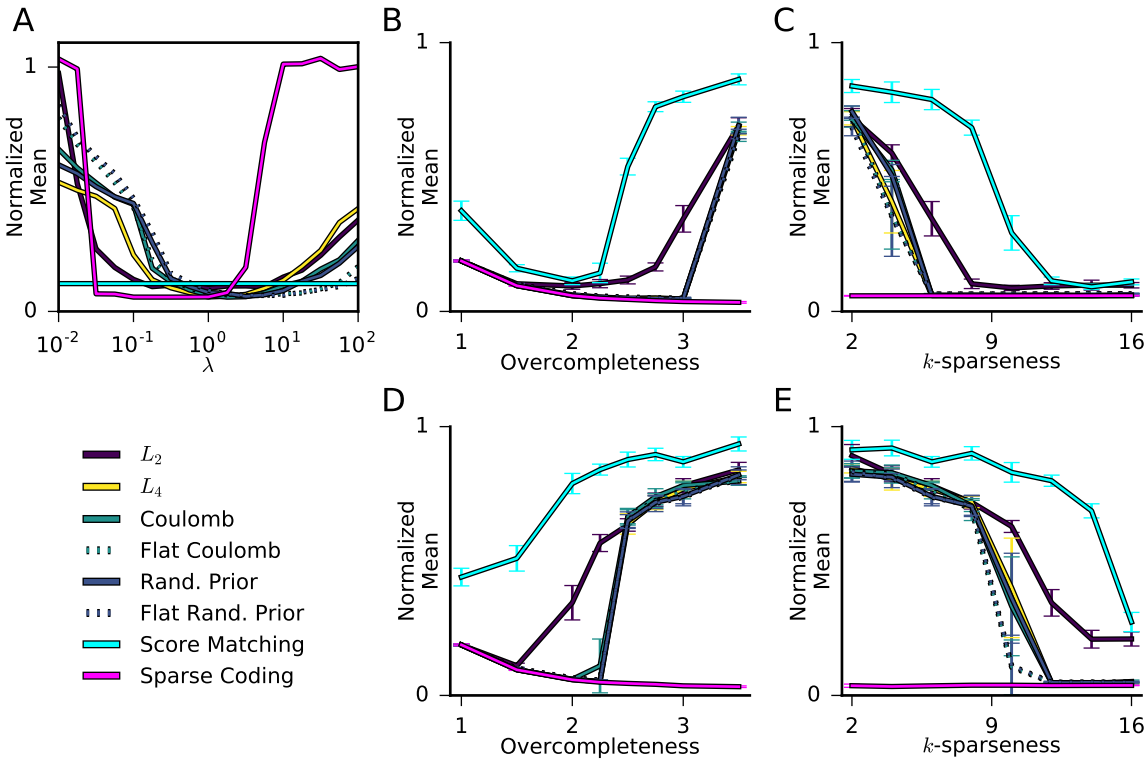

Figure 6.3: Comparison of different coherence control mechanisms in a known mixing-matrix recovery task. All mixing-matrices were generated from the Soft Coherence cost. **A** The mean of the two measures of recovery (lower is better), ΔP and median($p_{\min}$) (see section 6.6 for method for determining null values), for a 2-times overcomplete mixing matrix and $k = 12$. Color and line style are preserved across panels. Since score matching does not have a λ parameter, it is plotted at a constant. **B** Recovery performance ($\pm$ s.e.m., $n = 10$) at best λ as a function of overcompleteness at $k = 12$. **C** Recovery performance ($\pm$ s.e.m., $n = 10$) at best λ as a function of k-sparseness at 2-times overcompleteness. **D, E** Same plots as **B** and **C** at a point where methods do not perform as well: $k = 6$ and 3-times overcomplete.

Fig 6.3B shows how the methods perform as a function of overcompleteness at fixed $k = 12$. Score matching recovers well in a smaller range of overcompleteness as compared to other ICA methods. Besides the L_2 cost, all other ICA methods have nearly identical recovery. The L_2 cost's performance breaks down at lower overcompleteness. All ICA methods fail to recover the mixing matrix once the overcompleteness is large enough while sparse coding continues to be able to recover the mixing matrix. Since the number of bases being recovered changes

as the overcompleteness changes, it is not meaningful to compare the recovery metric across overcompleteness, but it meaningful to compare different models at fixed overcompleteness.

Fig 6.3C shows how the methods perform as a function of k-sparseness at fixed 2-times overcompleteness. Sparse coding performs well at all k-sparsenesses, but the ICA methods require larger k-sparseness. Score matching fails to recover at a lower k-sparseness than other ICA methods. Since the number of bases being recovered is fixed as a function of the k-sparseness, the recovery metric can be compared across k-sparseness and models.

Fig 6.3D-E show the methods in a regime ($k = 6$ and 3-times overcomplete, respectively) where ICA methods struggle to recover the mixing matrix whereas sparse coding continues to recover the mixing matrix well.

In summary, we find different ICA methods have different regimes of performance with score matching and the L_2 cost having the smallest ranges of applicability. Other ICA methods generally have similar performance. Score matching did not always perform as well as other ICA methods as a function of overcompleteness or k-sparseness, although it is a hyperparameter-free cost (no λ hyperparameter). In all cases, the more computationally costly sparse coding was able to recover the mixing matrix more consistently than ICA models. This shows that the linear inference in ICA models is not be able to recover latent representations for highly overcomplete representations.

Experiments on natural images

On a natural images data, we no longer have a ground truth mixing matrix or known prior, but the effects of coherence control on the distribution of bases learned can be evaluated. We train 2-times overcomplete ICA models on 8-by-8 whitened image patches at a fixed value of sparsity across costs found by binary search on λ. The score matching cost has no λ parameter to trade off sparsity versus coherence although it finds solutions of similar sparsity to the value chosen for the other costs. It is known that for natural images data sets, bases learned with ICA can be well-fit by Gabor filters [93]. Hence, we evaluate the distribution of the learned basis by inspecting the parameters obtained from fitting the bases to Gabor filters (see section 6.6 for details).

Our results show that the distributions of angles from the trained ICA models are in line with the theoretical results from section 6.2. Fig 6.4A shows the results of fitting a 2-times overcomplete models to 8-by-8 natural images patches from the Van Hateren database [118]. The L_2 cost has more pairwise angles close to zero compared to the other costs with the L_4 having the smallest coherence. Similarly, as shown in fig 6.4B, the Random Prior and Coulomb costs have lower coherence when the second order terms are removed and behave more similarly to the L_4 cost. For the range of sparsities which were considered, the visual appearance of the bases is similar to results from previous ICA work and similar across costs (L_4 bases are shown in fig 6.4C.

The tiling properties of the learned dictionaries were then visualized in more detail. Fig 6.5 shows the summary of these results for the L_2, L_4 and Flattened Coulomb costs. Fig 6.5A shows the coordinates of the center of the fit Gabor filter. The dimensions and rotation of

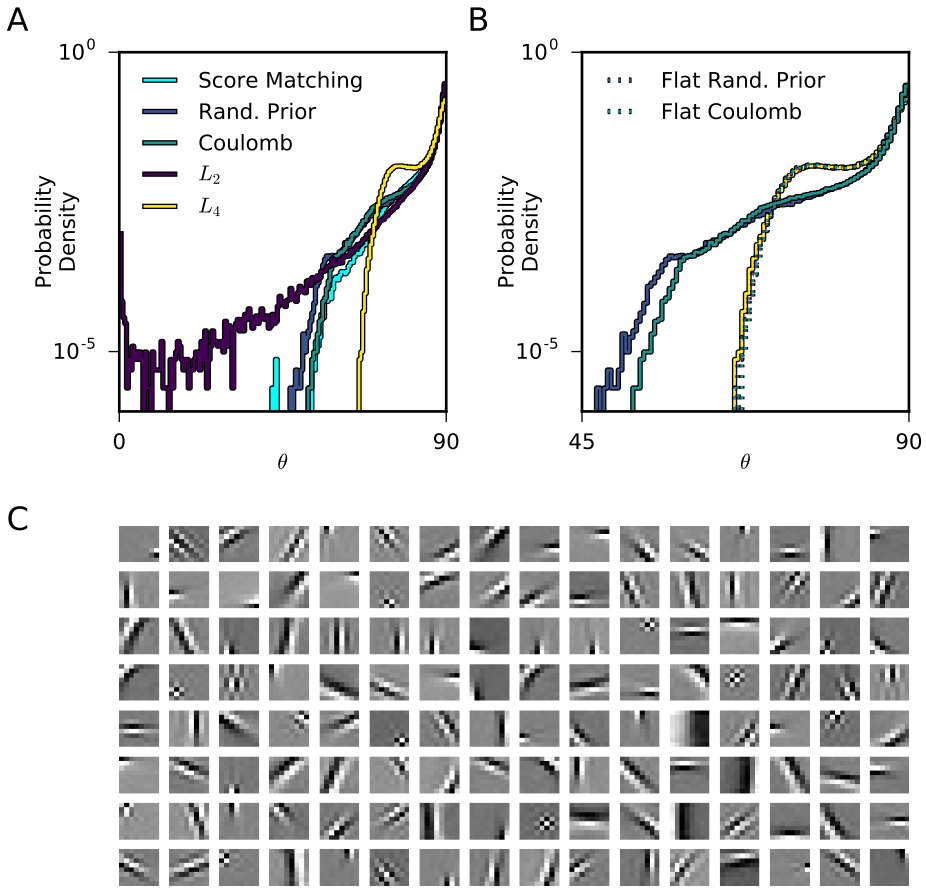

Figure 6.4: Results from fitting a 2-times overcomplete model on natural images. **A, B** Pairwise angle distributions across costs for 8-by-8 natural image patches for a fixed value of sparsity (y-axis log scaled). **B** Comparison between the Random Prior and Coulomb costs and the flattened versions. The L_4 distribution is also shown for comparison. **C** Bases learned with the L_4 cost.

the rectangle represent the envelope widths and planar rotation angle respectively. Fig 6.5B plots the planar rotation angle against the oscillation frequency of the Gabor. Fig 6.5C shows the envelope widths. The radius of the circle is proportional to the oscillation frequency. Qualitatively, the parameters of the fits to the L_2 cost bases are more clustered than the L_4 and Flattened Coulomb costs, which look similar.

6.6 Methods

A repository with code to reproduce the figures is online [119].

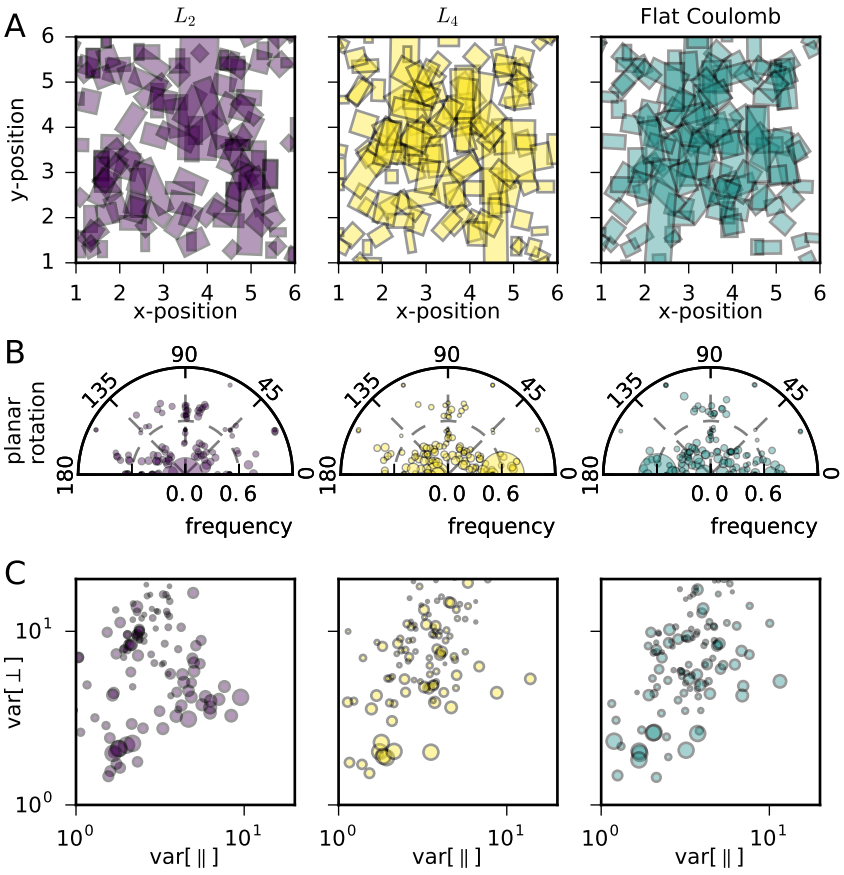

Figure 6.5: Results from fitting two times overcomplete ICA models on natural images with different costs. Color code as in previous figures. **A** Rectangle position: center of Gabor fit in pixel coordinates, rectangle rotation: planar-rotation of Gabor, rectangle shape: envelope width parallel and perpendicular to the oscillation axis. **B** Polar plots of planar-rotation angle and spatial frequency. Marker size scales with geometric mean of envelope widths. **C** Log-scale plot of envelope widths-squared parallel and perpendicular to the oscillation axis. Marker size scales with oscillation frequency.

Previously proposed coherence control methods

Quasi-orthogonality constraint

Hyvärinen et al. [114] suggest a quasi-orthogonality update which approximates a symmetric Gram-Schmidt orthogonalization scheme for an overcomplete basis, W, which is formulated as:

$$W \leftarrow \frac{3}{2}W - \frac{1}{2}WW^TW. \tag{6.19}$$

Reconstruction cost and the L_2 cost

Le et al. [10] propose adding a reconstruction cost to the ICA prior (RICA) as a form of coherence control, which they show is equivalent to a cost on the L_2 norm of the difference between the Gram matrix of the filters and an identity matrix for whitened data

$$\begin{aligned} C_{\text{RICA}} &= \frac{1}{N}\sum_{ij}(X_j^{(i)} - \sum_{kl} W_{kj}W_{kl}X_l^{(i)})^2 \\ \propto C_{L_2} &= \sum_{ij}(\delta_{ij} - \sum_k W_{ik}W_{jk})^2 = \sum_{ij}(\delta_{ij} - \cos\theta_{ij})^2, \end{aligned} \tag{6.20}$$

where W_{ij} is the component of the ith source for the jth mixture , $X_j^{(i)}$ is the jth element of the ith sample, θ_{ij} is the angle between pairs of basis, and δ_{ij} is the Kronecker delta.

The L_2 cost has also been proposed as a form of degeneracy control or coherence control [8, 9]. Equiangular tight-frames (ETFs) are a set of frames (overcomplete dictionaries) which have minimum coherence. The fact that an ETF has minimum coherence is used to motivate the L_2 cost as a form of coherence or degeneracy control. A matrix $W \in \mathbb{R}^{L\times D}$ is an ETF if

$$\sum_k W_{ik}\cdot W_{jk} = \cos\alpha,\ \forall i \neq j \tag{6.21}$$

for some angle, α, and

$$\sum_k W_{ki}W_{kj} = \frac{L}{D}\delta_{ij}. \tag{6.22}$$

The L_2 cost will encourage equation 6.22 to be satisfied, but does not encourage equation 6.21 to be satisfied as we show in Theorem 1.

Random prior cost

Hyvärinen and Inki [7] use a prior on the distribution of pairwise angles to encourage low coherence. The prior is the distribution of pairwise angles for two vectors drawn from a uniform distribution on the n-sphere[2].

$$C_{\text{Random prior}} = -\sum_{i\neq j}\log P(\cos\theta_{ij}) \propto -\sum_{i\neq j}\log(1-\cos^2\theta_{ij}) \tag{6.23}$$

Score Matching

Hyvärinen and Dayan [120] propose a training objective function for non-normalized statistical models of continuous variables called score matching. The score function is derivative of the log-likelihood of the model or data distribution with respect to the data

$$\psi(X;\Theta) = \nabla_X \log p(X;\Theta) \tag{6.24}$$

[2]For both the Random Prior and the Coulomb cost (section 6.2), we regularize the costs and their derivatives near $|\cos\theta| = 1$ by adding a small positive constant in the objective, i.e. $1-\cos\theta_{ij}^2 \to 1+|\epsilon|-\cos\theta_{ij}^2$.

The score matching objective is the mean-squared error between the model score, $\psi(X;\Theta)$, and data score, $\psi_{\mathcal{D}}(X;\Theta)$ averaged over the data, $\mathcal{D}$:

$$J(\Theta) = \frac{1}{2}\int_X p_{\mathcal{D}}(X)||\psi(X;\Theta) - \psi_{\mathcal{D}}(X;\Theta)||^2. \tag{6.25}$$

Model implementation

All models were implemented in Theano [121]. ICA models, with the exception of the Coherence cost, were trained using the L-BFGS-B [122] implementation in SciPy [123]. FISTA [124] was used for MAP inference in the sparse coding model and the weights were learned using L-BFGS-B. All weights were training with the norm-ball projection [10] to keep the bases normalized. A repository with code to reproduce the results will be posted . For ICA models with degeneracy costs, the degeneracy control cost with no sparsity penalty was used as the objective for Fig 6.2.

Datasets

k-sparse datasets Mixing matrices were generated by minimizing the Soft Coherence cost. Data was generated by taking k samples from a diagonal multivariate Laplacian distribution and combining them with the mixing-matrix.

Natural images dataset Images were taken from the Van Hateren database [118]. We selected images where there was minimal motion blur and minimal saturated pixels. 8-by-8 patches were taken from these images and whitened using PCA.

Recovery metrics

Permutation comparison: ΔP If the mixing matrix A is recovered perfectly, W^T will be a permutation of A. To estimate the closeness to a permutation matrix, the matrix $P_{ij} = |W_i \cdot A_j|$ was created. The maximum across rows of this matrix gives the element of A which is most similar to each element of W. The absolute difference from 1, averaged across columns was used as a metric for closeness to a permutation

$$\Delta P = \sum_j \left| \sum_i \delta_{\arg\max_k P_{ik},j} - 1 \right| \tag{6.26}$$

If W was generated exactly from a permutation of A, each column would have one maximum value and ΔP would be zero.

Permutation distribution comparison: $p_{\min}$ The median of the distribution of the smallest angles per row (largest inner product): median($p_{\min}$), of the matrix P_{ij}, measures how well the recovered bases match the bases from the mixing matrix.

$$\text{median}(p_{\min}) = \underset{j}{\text{median}} \arccos(\min_i P_{ij}). \tag{6.27}$$

For both metrics, the null value can be computer by comparing Ws and As where the Ws come from fits to the same prior data, but a different random initialization of A. The normalized mean of these two metrics is used as single measure of mixing matrix recovery.

Fitting Gabor parameters

We fit the Gabor parameters [125] to the learned bases using an iterative grid-search and optimization scheme which gave the best results on generated filters. The learned parameters were the center vector: $\{\mu_x, \mu_y\}$, planar-rotation angle: θ, phase: ϕ, oscillation wave-vector k, and envelope variances parallel and perpendicular to the oscillations: $\sigma^2_{\hat{x}}$ and $\sigma^2_{\hat{y}}$ respectively. Because they are constrained to be positive, the log of the parameters: k, $\sigma^2_{\hat{x}}$, $\sigma^2_{\hat{y}}$ are optimized.

$$\begin{aligned}
\hat{x} &= \cos(\theta)x + \sin(\theta)y \\
\hat{y} &= -\sin(\theta)x + \cos(\theta)y \\
\hat{\mu}_x &= \cos(\theta)\mu_x + \sin(\theta)\mu_y \\
\hat{\mu}_y &= -\sin(\theta)\mu_x + \cos(\theta)\mu_y \\
\text{Gabor}(x, y; \mu_x, \mu_y, \theta, \sigma_{\hat{x}}, \sigma_{\hat{y}}, \phi) &= \exp\left(-\frac{(\hat{x} - \hat{\mu}_x)^2}{2\sigma^2_{\hat{x}}} - \frac{(\hat{y} - \hat{\mu}_y)^2}{2\sigma^2_{\hat{y}}}\right) \sin(k\hat{x} + \phi)
\end{aligned} \tag{6.28}$$

The procedure for finding the best Gabor kernel parameters was to save the parameter set with best mean-squared error after the following iterations:

1. for different initial envelope widths, fit the center location for the envelope to the blurred absolute value of the basis,
2. for different planar rotations and frequencies, numerically optimize the rotation, phase, and frequency of the Gabor
3. for the best fits from above, re-optimize the centers, widths, and phases.

A repository with code to reproduce the fits is online [126].

6.7 Discussion

Learning overcomplete representations of data is often the first stage in data processing pipelines, in computational models of the brain, and many other fields including experimental

neuroscience. In neuroscience, such algorithms serve as models for efficient sensory coding because sensory areas in cortex have more neurons than afferent inputs. For all applications of dictionary learning, computational tractability is an issue, and for this reason, ICA-like algorithms with linear inference are particularly appealing. Here we investigated potential and inherent limitations of linear inference methods in overcomplete dictionary learning.

First, we compared different constraints for degeneracy control of the learned components. Any multidimensional method for extracting signal components needs a form of coherence control to prevent components from becoming co-aligned. Commonly, this is achieved by an orthogonality cost which may be implicit in the model. We show theoretically and by simulation, that the L_2 cost, which successfully achieves orthogonality in the complete case, exhibits pathological global minima in the overcomplete case.

We then suggest two novel costs which do not suffer from pathological minima in the overcomplete case. Among those, the L_4 cost provides solutions with the lowest coherence. The flattened Random Prior [7] and proposed Coulomb costs provide gentle coherence control in the sense that their influence is very small unless components are nearly co-aligned.

Further, we show that the proposed methods of coherence control can extend the regime in overcompleteness and sparseness of the standard ICA methods for dictionary recovery. However, even the improved methods begin to fail when overcompleteness grows beyond two-fold or if the data is k-sparse with small k. The difficulty with recovering the mixing matrix from k-sparse data is counterintuitive at first, because nonlinear inference methods do better as k is decreased. This is likely due to the inability of a linear transformation to recover k-sparse sources without explaining away which will compound as k is decreased.

All told, our study explores power and limitations of linear inference for overcomplete dictionary learning. We note that variations of the sparsity prior have not been systematically explored here, a potential topic of further investigation. The limitations of linear methods to yield highly sparse overcomplete representations might suggest a reason why cortex has dense, local, recurrent networks in sensory areas. The circuitry could provide the substrate for nonlinear inference of sparse sensory representations that possess an overcompleteness which has been estimated to be, depending on species, between ten- and many hundred-fold.

Chapter 7

Discussion

Theoretical neuroscience and machine learning have had an intertwined history. Many ideas in machine learning and particularly deep learning have their inspiration in biological neural networks (although they quickly stray from biological constraints). Today, the field of machine learning often contributes back to neuroscience through new methods for data analysis and intuitions and theories from machine learning and statistics.

This chapter will touch on the future research directions I would like to pursue in the intertwined areas of theoretical and computational neuroscience and machine learning.

7.1 Developing high-throughput, interpretable neural data analysis methods

As scientific datasets grow in size and complexity, our models and methods must also grow. Neural datasets are a microcosm of a larger scientific trend of larger datasets [127, 128].

In neuroscience, the important (potentially correlated) axes of data growth for any method are typically

- dataset collection duration per subject or animal,
- spatial and/or temporal sampling resolution,
- signal to noise ratio,
- number of functional or structural units captured, and
- complexity of the stimulus or behavior.

Measurement techniques which improve along any of these axes can lead to larger dataset sizes as measured in the dimensionality of a single sample and/or the number of samples. As the same time, the miniaturization of many recording methods means that data can

be recorded continuously (often 24/7) and in awake behaving animals at a high functional resolution.

These larger datasets mean that new questions can be asked and answered. For instance, there has been a trend in neuroscience to move away from trial-averaged analyses and towards methods that can do single trial analysis since trial-averaging has been shown to erase interesting neural dynamics [129]. This work has been facilitated both by larger, more precise measurements, but also better statistical and modeling methods.

There is a long history in computational in improving simple, interpretable methods to make them more powerful [130, 131]. This is schematized in fig 7.1 with traditional methods often having lower generalizable predictive performance which is often improved through model tweaks and additions. Although this is a simplification, it representative of many of the methods being developed for neural data analysis. An alternative and complementary approach is to first find methods most accurately relate neural data and stimuli or behaviors. At this point in time, deep networks are the best models of cortical signals recording during complex stimuli and and behavior.

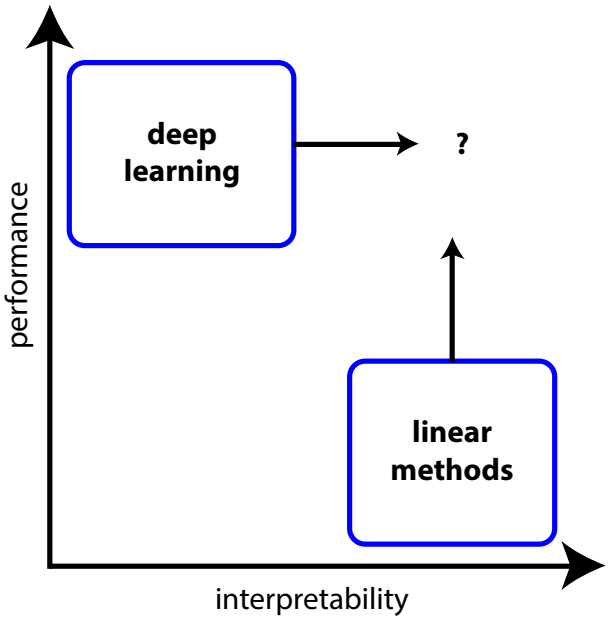

Figure 7.1: Directions for improving analysis methods. Deep learning methods typically have high performance as measured by variance explained or accuracy, but have uninterpretable parameters. On the other hand, linear methods or structured models are often interpretable, but often to not explain the neural data as well as deep networks.

There has been a small amount of work done in interpreting the representations learned by deep networks [132, 133]. Typically this is done in standard network architectures. It is often the case in other models, that structuring the representations can lead to more interpretable representations. These ideas could be applied to deep networks to encourage

them to learn more interpretable feature, e.g. statistically independent representations, factorized representations, or non-negative representations. How these representations form during learning is also not a well understood problem [134]. Understanding the dynamics of learning from both a computational and theoretical point of view would lead to better methods for producing more compressed representations, more structured representations, and potentially more interpretable representations. Finally, structured models and deep models have been combined in ways to create more powerful structured models [45, 65], but this combination could also be used to build more interpretable deep networks.

7.2 Coherence control in deep networks or neural systems

In both machine learning and biological systems it may be beneficial to control the coherence of the representations learned, where the coherence is the largest absolute value of the off-diagonal elements of the Gram matrix.

In the case of machine learning systems, high coherence representations mean that the representational capacity of a set of parameters has not been fully utilized. Similar features are being represented by multiple elements, which could be allocated elsewhere or removed. A high coherence solution may also sample the latent structure of a dataset in a biased way leading to overfitting. We have thus far only considered the effects of coherence control costs in ICA models. A similar cost could be applied to deep networks, graphical models, or dynamical systems models. It seems likely that this cost would be most useful in models with no recurrent, explaining-away effects such as deep networks. It would also be helpful, theoretically, to understand coherence control as a Bayesian prior so that it can be incorporated into models in a more principled way.

Biological systems may also benefit from coherence control for similar reasons. It is likely not the case that all excitatory neurons in cortex are learning different features since cells within a cortical column often have similar tuning [135]. It might be the case that coherence control operates laterally through inhibitory networks. This could operate in place of or in conjunction with explaining-away connections which are often hypothesized to be one of the functions of inhibitory networks in cortex. To implement coherence control in biological systems, the computations required for computing the cost and gradients must be mapped onto neural circuits. This may require addition modification or approximation to the cost function which can be used in machine learning models.

7.3 Biological implementations of machine learning

Understanding how to create biologically plausible implementations of machine learning algorithms has both scientific and practical value. Understand how the brain processes and acts in the world is one of the central goals of neuroscience. Exploring whether the brain is

implementing something similar to machine learning algorithms and what the algorithms' implementations looks like in biological hardware is one of the goals of computational and theoretical neuroscience. Formalizing this relationship often requires us to modify or approximate machine leaning models so that they can be implemented in simplified neural circuitry. There has been limited work in their field that is often highly specialize to one specific algorithm. Understanding the families of approximations and tricks required to translate machine learning algorithms to biological "hardware" is important for understanding how neural circuits may be processing information.

From a practical standpoint, the field of neuromorphic computing is looking to build computational devices which get around Moore's law by using non-Von Neumann architectures [136, 137]. This often leads to systems that are extremely energy efficient as compared to conventional computers, but are still very inefficient as compared to the biological brains. It is not always trivial to implement algorithms developed with a von Neumann architecture in mind in neuromorphic chips; neuromorphic chips often come with constraints on information transfer and processing that are similar to biological systems. Therefore, understanding methods for implementing algorithms in biology is also applicable to understanding how to implement these algorithms in emerging hardware platforms.

7.4 Contributions (past, present, and future) of physicists to neuroscience

(Bio)physicists' training in building analytic and computational models of systems has brought a new and useful set of tools and methods to neuroscience over the past hundred years [138, 47, 139]. The brain is a multi-scale, information processing machine which can extract and store ecologically relevant information from the environment and generate robust movements to interact with the environment. There are physical/energetic and information theoretic limits which can be used to understand how the brain might work. Often these questions can be used to ask whether biological system operate near fundamental limitations of information processing or control [140].

I view the current state of neuroscience similarly to how I imagine physicists thought about the laws of the universe before the Standard Model was developed. There is a vast amount of experimental data to be explained, and very few theories which can explain a diverse set of results. We are looking for theories which are both specific in that they can explain the detailed measurements in the brain and broad enough so that they are not single-purpose theories. Currently, I think that connections to machine learning are poised to provide the most insight into neural systems, but in the long term, I believe that the theoretical foundations of neuroscience will come from a variety of disciplines including physics, mathematics, statistics, and biology.

Bibliography

[1] Muwekma Ohlone Tribe of the San Francisco Bay Area. *Tribal History*. 2015. URL: http://www.muwekma.org/tribalhistory.html (visited on 05/08/2017).

[2] The Sogorea Te Land Trust. *The Sogorea Te Land Trust*. 2017. URL: http://sogoreate-landtrust.com (visited on 05/08/2017).

[3] J. Zylberberg, J. T. Murphy, and M. R. DeWeese. "A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields". In: *PLoS Comput Biol* 7.10 (2011), e1002250.

[4] T. P. Lillicrap et al. "Random feedback weights support learning in deep neural networks". In: *arXiv preprint arXiv:1411.0247* (2014).

[5] R. J. Williams and D. Zipser. "Experimental analysis of the real-time recurrent learning algorithm". In: *Connection Science* 1.1 (1989), pp. 87–111.

[6] M. a. Davenport et al. "Introduction to compressed sensing". In: *Preprint* 93 (2011), pp. 1–68.

[7] A. Hyvärinen and M. Inki. "Estimating overcomplete independent component bases for image windows". In: *Journal of Mathematical Imaging and Vision* 17.2 (2002), pp. 139–152.

[8] I. Ramırez, F. Lecumberry, and G. Sapiro. *Sparse modeling with universal priors and learned incoherent dictionaries*. Tech. rep. Citeseer, 2009.

[9] C. D. Sigg, T. Dikk, and J. M. Buhmann. "Learning dictionaries with bounded self-coherence". In: *IEEE Signal Processing Letters* 19.12 (2012), pp. 861–864.

[10] Q. V. Le et al. "ICA with reconstruction cost for efficient overcomplete feature learning". In: *Advances in Neural Information Processing Systems*. 2011, pp. 1017–1025.

[11] M. J. Wainwright and M. I. Jordan. "Graphical models, exponential families, and variational inference". In: *Foundations and Trends® in Machine Learning* 1.1–2 (2008), pp. 1–305.

[12] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[13] A. Yuille and D. Kersten. "Vision as Bayesian inference: analysis by synthesis?" In: *Trends in cognitive sciences* 10.7 (2006), pp. 301–308.

[14] J. M. Beck et al. "Probabilistic population codes for Bayesian decision making". In: *Neuron* 60.6 (2008), pp. 1142–1152.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[16] E. M. Mugler et al. "Direct classification of all American English phonemes using signals from functional speech motor cortex". In: *Journal of Neural Engineering* 11.3 (2014).

[17] M. S. Lewicki. "A review of methods for spike sorting: the detection and classification of neural action potentials". In: *Network: Computation in Neural Systems* 9.4 (1998), R53–R78.

[18] J. Sohl-Dickstein et al. "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". In: *Proceedings of The 32nd International Conference on Machine Learning*. 2015, pp. 2256–2265.

[19] D. J. MacKay. "Bayesian methods for adaptive models". PhD thesis. California Institute of Technology, 1992.

[20] K. Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological Cybernetics* 36.4 (1980), pp. 193–202.

[21] D. C. Cireşan et al. "Deep, big, simple neural nets for handwritten digit recognition". In: *Neural computation* 22.12 (2010), pp. 3207–3220.

[22] Y. LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[23] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, 2016.

[24] P. J. Werbos. "Applications of advances in nonlinear sensitivity analysis". In: *System modeling and optimization*. Springer, 1982, pp. 762–770.

[25] Y. LeCun. "Une procédure d'apprentissage pour réseau à seuil asymétrique (A learning scheme for asymmetric threshold networks)". In: *Proceedings of Cognitiva*. Vol. 85. 1985, pp. 599–604.

[26] D. Rumelhart, G. Hinton, and R. Williams. "Learning representations by back-propagation errors". In: *Nature* 323 (1986), pp. 533–536.

[27] J. Schmidhuber. *Who Invented Backpropagation?* 2015. URL: `http://people.idsia.ch/~juergen/who-invented-backpropagation.html` (visited on 04/03/2017).

[28] Y. N. Dauphin et al. "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization". In: *Advances in neural information processing systems*. 2014, pp. 2933–2941.

[29] I. Sutskever et al. "On the importance of initialization and momentum in deep learning." In: *ICML (3)* 28 (2013), pp. 1139–1147.

[30] Y. LeCun, C. Cortes, and C. J. Burges. *The MNIST database of handwritten digits*. 1998.

[31] N. Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[32] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *International conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[33] A. M. Saxe, J. L. McClelland, and S. Ganguli. "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks". In: *arXiv preprint arXiv:1312.6120* (2013).

[34] M. D. Zeiler. "ADADELTA: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).

[35] D. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[36] J. Bergstra and Y. Bengio. "Random search for hyper-parameter optimization". In: *Journal of Machine Learning Research* 13 (2012), pp. 281–305.

[37] J. Snoek, H. Larochelle, and R. P. Adams. "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems*. 2012, pp. 2951–2959.

[38] J. S. Bergstra et al. "Algorithms for hyper-parameter optimization". In: *Advances in Neural Information Processing Systems*. 2011, pp. 2546–2554.

[39] A. Doucet, N. De Freitas, and N. Gordon. "An introduction to sequential Monte Carlo methods". In: *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 3–14.

[40] J. Sohl-Dickstein, M. Mudigonda, and M. R. DeWeese. "Hamiltonian Monte Carlo without detailed balance". In: *International Conference on Machine Learning*. 2014.

[41] R. M. Neal. "Probabilistic inference using Markov chain Monte Carlo methods". In: (1993).

[42] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society. Series B (methodological)* (1977), pp. 1–38.

[43] C. J. Rozell et al. "Sparse coding via thresholding and local competition in neural circuits". In: *Neural computation* 20.10 (2008), pp. 2526–2563.

[44] I. Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[45] D. P. Kingma and M. Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[46] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. 2011.

[47] J. J. Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[48] P. Kanerva. "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors". In: *Cognitive Computation* 1.2 (2009), pp. 139–159.

[49] B. Olshausen. *Sparse coding and 'ICA'*. Tech. rep. 2008.

[50] R. P. Rao, B. A. Olshausen, and M. S. Lewicki. *Probabilistic models of the brain: Perception and neural function*. MIT press, 2002.

[51] R. VanRullen, R. Guyonneau, and S. J. Thorpe. "Spike times make sense". In: *Trends in neurosciences* 28.1 (2005), pp. 1–4.

[52] F. Rieke. *Spikes: exploring the neural code*. MIT press, 1999.

[53] K. E. Bouchard et al. "Functional organization of human sensorimotor cortex for speech articulation." In: *Nature* 495.7441 (2013), pp. 327–32.

[54] K. E. Bouchard and E. F. Chang. "Control of Spoken Vowel Acoustics and the Influence of Phonetic Context in Human Speech Sensorimotor Cortex". In: *Journal of Neuroscience* 34.38 (2014), pp. 12662–12677.

[55] K. E. Bouchard and E. F. Chang. "Neural decoding of spoken vowels from human sensory-motor cortex with high-density electrocorticography". In: *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*. IEEE. 2014, pp. 6782–6785.

[56] E. M. Mugler et al. "Direct classification of all American English phonemes using signals from functional speech motor cortex." In: *Journal of neural engineering* 11 (2014), p. 035015.

[57] F. Lotte et al. "Electrocorticographic representations of segmental features in continuous speech". In: *Frontiers in Human Neuroscience* 09.February (2015), pp. 1–13.

[58] W. Penfield and E. Boldrey. "Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation." In: *Brain: A journal of neurology* (1937).

[59] S. Kellis et al. "Decoding spoken words using local field potentials recorded from the cortical surface." In: *Journal of neural engineering* 7 (2010), p. 056007.

[60] X. Pei et al. "Decoding vowels and consonants in spoken and imagined words using electrocorticographic signals in humans." In: *Journal of neural engineering* 8 (2011), p. 046028.

[61] C. Herff et al. "Brain-to-text: decoding spoken phrases from phone representations in the brain". In: *Frontiers in Neuroscience* 9.June (2015), pp. 1–11.

[62] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *International Conference on Learning Representations* (2015).

[63] A. Graves, A. R. Mohamed, and G. E. Hinton. "Speech recognition with deep recurrent neural networks". In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pp. 6645–6649.

[64] D. Amodei et al. "Deep speech 2: End-to-end speech recognition in english and mandarin". In: *arXiv preprint arXiv:1512.02595* (2015).

[65] M. J. Johnson et al. "Structured VAEs: Composing Probabilistic Graphical Models and Variational Autoencoders". In: *arXiv preprint arXiv:1603.06277* (2016).

[66] S. Stober, D. J. Cameron, and J. A. Grahn. "Using Convolutional Neural Networks to Recognize Rhythm Stimuli from Electroencephalography Recordings". In: *Neural Information Processing Systems* (2014), pp. 1–9.

[67] M. Wand and T. Schultz. "Pattern Learning with Deep Neural Networks in EMG-based Speech Recognition". In: *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE* 35 (2014), pp. 4200–4203.

[68] A. Supratak, L. Li, and Y. Guo. "Feature Extraction with Stacked Autoencoders for Epileptic Seizure Detection". In: *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE* (2014), pp. 4184–4187.

[69] D. F. Wulsin et al. "Modeling electroencephalography waveforms with semi-supervised deep belief nets: fast classification and anomaly measurement." In: *Journal of neural engineering* 8.3 (2011), p. 036015.

[70] M. Yang et al. "Speech reconstruction from human auditory cortex with deep neural networks". In: *Telluride Neuromorphic Cognition Engineering Workshop*. 2015.

[71] K. He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[72] D. L. Yamins et al. "Performance-optimized hierarchical models predict neural responses in higher visual cortex". In: *Proceedings of the National Academy of Sciences* 111.23 (2014), pp. 8619–8624.

[73] L. McIntosh et al. "Deep learning models of the retinal response to natural scenes". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1361–1369.

[74] A. S. Benjamin et al. "Modern machine learning far outperforms GLMs at predicting spikes". In: *bioRxiv* (2017).

[75] J. Snoek, H. Larochelle, and R. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms." In: *Neural Information Processing Systems* (2012), pp. 1–9.

[76] I. J. Goodfellow et al. "Pylearn2: a machine learning research library". In: *arXiv preprint arXiv:1308.4214* (2013).

[77] J. R. Wolpaw et al. "Brain-computer interfaces for communication and control." In: *Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology* 113.6 (2002), pp. 767–91.

[78] A. Cruttenden. *Gimson's pronunciation of English*. Routledge, 2014.

[79] C. M. Reed and N. I. Durlach. "Note on Information Transfer Rates in Human Communication". In: *Presence: Teleoperators and Virtual Environments* 7 (1998), pp. 509–518.

[80] D. Marr. "A computational investigation into the human representation and processing of visual information". In: *WH San Francisco: Freeman and Company* 1.2 (1982).

[81] J. von Neumann. "The Computer And The Brain". In: (1958).

[82] L. Chua. "Memristor-the missing circuit element". In: *IEEE Transactions on circuit theory* 18.5 (1971), pp. 507–519.

[83] B. A. Olshausen and D. J. Field. "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583 (1996), p. 607.

[84] D. J. Klein, P. König, and K. P. Körding. "Sparse spectrotemporal coding of sounds". In: *EURASIP Journal on Advances in Signal Processing* 2003.7 (2003), p. 902061.

[85] E. C. Smith and M. S. Lewicki. "Efficient auditory coding". In: *Nature* 439.7079 (2006), pp. 978–982.

[86] A. Hyvärinen and U. Köster. "Complex cell pooling and the statistics of natural images." In: *Network (Bristol, England)* 18.2 (June 2007), pp. 81–100.

[87] N. L. Carlson, V. L. Ming, and M. R. DeWeese. "Sparse codes for speech predict spectrotemporal receptive fields in the inferior colliculus". In: *PLoS Comput Biol* 8.7 (2012), e1002594.

[88] H. Hosoya and A. Hyvärinen. "A hierarchical statistical model of natural images explains tuning properties in V2". In: *Journal of Neuroscience* 35.29 (2015), pp. 10412–10428.

[89] S. Shapero et al. "Optimal sparse approximation with integrate and fire neurons". In: *International journal of neural systems* 24.05 (2014), p. 1440001.

[90] E. Oja. "Simplified neuron model as a principal component analyzer". In: *Journal of mathematical biology* 15.3 (1982), pp. 267–273.

[91] C. Eliasmith and C. H. Anderson. *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press, 2004.

[92] P. Comon. "Independent component analysis, a new concept?" In: *Signal processing* 36.3 (1994), pp. 287–314.

[93] A. J. Bell and T. J. Sejnowski. "The "independent components" of natural scenes are edge filters". In: *Vision research* 37.23 (1997), pp. 3327–3338.

[94] B. A. Olshausen and D. J. Field. "Sparse coding with an overcomplete basis set: A strategy employed by V1?" In: *Vision research* 37.23 (1997), pp. 3311–3325.

[95] A. Hyvärinen. "Estimation of non-normalized statistical models by score matching". In: *Journal of Machine Learning Research* 6.Apr (2005), pp. 695–709.

[96] B. A. Olshausen and D. J. Field. "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583 (1996), pp. 607–609.

[97] D. J. Klein, P. König, and K. P. Körding. "Sparse spectrotemporal coding of sounds". In: *EURASIP Journal on Advances in Signal Processing* 2003.7 (2003), p. 902061.

[98] E. C. Smith and M. S. Lewicki. "Efficient auditory coding". In: *Nature* 439.7079 (2006), pp. 978–982.

[99] M. Rehn and F. T. Sommer. "A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields". In: *Journal of computational neuroscience* 22.2 (2007), pp. 135–146.

[100] J. Zylberberg, J. Murphy, and M. DeWeese. "A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields". In: *PLoS computational biology* (2011), pp. 1–33.

[101] N. L. Carlson, V. L. Ming, and M. R. DeWeese. "Sparse codes for speech predict spectrotemporal receptive fields in the inferior colliculus". In: *PLoS Comput Biol* 8.7 (2012), e1002594.

[102] A. Delorme, T. Sejnowski, and S. Makeig. "Enhanced detection of artifacts in EEG data using higher-order statistics and independent component analysis". In: *Neuroimage* 34.4 (2007), pp. 1443–1449.

[103] G. Agarwal et al. "Spatially distributed local fields in the hippocampus encode rat position". In: *Science* 344.6184 (2014), pp. 626–630.

[104] J.-i. Hirayama, T. Ogawa, and A. Hyvärinen. "Unifying blind separation and clustering for resting-state EEG/MEG functional connectivity analysis". In: *Neural computation* (2015).

[105] B. A. Olshausen. "Highly overcomplete sparse coding". In: *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics. 2013, 86510S–86510S.

[106] H. B. Barlow. "The ferrier lecture, 1980: Critical limiting factors in the design of the eye and visual cortex". In: *Proceedings of the Royal Society of London B: Biological Sciences* 212.1186 (1981), pp. 1–34.

[107] H. Spoendlin and A. Schrott. "Analysis of the human auditory nerve". In: *Hearing research* 43.1 (1989), pp. 25–38.

[108] C. A. Curcio and K. A. Allen. "Topography of ganglion cells in human retina". In: *Journal of comparative Neurology* 300.1 (1990), pp. 5–25.

[109] G. Leuba and R. Kraftsik. "Changes in volume, surface estimate, three-dimensional shape and total number of neurons of the human primary visual cortex from midgestation until old age". In: *Anatomy and embryology* 190.4 (1994), pp. 351–366.

[110] J. L. Northern and M. P. Downs. *Hearing in children*. Lippincott Williams & Wilkins, 2002.

[111] M. R. DeWeese, T. Hromádka, and A. M. Zador. "Reliability and representational bandwidth in the auditory cortex". In: *Neuron* 48.3 (2005), pp. 479–488.

[112] M. S. Lewicki and B. A. Olshausen. "Probabilistic framework for the adaptation and comparison of image codes". In: *JOSA A* 16.7 (1999), pp. 1587–1601.

[113] A. Hyvärinen and E. Oja. "A fast fixed-point algorithm for independent component analysis". In: *Neural computation* 9.7 (1997), pp. 1483–1492.

[114] A. Hyvärinen, R. Cristescu, and E. Oja. "A fast algorithm for estimating overcomplete ICA bases for image windows". In: *Neural Networks, 1999. IJCNN'99. International Joint Conference on*. Vol. 2. IEEE. 1999, pp. 894–899.

[115] T. Strohmer and R. W. Heath. "Grassmannian frames with applications to coding and communication". In: *Applied and Computational Harmonic Analysis* 14.3 (2003), pp. 257–275.

[116] S. Howard, R. Calderbank, and S. Searle. "A fast reconstruction algorithm for deterministic compressed sensing using second order Reed Muller codes". In: *Proc. IEEE Conf. Information Sciences and Systems* (2008), pp. 11–15.

[117] S. Smale. "Mathematical problems for the next century". In: *The Mathematical Intelligencer* 20.2 (1998), pp. 7–15.

[118] J. H. van Hateren and A. van der Schaaf. "Independent component filters of natural images compared with simple cells in primary visual cortex". In: *Proceedings of the Royal Society of London B: Biological Sciences* 265.1394 (1998), pp. 359–366.

[119] URL: `https://github.com/JesseLivezey/oc_ica`.

[120] A. Hyvärinen and P. Dayan. "Estimation of Non-Normalized Statistical Models by Score Matching." In: *Journal of Machine Learning ...* 6 (2005), pp. 695–709.

[121] Theano Development Team. "Theano: A Python framework for fast computation of mathematical expressions". In: *arXiv e-prints* abs/1605.02688 (May 2016).

[122] R. H. Byrd et al. "A limited memory algorithm for bound constrained optimization". In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208.

[123] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open source scientific tools for Python*. [Online; accessed 2017-03-15]. 2001–2017.

[124] A. Beck and M. Teboulle. "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems". In: *SIAM Journal on Imaging Sciences* 2.1 (2009), pp. 183–202.

[125] D. L. Ringach. "Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex". In: *Journal of neurophysiology* 88.1 (2002), pp. 455–463.

[126] URL: `https://github.com/JesseLivezey/gabor_fit`.

[127] *Allen Institute for Brain Science, Allen Brain Atlas*. URL: `http://www.brain-map.org/overview/index.html` (visited on 04/18/2017).

[128] *Collaborative Research in Computational Neuroscience*. URL: `http://crcns.org` (visited on 04/18/2017).

[129] K. W. Latimer et al. "Single-trial spike trains in parietal cortex reveal discrete steps during decision-making". In: *Science* 349.6244 (2015), pp. 184–187.

[130] T. Sharpee, N. C. Rust, and W. Bialek. "Analyzing neural responses to natural signals: maximally informative dimensions". In: *Neural computation* 16.2 (2004), pp. 223–250.

[131] J. W. Pillow et al. "Spatio-temporal correlations and visual signalling in a complete neuronal population". In: *Nature* 454.7207 (2008), pp. 995–999.

[132] J. Yosinski et al. "Understanding neural networks through deep visualization". In: *ICML Deep Learning Workshop*. 2015.

[133] A. Nguyen et al. "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3387–3395.

[134] I. J. Goodfellow, O. Vinyals, and A. M. Saxe. "Qualitatively characterizing neural network optimization problems". In: *arXiv preprint arXiv:1412.6544* (2014).

[135] P. E. Maldonado et al. "Orientation selectivity in pinwheel centers in cat striate cortex". In: *Science* 276.5318 (1997), pp. 1551–1555.

[136] G. Indiveri et al. "Neuromorphic silicon neuron circuits". In: *Frontiers in neuroscience* 5 (2011), p. 73.

[137] P. A. Merolla et al. "A million spiking-neuron integrated circuit with a scalable communication network and interface". In: *Science* 345.6197 (2014), pp. 668–673.

[138] A. L. Hodgkin and A. F. Huxley. "A quantitative description of membrane current and its application to conduction and excitation in nerve". In: *The Journal of physiology* 117.4 (1952), p. 500.

[139] S. P. Strong et al. "Entropy and information in neural spike trains". In: *Physical review letters* 80.1 (1998), p. 197.

[140] S. E. Palmer et al. "Predictive information in a sensory population". In: *Proceedings of the National Academy of Sciences* 112.22 (2015), pp. 6908–6913.