FMRI "MIND READERS": SPARSITY, SPATIAL STRUCTURE, AND RELIABILITY

MELISSA KRISTIN CARROLL

A DISSERTATION
PRESENTED TO THE FACULTY
OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE
BY THE DEPARTMENT OF
COMPUTER SCIENCE
Advisor: Robert E. Schapire

APRIL, 2011

Abstract

Over the last two decades, Functional Magnetic Resonance Imaging (fMRI) has revolutionized the study of the brain. This non-invasive technique produces snapshots of brain activity over time, allowing researchers to literally peer into the mind as it performs everyday tasks like reading or viewing images. Gradually the need has emerged for fMRI analysis techniques that model activity occurring at numerous locations throughout the brain simultaneously, and make predictions about what a person is doing or thinking solely from his or her brain activity, or *"mind read."* Machine learning techniques can accomplish both these goals, and thus have become a popular modeling choice; however, most standard machine learning algorithms were designed for problems in which there are relatively few candidate predictor variables, and the modeling objective is to make accurate predictions.

In fMRI data, the number of predictor variables can be very large, while the likely number of relevant predictors may be quite small. Furthermore, machine learning algorithms are increasingly being employed in the natural sciences, and while accurate predictions can serve to validate scientific models, the end goal of such modeling is usually to *interpret* the models to gain scientific insight. An emerging class of algorithms were designed to address the challenge of learning from and interpreting models with large predictor sets by building *sparse* models in which only a small subset of predictors are used. However, interpretation of such models still poses challenges; in particular, such models are often not *reliable* across datasets, limiting their usefulness as scientific models of brain functioning.

An additional challenge for fMRI modeling is that fMRI data are known to exhibit strong *spatial structure*, especially in the form of spreading of activity across localized areas of the brain. This phenomenon is generally understood, but the properties remain poorly specified and are known to vary by factors such as the person studied (subject), mental task, and brain region considered. Ultimately, better characterization of this spatial structure is warranted because it presents both a modeling confound and an inherently intriguing aspect of neural functioning.

In this thesis, we explore the interaction between prediction, sparsity, spatial structure, and reliability in fMRI models, addressing the following questions:

What is the relationship between prediction performance and model interpretation, specifically model reliability? We perform the first application of the Elastic Net sparse regression technique to fMRI data and find that reliability can be enhanced independently of prediction, particularly by accounting for the known spatial structure in the data.

How do we determine whether a model is reliable? We provide a novel argument that significance testing must be employed when evaluating model reliability, and introduce an evaluation approach that accounts for both the overall activity and spatial structure of a model, substantially affecting reliability estimation and showing that model weight smoothing can improve reliability.

What are the spatial properties of the fMRI response? The Elastic Net models support the hypothesis that the fMRI spatial profile is characterized by distributed clusters of localized activity, and we extend the original framework beyond individual voxels to explore a much larger feature space of spatial clusters, demonstrating the importance of flexible modeling to accommodate the uncertainty of fMRI spatial structure.

How can one tractably produce sparse models in an enormous feature space, such as that of candidate fMRI spatial clusters? We introduce a distributed implementation of the LARS-EN algorithm for solving the Elastic Net, which can exploit a High Performance Computing environment to efficiently search such spaces.

Our findings underscore the tremendous promise computational methods hold for elucidating brain function through fMRI, yet highlight several challenges that must be addressed as these techniques are standardized, refined, and expanded.

Acknowledgments

I have had the rare good fortune of benefiting from the advice of several leading researchers from various fields. First and foremost, I thank my primary advisor, Robert Schapire. A suggestion Rob made about spatial features during the fall of my first year served as the catalyst for the research that comprises most of this thesis, and he has steadily guided me since then. Rob's professionalism and high standards for research, writing, and teaching inspire all of his students to do their best work, and his warmth, caring, and humor make him a trusted mentor.

I was especially privileged to have been introduced to predictive fMRI modeling by two pioneers in the field, Kenneth Norman and James Haxby. Jim's views about spatial structure motivated my subsequent focus on this modeling aspect. Ken advised my early research in all but the most official sense, and his perspective has consistently led me in fertile new directions, as when his thoughts on reliability inspired an entire chapter of this thesis. I am indebted to Ken for the advice he has offered throughout my studies and as a reader on my committee. Likewise, I extend my thank you to Peter Ramadge and David Blei for taking the time to serve on my committee and offering valuable feedback that enhanced the quality of this work.

Intimately involved with this research were my advisors at IBM Research, Guillermo Cecchi and Irina Rish. Guille has played a direct role at every step, from working through the finest details to guiding me as I developed the overall story, and his example and teaching have had an enormous impact on my research and writing. Irina has been my guide through the intricacies of sparse modeling as well as a valued mentor and friend, and I very much appreciate the effort she has invested as a reader on my committee. Also at IBM, my collaborators Rahul Garg, Ravi Rao, and Aurelie Lozano contributed valuable research insight as well as general guidance. My past IBM supervisor, Charles Peck, offered generous advice, and my present supervisor, Richard Lawrence, has been graciously patient and encouraging.

The Princeton Program in Integrative Information, Computer, and Application Sciences (PICASso) generously funded two years of my graduate study, and I thank the PICASso staff members for their personal support of my work. I also received funding through NSF grant IIS-0325500.

Much of my machine learning education has been the product of my association with the Princeton Machine Learning group, all of whom have been generous with ideas and feedback. I especially thank Miro Dudík for his collaboration. I am also grateful to the Princeton Center for the Study of Mind, Brain, and Behavior (CSBMB) for data and computing resources, and the Neuroimaging Analysis Methods (NIAM) group for facilitating engagement with the wider fMRI community. I thank Francisco Pereira for his gracious advice, and the Norman Lab members for supplying a steady stream of insight, especially Greg Detre, who first oriented me to Multi-Voxel Pattern Analysis.

I am proud to have called the Princeton Computer Science department my second home for the last several years. Numerous faculty members significantly influenced my work through coursework and other interactions. The technical staff tirelessly provided computing resources and support, and the administrative staff always lent a friendly helping hand. Melissa Lawson consistently draws on her unrivaled knowledge to answer any question and fulfill any request, no matter how imposing, and visiting with her was always a welcome respite from long days of coding.

The greatest personal reward from this experience has been the friends I have made, to all of whom I am grateful. Several have especially directly influenced my research progress. Xiaojuan Ma, Haakon Ringberg, and Philip Shilane patiently tolerated me as an office-mate and provided more practical advice than they may realize. Michael Desmond, Amir Goren, William Josephson, Bob Kopp, and Frances Perry offered commiseration and gentle prodding. The other Computer Science women, especially Shirley Gaw and Ananya Misra, provided camaraderie, all my office-mates in 313 ensured engaging discussions (work-related of course), and the "ratpack" in my cohort year injected a healthy dose of levity into the graduate school experience.

My deepest gratitude, however, is reserved for my family. I thank my brother, David Carroll, for his much appreciated comic relief, and his wife, Alexandra, for her kindness. My extended family, most enthusiastically my grandfather, and also my grandmother, aunts, uncles, and cousins, exhibited a steady stream of encouragement that sustained my motivation. Finally, no one has provided support to such an extent and in so many ways as my parents, Robert and Mary Carroll. Throughout my academic career, they have worn the hats of advisor, mentor, personal assistant, and countless others, constantly putting my needs ahead of their own. It is no cliché to say that my accomplishments would truly not have been possible without their unfailing support, for which I am most thankful.

For my parents, Robert and Mary,
and grandparents, Edward, Joan, Herbert, and Margaret

Contents

List of Figures

Chapter 1

Introduction

1.1 A Glimpse of the Future

The setting is a courtroom. The defense attorney questions his client, who presents a compelling alibi and convincing grief over his wife's murder. The jury is sympathetic. The prosecutor then stands up to present his cross-examination. The jury is surprised to see the prosecutor quickly dismiss the defendant from the stand; he then requests permission to present a video to the jury. The video shows the defendant lying with his head inside a large, noisy machine. The prosecutor sits by the defendant's side in the video and asks him the same questions he had just answered on the stand: Where was he during the murder? Did he kill his wife? The defendant answers into a noisy microphone, and the jury hears his muffled responses, again professing his innocence. The video ends, and the prosecutor calls his next witness. He is a psychologist, and testifies to being an expert in the use of brain imaging techniques to determine the brain state of an individual; in particular, he is an expert in using brain images for lie detection. The expert presents colorful images in the shape of a head, which he says visualize the patterns of activity in the defendant's brain while he was answering the questions in the large machine. The expert then says that there is a 99% chance that the observed pattern of activity is consistent with deception. In other words, it is nearly certain that the defendant was lying while answering the questions. The jury

> are sent to the deliberation room, in which they must decide the fate of the defendant, taking into account the video and the interpretation of the expert witness .

It is possible that a scenario similar to that described above will become commonplace within our lifetimes, due to advances in the neuroimaging field of Functional Magnetic Resonance Imaging (fMRI) and the impressive "mind reading" capabilities afforded by state of the art computational and statistical techniques. The potential uses for mind reading tools are limitless, so there is naturally tremendous interest in developing and perfecting them; however, perhaps more enticing is the potential to use these techniques to gain a firm understanding of how the brain works. This elusive goal is both intuitively fascinating and a pre-requisite for building truly practical devices. fMRI lie detectors are not yet generally admissible in court, in part because our limited understanding of the brain makes it difficult to argue definitively that these methods measure what is claimed, especially when a life is at stake. This thesis will discuss ways we might better exploit fMRI, with the ultimate goal of making discoveries about brain function.

1.2 Background

1.2.1 Functional Magnetic Resonance Imaging

fMRI exploits properties of blood to infer the pattern of brain activity during a snapshot in time. Specifically, when the underlying units of brain activity, neurons, engage anywhere in the brain at any time, oxygen is needed to fuel the activation. This oxygen is transported by the blood; more precisely, the hemoglobin molecules in the blood transport oxygen molecules. Neural activity thus triggers increased blood flow to the surrounding area, and, locally, the ratio of oxygenated (transporting oxygen) to deoxygenated blood increases, a phenomenon that has been known for over 100 years [79]. The increase of this ratio in response to neural activation is well-characterized and is known as the Hemodynamic Response (HR), parameterized by a Hemodynamic Response Function (HRF). A canonical HRF, showing the response over time, is shown in Figure 1.1 (from [38]).

In 1990, Ogawa et al. [66] revolutionized the field of brain imaging by exploiting properties of hemoglobin to facilitate visualization of this hemodynamic response. The

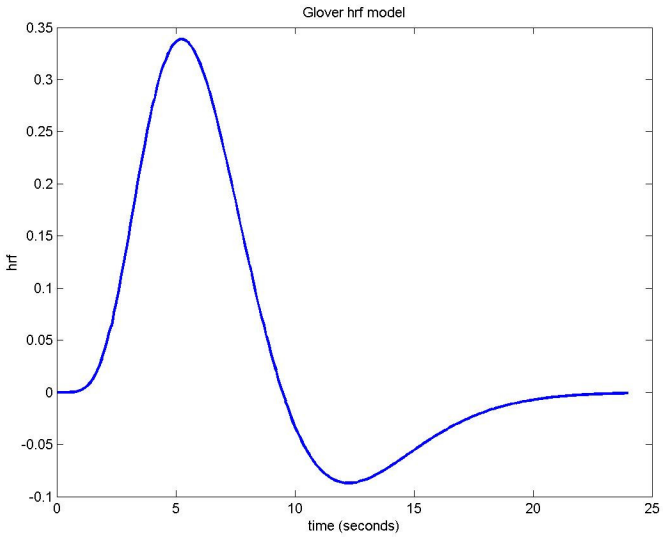


Figure 1.1: Canonical Hemodynamic Response Function (HRF), from [38].

hemoglobin molecules transporting the oxygen contain iron atoms, and their configuration is slightly altered when an oxygen molecule is attached. Since the iron is magnetic, the magnetic signal or Magnetic Resonance (MR), of the hemoglobin molecule differs between the oxygenated and deoxygenated states. The authors demonstrated that a Magnetic Resonance Imaging (MRI) scanner can measure these changes in the blood MR, known as the Blood Oxygen Level-Dependent (BOLD) signal, and hence infer the hemodynamic response. This scanner is essentially a large, noisy magnet. As described in the opening anecdote, to produce an image, the subject lies in the machine and the scanner reads the BOLD signal in his or her brain while the subject is engaged in some activity. Studies have confirmed the relationship between the neural HR and BOLD signal [55], allowing BOLD to serve as a proxy for direct measurement of neural activity. fMRI thus differs from standard structural MRI in that it provides snapshots of activity over time, while the subject is engaged in an activity of interest. This scanning technique is considered non-invasive, that is, no side effects for the subject have been found, assuming certain safety precautions are taken. This non-invasiveness, along with desirable properties of the images derived from them, have resulted in this fundamental technique developing over the last 20 years into a standard paradigm for investigating neurological properties.

At the conclusion of a scanning session, the BOLD signal measurements across the brain are combined into a dataset of brain images over time. Each image is three-dimensional; just as two-dimensional images are discretized into *pixels*, three-dimensional images are discretized into *voxels*. In typical fMRI scans, each voxel corresponds to a region of roughly $2\text{mm} \times 2\text{mm} \times 2\text{mm}$, though this size is often adjusted by the experimenter. For a typical scan, the images will be produced by taking 2D images at approximately 34 positions, or slices, from the front to the back of the brain. Each of these 2D images is typically on the order of 64×64 pixels, leading to a 3D image of size $64 \times 64 \times 34$ voxels. Given the brain shape, however, most of these voxels will not correspond to meaningful brain locations. The specific number of valid brain voxels in a typical scan varies depending on the size and shape of the subject's brain, the exact spatial resolution of the scan, and processing techniques applied to the raw image, but will usually be on the order of 10^4 voxels. A series of these 3D images is taken over the course of the experiment. The time between images is known as the Time to Repetition (TR), and images taken over time are often referred to as *TRs*. There is usually a trade-off between the spatial resolution of the images, and the temporal resolution of the TRs, but a typical TR is approximately once every 1-2 seconds. The number of images in the dataset thus depends on the TR and the duration of the scanning session. Since scanning requires the subject to lie in a very tight and noisy space, durations are usually minimized. Thus, the number of TRs tends to be quite small relative to the number of voxels, and is usually on the order of hundreds. A standard experimental design involves the subject engaging in some behavioral or cognitive (mental) task of interest multiple times, so as to produce sufficient examples from which to detect statistical patterns. These repeated sessions are called *runs*. The number of runs can vary significantly depending on the goals of the experimenter.

Once the scanning session is complete and the dataset of images over time (binned into runs) exists, considerable processing of the data must take place before the data can be analyzed, since numerous noise sources confound the already weak BOLD signal. The noise sources include physical properties of the scanner itself, properties of conversion of the signal from Fourier to real space, movement on the part of the subject, and interference from the subject's breathing and heart rate. Sophisticated techniques to remove these various noise artifacts have been developed and included in standard pre-processing packages, notably AFNI [3] and FSL [85].

1.2.2 Applications and Analysis: General Linear Models (GLM)

Once the pre-processing has been performed, there are a nearly unlimited variety of statistical approaches that can be used to derive insight into neurological functioning. The canonical question researchers seek to answer is: What patterns of brain activity are most strongly associated with a behavioral or mental task of interest? More specifically, researchers seek to find those voxels or sets of voxels that are most strongly correlated with a signal representing the strength of an internal or external task or stimulus. The task or stimulus can be any phenomenon scientists seek to better understand, such as viewing images of different types of objects, reading sentences, pressing a button, performing arithmetic computations, or watching a sad movie.

The field that seeks to match fMRI brain images to mental tasks or behaviors is known as *brain mapping*. News stories in the popular media now routinely feature headlines announcing some behavior for which the "brain region" responsible has been determined. In most cases, these results were yielded from fMRI experiments. The number of scientific hypotheses generated and, in many cases, supported by the technique is staggering. Some phenomena that have been well-studied using fMRI include processes underlying decision making [18], understanding another person's point of view [82], and, still only vaguely understood, falling in love [27].

The simplest approach to discovering such associative patterns from fMRI is to apply standard inferential statistics. The most basic model, which is applicable for tasks involving two conditions, for example viewing a face or not viewing a face, is a t-test, which measures whether the activation levels observed at a given voxel are differently distributed within the two conditions. The generalized version of a t-test is the General Linear Model (GLM) [46] approach, which derives a model of each voxel's activation from the various task or stimulus signals at a given time (known as the *design matrix*), and returns a model significance estimation reflecting the strength of the association between voxel and task. If there are K tasks or stimuli, N TRs, and M voxels, the GLM is computed as:

$$\mathbf{Y} = \mathbf{XW} + \mathbf{E}, \tag{1.1}$$

where $\mathbf{X}$ is a $T \times K$ design matrix, $\mathbf{W}$ is a $K \times M$ matrix of regression coefficients, and $\mathbf{E}$ is a $T \times M$ error matrix, typically modeled as a low-order auto-regressive process.

While there has been extensive effort to enhance the statistical validity of GLM approaches, numerous challenges remain. Recently, Vul et al. [94] revealed a number of peer-reviewed findings in the literature that may suffer from statistical confounds induced by the modeling approaches. However, despite such limitations and the simplicity of these techniques, GLM-based models have been the most common fMRI modeling approach used for most of the last approximately 15 years.

1.2.3 Spatial Structure and Statistical Parametric Maps (SPM)

GLM-based techniques measure the *univariate* relationships between individual voxels (or groups of voxels) and the task of interest. Their use clearly involves performing a very large number of significance tests, so any significance estimation should be adjusted to account for the large number of comparisons performed. In the statistics literature, standard techniques exist for multiple comparison adjustments, the most common of which is the Bonferroni correction [1].

However, standard multiple comparison adjustments are too conservative for fMRI data because they fail to account for the *spatial structure* in such data. More specifically, the voxel (commonly 8mm^3) is almost never the relevant unit of analysis, due to the lack of correspondence between the voxel grid and underlying anatamo-functional spatial regions. There is both microscopic structure, in that millions of neurons are contained within a single voxel, and macroscopic structure, in that broader regions of activity frequently share a response profile. The spatial structure is further obscured by the diffuse nature of the vascular hemodynamic response, which has a point spread dispersed over several voxels. Factors such as head movement and, in multi-subject analysis, differences in brain size and shape, further smear the signal. To mitigate the effects of these factors, data are typically *smoothed* prior to analysis with a simple Gaussian kernel of fixed variance within the range of $4-8\text{mm}^3$. While it is known that this kernel is suboptimal [28], the spatial smearing effect is not well quantified, necessitating approximation.

In fact, this somewhat poorly understood spatial structure presents many open research questions itself. At the lowest level, the optimal smoothing parameters are sought. At a

higher level, spatial properties of the response can be used to gain further insight into the nature of both neuronal response and psychological functioning. For instance, when viewing a male face versus a female face, is the spatial scale at which the strongest contrasts in activation are occurring smaller than when viewing a human face versus a cat face? Do these spatial properties vary depending on the region of the brain, or the subject?

While these questions are intriguing, on a more practical level, the spatial properties of fMRI create some difficulties in analysis; for instance, due to the presence of this spatial structure, voxels exhibit a high degree of spatial auto-correlation, meaning nearby voxels are highly correlated. Since standard corrections for multiple comparisons assume all tests are independent, this inter-dependence between voxels leads to over-penalization. A technique developed to address this spatial auto-correlation is Statistical Parametric Mapping (SPM) [32], in which a GLM analysis is performed and significance is adjusted using Random Field Theory (RFT) [2]. This model assumes that the GLM error term $\mathbf{E}$ is being generated by a process that can be approximated by a random field with a multivariate Gaussian distribution. This correction essentially views the spatial map as a mixture of independent "blobs," which is appropriate considering both the known spatial structure and the spatial structure imposed by smoothing. This RFT-based adjustment results in a more balanced number of independence corrections, making SPM a standard technique for producing univariate statistical significance maps.

1.2.4 Rationale for Predictive Model Building

One very active target of fMRI research is *visual object recognition*, the processing underlying viewing of images of different types of objects. Numerous studies have led to the discovery of brain areas that appear to selectively respond to images of faces (FFA) and places such as houses (PPA) [51]. These initial findings were so intriguing and encouraging that researchers began seeking localized areas associated with even more specific objects, such as shoes, chairs, or scissors. The tendency towards more specific functional localization began to prompt unwelcome comparisons of such approaches to *phrenology* [92], a discredited 19th century theory that postulated that personality traits corresponded to specific regions of the brain, which could be inferred by feeling bumps on an individual's skull.

In 2001, Haxby et al. [43] published a seminal paper showing clear evidence that findings of localization for such specific tasks might not be capturing the full story. The authors had subjects view images of 8 types of objects, including faces, houses, cats, scissors, and bottles. Images of one type of object were viewed over blocks of TRs, which were repeated over 12 runs. The authors then used a standard technique, split-sample correlation, to analyze the data. The brain images were averaged within blocks over all TRs in half of the runs, and all blocks in the other half, resulting in 16 images corresponding to activity patterns for the 8 objects in 2 average run sets. The authors measured the correlation between each of the 8 images in one run set and each of the 8 images in the other run set. Furthermore, they measured this correlation not just for the entire image, but for certain regions hypothesized to selectively respond to particular objects. The authors found that there was a strong enough correlation between fMRI images to determine when a pair of fMRI images corresponded to the same object class, even among those brain regions that reached their maximal signal in response to a *different* object class. This finding underscored the point that attempting to use univariate statistics to discover highly localized brain regions associated with a certain task may fail to account for the *distributed* information present throughout the brain. In addition, patterns were consistent enough across runs that, using just the simple technique of examining cross-correlations, the researchers were able to make retrospective *predictions* about the type of object being viewed when an image was scanned. In a sense, the researchers had effectively "read the mind" of subjects by inferring their mental state entirely from their brain image.

In 2005, Kamitani and Tong [50] used a predictive technique to reveal further insight into the nature of the neural response. They explored brain images associated with the viewing of lines at different orientations. The neural response to this task is actually well-characterized at the neuronal level, and is known to occur at the scale of structures called neuronal columns, of which there are many within a typically-sized fMRI voxel. Given the poor spatial resolution of fMRI relative to the known scale of functioning, the research community had generally assumed that this signal could not be observed with fMRI. Certainly, this limitation had been the case when standard univariate modeling approaches were employed. Kamitani and Tong, however, instead used a technique, Linear Discriminant Analysis (LDA), to build a *multivariate* model that incorporated information from all of the voxels in a region. They trained the model to learn a pattern among the voxels that co-varied with the line orientation, and used the model to predict

what line orientations were being viewed when a separate set of images was scanned. Their model proved to be excellent at predicting the line orientation, despite using voxels as predictors. They showed that this result was achieved due to exploitation of slight discrepancies in the activation levels, reflected in the BOLD signal, across voxels, due to a mismatch between the locations of the arbitrary voxels and the real underlying neural columns. The multivariate model was able to learn to associate the specific pattern of activity across all of the voxels at once with the line orientation. Univariate models consider only one voxel at a time and hence are unable to learn such multi-voxel patterns. As discussed by Norman et al. [64], multivariate techniques essentially provide greater sensitivity to a weak and noisy true signal.

Thus, Haxby et al. found that patterns consistent across runs and distributed throughout the brain exist and might not be discovered using univariate approaches, and Kamitani and Tong used techniques that incorporate data from multiple voxels to make predictions. Furthermore, Kamitani and Tong pointed to the strong prediction performance achieved by their model as evidence that their model had captured a pattern truly associated with the task of interest. Any scientific model should likewise be *valid*, that is, correspond to some known ground truth or an accepted approximation. In fMRI, the ground truth activation pattern is unknown, so a cognitive or behavioral state can be used as a proxy, and the extent to which the state is predicted serves as a measure of validity that is intuitive, easily interpreted, and relatively straightforwardly quantified, making it quite appealing to scientists.

The notion of making predictions from the brain data, or "mind reading," can also lead to creative uses that yield some fascinating discoveries about the nature of neural functioning. Polyn et al. [76] asked subjects to memorize lists of words from different categories and, in a separate step, recall as many words as they could off the top of their head. Scans were conducted during both the first (learning) phase and the second (recall) phase. The authors trained models from the images in the learning phase that predicted (retrospectively) the category of the word being learned at a given time; however, the researchers then used the same model to make predictions from the *recall* data, and were able to successfully predict the category of word being recalled just *before* the subject recalled it. Thus prediction performance was used to demonstrate that the activity evident when recalling a word is consistent with the pattern of activity when the word was learned, implying that when remembering something, one essentially re-instates a brain

state similar to that present when the item was learned. Another creative application is the use of predictive techniques to reveal sub-conscious processes, such as the processing of images of which the subjects were not consciously aware [44].

Predictive models can also reveal insight into the encoding of concepts. Mitchell et al. [62] used a two-step learning procedure in which subjects were scanned while thinking about objects represented by a set of nouns. The nouns were converted to a representation in a basis defined by their frequency of co-occurrence with a pre-defined set of action verbs, and the patterns of brain activity associated with these basis features were learned. Using this basis set, the patterns could be generalized and used to predict brain activity when thinking of nouns that were not used for training. The strong prediction performance was used as a measure of validity of the verb-defined encoding basis hypothesis, suggesting that we may indeed encode object concepts based on the types of actions typically performed on them. This conceptual decoding has also been applied to visual data. Kay et al. [52] represented complex visual stimuli in a basis set of spatial filters that captured statistical properties of the images, including space, orientation, and spatial frequency. They trained models of brain activity along these spatial dimensions from a set of training images, and found that the models generalized to images not used for training. Strong prediction performance not only implies the potential existence of a visual decoder that can "read out" an image being viewed from a subject's brain activity, but serves as partial validation of hypotheses regarding the spatial filter set used by the brain to encode images.

Thus, there has been considerable interest in exploiting predictive modeling techniques to gain insight into aspects of neurological functioning that can be difficult or impossible to study without such approaches. Naturally though, the very notion of "mind reading" is intuitively enticing, even to non-scientists. From a practical perspective, the ability to predict someone's thoughts directly from their brain activity could revolutionize fields like medicine, in which neuro-prostheses can aid disabled individuals, although the state of the art in this field does usually rely on neuroimaging techniques with greater temporal resolution than fMRI. From a business perspective, discoveries that sub-conscious processes can be detected using these techniques provides plenty of opportunities in the nascent field of neurologically-based marketing techniques (Neuro-marketing). Finally, as discussed in the opening anecdote, several groups, including the federal government, have a strong interest in developing accurate fMRI-based lie detectors. Investment in

related research is well underway, and early results present moderate promise [20], but the admissibility of fMRI lie detection evidence in court is being debated among legal scholars.

1.2.5 Predictive Modeling Approaches

The broad approach of using multivariate predictive models of mental states from fMRI data is sometimes referred to as Multi-Voxel Pattern Analysis (MVPA) [64] to differentiate it from the more traditional GLM-based approaches of predicting univariate voxel activity from task-related design matrices. The term MVPA thus refers more to the spirit or end goals of the modeling approach and less to the specific modeling techniques employed. However, *machine learning* methods, which produce multivariate predictive models, are a natural fit for this domain.

The standard Machine Learning framework involves seeking patterns among a set of training examples, or *instances*, that are represented by vectors of *features* or predictors, that yield high prediction of a set of instance labels (classification) or a real-valued label vector (regression). Machine learning researchers have developed countless algorithms and modeling strategies for learning models from a set of labeled data and making predictions on held-out data. Some of the earliest direct applications of the machine learning framework and algorithms to make predictions from fMRI data include the following: Cox and Savoy [19] used Support Vector Machines (SVMs) to predict the category of object viewed from distributed brain voxels; Hanson et al. [42] used Neural Networks to build a more sophisticated predictive model from the Haxby et al. [43] data; and Mitchell et al. [61] evaluated several machine learning-based approaches, including Gaussian Naive Bayes, SVMs, and k-Nearest Neighbor.

The commonalities across all these approaches, which differ primarily in the specific learning algorithm employed, are in the representation of the data and the model evaluation procedures. Figure 1.2 illustrates the most typical paradigm used to frame fMRI mental state prediction in the Machine Learning paradigm. To generate the dataset used for training, each TR becomes an instance, each brain voxel becomes a feature, and the task conditions or continuous response of interest over time serves as a label. Thus, predictions are made from *individual* voxels at *individual* TRs, allowing characterization of the instantaneous mental state of a subject. Predicting from individual voxels potentially

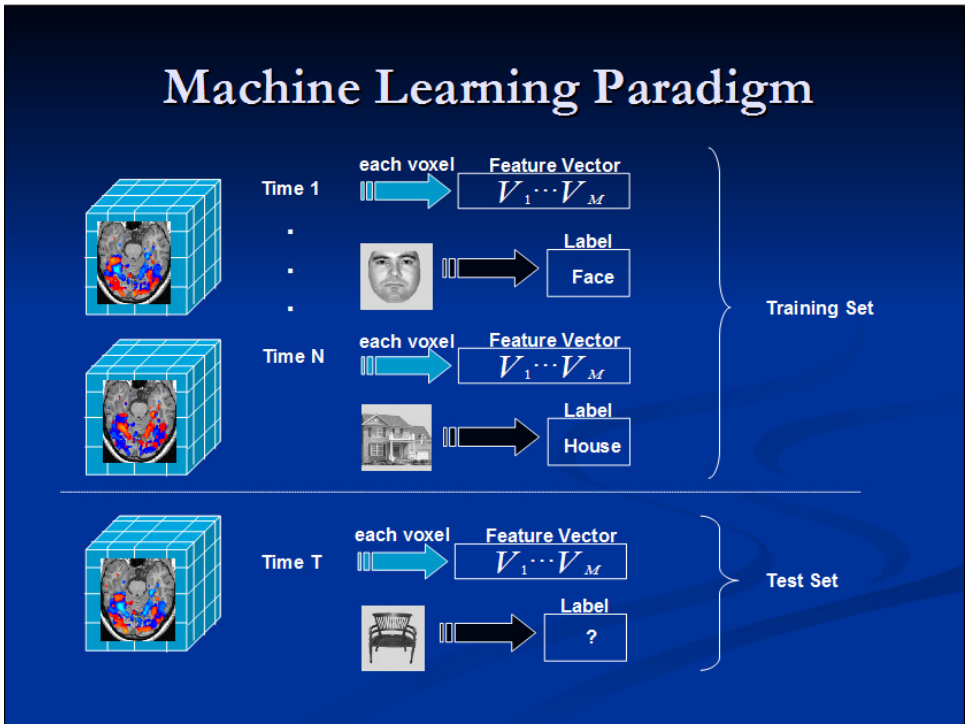


Figure 1.2: Prototypical Machine Learning Approach to fMRI Predictive Modeling.

enables recovery of very specific signals, such as line orientations. Unfortunately, since subjects' brains are different shapes and sizes, most modeling across subjects requires applying some form of smoothing to the data. Therefore, a drawback of the sensitivity of MVPA models is that models are usually limited to within individual subjects [64].

Classification or regression is employed, depending on whether the behavioral or mental condition predicted is discrete, such as viewing one of several types of objects, or continuous, such as a real-valued score reflecting how strongly some physical stimulus is being applied. Prediction is typically defined as the percent *accuracy* for classification, or *correlation* (e.g., Pearson) with response for regression. Machine Learning requires both a *training dataset*, from which the model is learned, and a *test dataset* on which prediction performance is evaluated. Ideally, these two datasets will be similar enough that prediction is possible, yet not so similar as to confound the evaluation. A standard approach in the machine learning literature is to randomly sub-divide the available data; however, data from TRs adjacent in time are not independent, since there is a strong degree of temporal correlation in fMRI. Therefore one seeks to avoid testing models on TRs adjacent to any TRs used for training, precluding the use of random assignment.

Fortunately, the division of the experiment into runs addresses this confound by providing natural data subsets. The most common training approach is "leave-one-run-out" evaluation, in which models are trained using all runs except one and tested on the *held-out* run's data. This procedure is then repeated, with each run serving as the held-out set, and prediction performance is aggregated over all such evaluations.

1.2.6 Feature Selection

The resulting training dataset is of size ~ 500 instances by $\sim 30,000$ features, making the feature set very large relative to the number of training instances. In the machine learning literature, this *high-dimensional* scenario is known to lead to *over-fitting*, in which models are overly sensitive to noise in the training data and fail to generalize to test datasets, hurting prediction performance. In such cases, it is often desirable to reduce the size of the feature set, which may not only mitigate the effects of over-fitting, but also leads to more *parsimonious* models. In the 14th Century, the philosopher William of Occam stated that a scientific model should capture the true phenomenon as succinctly, or parsimoniously, as possible, a principle known as Occam's Razor. In our models, we especially seek to find voxels (or groups of voxels) that are *relevant*, or most strongly correlate with a mental task of interest, with the inherent assumption that most voxels are *irrelevant* for a given task. A parsimonious model should therefore include parameters only for the relevant voxels.

This need to achieve model parsimony is part of a larger challenge that emerges from a slight disconnect between the historical basis for the modeling algorithms, and the present modeling goals. Most of the common state-of-the-art statistical learning algorithms were motivated by engineering applications, such as text or image processing, in which accurate prediction performance was the only goal. In these cases, model properties, such as parsimony, are irrelevant as long as the model achieves good prediction performance on held-out data. However, since prediction is a compelling measure of validity, such algorithms are increasingly being applied in many of the natural sciences, for example genomics, with the goal of understanding the underlying natural system. In such cases therefore, the ultimate objective is not necessarily prediction, but *interpretation* of the model to glean insight. Classic machine learning algorithms were designed to achieve highly accurate prediction performance without regard to model interpretation.

In cases in which model parsimony is desired, this disconnect between the algorithm objective and the modeling goals is especially germane. Machine learning algorithms directly optimize prediction of the *training* set and therefore do not directly consider generalization to unseen data. An optimal model of the training data will often exploit as much information as possible by assigning weights to all or most predictors. While frequently the optimal model is naturally parsimonious, most classic machine learning algorithms do not explicitly seek parsimony, so other approaches must be used if further feature set size reduction is required.

The most intuitive method for reducing the feature set size is to select a subset of the features prior to training using a *feature selection* method. The most basic form of feature selection performs a univariate test for inclusion of every voxel, for instance Pearson correlation between the voxel time series and the response time series, and retains only the k highest-scoring voxels in the feature set. Recall, however, that we seek to build *multivariate* models due to the increased sensitivity of such models to obscured patterns in the data. By employing univariate techniques to prune the feature set, we may be throwing out some of the information that is only revealed through multivariate models. Pereira and Gordon [73] describe several other limitations of this approach, including the corrections that must be performed to adjust for multiple comparisons. In addition, univariate feature selection techniques require choosing an appropriate number k of voxels to select, which can be highly computationally intensive, as described below.

1.2.7 Regularization and Sparse Modeling

An alternative to methods that pre-select features using some criterion prior to modeling, sometimes referred to as *filtered feature selection*, is to select the features within the modeling stage, known as *embedded feature selection*, using some multivariate information to inform the pruning. These approaches are derived from two fundamental model properties: bias and variance. The *variance* of a model is generally a measure of how sensitive the model is to randomness in the data. If parameters are learned for the exact same set of features from two datasets that are identical except for some small perturbations, variance captures the consistency of the model across the two datasets. *Bias* is a measure of the divergence between the prediction of the model on training data, and the target training label to be predicted.

Models that incorporate a very large number of features tend to be over-fit to the data, depending on the number of training instances and other data properties. These models actually exhibit very low bias, because the large number of parameters can yield a model that approximates the training label quite precisely; however, parameters learned for the same features on another dataset, such as data from another run, tend to be quite different, making the variance quite high. It is this model variance that leads to the poor generalization performance. Smaller feature sets, in which there are fewer parameters to adjust, naturally tend to yield less variable models. From the training point of view, however, these models do not as accurately capture the true generating signal (for the training data), and hence the model bias is higher. Yet while the bias for the *training* data is greater, a slight sacrifice in this accuracy is warranted if the model has better generalization, and thus likely better captures the truly consistent signal. To reduce over-fitting, therefore, methods can seek a trade-off, in which model simplification increases bias yet decreases variance. To force the learning algorithm to increase bias, which is contrary to its typical goal, the method most commonly used is *regularization*. With regularization, a constraint is placed on the solution in a way that forces "simpler" models with lower variance. To illustrate, consider the standard linear regression objective with N examples and M predictors:

$$\min \sum_{n=1}^{N} (\mathbf{y}_n - \beta^T \mathbf{X}_n)^2 \tag{1.2}$$

where we have N data points, $\mathbf{y}$ is our response vector (e.g., mental task), $\mathbf{X}$ is our data (e.g., brain images), and we seek to learn β that minimizes the Residual Sum of Square error. Now consider the following modified version of Eq. (1.2):

$$\min \sum_{n=1}^{N} (\mathbf{y}_n - \beta^T \mathbf{X}_n)^2 \quad \text{s.t.} \quad \sum_{i=1}^{M} \beta_i^2 \leq s. \tag{1.3}$$

In Eq. (1.3), an additional constraint has been imposed on the solution, specifically that the sum of the squares of the real β values must not exceed value s. In other words, models trained with this objective will have smaller weights for most parameters. This form of regularization is *ridge regression* [45], the most common form of regularization. The constraint imposed by ridge is a bound on the l_2-norm [22] of the β weights; hence, this regularization is often called l_2-regularization. By constraining the weights in this manner, the model no longer as closely captures the true function, increasing bias, but the

differences in prediction across datasets are not as great either, improving generalization performance.

Ridge thus reduces the values of the β weights, but recall that ideally most features in our fMRI models will not be included in the model at all, having been given weight 0. Models in which most features have value 0 are *sparse* within the feature space. It is understood that l_2-regularization does not yield sparse models. In fact, the only constraint that is proven to minimize model size, as in the number of non-zero features, is a bound on the l_0-norm of the weights, corresponding to direct minimization of the count of non-zero features. Unfortunately, no tractable algorithm exists for solving l_0-regularized regression.

Fortunately, l_1-regularization, which constraints the sum of the absolute values of β weights, can be tractably applied. Consider the variant of Eq. (1.3) in Eq. (1.4), which includes exactly that constraint. This form of regularization is known as the Lasso [87], and solves an l_1-constrained least-squares minimization:

$$\min \sum_{n=1}^{N} (\mathbf{y}_n - \beta^T \mathbf{X}_n)^2 \quad \text{s.t.} \quad \sum_{i=1}^{M} |\beta_i| \leq s. \tag{1.4}$$

The objective in Eq. (1.4) can also be written as an unconstrained mixed-norm minimization:

$$\underset{\beta}{\operatorname{argmin}} |\mathbf{y} - \mathbf{X}\beta|^2 + \lambda ||\beta||_1. \tag{1.5}$$

The multi-dimensional diamond shape of the l_1-penalty in coefficient space, contrasted with the spherical l_2-penalty, is responsible for making it likely that solutions will lie close to the axes and hence be sparse. Baraniuk [6] provides a nice comparative visualization of the penalties, as well as an overview of *compressive sensing*, a signal processing field that uses sparse methods to allow recovery of a signal using a small number of measurements. While the diamond shape of the Lasso penalty enforces sparsity, it does make the penalty not strictly convex, necessitating special algorithms to solve it. Despite the relatively recent introduction of l_1-regularization into the modeling literature, numerous algorithms have already been developed to solve it.

Regardless of whether filtered or embedded feature selection is used, such model optimization always requires one additional step. Filtered feature selection techniques require one to determine the number of features k to select, while embedded feature se-

lection techniques employing regression require one to select the norm bound parameter s. Certain algorithms for embedded feature selection offer a strong advantage in this regard, since they compute the full *regularization path* for a dataset, i.e., the optimal model given all possible model sizes, using the same amount of computation required to run one-pass of ordinary least squares regression. Specifically, the LARS algorithm [24] is a popular choice for solving the Lasso, since it provides the full l_1-regularization path. The models in the regularization path can then be evaluated to easily choose an optimal number of predictors.

Whether optimizing k selected features or the bound s though, care must be taken to prevent contamination of the test dataset. Specifically, a first inclination for optimizing these parameters may be to evaluate all models on the test set and simply retain the best predicting model; however, that model may then be highly over-fit to the test data, artificially inflating the prediction performance. To avoid this confound induced by "peeking" at the test data, *cross-validation* is used, in which the data are divided into training, test, and *optimization* sub-datasets, reserving an entire set of examples for choosing the model size [74]. As with leave-one-run-out training, k-fold cross-validation performs k divisions of the dataset into training, test, and optimization folds. While widely used in the broader machine learning community, k-fold cross-validation does present a challenge for some fMRI applications, specifically in those experiments for which only 1 or 2 runs were performed, requiring some care to perform data sub-divisions.

1.2.8 Predictive Model Reliability

Since bias is a measure of the (inverse) accuracy of the prediction of the model on *training* data, it is closely related to the prediction performance on *test* data, which serves as a measure of model validity. In some fields, such as Psychology, considerable effort is devoted to developing efficient measures of constructs. The validity of such measures is crucial, so the correspondence between the measure and an established measure of the construct is used to evaluate the model. In addition to validity, however, it is considered important that such measures are *reliable*, meaning that if the measure is re-applied, for example to the same subject at a later time or by a different rater, the scores should be consistent. Reliability is hence somewhat analogous to (inverse) model variance; however, unlike bias and variance, the relationship between reliability and validity is

generally not antagonistic. Rather, the two aspects are considered equally important properties when evaluating a new measure, or model.

An fMRI model can be viewed as a measure of the extent to which a brain is engaged in a particular task; thus, it should be both valid and reliable. Since fMRI analysis has traditionally been grounded in Psychology, some effort has been devoted to measuring the reliability of standard GLM-based models. Genovese et al. [37] discuss a method for evaluating how reliably a voxel is deemed "active" across repeated runs. Maitra et al. [56] incorporate spatial information to detect *localized regions* that are active, rather than single voxels, with the goal of reducing the number of false positive identifications of significantly active voxels to improve consistency across runs. Strother et al. [86] were among the first to consider both reliability and prediction as model evaluation metrics, although they primarily focused on the trade-off between the two metrics observed when increasing or decreasing the complexity of the representation of the data.

Within the context of predictive multivariate models, however, relatively little attention has been devoted to reliability, which is unfortunate since, regardless of the feature selection technique used, the reliability of such models is often quite poor. For instance, it is typically the case that when feature selection is performed on models learned from runs by the same subject for the same task in the same experiment, the two sets of selected voxels display subjectively poor overlap [73]. The under-exploration of reliability and the poor reliability itself are again related to the difficulties with the interpretation of machine learning-based models. If prediction is the only goal, the consistency of the model across data subsets could be considered irrelevant. If however the model is being used to gain insight into scientific data, as is usually the case with fMRI models, the intent is for the model to fully describe those aspects of the data that are consistent across data subsets in the most parsimonious manner. It could be argued that reliability is capturing this property.

Even if prediction were the only goal, as in the case of lie detection, the poor reliability may be a sign of problems with the model that the prediction score alone might not be sensitive enough to detect. Since poor reliability may indicate poor model parsimony, it could reflect an inefficiency in the *representation* of the data that, if addressed, could improve prediction performance directly. Indeed, it is generally understood that a voxel is not an ideal feature to use for modeling, since it is arbitrary and not directly associated with any meaningful unit. The scale of voxels is much larger than that at which most

neural processing occurs, but much smaller than established functional regions of activity. In addition, the spatial structure of response creates a high degree of inter-dependence between voxels. A model that represents the data with more meaningful, independent features might better capture the data and hence be more predictive as well as more reliable.

1.3 Thesis Outline and Contributions

Thus multivariate predictive fMRI models offer great promise for both neuro-scientific discovery and practical applications, but there is a disconnect between the methods being used to make predictions and our goal of gaining insight into brain function. Most notably, in the haste to build more accurately predicting models, the field has largely overlooked the observed poor model reliability, while it seems likely that better understanding what makes a model reliable could significantly impact modeling in general. In addition, the spatial structure of the brain has largely been viewed as a modeling challenge to be overcome, but the structure is such a dominant aspect of fMRI response that better characterizing it can only improve our understanding of the brain as a whole. We choose therefore to focus on these under-explored areas, with an emphasis on the interplay between prediction, reliability, and spatial structure, using sparse modeling as a framework due to its success in related domains. An overarching theme is the importance of both *prediction* and *interpretation* in fMRI modeling. In addition, we will repeatedly encounter a spatial pattern of brain activity characterized by the interaction between spatially *distributed* and *localized* processes, highlighting the redundant information processing patterns that successful models will need to address. The thesis is organized around the following questions:

How do prediction performance and reliability interact, and to what extent can jointly optimizing both reveal new insights? Chapter 2 describes the first application of Elastic Net regularized regression [103] to predictive fMRI modeling. Elastic Net was developed within the genomics community to address limitations of Lasso and, in particular, the popular Least Angle Regression (LARS) [24] algorithm for solving it, especially pertaining to model reliability. Elastic Net accomplishes its objectives by requiring two parameters, one that controls the number of voxels in the resulting model, and another the degree to which correlated voxels are included. Our findings highlight the functional

significance of patterns of distributed clusters of localized activity, and underscore the importance of models that are both predictive and interpretable. Specifically, we find that:

1. This technique produces highly predictive models of fMRI data that provide evidence for the spatially distributed nature of neural function that justifies the use of MVPA techniques.

2. By manipulating inclusion of correlated voxels, model reliability can be improved without compromising predictability. In other words, reliability can be manipulated *independently* of prediction.

3. Models must incorporate spatially localized clusters of activity.

We seek to improve model reliability, but what do we really mean by "reliability?" What does it mean for two brain states to be similar? Before we can optimize reliability, we need to determine precisely how to measure it. The existing literature strongly lacks effort to determine the *significance* of model reliability estimates. Chapter 3 tackles the issue of measuring reliability, with an emphasis on measures of map similarity and factors that must be accounted for when measuring reliability. Specifically, we compare three similarity metrics (non-zero overlap, weighted overlap, and correlation), and introduce a procedure for estimating the significance of these measures, accounting for the value distribution and spatial structure of the maps. We also perform what we believe to be the first in-depth evaluation of the prediction performance and reliability of predictive brain maps to which spatial smoothing is directly applied post-training. We find that:

1. Just as significance testing is a crucial part of model validation, the significance of any model reliability estimate must be measured.

2. Knowledge of neuro-physiology suggests that significance estimation must account for both the overall level of activity and the spatial structure in the model, and we introduce a method that meets both requirements.

3. The more accurate significance estimates derived from our method dramatically impact comparative model reliability estimates.

4. We use this new reliability evaluation framework to demonstrate that blindly smoothing learned maps post-training improves model reliability, independently of prediction performance, up to an extent that impairs both prediction and reliability.

Thus such smoothing should be used as a baseline when evaluating spatially-based modeling procedures.

How can the spatial structure in fMRI be properly captured and exploited when the properties of the structure are difficult to determine? One possibility is a brute force search for predictive spatial features among the vast space of candidate spatial filter parameters. Brute force search is rarely employed for fMRI due to the computational challenges involved in processing large datasets, but High Performance Computing (HPC) environments are ideally suited to such challenges, and facilitate the implementation of approaches previously considered infeasible. Thus HPC often paves the way for new ways of looking at problems, yet despite being a natural fit for fMRI data, it is underutilized in the field. In Chapter 4, we demonstrate the potential for HPC to expand the solution space by detailing the implementation and evaluation of a brute force search through spatial feature space that relies on a distributed learning algorithm, which we describe in detail in Chapter 5. This filter-based approach provides a test bed for optimizing the spatial parameters of predictive models and interpreting the models to gain insight into the spatial structure of fMRI response. The key findings are:

1. Gaussian spatial filters, while simple, are highly correlated across spatial scales, complicating the learning of models from the resulting feature sets, and necessitating algorithmic adjustments.

2. Without such additional effort, Gaussian spatial filters do not provide a clear advantage in terms of prediction or reliability over models learned with standard voxel-only features with or without smoothing.

3. Despite these drawbacks, the interpretability of the approach allows the revelation of a correlation among prediction performance, model size, and spatial structure that re-affirms the characterization of predictive fMRI response as distributed clusters of localized activity.

How do we address the general computational challenge of finding patterns in large datasets? Chapter 4 demonstrated the application of HPC to fMRI in particular, but the challenge of finding sparse patterns among large feature sets is applicable in many domains. Chapter 5 introduces a distributed implementation of the LARS-EN Elastic Net training algorithm for use on datasets with very large sets of features that are difficult to store in memory, such as that encountered in Chapter 4. The algorithm is generic, so it can

be used for any application presenting very large feature sets. We detail the parallelization of the algorithm, and discuss alternative implementation choices and ramifications. In addition to providing an implementation specification, we perform timing experiments to illustrate scaling properties of the implementation. These results show that:

1. This implementation successfully computes LARS-EN on datasets distributed by predictors without adding significant computational or communication overhead.
2. The implementation can achieve speedup in performance by exploiting parallel processing.

Chapter 6 concludes by elaborating on the main findings of this thesis, describing preliminary and future work beyond that discussed in the preceding chapters, and speculating on future directions relating to predictive modeling, sparse methods, spatial structure, reliability, and the overall fMRI field.

Together, these chapters highlight several issues that must be addressed in MVPA analyses, but also underscore the great promise and potential for the creative application of computational methods for analyzing fMRI data. Going forward, the most significant advances in the field will likely come from refinement of modeling goals, and greater interaction between the experimental and modeling aspects of fMRI investigation.

Chapter 2

Elastic Net for Predictive fMRI Modeling

2.1 Introduction

In the absence of formal neuronal theories of global brain function, the analysis of Functional Magnetic Resonance Imaging (fMRI) has been frequently reduced to modeling the relationship between specific image voxels and the associated mental tasks. The highest priority of early fMRI analysis was the *interpretation* of analyses to infer relevant voxels, usually involving testing hypotheses that related localized Regions of Interests (ROIs) to function. Over time, the putative regions became more localized and hypothesized functions more specific, such that, for instance, different brain regions were associated with viewing a face versus viewing a house. However, Haxby et al. [43] published a seminal paper in which models built solely from sub-maximally responding voxels were able to discriminate between mental states, underscoring both the extent of distribution of brain function and the need for models that accurately *predict* mental states from fMRI data. Since then, a diverse collection of sophisticated predictive modeling methods have been introduced to the fMRI literature [19, 64], achieving impressive prediction performance that has surprised many neuroscientists. Interest in predictive modeling has been so strong that a competition, the PBAIC [75], has been introduced to reward the most accurately predicting models. However, although predictive accuracy is vital to model assessment, it is important to keep in mind the ultimate objective of fMRI data analysis

that underscores neuroscientific discovery, and thus include model interpretability as a necessary evaluation criterion.

It is well known in statistical data analysis that proper variable selection is as critical for prediction as for interpretation [87]; therefore one development that has contributed to strong predictive performance is *sparse* modeling, in which resulting models use information from only a relatively small subset of predictive variables. Standard predictive models from fMRI data are built using individual image voxels as predictors and single time point (time-to-response, or TR) volumes as examples [61], leading to datasets typically consisting of a large number (e.g., 10^4) of predictors but many fewer examples (e.g., 10^2 or 10^3). Learning statistical models from such data is particularly challenging since it is easy to *over-fit* the training data and produce models that generalize poorly. However, the over-fitting problem can be tempered by reducing the dimensionality of the data, and thus prediction performance can be significantly improved by employing methods that identify the relevant predictive variables or combinations of them. Many predictor selection techniques (e.g., as in [61]) use a straightforward *filtering* approach, treating the selection stage as separate from the modeling stage, but sparse modeling approaches combine the selection and modeling states into one process, sometimes called *embedded* selection. Other methods, such as ICA [11], use *dimensionality reduction* to extract new predictors by linearly combining voxels. Sparse modeling methods, such as Lasso [87] (for regression) and Sparse PCA [90] (for dimensionality reduction), compare favorably to non-sparse methods on prediction performance, since they incorporate multivariate information into the selection process and, in some cases, through their incremental nature facilitate the optimization of predictors.

In principle, a sound fMRI model should exclude all irrelevant voxels while retaining all relevant ones, and the set of selected voxels will be reliable or *robust*, i.e., consistent across multiple well-designed experiments exploring a task. Therefore, a voxel selection method should both facilitate the optimization of the number of included voxels, and produce robust models. Although the prediction performance of a model is a good measure of model validity, Zou and Hastie [103] suggest that prediction performance alone does not guarantee validity. They suggest that in datasets in which many relevant predictors are correlated with each other, such as genetics data, existing sparse techniques may tend to include only one representative predictor from each cluster of correlated predictors. The authors introduce a new algorithm, the Elastic Net, which they demonstrate matches, and

in some cases surpasses, the prediction performance of sparse methods such as Lasso, yet can achieve the *grouping effect* of assigning similar weights to correlated predictors. While other sparse regression algorithms may achieve the grouping effect to some degree, Elastic Net uses a strictly convex loss function that guarantees identical predictors will be assigned the same weights, and offers the implementer the ability to adjust the degree to which the grouping effect is enforced, along with control over the number of selected predictors. Since activity levels of individual voxels are known to correlate highly with each other, some sparse modeling methods, such as Lasso, might fail to include all of the relevant voxels, but Elastic Net in such cases appears to fulfill interpretability goals of voxel selection while still surpassing the prediction performance of traditional non-sparse methods.

To reconcile traditional fMRI analysis approaches, which were guided by the intuition that functional units are localized regions rather than voxels, with the success of models of spatially distributed activity, we hypothesized that true neural response will be marked by distributed patterns of localized clusters of activity. The Elastic Net parameter controlling the "grouping" effect, usually labeled λ_2, is a particularly interesting variable for testing this hypothesis. Since nearby voxels tend to be highly correlated with each other, we hypothesized that as this grouping parameter is adjusted to increase the degree to which groups of correlated voxels are included, the model would feature more localized clusters, and hence the "scattering", or "spatial distribution," of the model voxels (formally defined later) would decrease.

In this chapter, we describe the Elastic Net method and evaluate its behavior on an fMRI predictive modeling task, from the PBAIC 2007, with regard to model prediction performance and robustness, as well as to a novel spatial distribution metric we introduce herein. Our main results are that optimizing the number of voxels included in the model via cross-validation on the data instead of choosing a fixed number of voxels is vital to prediction and, by facilitating this optimization, Elastic Net outperforms the traditional regression method of Ordinary Least Squares (OLS) in prediction performance; for a fixed grouping parameter, better predictive performance of cross-validated models is positively correlated with increased spatial distribution; increasing the grouping parameter tends to increase the robustness of a model; and models do become less spatially distributed as the grouping parameter is increased.

In summary, this work demonstrates the promise of Elastic Net for fMRI data analysis, presents experimental results supporting our hypothesis about distributed patterns of localized clusters of neural activity, and illustrates the importance of producing models that are both predictive and interpretable. These results were first detailed in [12].

2.2 Methods

2.2.1 Elastic Net

In this section, we provide a formal description of the Elastic Net method. When both the fMRI data and predicted mental states are quantified as real-valued time series, as in the PBAIC data, a common approach is to formulate the prediction task as a regression problem, in which individual TRs are viewed as independent and identically distributed (i.i.d.) samples (a simplifying assumption), the voxel activity levels are the predictive variables (*predictors*), and the mental state is the predicted, or *response*, variable. Formally, let $X_1, \cdots, X_M$ be a set of M predictors, let Y be the response variable, and let N be the number of samples; $\mathbf{X} = (\mathbf{x}_1 | \cdots | \mathbf{x}_M)$ denotes the $N \times M$ data matrix, where each $\mathbf{x}_i$ is an N-dimensional vector consisting of the values for predictor X_i for all N instances, while the N-dimensional vector $\mathbf{y}$ denotes the corresponding values for the response variable Y. Many existing regression techniques, including Elastic Net, assume a preprocessing step that performs location and scale transformations, so that the response variable is centered to have zero mean and all predictors have been standardized to have zero mean and unit length:

$$\sum_{i=1}^{N} y_i = 0, \ \sum_{i=1}^{N} x_{ij} = 0 \text{ and } \sum_{i=1}^{N} x_{ij}^2 = 1, \ 1 \leq j \leq M.$$

Then the linear regression problem is to learn the coefficients β_i in the following model:

$$\hat{\mathbf{y}} = \mathbf{x}_1 \beta_1 + \cdots + \mathbf{x}_M \beta_M = \mathbf{X}\beta \tag{2.1}$$

where $\hat{\mathbf{y}}$ is an approximation of $\mathbf{y}$. A standard approach is to use the Ordinary Least Squares (OLS) regression which finds a set of β_i that minimize the sum-squared approximation error $||\mathbf{y} - \mathbf{X}\beta||_2^2$ of the above linear model. More advanced techniques include

regularized regression methods that add a *regularization constraint* of some form to the basic least-squares minimization problem in order to avoid over-fitting and improve the prediction accuracy. This constraint usually takes the form of a bound on norms of the coefficients of the model, i.e., the β values in Eq. (2.1). The two most common types of regularization imposed are bounds on the l_1- and l_2-norms, i.e., the sum of the absolute values or squares of the coefficients respectively. Note that from a Bayesian point of view, regularized regression can be viewed as finding regression coefficients that maximize the model's posterior probability under an assumption about the prior on the coefficient values; for example, l_2-regularization corresponds to a Gaussian prior and l_1-regularization corresponds to a Laplace prior. These norms are specific cases of l_q-norm, denoted $||\mathbf{z}||_q$, where $q \geq 1$:

$$||\mathbf{z}||_1 = \sum_{i=1}^{M} |z_i|, \; ||\mathbf{z}||_2 = \sqrt{\sum_{i=1}^{M} z_i^2}, \; ||\mathbf{z}||_q = \left(\sum_{i=1}^{M} |z_i|^q \right)^{1/q}. \tag{2.2}$$

Several state-of-the-art regularized regression methods exist, differentiated mainly by the type of regularization they employ. Examples include: *Ridge regression* [45], which uses l_2-norm regularization, *Lasso* [87], which uses l_1-norm regularization, and *Bridge regression* [30, 34], which uses l_q-norm regularization, with Ridge and Lasso corresponding to $q = 2$ and $q = 1$, respectively. Interestingly, in the l_q-norm regularization family where $q \geq 1$, only the l_1-norm regularization can produce a *sparse* model [26], i.e., a model in which only a small subset of the predictors have nonzero coefficients [87]. Therefore, most of the modern sparse modeling methods include l_1-norm regularization.

The *Elastic Net (EN)* regression [103] was designed to produce models that achieve both sparsity and the grouping effect mentioned in Chapter 1 by using a weighted combination of l_1- and l_2-norm penalties on top of the least-squares problem, resulting in a strictly convex loss function, which can be written formally as minimizing the following objective function:

$$L_{\lambda_1,\lambda_2}(\beta) = ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda_1 ||\beta||_1 + \lambda_2 ||\beta||_2^2. \tag{2.3}$$

In addition to the grouping effect, Elastic Net also offers the advantage of being able to find a solution when $M \gg N$, i.e., the number of predictors is greater than the number of instances. In such cases, Lasso is limited to selecting at most N predictors.

It is easy to see from Eq. (2.3) that Elastic Net becomes equivalent to Lasso when $\lambda_2 = 0$ and $\lambda_1 > 0$, while for $\lambda_1 = 0$ and $\lambda_2 > 0$ it is equivalent to Ridge regression. When both λ_1 and λ_2 are zero, the Elastic Net problem simply reduces to OLS regression. Also, as shown in ([103, Eq. 16]), when λ_2 approaches infinity, the Elastic Net becomes equivalent to *univariate soft thresholding*, i.e., to correlation-based voxel selection with a particular threshold value.

Considerable effort has been made to understand the conditions under which Lasso can learn a known ground truth model, in which predictors are known to be "relevant" (i.e., truly included in the ground truth model) or "irrelevant." In such cases, the primary barrier to successful pattern recovery is the presence of irrelevant predictors that are correlated with relevant predictors. More specifically, consistent true model selection has been shown to be achievable only under certain conditions regarding the ability to predict an irrelevant predictor from the relevant predictors, known as the *Irrepresentable Condition* (IC) [102, 59]. Some work has been done to extend these analyses of Lasso behavior to Elastic Net. In particular, Yuan and Lin [101] derive the Elastic Net variant of IC, the *Elastic Irrepresentable Condition* (EIC), which is less stringent than IC. Jia and Yu [49] generalize EIC to non-fixed dimensions and demonstrate that it results in Elastic Net being more likely than Lasso to consistently select the true model. They also confirm the grouping effect by showing that when predictors are highly correlated, EN is more likely to pull in all correlated predictors. De Mol et al. [21] have also characterized the consistency of Elastic Net in cases in which multiple target models are learned simultaneously, and have made suggestions for selecting the λ_2 parameter; however, they assume a random design matrix, which is used in the compressive sensing framework yet has limited applicability to real-world data such as fMRI.

2.2.2 LARS-EN Algorithm

Many algorithms exist for solving the Lasso, one of which is Least Angle Regression (LARS) [24]. The LARS algorithm is very similar to Forward Stagewise regression, which is a "cautious" version of "forward stepwise regression" [95], a simple iterative approach to variable selection and regression. However, LARS is more efficient than Forward Stagewise as it makes larger steps, which are still cautious compared to the straightforward greedy method. LARS starts with an empty set of predictors and selects

the one having the largest absolute correlation with the response; however, it proceeds along the selected direction only up to the point that another predictor becomes equally correlated (in the absolute sense) with the *current residual*. Then, LARS chooses a new direction equiangular between the two predictors and continues moving along this direction until some third predictor enters the "most correlated" set (also called the *active set*). LARS chooses the new direction equiangular between the three active predictors, and so on, until it includes the desired number of predictors, specified as an input to the algorithm. Efron et al. [24] showed that, under a very minor modification, LARS finds an *optimal* solution to the Lasso problem. This result and its computational efficiency have made LARS the algorithm of choice for solving various sparse regression problems.

The Elastic Net functional in Eq. (2.3) can be easily transformed into the Lasso functional as follows [103]. Let $\mathbf{X}^*$ be an "augmented" version of $\mathbf{X}$ of size $(N+M)\times M$ and $\mathbf{y}^*$ an augmented version of $\mathbf{y}$ of size $(M+N)\times 1$ where:

$$\mathbf{X}^* = \frac{1}{\sqrt{1+\lambda_2}}\begin{pmatrix}\mathbf{X}\\ \sqrt{\lambda_2}\mathbf{I}\end{pmatrix} \tag{2.4}$$

$$\mathbf{y}^* = \begin{pmatrix}\mathbf{y}\\ \mathbf{0}\end{pmatrix} \tag{2.5}$$

Let $\gamma = \frac{\lambda_1}{\sqrt{1+\lambda_2}}$ and $\beta^* = \sqrt{1+\lambda_2}\beta$. Then:

$$L(\gamma,\beta^*) = ||\mathbf{y}^* - \mathbf{X}^*\beta^*||^2 + \gamma||\beta^*||_1 \tag{2.6}$$

Since the functional in Eq. (2.6) takes the same form as the Lasso functional in Eq. (1.4), the Elastic Net optimization problem in Eq. (2.3) can be solved with a simple modification of LARS called LARS-EN, described by Zou and Hastie in [103]. LARS-EN essentially uses LARS to minimize the above functional, while exploiting the specific structure of the augmented data matrix to improve the efficiency.

LARS-EN has two input parameters: the *grouping* parameter λ_2 and the *sparsity* parameter k that specifies the maximum number of *active* predictors, i.e., the predictors having nonzero coefficients in $\hat{\beta}$ (also called the *active set*). It can be shown [24] that each value of k corresponds to a unique value of λ_1 in Eq. (2.3), with larger λ_1 (i.e., larger weight on l_1-norm penalty) enforcing sparser solutions and thus corresponding to a *smaller* number of nonzero coefficients k. We will use this term k to denote the active set

size, i.e., the number of voxels selected. LARS-EN produces the collection of solutions, called the *regularization path*, for all values of k varying from 1 to its specified maximum value. As such, the sparsity parameter is also referred to as the *early stop* parameter since it serves as a stopping criterion for the LARS-EN incremental procedure, which adds predictors to the active set at each iteration (though removal is also possible). Note that LARS-EN, like the original LARS, is highly efficient, as it finds the entire regularization path at the cost of a single OLS fit. In addition, knowing the regularization path facilitates choosing the best solution β and its corresponding parameter k using cross-validation (described in section 2.2.5).

2.2.3 Data

The data used in these experiments were supplied by the 2007 Pittsburgh Brain Activity Interpretation Competition (PBAIC) [75] (see reference for more detail). Subjects were engaged in a Virtual Reality game, during which they had to perform a number of tasks, designed around the theme of "anthropology field work" in a hypothetical neighborhood. The field work included, among others, the acquisition of pictures of neighbors with particular characteristics (e.g., a piercing), the gathering of specific objects (e.g., fruits, weapons), and the avoidance of a growling dog. Functional MRI data were recorded for three independent runs (i.e., sessions/games) for each of 3 subjects. Each run consisted of fMRI data for the 33,000-35,000 voxels (depending on the subject) over 704 time points (TRs) each. Besides fMRI data, 24 real-valued *response* variables were provided. Several objective response variables (e.g., picking up the objects, having a dog in the picture, etc.) were measured simultaneously with the functional data, while a few subjective response variables (e.g., being annoyed or angry) were estimated off-line. All experiments were performed using fMRI data that had been passed through a high-pass filter (removing 8 Fourier modes) and response vectors that had been convolved with a standard hemodynamic response function (HRF). The task was to learn predictors for the response variables given the fMRI data and the corresponding "labels" (responses) for two runs; the models were then evaluated by PBAIC organizers on the third run.

2.2.4 Metric Definitions

Spatial Distribution Metric

We hypothesized that models would be marked by patterns of distributed clusters of localized neural activity. To evaluate model distribution and clustering, we computed a *spatial distribution metric* that estimates the spread of voxels throughout the brain. This metric is an adapted version of Thiel's redundancy measure, usually utilized to characterize spatial point patterns [68]. We first translated the β values obtained by the corresponding regression model back to their original (x, y, z) coordinates in brain space, i.e., the spatial maps of the models. We then calculated the degree of spatial distribution as follows:

1. The maps were binned using a fixed grid of $3 \times 3 \times 3$, the minimum bin size yielding a meaningful number, resulting in B bins (5808 in this case).

2. A normalized distribution was computed, such that each bin b was represented by $p_b = \frac{\sum_{i \in b} |\beta_i|}{Q}$ where $Q = \sum_{j=1}^{A} |\beta_j|$ and A is the number of nonzero β weights, or number of active voxels, in the model.

3. The entropy of the distribution was computed as

$$H = -\sum_{b=1}^{B} p_b \log p_b.$$

4. The final distribution measure was computed as $d = \frac{H}{H_0}$ where $H_0 = \log(A)$ corresponds to the maximum entropy.

Thus the spatial distribution varied between 0 and 1, tending towards 0 for maximally clustered models, in which all or a majority of the β weight mass was located in one 27-voxel bin, and tending toward 1 for maximally distributed models, in which the weight mass was evenly spread among as many 27-voxel bins as possible. Spatial distribution was computed using this metric for each Elastic Net run using the resulting model's β weights. Note, therefore, that the same voxels trained using Elastic Net with different λ_2 values result in slightly different spatial distribution scores, as the voxels were weighted differently. The performance scores naturally differ as well. The supplementary material

associated with [12] provides further details about the derivation of this algorithm and empirical tests of its validity.

Robustness Metric

As mentioned in Section 2.1, we hypothesized that greater inclusion of voxels from within correlated clusters would result in greater overlap in included voxels between two models generated on different datasets. For computing this overlap, or robustness, across experimental runs, we thus counted the total number of unique voxels selected over both experimental runs, and the number of voxels co-occurring in the two models. In our results, we discuss robustness for cases in which k was variable and optimized using cross-validation; therefore, we report robustness as the percent of the total number of unique voxels included in either model that appeared in both models; i.e., if n_1 and n_2 are the number of voxels selected in models 1 and 2 respectively, and n_b the number selected in both models, the robustness score is computed as:

$$\frac{n_b}{(n_1 + n_2 - n_b)} \tag{2.7}$$

2.2.5 Experiments

For each experiment, 144 models were trained: one for each of the 3 subjects, 24 response vectors, and 2 fMRI runs of 704 TRs. *Prediction performance* of a particular model, or set of β weights, was always measured as the Pearson correlation between the prediction of a model on a dataset $\mathbf{X}$ and the response vector $\mathbf{y}$ corresponding to the time points in $\mathbf{X}$. The specific data $\mathbf{X}$ used for testing will vary as described below.

We sought to examine model properties while varying three dimensions of the training: voxel selection, sparsity parameter selection, and learning. We will describe the choices considered along these dimensions in the following sections.

Voxel Selection

Using a filtering approach to selection is similar to setting the λ_2 parameter to ∞, as only univariate associations with the response vector are considered. Therefore, we might

expect the extent of inclusion of voxels from these clusters to increase on a continuum from low λ_2 values through filtering selection methods. In addition, we sought to compare filtered and embedded voxel selection as well as naive, or random, voxel selection.

For all model runs, the following methods were first used to obtain a ranking of voxels:

1. **Correlation-based (filtering)**: A Pearson correlation coefficient was obtained for each voxel with the response vector of interest. Voxels were ranked by the absolute value of this coefficient.

2. **Elastic Net/Lasso-Based**: Elastic Net was trained on the full dataset of 30,000+ voxels with $k = 1000$. The λ_2 parameter was selected from among $0.0, 0.1,$ and 2.0 as part of all experiments, in which 0.0 corresponds to pure Lasso, and some experiments were also performed with λ_2 values of 5.0 and 7.0 (some additional experiments revealed that results with intermediate λ_2 values were consistent with an interpolation between 0.1 and 2.0 and hence are not reported). The λ_2 value is indicated in all results. Recalling that, on some iterations, LARS-EN removes voxels, the number of iterations taken to achieve an active set size of 1000 routinely slightly exceeds 1000 iterations. However, for simplicity, the voxels that were active during the first iteration in which the active set size was 1000 were selected as the top 1000 voxels and their rank was approximated as the order in which they were last added to the active set.

3. **Random**: For comparison purposes, random rankings were generated by randomly ordering the voxels. An individual random ordering was used for each of the 144 models.

Each of these 3 methods was used to obtain a *ranking* for the voxels, which was used as described in the following sections.

Sparsity Parameter Selection

As described above, we use the term k to denote the number of voxels "selected", i.e., given non-zero weights. In particular, we consider the effect of using *cross-validation* to select an appropriate number of voxels, comparing such an approach to choosing a fixed number of voxels. Cross-validation refers to the optimization of model-building parameters based on prediction performance on held-out data, for instance data from a

separate experimental run, for maximizing generalization of the selected model to new datasets. Cross-validation can be used with any voxel selection strategy; however, the fact that Elastic Net produces the full regularization path greatly facilitates selection of an optimal k, which we will call $\hat{k}$, since models for every value of k are learned efficiently. We simply select, for each response variable, the β values from the LARS-EN iteration that produced the model with highest prediction performance on the held-out data.

Selecting the optimal parameter using different data than that used for training is important to ensure that the model is not over-fit to the training data. The PBAIC data, containing two runs, allows such testing on held-out data. To do so, we can train a model on the data for the run we are modeling and test the model on the data from whichever of the two runs, for the given subject, was not used for training. Prediction performance results calculated using the above approach may be confounded, however, by the fact that the $\hat{k}$ value was chosen by considering performance on the *same dataset* for which performance was evaluated, i.e., "peeking" at the test data. Thus, the performance results in particular will not reflect true generalization error, for which a third dataset would be needed. The test data could be randomly sub-divided into an optimization dataset, used for determining $\hat{k}$, and a test dataset, used to evaluate prediction performance, but making such a division fairly is somewhat difficult for data from structured experiments, like the PBAIC video game runs. The most straightforward way of measuring generalization error would be to apply the models to data from a third run of the experiments; however, unfortunately, as of this writing, the third run response vectors for the PBAIC data have not yet been made public. Therefore, the sub-division strategy must be employed for PBAIC despite the structure in the data.

We seek primarily to broadly compare choosing an appropriate $\hat{k}$ value to blindly choosing a fixed number of selected voxels. Specifically, we consider the following parameter selection approaches at various times, indicating the approach used in the results. The specific details of each selection strategy are dependent on the training algorithms, which are described in the following section.

1. **Fixed** k: Set $k = v$, where $v \in \{10, 300, 500, 1000\}$, by training the model using only the top v voxels obtained through the ranking method. These values for v were chosen somewhat arbitrarily, but the results are representative.

2. **Cross-validated** $\hat{k}$ **with "peeking"**: Choose $\hat{k}$ based on prediction performance on the 704 TRs in the full run (for the given subject) not used for training. Note that

a different $\hat{k}$ value is selected for each of the 144 models (one for each of the 3 subjects, 24 response vectors, and 2 fMRI runs of 704 TRs).

3. **Cross-validated $\hat{k}$ with "no peeking"**: Choose $\hat{k}$ for each model based on prediction performance on an *optimization* dataset, constructed as follows: For each of the 144 models, the test set (run 1 or 2, whichever was not used for training) was further divided into 4 optimization and sub-test datasets using 4-fold cross-validation. Each optimization dataset included a random, not necessarily temporally contiguous, 528 optimization examples and 176 test examples, where the same dataset randomization was used for each model tested. Thus each test example was predicted exactly once using models optimized on a subset of the other test examples. A $\hat{k}$ value was selected specifically for each combination of 144 models and 4 optimization sets.

While it seems ideal to use the more rigorous "no peeking" approach exclusively, we sometimes use the "peeking approach" for two reasons:

1. Since prediction performance is the metric being optimized, it is the only metric we consider for which the results are potentially confounded by peeking. Since the models obtained with peeking are optimized over a greater number of examples (704 versus 528), these results are preferable when considering non-confounded metrics, such as robustness and spatial distribution.

2. The no-peeking approach is more computationally burdensome. For some experiments requiring many models to be learned, we use the peeking approach to illustrate an approximate trend, and then show results without peeking that demonstrate the validity of the approximation.

Model Training

We experiment with the following training algorithms. The details specific to each parameter selection strategy are indicated.

1. **Linear regression (OLS)**: Only the fixed and "no peeking" selection strategies were used. First, OLS was performed on a set of v voxels against the convolved response vector. When 1000 voxels are used, OLS results should not be considered valid, since the data matrix is not invertible because the number of voxels (1000)

exceeds the number of TRs (704). Singularity problems can also emerge when N is close to v. Therefore, v was limited to 10, 300, or 500 voxels. The results of these models were used directly, with no further optimization, when using fixed k values, i.e., $k = v$. For the "no peeking" cross-validation, however, prediction performance was then averaged over the 4 optimization datasets for each of these 3 candidate values of v, and the final value of v for each model was chosen to be the candidate v with maximum average prediction performance over the optimization sets. For either voxel selection strategy, prediction performance was then evaluated on the corresponding held-out test examples for each of these selected models.

2. **Ridge regression**: Elastic Net results for a given λ_2 value are equivalent to applying Ridge regression with that λ_2 value to the pre-selected group of voxels. Since computing the Ridge regression model for datasets this size is typically infeasible, a common approach is to pre-select voxels, typically using univariate correlation [16], which is one of the voxel selection strategies studied herein. Therefore, to test Ridge regression, the same training procedures (for fixed and "no peeking" cross-validation) used for OLS were applied using the closed-form ridge solution [45] instead of OLS, except that, since Ridge uses regularization, no singularity is encountered when $v > N$, so candidate v values of 10, 300, 500, and 1000 were tested.

3. **Elastic Net regression**: For all approaches, an Elastic Net model was learned using the full set of 1000 voxels (depending on the voxel selection approach). When training using voxels that had been pre-selected using Elastic Net, the λ_2 parameter in this final training was set to the same value as was used in the selection phase. For the other selection methods, λ_2 was varied as part of the experiments and indicated in the results. For the fixed k approach, the models for the candidate v values of 10, 300, 500, and 1000 were retained and evaluated (Elastic Net also does not suffer from a singularity when $v > N$). For the cross-validated approaches, $\hat{k}$ was chosen as the k value of the best predicting model over all candidate models, of which there was one per iteration. The number of candidate models was approximately 1000, yet sometimes slightly greater due to dropped variables, but always $\hat{k} \leq 1000$. For the "peeking" cross-validation approach, prediction of the test set was used to select the model, while the mean prediction performance over the optimization sets was used to select the model for the "no peeking" approach. Note that, regardless

of the selection strategy, while the initial ranking was obtained using the full voxel set, the final learned model was trained using only a subset of 1000 voxels. Thus the resulting model for a fixed value of $k = v$ is not equivalent to the model that would be obtained by setting $k = v$ when training on the full set of original voxels, since voxels can be dropped and/or re-added to the active set at any time. We did, however, observe similar trends when training using the full voxel set.

4. **Lasso regression**: Since Lasso is equivalent to Elastic Net with $\lambda_2 = 0.0$, Lasso was evaluated using the same approaches as for Elastic Net but with a λ_2 of 0.0. Lasso is applicable when $N > M$, as in this case; however, in such cases, Efron et al. [24] state, "a Lasso fit can have no more than $N-1$ (mean centered) variables with non-zero coefficients.". While LARS-EN does not suffer this problem even when $\lambda_2 = 0.0$, it does become relatively unstable as the λ_2 value is very low, making evaluations of more than 300 voxels computationally infeasible in this setup. Therefore, these experiments only considered values of v up to 300, instead of the 1000 for Elastic Net, so relative prediction performance and robustness measures are approximations.

2.3 Results

The goal of our study was to exploit the flexibility of Elastic Net to explore the four axes of prediction, interpretation, distribution and localization. For this, we experimented with both fixed and cross-validated k values, as well as primarily two λ_2 values, 0.1 and 2.0. Since cross-validating the k value is most likely to affect prediction performance, and predictive models are associated with spatially distributed patterns, it makes sense to consider k along with these properties. Likewise, since we hypothesized that manipulating λ_2 would affect robustness, which relates to interpretation, and since analyses focused more on interpretation tend to consider localized structures, we will consider λ_2 along with interpretation and localization.

2.3.1 Prediction, Cross-Validation, and Spatial Distribution

A goal of our prediction experiments was to compare Elastic Net, as a predictive modeling tool, to the traditional regression method of Ordinary Least Squares (OLS). In addition, we sought to compare Elastic Net, run with a non-zero λ_2 value, to Lasso, which can be considered Elastic Net run with a λ_2 value of 0.0. Recall that OLS requires choosing a fixed number of voxels before training, such as the values considered in our experiments (10 and 300), while Elastic Net facilitates the selection of a model-dependent optimal number of voxels. In Figure 2.1, therefore, we compare OLS using these fixed values of k to Elastic Net and Lasso, with $\hat{k}$ determined through cross-validation. Due to the amount of computation involved in running the various voxel selection versus training permutations, we show only "peeking" cross-validation results in this figure.

Elastic Net and Lasso perform comparably to each other, yet the ability to easily optimize the parameter k can significantly improve prediction performance on the test (or optimization) dataset. This improvement, regardless of any voxel pre-selection method applied, causes both Elastic Net and Lasso to significantly outperform OLS in prediction.

Of course, the most interesting results are those obtained when the voxel selection strategy is typical for the employed training approach, i.e., univariate correlation for OLS or Ridge, and Elastic Net or Lasso for models trained with that approach. In addition, unlike the other types of results we will discuss, prediction results may be confounded by using "peeking" cross-validation. To illustrate that these prediction trends generalize to the more rigorous "no peeking" cross-validation approach, we show in Figure 2.2 the results of these "matched" voxel selection/training strategies when peeking versus not peeking. The Optimization bars show the mean prediction performance of the selected models on the 4 optimization sets. Note that selected models had the highest mean prediction performance on these optimization sets, so these results are essentially "peeking." The Test bars show the mean prediction performance of the selected models on the held-out test examples for each optimization set, i.e., "no peeking." Importantly, note that the trend for sparse methods to significantly out-perform non-sparse methods, while performing similarly to each other, holds when using these more rigorous standards as well. Also, interestingly, while not shown, it was observed that most of the models selected by OLS and Ridge using the "no peeking" approach used only 10 voxels despite 300 and 500 being candidates as well (along with 1000 for Ridge), indicating the importance of a small subset of voxels.

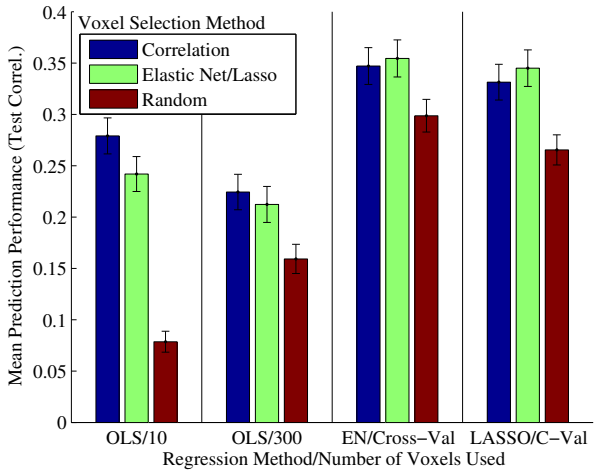


Figure 2.1: *Elastic Net and Lasso, when optimized, produce more predictive models than OLS with fixed numbers of selected voxels, regardless of the initial voxel selection approach.* Prediction performance across the 3 voxel selection methods is shown for OLS, using 2 sample values for the number of voxels selected (10 and 300), and Elastic Net and Lasso, when "peeking" cross-validation is used to select an optimal number of voxels. Mean correlation of model predictions with test data are shown, with standard error bars, averaged over the 24 response vectors, 3 subjects, and 2 "peeking" cross-validation runs. For the Elastic Net voxel selection used by OLS and Elastic Net (but not Lasso), and for the Elastic Net runs, a λ_2 value of 2.0 was used.

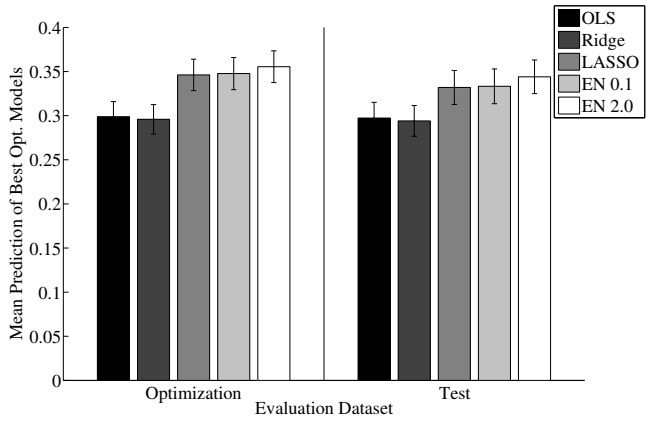
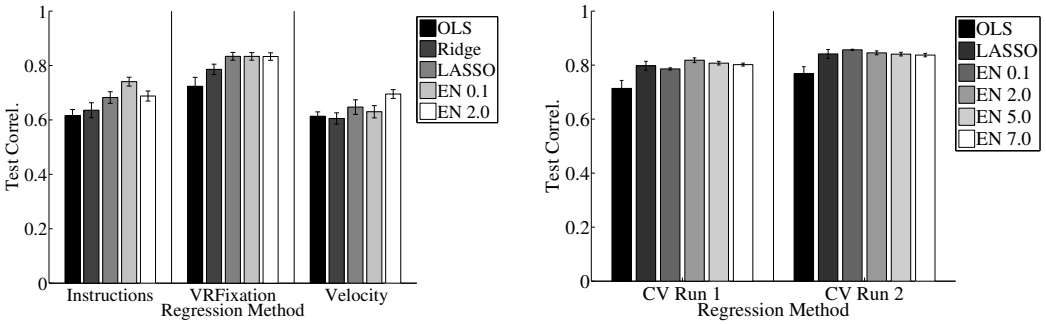


Figure 2.2: *Sparse models outperform non-sparse models when cross-validating on optimization data and testing on held-out test datasets.* Mean prediction performance of the best performing optimization models, on both the optimization datasets and the test datasets, is shown for each method, with 95% confidence interval bars, averaged over the 3 subjects, 24 response vectors, 2 runs, and 4 cv folds.



(a) Best Predicted Responses Only (b) VRFixation Subject 1 Run 1 Only

Figure 2.3: *The prediction performance of Elastic Net is similar to that of standard sparse and non-sparse methods, regardless of the λ_2 value. Models generated for the exact same dataset are very similarly predictive regardless of EN λ_2 value.* Mean correlation of model predictions with test data are shown, with 95% confidence intervals, for each modeling approach and/or Elastic Net λ_2 value. (a) Within responses, averaged over 3 subjects, 2 runs, and 4 test subset evaluations; (b) Within subject, response, and run ("CV Run"), averaged over the 4 test subset evaluations.

In addition, Figure 2.3 illustrates that this trend for sparse methods outperforming non-sparse methods, but λ_2 not significantly affecting prediction, continues as λ_2 is increased to higher values, and exists within responses and even individual subjects. These results were computed using the "no peeking" cross-validation approach on the 3 best predicted PBAIC responses: Instructions, Velocity, and VRFixation. Figure 2.3(b) demonstrates that models built on the same dataset using various λ_2 values have very similar prediction performance.

Our spatial distribution metric reveals a likely key reason for the improvement in prediction performance associated with selecting an optimal number of voxels. As Figure 2.4 shows, in most cases, cross-validating, compared to using a fixed number of voxels, produces models that are significantly less spatially distributed; this finding suggests that, in many cases, very poor performance is associated with models over-fit to training data due to the inclusion of many irrelevant voxels, which are most likely distributed randomly throughout the brain. However, note that in Figure 2.1, random voxel selection, in which small random subsets of the full set of voxels were selected, resulted in prediction performance that was comparable to that of the more principled approaches, especially as a larger, fixed number of voxels were used. These seemingly disparate results can be reconciled by examining the effects of cross-validation. After cross-validating to remove

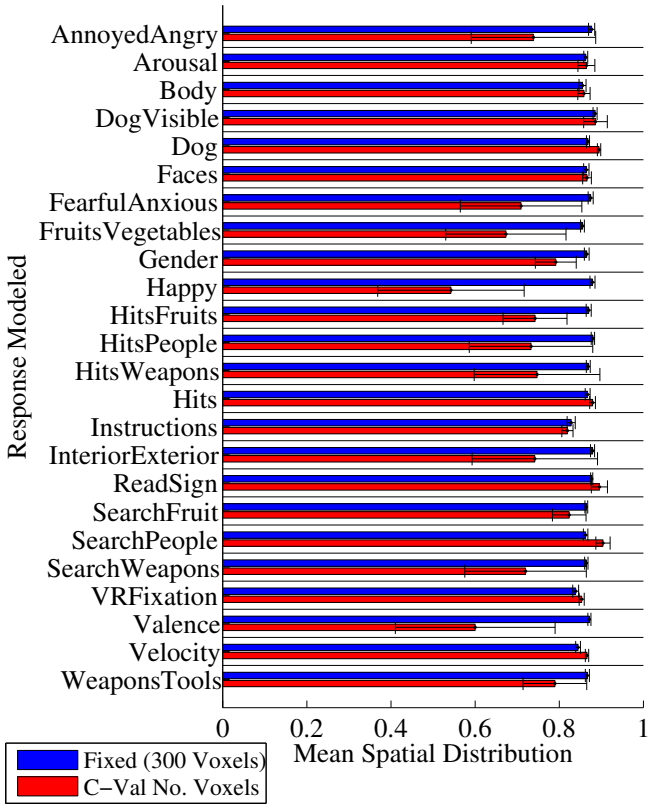


Figure 2.4: *Selecting an optimal number of voxels through cross-validation, rather than selecting a fixed number of voxels, frequently results in models that are significantly less spatially distributed.* Mean spatial distribution values of learned models for each predicted response vector are shown, with standard error bars, averaged over the 3 subjects and 2 "peeking" cross-validation runs (C-Val.). An Elastic Net λ_2 value of 2.0 was used. For Fixed, 300 voxels were used. For C-Val, the number of voxels varied; averages by feature are shown in Figure 2.6(c).

the randomly distributed irrelevant voxels, we observe in Figure 2.5 the importance of representations that incorporate information from distributed yet relevant voxels. When using a fixed number of voxels (Figure 2.5(a)), greater model spatial distribution is correlated with poorer prediction performance, but when the number of voxels is chosen using cross-validation (Figure 2.5(b)), the reverse is true: better predicted response vectors tend to be those for whose optimal learned models are more spatially distributed. These results, and the strong performance of random voxel selection provide strong evidence for the distributed nature of pattern representation in the brain.

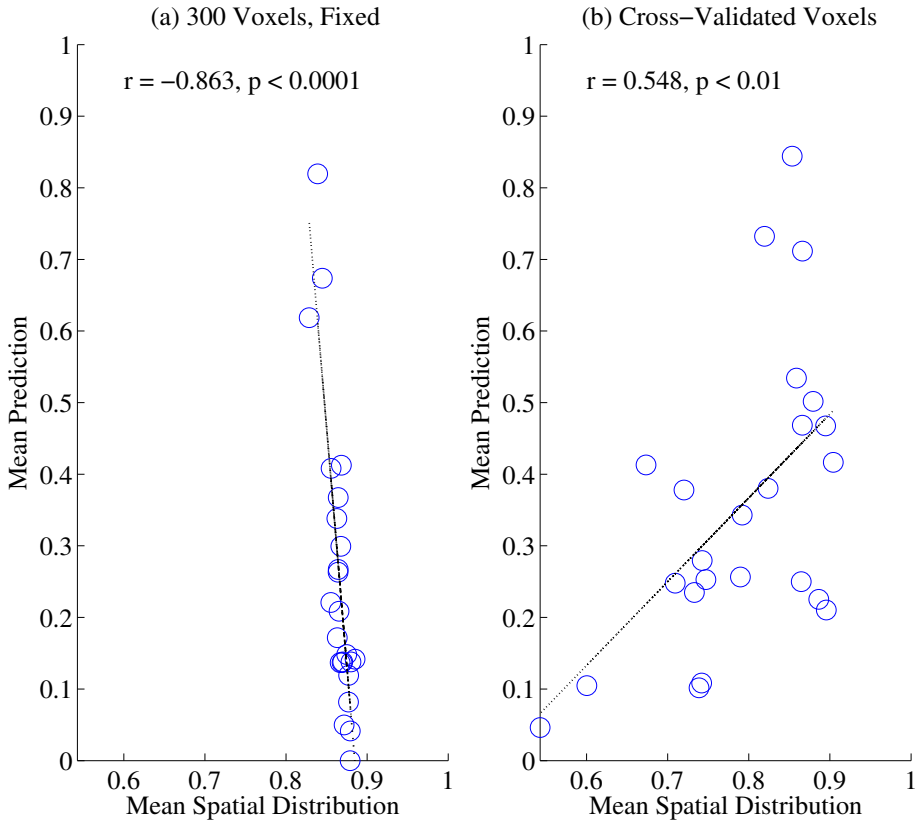


Figure 2.5: *When selecting a fixed number of voxels (a), higher spatial distribution of learned models is associated with poorer prediction performance; when the optimal number of voxels is selected through cross-validation (b), the reverse is true.* Mean model spatial distribution values are shown plotted against their matching mean prediction performance (correlation with test data). Values are averaged over the 3 subjects and 2 "peeking" cross-validation runs. A fit linear regression line is overlaid and correlation statistics are indicated. An Elastic Net λ_2 value of 2.0 was used.

2.3.2 Robustness and Grouping

Having examined prediction performance using the standard criteria, we now consider robustness, our metric defined earlier. Recall that the main effect of increasing λ_2 is the inclusion of more voxels that are correlated with other relevant voxels and hence omitted by traditional sparse methods that only consider prediction performance. Indeed, as shown in Figure 2.6a, increasing the λ_2 value from 0.1 to 2.0 may slightly improve prediction performance, but ultimately has little effect on prediction. However, since selection of voxels from within correlated clusters is likely arbitrary, including more voxels from

such clusters should increase the frequency with which voxels are selected for separate experimental runs. As is clearly evident in Figure 2.6(b), increases in λ_2 are usually associated with increases in the *robustness* (as described in section 2.2) across the 2 cross-validation runs for each response vector + subject combination; moreover, as Figure 2.6(c) shows, this change is frequently associated with the inclusion of a greater number of voxels; these additional voxels are likely those relevant voxels highly correlated with other relevant voxels. Hence, by including more relevant yet correlated voxels, increasing λ_2 improves model robustness without compromising prediction performance. We also found that model robustness directly correlates with model prediction performance ($r = 0.677$, $p < 0.001$ when $\lambda_2 = 0.1$; significant also for $\lambda_2 = 2.0$), suggesting that these two measures point to certain response variables as being generally "easier" to model, producing models that are both more predictive and more robust.

2.3.3 Localization

We have shown that controlling the learning algorithm to select an optimal number of voxels for prediction helps reveal information about the spatial structure of neural response, namely the relationship between prediction, robustness, and spatial distribution. As Figure 2.6 depicts, the additional control provided by Elastic Net's λ_2 parameter enables adjustment over model properties not directly related to prediction performance; this control too turns out to facilitate testing of neuroscientific hypotheses. In particular, recall the hypothesis that clusters of correlated voxels would exist and would frequently be localized within the brain. As Figure 2.7 demonstrates, increases in λ_2, shown to be associated with greater inclusion from among correlated clusters of voxels, are also associated with decreases in model spatial distribution. Since our spatial distribution metric is essentially independent of the number of voxels included in the model, this decrease in spatial distribution implies the inclusion of more spatially proximal groups of voxels; therefore, the correlated clusters from which more voxels are included are likely to be spatially proximal, consistent with neuroscientific intuition.

This is further exemplified in Figures 2.8(a) and 2.8(b): for the Instruction feature (auditory playback of instructions to begin a new block), the figures show the density maps of the absolute value of the regressors, normalized to the subject's anatomy, for $\lambda_2 = 0.1$ and $\lambda_2 = 2.0$. The higher value of the grouping parameter implies a relatively

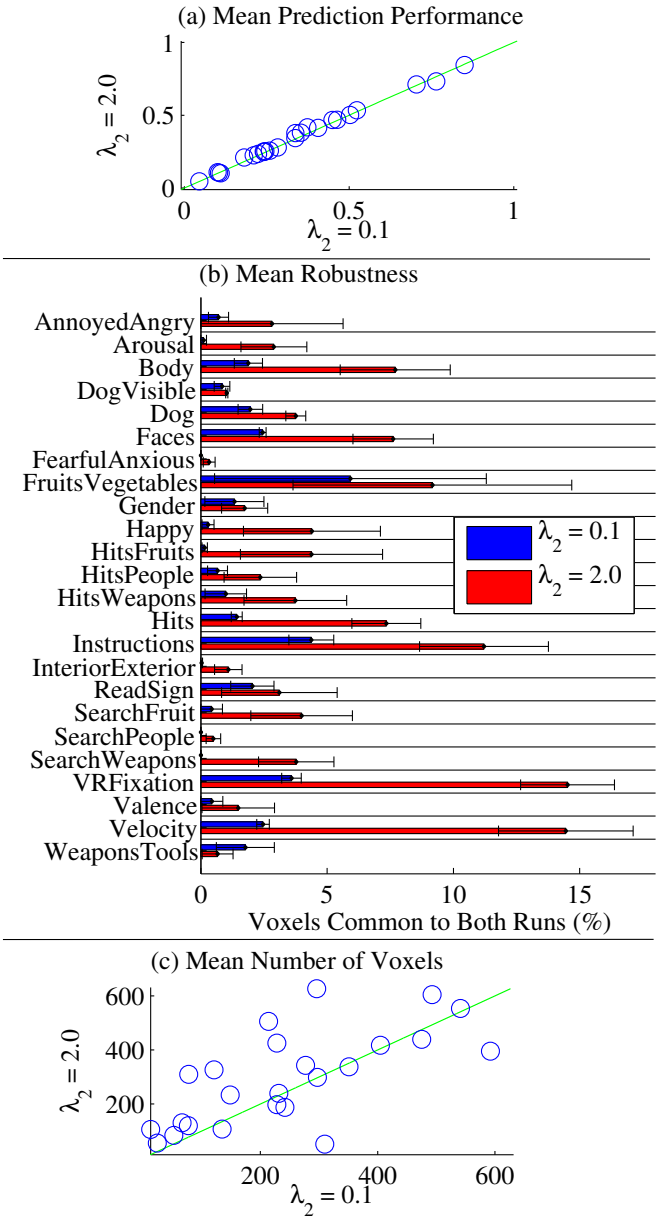


Figure 2.6: *Even among equally predictive, cross-validated models (a), increasing the* λ_2 *parameter increases model robustness (b) while slightly increasing the number of included voxels (c).* Changes in prediction, robustness, and number of non-zero voxels as λ_2 is increased are shown. (a) "Peeking" test correlation for $\lambda_2 = 0.1$ versus $\lambda_2 = 2.0$ for each of the 24 response vectors. (b) Robustness by λ_2 value and predicted response vector; bars reflect standard error. (c) Number of selected voxels for $\lambda_2 = 0.1$ versus $\lambda_2 = 2.0$ for each of the 24 response vectors. In (a) and (c), the unit line indicates no change and points above and to the left reflect increase. Means in (a) and (c) are over the 3 subjects and 2 "peeking" cross-validation runs, in (b) over the 3 subjects.

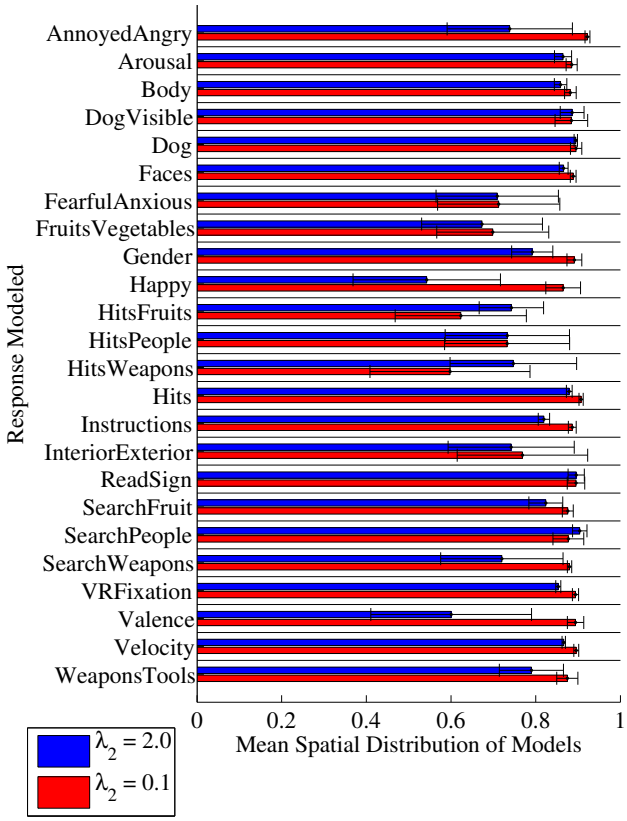
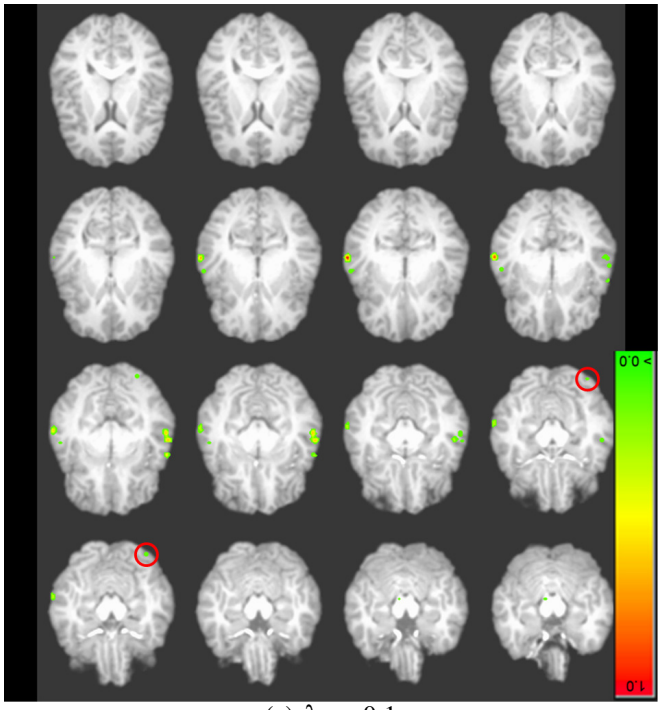


Figure 2.7: *Decreasing the λ_2 parameter is associated with more spatially distributed (less locally clustered) models.* Mean spatial distribution of learned models for each predicted response vector are shown, with standard error bars. Means in both plots are over the 3 subjects and 2 "peeking" cross-validation runs.

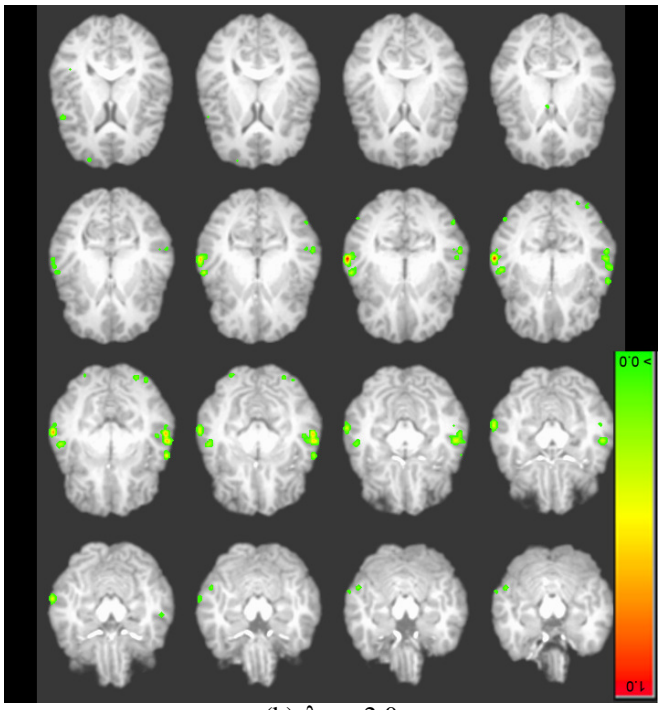
small number of locally extended clusters, whereas the smaller value produces a more globally distributed ensemble of locally restricted clusters. Observe that the maps are not threshold versions of each other: the $\lambda_2 = 2.0$ map tends to overlap with the $\lambda_2 = 0.1$ one, but it takes more contiguous territory; the latter is more spotty and distributed (note that, due to cross-validation, the maps do not necessarily have the same number of active voxels).

2.4 Discussion

Indisputably, prediction is an essential component of scientific modeling, and great effort should be put into maximizing it; however, as shown in this chapter, equally predictive



(a) $\lambda_2 = 0.1$



(b) $\lambda_2 = 2.0$

Figure 2.8: *A higher λ_2 produces maps that are still distributed yet exhibit larger localized clusters; some clusters found with lower λ_2 are not found with the higher value.* Absolute values of β weights for the Instruction feature, subject 1, for (a) $\lambda_2 = 0.1$ or (b) $\lambda_2 = 2.0$ (radiological view), with associated colorbar.

models can still be markedly different. In fMRI analysis, the core goal underlying predictive modeling is production of a model that can be interpreted to pinpoint all relevant voxel activity and exclude all irrelevant activity. Therefore, it is crucial to not lose sight of the interpretation of the resulting models in the quest to optimize prediction performance.

Innumerable techniques have been developed for choosing an appropriate set of voxels from which to build models. Not surprisingly, these techniques are most often evaluated based on the prediction performance of the resulting models. As our results confirm, selecting a set of voxels that optimizes prediction performance is indeed critical for model interpretation. When fixed numbers of voxels are chosen, the resulting models are frequently over-fit to training data, and therefore implicate many voxels that are actually irrelevant. Since these spurious voxels will tend to be randomly distributed throughout the brain, these models might misrepresent more general neuroscientific characteristics, for instance by overestimating the spatial distribution of response. Elastic Net is able to select an optimal number of voxels by automatically selecting voxels as part of the modeling process and computing the full regularization path, which facilitates cross-validation. The standard error bars for spatial distribution in Figure 3 indicate considerable variability in spatial distribution even when cross-validating when $\lambda_2 = 2.0$, yet our results indicating increased model robustness as λ_2 is increased suggest that with higher, better optimized, values of λ_2, the spatial distribution of the models for a given response vector might converge.

When choosing a validated set of voxels, the true importance of spatial distribution becomes clear; the most predictive models draw on information from the most distributed regions of the brain. However, we should be conservative when drawing conclusions about the relationship between spatial distribution and prediction performance from these results. We found no significant correlation between the degree to which a particular response variable's mean model spatial distribution increased when cross-validation was used, and the corresponding improvement in prediction of that response variable. Perhaps these results imply that modeling highly distributed neural responses is easier than modeling highly localized responses; such a finding would be consistent with the observed exclusion of voxels from localized clusters when low λ_2 values are used. Therefore experimenting with even higher λ_2 values may address this question. Still, in conjunction with the inferior yet impressive predictive performance of random voxels, these findings

underscore that the highly distributed nature of neural response necessitates the use of multivariate methods, not tied to localized regions, which can be validated by prediction performance.

We have also shown that being preoccupied with prediction performance can be equally destructive. Models that function as highly predictive "black boxes" might be useful for neuro-engineering "mind reading" efforts, but for informing neuroscience, these models should also be reliable and valid. At the least, we would expect useful models of the PBAIC response vectors to incorporate the same, or very similar, sets of voxels when trained on data from the two experimental runs. Elastic Net goes beyond other sparse modeling approaches by facilitating, with only one additional parameter, an increase in model robustness without compromising prediction performance. Examining the findings more closely by manipulating the λ_2 parameter, we observe that neural response is marked by clusters of correlated voxel activity. Existing sparse methods, which effectively use a very low λ_2 value, will tend to include only one voxel from each cluster; the voxel chosen is in effect arbitrary due to minor fluctuations in the dataset. By selecting more voxels from within these clusters that are redundant for prediction yet relevant to the task, Elastic Net with a higher λ_2 value achieves more robust, and hence valid, models.

We can be even more emphatic, but speculative, in our interpretation of the results, by extrapolating the observation that, at least for a good feature such as Instruction, two alternative models (as shown in Figures 2.8(a) and 2.8(b)) can achieve an almost identical prediction performance: the only possible way to avoid this model degeneracy is to introduce further functional constraints, including local and global neural dynamics, and perhaps challenge the idea that brain function arises from sufficiently repeatable spatio-temporal patterns. By the same token, the fact that the maps are not thresholded versions of each other also emphasizes that brain processes occurring at several temporal and spatial scales can be detected by fMRI. To make progress along these challenging lines, however, we need stronger and more encompassing theoretical tools than we have at our disposal.

These results suggest the promise of Elastic Net for fMRI modeling, although the field is still quite far from producing models that are as predictive, and certainly as robust, as needed to answer the pressing modern neuroscientific questions. For instance, regardless of λ_2 value, the models built on the PBAIC 2007 data all showed poor robustness, with no response vector averaging better than 17% overlap. This finding highlights the important

point, however, that models produced are only as good as the data used to train them. While the PBAIC experiments were run carefully, the experimenters will agree that the response variables analyzed are highly challenging to model. The response variables are sometimes overly broad and sometimes overly specific, and little is known about how they might map to underlying function. Our results reflect this variability among the response variables, with some responses clearly more "easily" modeled than others. The diversity and complexity of these response variables are in fact the very reasons they appear in such a generalized competition; the experimenters have been highly impressed by even the prediction capabilities observed using all methods thus far. We intend to continue exploring the issue of robustness empirically as well as theoretically on better understood fMRI datasets.

Still, the effect of λ_2 on robustness is clear from these results, and the control provided by λ_2 is shown in this paper to have value even beyond improving robustness. This parameter has a well-understood effect on the resulting models; as such, it is an attractive candidate independent variable for testing neuroscientific hypotheses. In our experiments, this variable served as a proxy for the degree to which members of correlated clusters are included in models. Since greater inclusion is associated with decreased spatial distribution, we can conclude that these correlated clusters are frequently localized in space. Combined with our findings about the importance of spatial distribution, a picture emerges of neural response characterized by patterns of localized clusters distributed highly throughout the brain. Many existing techniques exist solely to attempt to extract these localized clusters directly, for instance by setting thresholds on the extent of cross-correlation [29]. Elastic Net, by selecting voxels automatically during modeling, offers the advantage of automatically selecting these clusters without the need for separate methods or thresholds, and does so in a multivariate context, on a per response variable basis, so that all clusters relevant to a task, but none that are irrelevant, are selected.

As demonstrated preliminarily in this chapter, once a model has proven itself trustworthy, the next step is naturally to extract information from it. Models exhibiting satisfactory prediction performance and robustness can be easily explored visually by plotting the β weights from the models according to corresponding voxel location. The presence of voxels within specific ROIs can be determined and techniques can be developed to explicitly "cluster" the voxel sets so as to visually isolate the localized clusters. It might also be interesting to compare the "clusters" generated by Elastic Net with the

"dimensions" extracted through dimensionality reduction techniques such as ICA and Sparse PCA. In addition, we have shown that the spatial distribution metric introduced in this chapter, while simple, can reveal insights about neural functioning. Subsequent efforts might use this measure to explore differences in neural representation across tasks and between subjects. Predictive modeling techniques clearly offer great promise for knowledge discovery, but this ultimate goal of their production must be considered.

Chapter 3

Reliability Evaluation and Map Smoothing

3.1 Introduction

Chapter 2 demonstrated that sparse modeling offers two advantages for building predictive fMRI models: 1) they facilitate the selection of an appropriate number of voxels for model inclusion, leading to models that may have better generalization to other data subsets and 2) they yield more parsimonious models that can aid interpretation. It was also shown that, in particular, the Elastic Net method for training sparse models [103] can improve the reliability of such models across data subsets even among equally well-predicting models. These results contrast somewhat with the scarce existing work examining the relationships between model prediction and reliability, specifically that of Strother et al. [86], which primarily focused on the trade-offs between prediction and reliability as model complexity is manipulated. This trade-off is similar to the bias-variance trade-off in statistics. On the contrary, Chapter 2 showed that Elastic Net can improve reliability without sacrificing prediction performance.

Elastic Net achieves this effect by adding an l_2-regularization penalty to l_1-regularized regression, which essentially spreads weights over clusters of correlated relevant predictors. This weight spreading is especially useful in domains in which the underlying correlation structure among predictors is not necessarily easily parameterized, as in sets of genes; however, it is well known that there is a high degree of spatial auto-correlation

among fMRI voxels due to neural, functional, and physical processes, which is factored into traditional Statistical Parametric Mapping (SPM) approaches [32]. Chapter 2 showed that much of the increased reliability achieved with increasing l_2-regularization did indeed result from the spreading of model weights over spatially localized clusters of relevant voxels.

Rather than regularize blindly, one can in fact exploit this knowledge of spatial structure by applying an explicit spatial regularization to the model [91, 97, 7]. In fact, the Fused Lasso [88] is a Lasso variant developed specifically for domains, such as image processing, in which some form of variable ordering is known, for example ordering by spatial coordinates. Models trained using regularization that smears model weights, whether through indirect or direct spatial regularization, yield smoother-appearing activation (parameter) maps that sometimes show better prediction performance. The rationale for applying regularization in the modeling step is that both the predictive information in a voxel and its spatial information are incorporated jointly into the voxel weight; however, such regularization often carries a heavy computational burden that may require customized learning algorithms, precluding the use of standard algorithms such as LARS [24], which facilitate cross-validation for model selection.

A much simpler alternative would be to train the model using a more generic learning algorithm and then apply blind spatial smoothing to the weights. Smoothing *images* prior to model building is a standard technique used to reduce noise, especially when comparing across subjects. In fact, Strother et al. [86] found that smoothing as a preprocessing step alone can improve reliability across subjects and trials; however, there appears to be no prior examination of the effects on prediction and reliability of directly smoothing the weights of a learned predictive model *after* training. This paucity of existing literature is not surprising, since this approach has generally been assumed to be detrimental to prediction performance, given the sensitivity of single-voxel multivariate modeling. Furthermore, most of the literature on methods that employ a form of spatial "smoothing," including regularization, contain evaluations of prediction performance and perhaps map appearance, but not direct evaluations of reliability. In addition, when reliability is evaluated for such cases, little attention is usually paid to the metric used for evaluation.

The robustness metric used in Chapter 2 for measuring the reliability of models across fMRI runs is a natural one for evaluating sparse models, as it measures the degree of

overlap between the two sets of voxels selected, or given *non-zero* weights, by calculating the percentage of voxels selected in either set that are selected in both sets. For less sparse models, however, this metric is confounded. Overlap scores can be boosted simply by increasing the number of voxels selected until the intersection of the two sets equals the union and overlap is 100%. Unfortunately, smoothing can drastically increase the number of non-zero voxels in a map, leading to such a sparsity confound.

Non-zero overlap is essentially a *similarity* metric between vectors. Similarity, or distance metrics, arise in numerous problems, and countless such metrics exist. Therefore, one solution to the sparsity confound may be to employ an alternative metric. The raw overlap metric ignores the actual values of the voxels with non-zero weights, so a simple variation is to consider the *weighted overlap*, which calculates the proportion of total weight mass devoted to the common set of voxels. The weighted overlap metric captures the *specificity* of model weights better than the non-zero overlap metric. Encouragingly, scores for this metric almost universally exceed non-zero overlap scores, especially when no smoothing is applied, suggesting that the most relevant voxels are among the most reliably selected. An even simpler similarity metric used by Strother et al. and others is the Pearson *correlation* between the two map weight vectors. As we will show, unlike the overlap metrics, correlation is independent of the number of voxels selected, making it useful for both sparse and non-sparse models. Unfortunately, correlation scores lack some of the clear interpretation of the overlap measures, especially for sparse models.

Needed therefore is a metric that conveys some notion of "significance." Most commonly, this requirement is satisfied by a test of statistical significance, or p-value. Fury et al. [35] describe one such approach commonly used in the genomics literature for the analogous problem of evaluating the overlap between two sets of selected genes. As they discuss, the hypergeometric distribution parameterizes the overlap between two sets, and a p-value for this distribution can be obtained by performing a Fisher's Exact [96] test on the cross-tabulation matrix corresponding to counts of predictors selected in neither set, one set, the other set, or both sets; however, they also demonstrate that this score can also be influenced by the number of variables selected relative to the total set of candidate variables. In addition, this approach can only estimate the significance of the non-zero overlap metric, which has limited utility for non-sparse models. For correlation, a p-value can also be parametrically obtained, but this computation assumes that the variables are independent and normally distributed. While in many cases the variables

are indeed normally distributed, we know unequivocally that they are not independent. In fact, we are exploiting these correlations when we increase reliability by regularization or smoothing.

Therefore, to estimate significance, we must have an estimate of the distribution of metric scores, but analytical estimation of the distribution is difficult. A common approach in statistics for estimating distributions when analytical estimation is difficult is *resampling* [40]. Resampling describes a broad range of approaches that vary depending on the problem being addressed. For instance, to estimate the precision of model parameters given a small amount of data, *bootstrapping* [25] and related methods are often used to generate new datasets from permutations of the available data. Another common goal is to estimate the significance of a particular classification of examples using *permutation testing* [67]. In this approach, each example is assumed to have a true label, and these labels are "shuffled" randomly and repeatedly to generate a series of artificial labeling schemes. The artificial estimates are used to arrive at a non-parametric estimate of some distribution relating to classification accuracy, such as the margin between classes or model accuracy.

In our case, we are essentially evaluating the accuracy of the assignment of voxel weights to voxel locations using reliability between models as a proxy measure, since the ground truth is unknown. Estimating the significance of this reliability hence entails estimating the significance of the *joint* assignment of weights to locations between two models, which does not map directly onto one of the standard resampling approaches. The closest mapping would be to consider voxel weights as samples and their locations as labels; however, rather than estimate the significance of the specific assignment of labels, we seek an estimate of the significance of the degree to which the label assignments are the same between two models, regardless of the specific labels. Our approach, however, can still borrow from the permutation testing framework by generating numerous new sample surrogate models by repeatedly randomly shuffling the locations of the voxel weights, and evaluating the distribution of the reliability score across these new samples. By doing so, we are essentially stating that our null hypothesis is that the two specific sets of weight values were responsible for the observed reliability score.

This basic null hypothesis, which assumes independence among samples, seems satisfactory given the dependence of the overlap measures on the number of selected (non-zero) voxels. Unfortunately, our samples are not independent; they correspond to voxel

locations, which, as we know, exhibit high auto-correlation. Like the Pearson correlation p-value, this simple null hypothesis does not account for the 2-point correlations, or covariance structure, in the data. In most domains, the process generating this structure is unknown, but, for fMRI data, we know it to be a largely spatial process. Since we exploit this knowledge indirectly with Elastic Net and directly with spatial regularization or smoothing, we would expect this data property to be manifested in the learned models as well, so it is natural that we incorporate this information into our null hypothesis.

In this work, we compare these three reliability metrics (non-zero overlap, weighted overlap, and correlation), and introduce a surrogate-generating procedure for significance estimation that preserves the exact value distribution of a map and the approximate spatial spectrum of the map. We compare this approach to the non-spatial resampling approach and the parametric overlap significance estimate. We also perform, to our knowledge, the first in-depth evaluation of the prediction performance and reliability of predictive activation maps to which spatial smoothing is directly applied post-training. We find that smoothing with a small spatial kernel does not necessarily impair prediction performance; the overall activity level in a model has a direct effect on the expected similarity between corresponding maps, and spatial structure increases the variance of this expected similarity, so both properties must be considered when evaluating the reliability of maps incorporating spatial information; and even with stringent criteria, smoothed maps are significantly reliable and, for certain tasks for which well-predicting models are more difficult to learn, more meaningfully reliable than maps obtained through Elastic Net regularization alone. Weight smoothing should therefore be used as a baseline comparison when indirect or direct spatial regularization is evaluated. Most importantly, the information captured by our significance estimation procedure is found to dramatically impact reliability comparisons, so such a procedure should be employed whenever estimating reliability.

3.2 Methods

3.2.1 Definitions

We define the following terms:

Image: An image $G \in \mathbb{R}^{X \times Y \times Z}$ contains one fMRI scan received directly from the scanner with or without pre-processing (which will be specified when applicable) where X, Y, and Z are the 3-D scanning resolution parameters.

Brain mask: A bit array in which "on" bits correspond to valid brain locations. For a brain of size $X \times Y \times Z$ with M valid brain voxels, the brain mask $\mathbf{B}$ will be of size $X \times Y \times Z$ and will have M non-zero bits. Applying $\mathbf{B}$ to an image produces a vector $\mathbf{v} \in \mathbb{R}^M$. If the 3D image space is of size $64 \times 64 \times 34$, $M \sim 30,000$.

Dataset: The full set of images collected during the course of an experiment after applying a brain mask to convert the images into vectors. The resulting dataset is a matrix $\mathbf{X} \in \mathbb{R}^{N \times M}$ where N is the number of time points (TRs) (one image per time point) and M is the number of valid brain voxels.

Smoothing filter/kernel: A smoothing kernel $\mathbf{K} \in \mathbb{R}^{a \times b \times c}$, where a, b, and c are arbitrary dimensions, is a filter that when convolved with an image or map has the effect of smoothing the values in the image or map. In this chapter, we will focus on Gaussian-parameterized kernels.

Map: $\mathbf{M} \in \mathbb{R}^{X \times Y \times Z}$ where X, Y, and Z are the scanning resolution parameters of the images; however, unlike an image, in which the values correspond to the actual fMRI data at a time point, the values in a map have been assigned by some process, human or computational, as a model of the data. In this chapter, we will deal primarily with *learned maps*, in which the values correspond to parameters, or weights, of a learned model, and *smoothed maps*, in which a smoothing kernel has been applied to a map.

3.2.2 Data

PBAIC 2007

Experiments were run on the PBAIC Competition data described in Section 2.2.3.

Pain

Our analysis was performed on the fMRI dataset originally presented by Baliki et al. in [5], consisting of 14 subjects and 240 TRs of 2.5s each, divided evenly between 2

runs. The subjects in the scanner were asked to rate their pain level (using a finger-span device) in response to a painful stimulus applied to their back. An fMRI-compatible device was used to deliver fast-ramping (20C/s) painful thermal stimuli (baseline 38C; peak temperatures 47, 49, and 51C) via a contact probe. During each session, nine such stimuli were generated sequentially, ranging in duration from 10s to 40s, with similar-length rest intervals in between. The actual applied temperatures as well as the subject's perceived rating of the temperature were recorded. The data were acquired on a 3T Siemens Trio scanner with echo-planar imaging (EPI) capability using the standard radio-frequency head coil. Each volume consisted of 36 slices (slice thickness 3mm), each of size 64×64 covering the whole brain from the cerebellum to the vertex. Maps were sub-sampled to dimensions $46 \times 55 \times 46$, such that each voxel was exactly 4mm^3. The total number of brain voxels ranged from 26,000-28,000 depending on the subject. The standard fMRI data preprocessing was performed using FSL FEAT [85], including, for each subject: skull extraction using a brain extraction tool (BET), slice time correction, motion correction, spatial smoothing using a Gaussian kernel of fullwidth half-maximum 5mm, nonlinear high-pass temporal filtering (120 s), and subtraction of the mean of each voxel time course from that time course. Pain and visual ratings were convolved with a generalized hemodynamic response function (gamma function with 6s lag and 3 s SD). In these analyses, predictive models were trained for 2 temporal response vectors: the actual temperature of the stimulus and the subject's rated perception of experienced pain.

Synthetic Data

A synthetic dataset was used to explore properties of reliability metrics that are computationally difficult to manipulate with real data. Since we are most interested in models exhibiting both some sparsity and some spatial correlation, often parameterized as Gaussian "blobs," we used a synthetic dataset generated from a relatively small set of Gaussian spatial fields. Our synthetic dataset consisted of 10 datasets generated from 2 class template vectors, $\mathbf{C}_1 \in \mathbb{R}^{800}$ and $\mathbf{C}_2 \in \mathbb{R}^{800}$. Each template consisted of 20 spatial Gaussian "blobs" (distributions) randomly localized on a surface of 800 voxels, a 1-dimensional "brain" map. The standard deviations of the blobs were chosen randomly from values $\{1.0, 2.0, 3.0\}$ and each blob mean corresponded to the voxel on which the blob was centered. Each blob was $l_\infty-$normalized to have a peak value of 1.0; however, since the blobs can overlap, the template amplitude at a given voxel may be more than 1.0.

The dataset consisted of 220 examples, evenly distributed between the 2 classes, where each example was generated by choosing a random subset of the template blobs for that class, each with a 50% chance of being selected, and white noise of standard deviation 0.1 was added. In addition, spatially correlated noise was generated by applying a band-pass filter to white noise with standard deviation 1.0. Examples drawn from $\mathbf{C}_1$ were labeled positive and the $\mathbf{C}_2$ negative; we can therefore approximate the discriminative function as $\mathbf{C}_1 - \mathbf{C}_2$, though in practice, since only one class template is necessary for discrimination, the algorithm will likely learn a hybrid of these 2 templates. This approximate template is used for illustrative purposes. Figure 3.7(a)(i) visualizes this "ground truth" model parameter template.

3.2.3 Map Smoothing

We investigated the properties of smoothing maps (rather than images prior to learning) by learning models from the data and then applying smoothing to the learned parameter map. Smoothing was obtained by convolving the learned map with a Gaussian spatial kernel. The kernel was applied at all positions in which at least half of the affected voxels were brain voxels (for real data) or within the space of valid "voxels" (for synthetic data). Invalid voxels were treated as 0-valued. For real data, 3 spatial kernels were considered, with sizes corresponding exactly (for Pain) and roughly (for PBAIC) to standard deviations of size 2mm, 6mm, and 10mm. For synthetic data, a kernel with standard deviation 2 voxels was used. As discussed further in Section 3.3, applying even a small amount of smoothing in this way dramatically increases the number of selected voxels to such an extent that the model is no longer sparse. Therefore, experiments were also performed after thresholding these smoothed maps so that weight values below 0.01 were set to 0.

3.2.4 Model Training and Prediction Evaluation

A separate model was learned for each subject, task, and run in a given dataset. For each model, the LARS-EN Elastic Net training algorithm [103, 24] was applied to the full set of voxels and run until a given number of voxels were selected. Since LARS algorithms drop predictors under certain conditions, the number of iterations may not

equal the final number of selected voxels. As discussed in Chapter 2, cross-validation generally results in models with a greater number of voxels as λ_2 is increased, so more training iterations are required and hence more voxels are needed to obtain an equivalent test correlation. The simplest approach would be to train the model until some very high number of voxels is selected; however, in practice, LARS-EN computation can become much less efficient as the number of selected voxels increases, and if the target number of voxels greatly exceeds the range needed for prediction, the algorithm can become very unstable, frequently dropping voxels and greatly increasing the number of required iterations. Therefore, models were trained for more iterations when a higher λ_2 value was used, but the results across λ_2 values are comparable.

Prediction was then evaluated on data from whichever run was not used for training. LARS greatly facilitates selection of the best predicting model by efficiently providing the entire regularization path, that is, the model learned for each possible number of selected voxels. Each model can then be evaluated and the best predicting one retained. Since prediction performance scores can be confounded by optimizing the number of selected voxels using the same data used to evaluate prediction scores, it is desirable to hold out one portion of the test data for such optimization and one for evaluation of generalization performance. Ideally, one entire experimental run would be used for model selection and a third run would be used for evaluating prediction. Unfortunately, no labels have been released for a third PBAIC 2007 run, and the Pain experiment included only two runs, so an alternate cross-validation approach was used. While the temporal structure in some of the PBAIC response variables can confound the standard k-fold cross-validation approach, we chose to use 2-fold cross-validation, in which each run was divided equally into the earlier temporal half and later temporal half, so that, with the exception of two data points, the optimization dataset includes no time points temporally contiguous with any test time points. The resulting optimization sets contained 352 TRs each for the PBAIC data and 60 TRs each for the Pain data. Prediction was measured by concatenating the predictions for each of the 2 optimization folds in a given run and measuring the Pearson correlation between this concatenated vector and the true response vector.

For PBAIC, the following λ_2 values were tested: 0.1, 2.0, 4.0, and 6.0. Using empirical estimates of the number of required voxels, for 0.1 the model was trained for up to a model size of 1500 voxels; for 2.0, 4.0, and 6.0, up to 2500 voxels. For simplicity, many results are visualized only for the 3 best predicted PBAIC tasks, which are Instructions,

VRFixation, and Velocity, and 3 moderately-well predicted PBAIC tasks, which are viewing of Body, viewing of Face(s), and Hits. For Pain, λ_2 values of 0.1 and 2.0 were tested, with up to 1000 voxels selected due to the fewer training instances in that dataset.

For the synthetic data, 51 models were trained for each of the 10 datasets, exploring the use of all λ_2 values between 0.1 and 5.0 at increments of 0.1. Models were trained until all 800 voxels were included. Lasso, or Elastic Net with a λ_2 of 0.0, is not valid when the number of selected predictors exceeds the number of examples (110 in this case) so results for $\lambda_2 = 0.0$ are not reported. This range of λ_2 values was used mainly to demonstrate variability in the metrics tested. Every 20 iterations, the reliability for each of the 45 pairs of models trained to that point for the same λ_2 value were computed. For all results shown for this dataset, the mean scores over these 45 pairs are displayed. Note that since all comparisons were made between models trained for a specific number of iterations, cross-validation is not applicable. The test accuracy of the binary classification task, averaged across all 9 non-training datasets over all 10 datasets, was always near 100%, given the ease of the task; therefore, ground truth reliability evaluation was performed instead, as described in the following section.

3.2.5 Reliability Evaluation

We are concerned with reliability defined as the similarity between the two parameter maps learned on two corresponding runs; however, when employing cross-validation, two models are selected for each run (one for each optimization fold). Therefore, for each of the models compared (across all subjects, tasks, and regularization/smoothing parameters), we measured reliability across all 4 pairs of different-run optimization datasets. Since these maps are not independent, we averaged the results to arrive at one score.

Reliability Metrics

We explore reliability evaluation along two orthogonal dimensions: metric and null hypothesis. Metrics are essentially similarity metrics between two maps, while the null hypothesis is used to generate surrogates over which to calculate a significance score. The following notation is used in formalizing the metrics:

- $\mathbf{M}_1$ and $\mathbf{M}_2$: the two brain maps being compared (which may be generated by any means, but for our purposes are learned parameter maps).
- $\mathbf{v}_1 \in \mathbb{R}^M$ and $\mathbf{v}_2 \in \mathbb{R}^M$: vectors where M is the number of valid brain voxels, produced by applying the same brain mask to $\mathbf{M}_1$ and $\mathbf{M}_2$.
- $R_1 = v : \mathbf{v}_1(v) \neq 0$, the set of indices of the "relevant" (non-zero) voxels in $\mathbf{v}_1$
- $R_2 = v : \mathbf{v}_2(v) \neq 0$, the set of indices of the "relevant" (non-zero) voxels in $\mathbf{v}_2$.
- $B = R_1 \cap R_2$, the set of indices in both R_1 and R_2.
- $n_1 = |R_1|$, the number of non-zero voxels in $\mathbf{v}_1$.
- $n_2 = |R_2|$, the number of non-zero voxels in $\mathbf{v}_2$.
- $n_b = |B|$, the number of elements in B.
- w_1, the sum of the absolute values of $\mathbf{v}_1$:

$$w_1 = \sum_{v \in \mathbf{v}_1} |v| \tag{3.1}$$

  where $\mathbf{v}_1(v)$ denotes the element of $\mathbf{v}_1$ at index v.

- w_2, the sum of the absolute values of $\mathbf{v}_2$:

$$w_2 = \sum_{v \in \mathbf{v}_2} |v| \tag{3.2}$$

- w_b: the sum of the absolute values of the weights of the elements in B when they appear in both $\mathbf{v}_1$ and $\mathbf{v}_2$. If $\mathbf{v}(i)$ denotes the vector of elements in vector $\mathbf{v}$ at indices i:

$$w_b = \sum |\mathbf{v}_1(B)| + \sum |\mathbf{v}_2(B)| \tag{3.3}$$

The metrics are as follows:

- **Non-zero overlap**: the same metric used in Chapter 2

$$\frac{n_b}{(n_1 + n_2 - n_b)} \tag{3.4}$$

- **Weighted overlap**: similar to Non-Zero Overlap, but weighted by the actual values of the voxels and normalized, such that voxels with higher coefficients receive greater weight

$$\frac{w_b}{(w_1 + w_2 - w_b)} \tag{3.5}$$

- **Map Correlation**: Pearson correlation between $\mathbf{v}_1$ and $\mathbf{v}_2$ (hence technically "vector" correlation):

$$\frac{M\sum_i \mathbf{v}_{1,i}\mathbf{v}_{2,i} - \sum_i \mathbf{v}_{1,i}\sum_i \mathbf{v}_{2,i}}{\sqrt{M\sum_i \mathbf{v}_{1,i}^2 - \left(\sum_i \mathbf{v}_{1,i}\right)^2}\sqrt{M\sum_i \mathbf{v}_{2,i}^2 - \left(\sum_i \mathbf{v}_{2,i}\right)^2}} \tag{3.6}$$

Null Hypotheses

Assume we have calculated a reliability score ψ between two maps $\mathbf{M}_1$ and $\mathbf{M}_2$ using one of the three above metrics. We now wish to estimate the statistical significance of ψ. To do so, we assume a null hypothesis $\mathcal{H}_0$ governing the assignment of voxel weights to voxel locations, reflecting a key property of the maps $\mathbf{M}_1$ and $\mathbf{M}_2$ that we wish to retain when resampling. Using the resampled surrogates, we obtain a distribution $\mathcal{D}$ over reliability scores from which ψ is assumed to be drawn (occurs due to chance). We assume, for simplicity, a Normal distribution $\mathcal{D}$ and use this distribution to obtain a z-score for ψ. A high z-score implies that $\psi \gg \mathbf{E}_{\mathcal{D}}\,\psi$.

We explore two null hypotheses, one more specific than the other, and use each to create a set of *surrogate* maps. The null hypotheses and procedures used to generate the surrogates are the following:

1. **Non-Spatial**: Voxel locations are assumed to be independent, such that vectors with the exact same sets of weight values in $\mathbf{v}_1$ and $\mathbf{v}_2$ are expected to yield the observed reliability. Note that the metrics above only require the vectorized form of the maps being compared, and thus only the vector surrogate need be generated; since spatial information is not considered for this null hypothesis, all computations can take place on the vector forms. To generate a surrogate, the indices of values are shuffled uniformly; in other words, the weights are randomly re-assigned to other locations. If $\mathbf{I} = \{1,\ldots,M\}$ are the indices of a vectorized map $\mathbf{v} \in \mathbb{R}^M$ and randperm($\mathbf{I}$) returns a vector $\mathbf{r} : \mathbf{I} \rightarrow \mathbf{I}$ in which the elements of $\mathbf{I}$ are randomly permuted, the surrogate $\tilde{\mathbf{s}} \in \mathbb{R}^M$ for $\mathbf{v}$, initialized to all 0, is calculated as:

$$\mathbf{r} = \text{randperm}(\mathbf{I}) \tag{3.7}$$

$$\tilde{\mathbf{s}}(\mathbf{r}(i)) = \mathbf{v}(i), i \in \mathbf{I} \tag{3.8}$$

2. **Spatial (FFT)**: The voxel locations exhibit spatial dependency, such that maps with both the exact same set of *voxel weight values* in the maps $\mathbf{M}_1$ and $\mathbf{M}_2$ and the same covariance structure, reflected in *spatial structure*, are expected to yield the observed reliability. Note that this null hypothesis therefore considers the original *map*, not vector, though it again needs only to return a surrogate vector. To generate a surrogate, the indices of the values in the maps are shuffled in such a way that the spatial structure of the maps, estimated with a Fast-Fourier Transform (FFT), are retained as closely as possible. Since it is difficult to generate a map with both the exact same value distribution and the exact same spatial structure of the original map, a trade-off is made. The specific value distribution must take precedence, as most maps generated strictly from a spatial spectrum fail to maintain the sparsity of the original map, significantly skewing the reliability estimate. We therefore generate maps with the same spatial spectrum as the original map and retain the spatial trends while substituting the exact original value distribution. If $\mathbf{M}^{X\times Y\times Z}$ is the original map, we generate a surrogate $\tilde{\mathbf{s}} \in \mathbb{R}^M$ as follows:

   (a) Obtain the spatial frequency spectrum $\mathbf{F} \in \mathbb{R}^{X\times Y\times Z}$ for $\mathbf{M}$ by performing an FFT on the map and taking the absolute value of the resulting array of amplitudes.

$$\mathbf{F} = |\text{FFT}(\mathbf{M})| \tag{3.9}$$

$$\tag{3.10}$$

   (b) Generate a new map $\mathbf{F}^* \in \mathbb{R}^{X\times Y\times Z}$, initialized to all 0, by randomizing the phase of each frequency in $\mathbf{F}$, so for $i = \{1,\ldots,X*Y*Z\}$:

$$r \sim uniform([0,1)) \tag{3.11}$$

$$\mathbf{F}^*(i) = \mathbf{F}(i)\left(e^{(\sqrt{-1})2\pi r}\right) \tag{3.12}$$

(c) Compute a temporary surrogate $\mathbf{T} \in \mathbb{R}^{X \times Y \times Z}$ by taking the inverse FFT (IFFT) of $\mathbf{F}^*$:

$$\mathbf{T} = \text{IFFT}(\mathbf{F}^*) \tag{3.13}$$

(d) Let $\mathbf{v} \in \mathbb{R}^M$ be the vectorized form of map $\mathbf{M}$ after applying a brain map and let $\mathbf{t} \in \mathbb{R}^M$ be the vectorized form of $\mathbf{T}$ after applying the same brain map. Re-create the original value distribution in $\mathbf{M}$ by sorting the real values of $\mathbf{t}$, storing the sorted indices $\mathbf{I}$ as a rank ordering of locations, and sorting the weight values in $\mathbf{v}$. The ordered $\mathbf{v}$ values will be assigned to the rank-ordered locations $\mathbf{I}$. Let $\mathbf{s} \in \mathbb{R}^M = \text{sort}(\mathbf{t})$ return the elements of $\mathbf{t}$ in ascending order of their real values. Let $\mathbf{I} = \text{index}(\mathbf{s})$ return a vector the indices in $\mathbf{t}$ from which each subsequent element of $\mathbf{s}$ was drawn. Letting $\mathbf{t}(\mathbf{I})$ denote the elements of $\mathbf{t}$ at indices $\mathbf{I}$, $\mathbf{t}(\mathbf{I}) = \mathbf{s}$. The surrogate vector $\tilde{\mathbf{s}}$, initialized to all 0, is computed as:

$$\mathbf{I} = \text{index}(\text{sort}(\mathbf{t})) \tag{3.14}$$

$$\mathbf{v}^* = \text{sort}(\mathbf{v}) \tag{3.15}$$

$$\tilde{\mathbf{s}}(\mathbf{I}) = \mathbf{v}^* \tag{3.16}$$

Therefore, $\text{index}(\text{sort}(\tilde{\mathbf{s}})) = \text{index}(\text{sort}(\mathbf{t}))$ so that the map form of $\tilde{\mathbf{s}}$ exhibits the same relative spatial pattern as the generated map $\mathbf{T}$, based on the ordered locations of the real values of the two maps, but the set of values in $\tilde{\mathbf{s}}$ will be exactly the same as the values in the original map $\mathbf{M}$.

Note that an alternative "blob shift" method could be used to directly generate surrogates when a known underlying spatial process was used to generate the actual maps, as in the case of smoothing the parameter weights with a uniform Gaussian filter. In that case, surrogates may be obtained by smoothing the surrogates obtained using the non-spatial approach with the same kernel as was used to produce the actual map and, as in the FFT approach, replacing the surrogate values with the correspondingly ranked values in the actual map. The surrogates obtained from this approach, and their spatial spectra, are very similar to those obtained by the FFT-based method; however, the FFT-based method offers the strong advantage of being more general, applicable to maps for which the generating spatial process is unknown, as will usually be the case. In these cases, the "blob shift" approach reverts to the non-spatial generating process.

Significance Estimation Procedure

The complete procedure for producing a reliability estimate between two maps $\mathbf{M}_1$ and $\mathbf{M}_2$, given a chosen null hypothesis H_0 from among the two choices above, is therefore the following:

1. For each of the two maps, create a set of 20 surrogate vectorized maps using the procedure for H_0.
2. For each metric χ considered from the 3 choices above:
   (a) Calculate a score ψ for the real maps using the equation given for χ.
   (b) For each of the 400 (20×20) pairs of surrogates from different maps, calculate a score $\tilde{\chi}$.
   (c) Compute the mean μ and standard deviation σ over all surrogate scores $\tilde{\chi}$.
   (d) Compute a z-score for the real score as $z = \frac{\psi - \mu}{\sigma}$.

One can then use z as a measure of the significance of the observed reliability, given the metric chosen and the null hypothesis assumed.

It can be useful to consider this approach conceptually. For simplicity, consider only the non-spatial null hypothesis. If two vectors are exactly the same, the raw reliability metrics will be maximized, but the specific z-score will depend on the mean score among the surrogates, which depends, necessarily, on the data properties and metric considered. For instance, if two vectors are all 0 except for a 1 at one common location, the raw non-zero overlap will be 100%. The surrogate maps, however, will all have exactly one non-zero voxel, but that location will be random, so the majority of sample non-zero overlap scores will be 0% and the z-score will be quite high. In fact, it is possible that every observed sample will be 0%, so the standard deviation σ of the estimate will be 0 and the z-score will be ∞.

In contrast, if the two vectors have a 1 at each location except one common location, the observed non-zero overlap will still be 100%, but the surrogates will have all but one location commonly selected, so the sample non-zero overlap will be very close to 100%, and the z-score likely lower; however, it is equally likely that the standard deviation σ will be 0, so the z-score would still be ∞. While this phenomenon may appear to be a confound, it is in fact an accurate reflection of the significance of the observed

overlap. If there are 100 voxels and all but one is non-zero, the expected non-zero overlap is $98/100 = 98\%$ with nearly no variability. The observed overlap of 100% is in fact significant in this context. We will observe that smoothing results in a scenario much like this example, leading to such high-mean, low-variance estimated scores, but yielding appropriately high z-scores as a result.

Fisher's Exact

We also consider one other approach to significance estimation of non-zero overlap, the Fisher's Exact test for the hypergeometric distribution [35], discussed briefly in Section 3.1 and defined as:

$$P(k) = \frac{\binom{n_1}{n_b}\binom{M-n_1}{n_2-n_b}}{\binom{M}{n_2}} \tag{3.17}$$

$$P(b) = \sum_{k=0}^{\min(n_1,n_2)} p(k) - \sum_{k=0}^{n_b-1} p(k) \tag{3.18}$$

$P(b)$ can be obtained by computing a Fisher's Exact (or chi-square) test on the following cross-tabulation matrix:

$$c = \begin{bmatrix} n_b & (n_1-n_b) \\ (n_2-n_b) & (n-n_1-n_2+nb) \end{bmatrix} \tag{3.19}$$

The null hypothesis in this case is that the columns of c are generated independently, hence the proportion in any entry of table c is the product of the proportions in each dimension of the matrix. When using this test, we report the F-score.

3.2.6 Ground Truth Comparison Metrics

The reliability estimation procedure described above is *generic* in that it works for any two maps regardless of how the maps were produced. We primarily consider maps that

have been learned through a modeling procedure, but they could just as easily be maps that a human created, whether through expert knowledge or artificial design, or through some other technique. Learned maps and designed maps can also be directly compared using this approach.

To further explore the properties of the reliability evaluation approaches, we can exploit both this flexibility in the metrics and estimation approaches, and our knowledge of the "true" parameter map on the synthetic data, to evaluate the correspondence not only between two learned maps, but between each learned map and the *ground truth* map. We can then use this relationship to demonstrate an interesting and appealing property of both the overlap metric and the resampling approach to evaluating reliability. We define:

- $\tilde{\mathbf{v}} \in \mathbb{R}^M$, the vectorized form of the learned map (note that for the synthetic data, the vectorized form is equivalent to the map since there is only one non-singular spatial dimension in the map).
- $\mathbf{v} \in \mathbb{R}^M$, the vectorized form of the true map.
- $\tilde{R} = v : \tilde{\mathbf{v}}(v) \neq 0$, the set of indices of non-zero voxels in $\tilde{\mathbf{v}}$.
- $R = v : \mathbf{v}(v) \neq 0$, the set of indices of non-zero voxels in $\mathbf{v}$.
- $\tilde{n} = |\tilde{R}|$, the number of non-zero voxels in $\tilde{\mathbf{v}}$.
- $n_t = |R|$, the true number of non-zero voxels. $n_t = 358$ for this synthetic dataset.
- $B_t = \tilde{R} \cap R$, the set of true positives.
- $n_c = |B_t|$, the number of true positives.

We can compute the non-zero overlap with ground truth by replacing n_b in Eq. (3.4) with n_c, n_1 with $\tilde{n}$, and n_2 with n_t.

When the ground truth is known, it is common to compute the sensitivity and specificity of the methodology. We can compute these metrics as follows:

1. Sensitivity:
$$sens = \frac{n_c}{n_t} \tag{3.20}$$
2. Specificity:
$$spec = \frac{n_c}{\tilde{n}} \tag{3.21}$$

From Eqs. (3.4), (3.20), and (3.21), we see that the **non-zero overlap** ψ between a learned vector and the ground truth vector has the appealing property of being directly related to sensitivity and specificity:

$$\psi = \frac{1}{\frac{1}{sens} + \frac{1}{spec} - 1}. \tag{3.22}$$

3.3 Results

3.3.1 Non-Spatial Null Hypothesis

We begin by evaluating the relevance of the simpler of the two null hypotheses, which considers only the set of map weight values, without regard to spatial structure. We demonstrate the relationship between properties of the set of weight values, in particular the number of non-zero voxels, and the 3 metrics described above, with or without significance estimation. We pay special attention to the inverse sparsity level, approximated as the number of Elastic Net training iterations, and smoothing, both of which increase the number of non-zero voxels. We also consider the effects of increasing the λ_2 value. To rigorously evaluate the relationship between the number of training iterations and the reliability scores, we focus first on the synthetic data, for which it is feasible to evaluate reliability over the full regularization path.

The overlap metric is the number of elements in the intersection of the two sets of voxels divided by the number in their union; thus, if the intersection is equal to the union, the overlap will be 100%. Therefore, the simplest way to increase the score on this metric and its weighted variant is to increase the number of selected voxels. Figure 3.1 confirms this trend by showing the mean of the 2 overlap metrics for the synthetic data as the number of training iterations increases. Separate series are shown for every λ_2 value tested. Figure 3.1(a) shows that as the number of selected voxels approaches its maximum of all 800 voxels, the overlap tends to increase towards 100%, regardless of the λ_2 value. For some ranges, there is variability in the metric, especially for lower (darker-colored) λ_2 values, but this variability completely washes out as the number of selected voxels increases. This metric does have some merit though: within the range where there is the most variability, the scores reach a local maximum, then dip before again increasing

monotonically. This non-monotonicity likely reflects the point at which increases in the size of the intersection of the sets shifts to being due mostly to the increase in the union of the two sets. Similar results are obtained for weighted overlap in Figure 3.1(b).

Figures 3.1(c-d) show the same two metrics plotted after smoothing is applied to the parameter weights. Note that, with smoothing, the overlap scores approach 1.0 much more rapidly; in addition, after the transition point, lower λ_2 values actually result in greater overlap.

The overlap metrics are therefore confounded, but, given their desirable interpretation properties, it seems over-zealous to discount them entirely. What is needed is a metric for evaluating for statistical significance, or meaningfulness, of the observed overlaps. The described non-spatial null hypothesis resampling approach can be used to provide such an estimate. Figures 3.2(a-b) show that z-scores for both non-weighted and weighted overlap are not confounded by the number of selected voxels, and in fact are penalized as the observed overlap has a greater likelihood of being due to chance. The same peak observed in Figures 3.1(a-b) is again evident in 3.2(a-b), but the score is not overcome by the relative increase in the size of the intersection of the two sets relative to the size of the union. The local maximum in Figures 3.1(a-b) becomes a global maximum, reflecting the truly most reliable point. Also, note again that there is a high variability in the score among λ_2 values, particularly the lowest values, indicating that λ_2 can truly influence reliability.

Figures 3.2(c-d) show that these scores approach 0 more rapidly when smoothing is applied to the weight maps, since the maximum number of voxels is reached more quickly.

The significance estimation approach that employs the Fisher's Exact calculation is also, in effect, measuring the significance of the non-zero overlap. Therefore, we can assess the validity of our significance estimation approach by comparing our results for the non-spatial non-zero overlap z-score to results obtained using the Fisher's Exact approach. Figure 3.3 shows that the trend in the F-score produced by this measure is in fact strikingly similar to that of the z-score produced by the resampling method, which is duplicated from Figure 3.1(a), although the Fisher's measure results in F scores that are greatly inflated relative to the estimates obtained non-parametrically, and reach 0 more rapidly. The convergence of the two trends points to a true significance of the overlap, but the differences underscore the potential dependence of the F-score on the number of

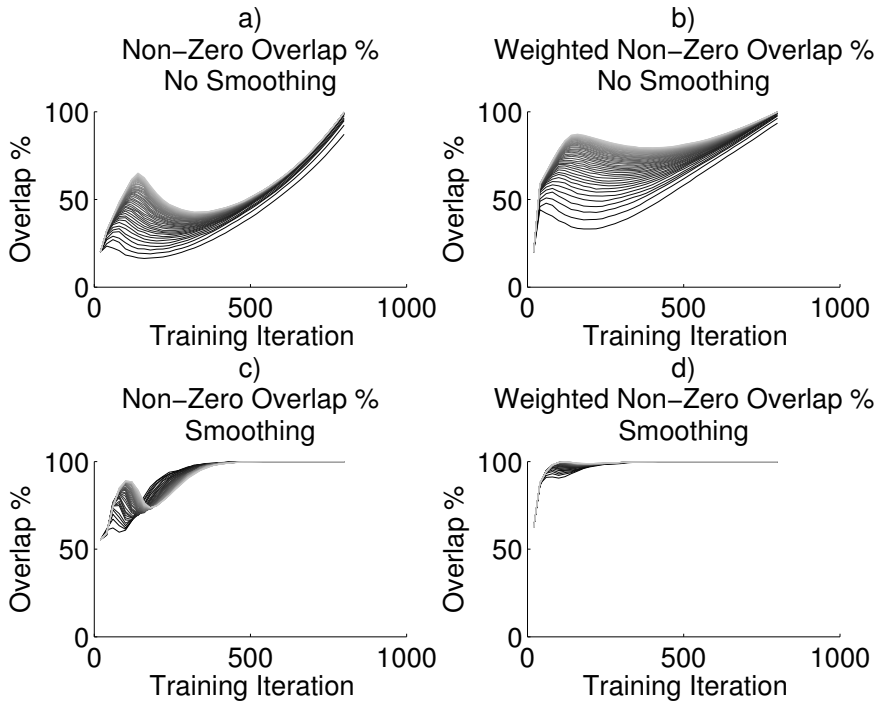


Figure 3.1: *Uncorrected overlap metrics are dependent on the number of non-zero voxels, rapidly approaching 1.0 for smoothed maps.* Uncorrected scores by metric (overlap or weighted overlap) and smoothing (with or without); all lines are mean score for metric on synthetic data versus number of training iterations; series are λ_2 values, with lighter colors corresponding to higher values.

selected voxels cautioned by Fury et al. [35]. Note again, though, that the differences among λ_2 values are still highly significant.

An alternative reliability metric to overlap scores is map Pearson correlation. Figure 3.4 illustrates, on the synthetic data, the appealing property that correlation is not linearly dependent on the number of selected voxels. In fact, it displays a trend similar to that of the z-scores, with a slight peak before a gradual monotonic decrease, though the peak here occurs slightly later than in the overlap measures. It also exhibits another informative property: high variability among the λ_2 values.

3.3.2 Overlap, Sensitivity, and Specificity

In this section, we use the procedure we described for comparing learned maps with ground truth maps to reveal an interesting relationship between the overlap metrics and

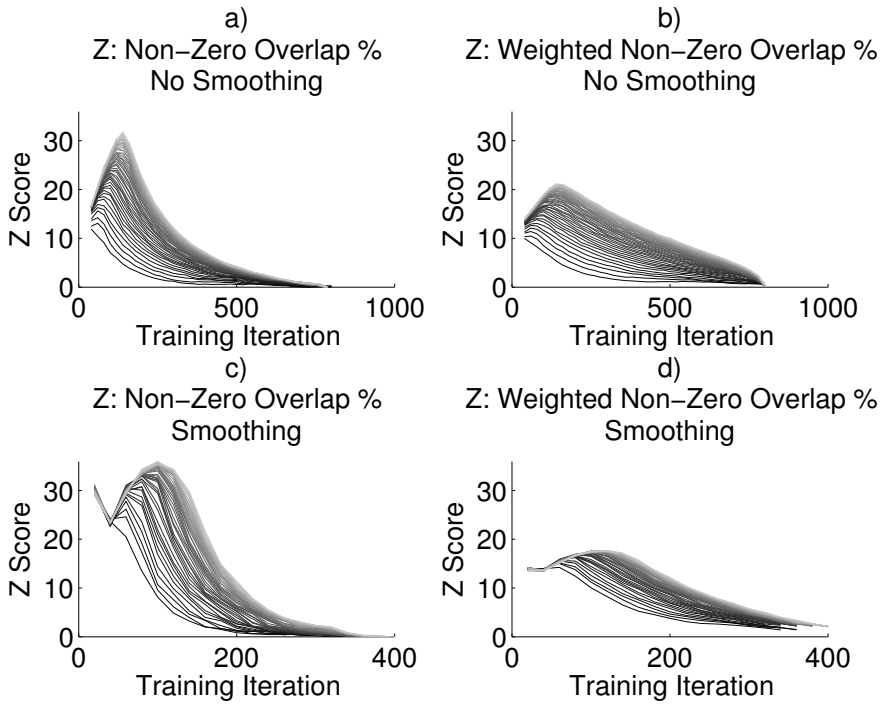


Figure 3.2: *Non-Spatial Z-Scores are not linearly dependent on sparsity.* Non-Spatial Z-Scores by metric (overlap or weighted overlap) and smoothing (with or without); all lines are mean score for metric on synthetic data versus number of training iterations; series are λ_2 values, with lighter colors corresponding to higher values.

sensitivity and specificity, especially when taking significance into account. From Eq. (3.22), as the sensitivity approaches 1 (or 100%), the overlap will approach the specificity and vice versa. When we plot the mean overlap between the learned synthetic parameter maps and the ground truth along with sensitivity and specificity in Figure 3.5, we see results consistent with this relationship. Sensitivity in (a)(ii) starts low and grows linearly with the number of voxels selected, approaching the maximum of 100%. Specificity in plot (a)(iii) starts out high, as the first few voxels chosen are likely to be true positives. Overlap in (a)(i) starts off similarly to sensitivity but slowly crosses over to closely match specificity. The same trend is true for both non-smoothed and smoothed parameter maps. In Figure 3.5(b), however, we consider the significance estimates of these measures. We use the non-spatial resampling approach to obtain z-scores for these three measures. The figures reveal two intriguing properties: (1) the sensitivity and specificity z-scores are nearly identical, even when smoothing is applied, and (2) the overlap z-scores trend is

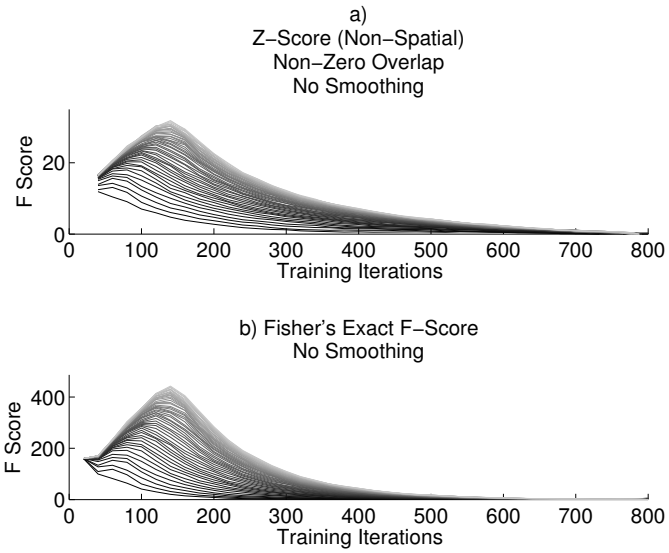


Figure 3.3: *The trend from the Fisher's Exact estimate is nearly identical to that of the non-zero overlap z-score.* (a) Replicate of Figure 3.2(a); (b) F-Score for Fisher's Exact Test on overlap cross-tabulation matrix. All lines are mean score for metric on synthetic data versus number of training iterations; series are λ_2 values, with lighter colors corresponding to higher values.

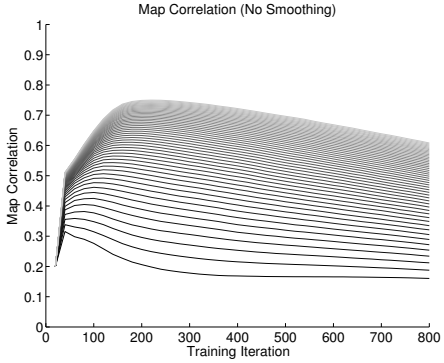


Figure 3.4: *Map correlation is less dependent on the number of non-zero voxels.* Reliability measured as Pearson correlation for synthetic data; all lines are mean score for metric versus number of training iterations; series are λ_2 values, with lighter colors corresponding to higher values.

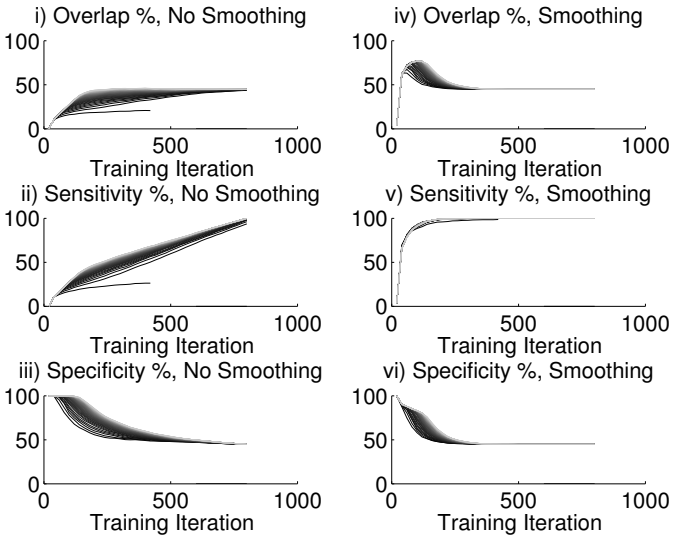
strikingly similar to that of sensitivity and specificity, though slightly higher, even when smoothing is applied.

Most interestingly, Figure 3.6 shows Figures 3.2(i) and 3.5(b)(i) side by side, illustrating that the z-scores for overlap between the learned maps and the ground truth are quite similar to those for the reliability between pairs of vectors of the same sparsity. While the reliability scores reach higher values as λ_2 increases, both scores have a global maximum at nearly the same point. These results suggest that reliability between learned maps provides some information about model validity.

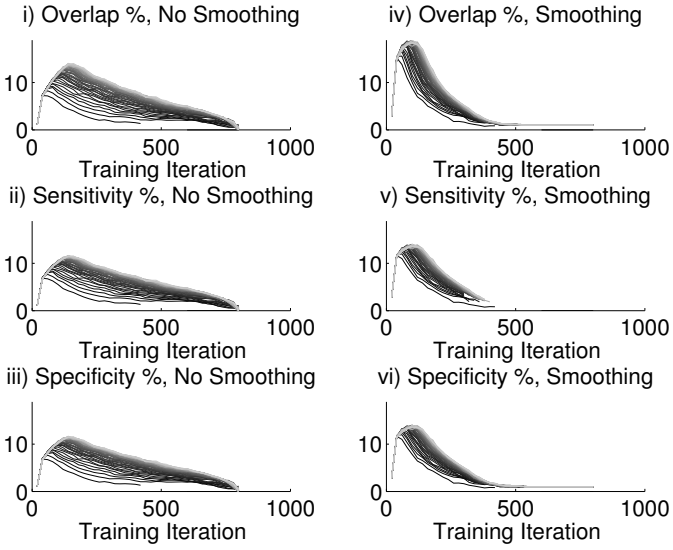
3.3.3 Spatially-Based Surrogate Generation Procedure

Subsequent results will hinge on the ability of the estimation procedure to approximate the distribution $\mathcal{D}$, which in turn relies on an accurate retention of the key map properties when producing the surrogate maps. The non-spatial generating procedure by definition retains the key property under H_0, which is the specific set of weight values. The spatially-based generating procedure retains this property as well, but, as a trade-off, approximates the spatial structure of the map. We therefore begin by examining the impact of this trade-off, by evaluating how well the FFT-based approach retains the spatial structure of the map.

We first consider the surrogate maps obtained for the synthetic dataset, for which the "ground truth" map is known and can provide insight. Figure 3.7(a)(i) shows the ground truth ("template") parameter map for the synthetic dataset and (ii) shows the map learned using Elastic Net using a λ_2 of 2.0. (iii-iv) show 2 representative sample surrogates generated from this learned map using the FFT approach. Figure 3.7(b) shows the absolute value of the spatial spectra for the maps in Figure 3.7(a). It is clear that there is a great amount of spatial structure in the data that Elastic Net has failed to capture; however, the spatial approach has generated surrogates with similar spatial spectra to the learned maps and, by definition, identical value distributions. Furthermore, as Figure 3.7(d) shows, applying even a small amount of smoothing results in a learned map with a spatial spectrum more closely matching that of the ground truth map. The similarity between the spatial structure in the surrogates and that of the learned map is much more evident in this case, demonstrating that the spatial surrogate-generating procedure captures relative differences in map spatial structure.



(a) Uncorrected Ground Truth Scores



(b) Ground Truth Z-Scores

Figure 3.5: *Uncorrected ground truth overlap combines sensitivity and specificity, which are intuitively different; however z-scores (non-spatial) for sensitivity and specificity are nearly identical to both each other and non-zero overlap z-scores.* (i) Non-Zero Overlap with ground truth (uncorrected or Z), (ii) Sensitivity (uncorrected or Z), (iii) Specificity (uncorrected or Z). All lines are mean score for metric on synthetic data versus number of training iterations; series are λ_2 values, with lighter colors corresponding to higher values.

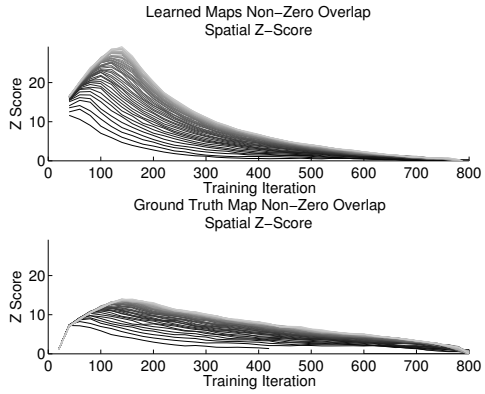


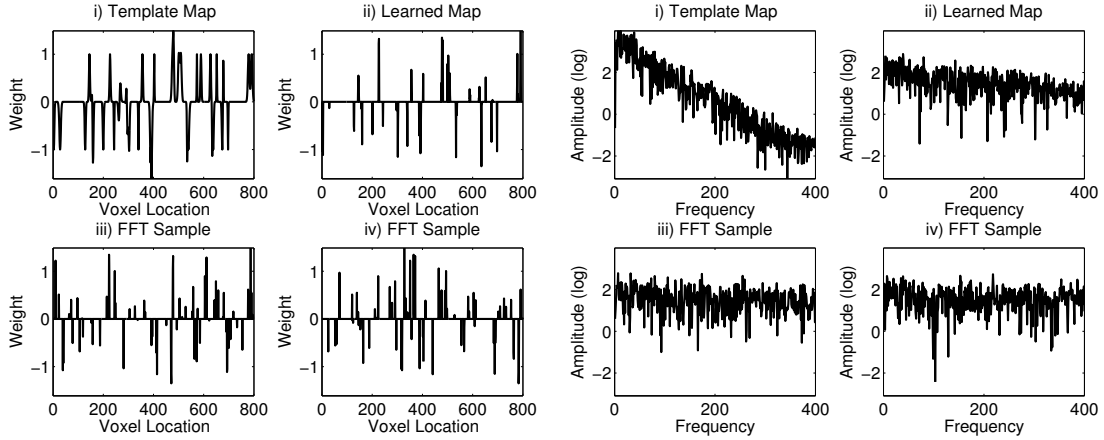
Figure 3.6: *Z-scores for overlap between two learned maps and overlap with ground truth are very similar.* (a) Non-Zero Overlap between learned un-smoothed maps (Figure 3.2(i)); (b) Non-Zero Overlap between learned map and ground truth map (Figure 3.5(b)(i)); All lines are mean score for metric on synthetic data versus number of training iterations; series are λ_2 values, with lighter colors corresponding to higher values.

Figure 3.8 shows the spatial spectra for the actual and surrogate maps for the real data. The effects of the value vs. spatial preservation trade-off are more evident here: the surrogate maps exhibit some loss of spatial information; however, overall trends remain. While the surrogates do underestimate the spatial structure, the underestimation is consistent, so surrogate maps generated from two maps with different spatial structure should retain the relative differences between the original maps and maps will not be over-penalized for spatial structure in the resulting z-scores.

An example of an FFT-based surrogate is shown in Figure 3.9. Figure 3.9(a) is the actual map learned for one subject for the PBAIC Instructions task with $\lambda_2 = 2.0$, trained until 2500 voxels were selected, and then smoothed with a 2mm filter. Figure 3.9(b) is an example surrogate for this map, generated with the FFT-based approach. The set of weights are exactly the same, but have been re-distributed while retaining the approximate spatial structure.

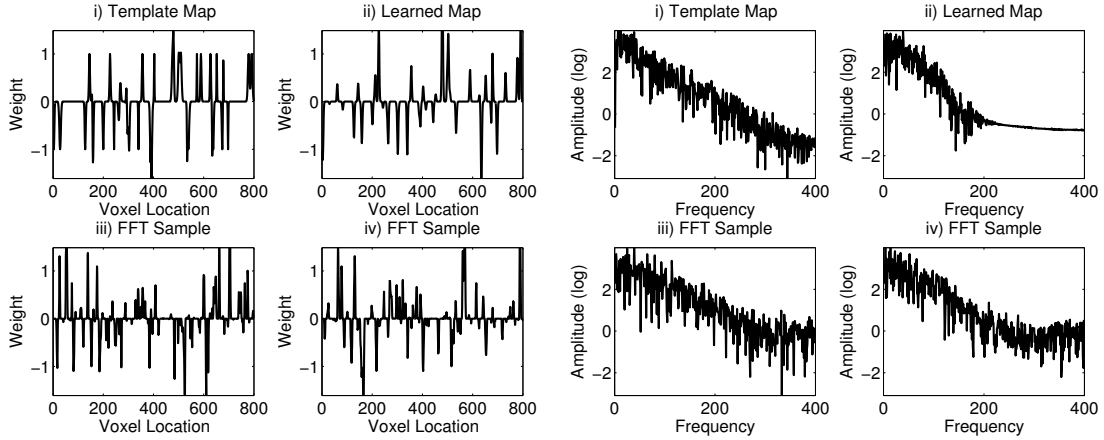
3.3.4 Smoothing and Prediction

Having established some validity for the reliability estimating procedure, we now use the method to explore reliability properties of real fMRI maps. We will consider maps that vary on two dimensions, λ_2 training value and smoothing parameter. In Chapter 2



(a) Maps: $\lambda_2 = 2.0$

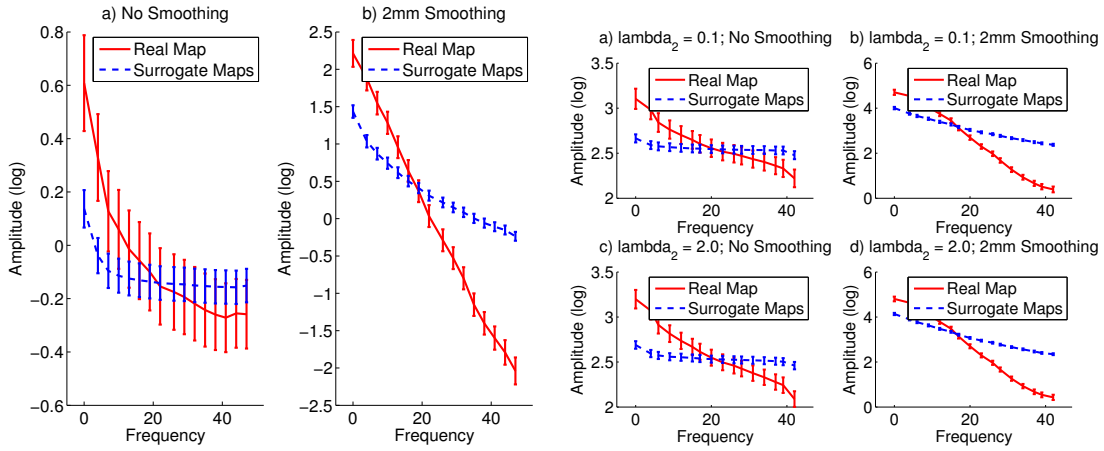
(b) Spatial Spectra: $\lambda_2 = 2.0$



(c) Maps: $\lambda_2 = 0.1$ Smoothed

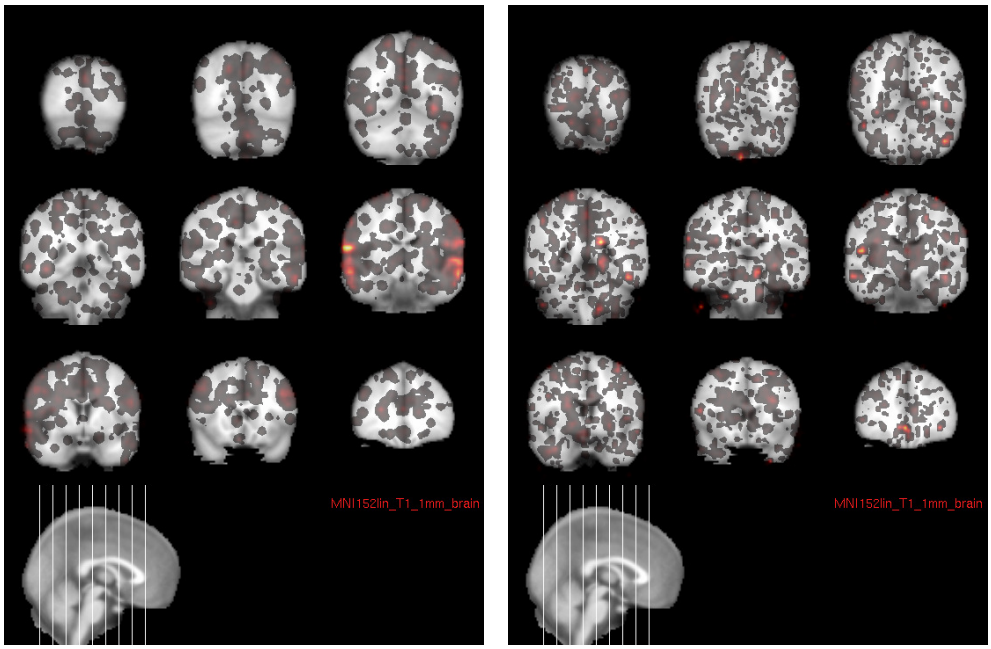
(d) Spatial Spectra: $\lambda_2 = 0.1$ Smoothed

Figure 3.7: *FFT-based surrogates have similar spatial structure to the actual learned maps, which do not always capture the spatial structure in the ground truth map.* Maps and associated spatial spectra (absolute values) for spatial surrogate maps generated from learned maps on synthetic data. (i) corresponds to the ground truth map for the dataset, (ii) to the learned map, and (iii-iv) to the surrogates.



(a) PBAIC Spectra (b) Pain Spectra

Figure 3.8: *For real maps, FFT-based surrogates lose some spatial structure, but retain much of the comparative spatial information.* 3D spatial spectra, collapsed to 1D, for real and FFT-based surrogate maps. PBAIC results (a) averaged over all 3 subjects, 24 tasks, and 4 λ_2 values; Pain results (b) shown by λ_2 and averaged over all 14 subjects and 2 tasks; standard error bars are shown.



(a) Original Map (b) Surrogate Map

Figure 3.9: *An original smoothed map on real data and sample FFT surrogate are shown.* (a) PBAIC Subject 1 Run 1 Instructions trained with $\lambda_2 = 2.0$ and smoothed with a 2mm filter; (b) FFT-based surrogate for map in (a).

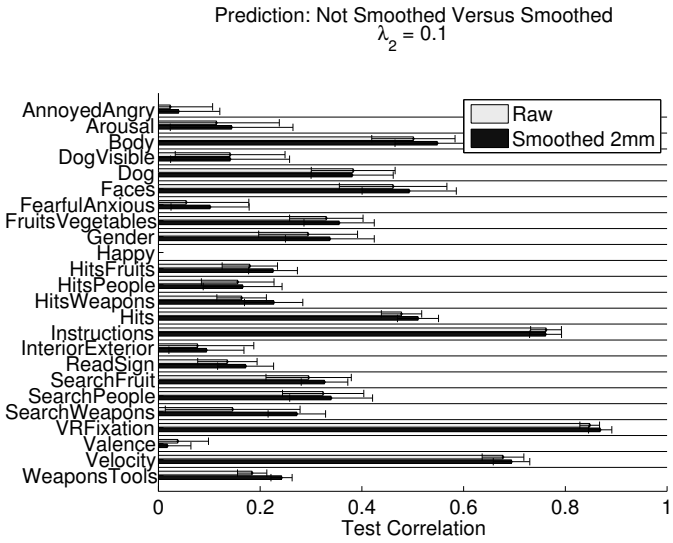
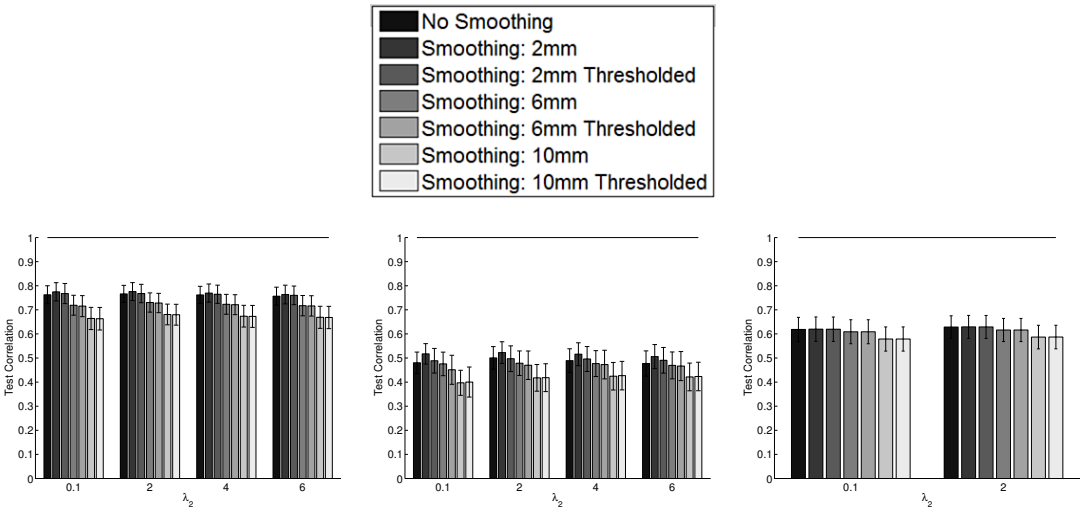


Figure 3.10: *A small amount of map smoothing does not impair prediction performance.* Test correlation (cross-validated) for all PBAIC 2007 tasks for λ_2 of 0.1, with or without applying a 2mm smoothing filter to the weights, averaged over all 3 subjects and 2 runs. Bars reflect 95% confidence.

we showed that prediction performance remains stable as λ_2 is increased (we expand those results in Appendix Section A.1). Intuition, however, suggests that smoothing maps should impair prediction, so we focus here on assessing the prediction performance of smoothed maps, beginning in Figure 3.10 with the effect on all 24 PBAIC tasks. This figure shows that applying a relatively small amount of spatial smoothing, on the order of 2mm, does not impair prediction performance. In fact, in most cases when prediction is significantly affected, smoothing actually improves the performance.

As the amount of smoothing is increased, however, prediction performance does begin to suffer. Figure 3.11 shows the effects of both increasing λ_2 and increasing the amount of smoothing for 3 sets of predicted tasks: the best predicted PBAIC 2007 responses (Instructions, VRFixation, and Velocity), the 3 more moderately predicted PBAIC responses (Body, Faces, and Hits), and the 2 Pain tasks (actual temperature and perception). Note that prediction for the 2 Pain tasks have been aggregated in this figure; prediction performance for the two tasks was not significantly different. For all sets of tasks, a *small* amount of smoothing does *not* affect prediction, but prediction does begin to degrade as more smoothing is applied.



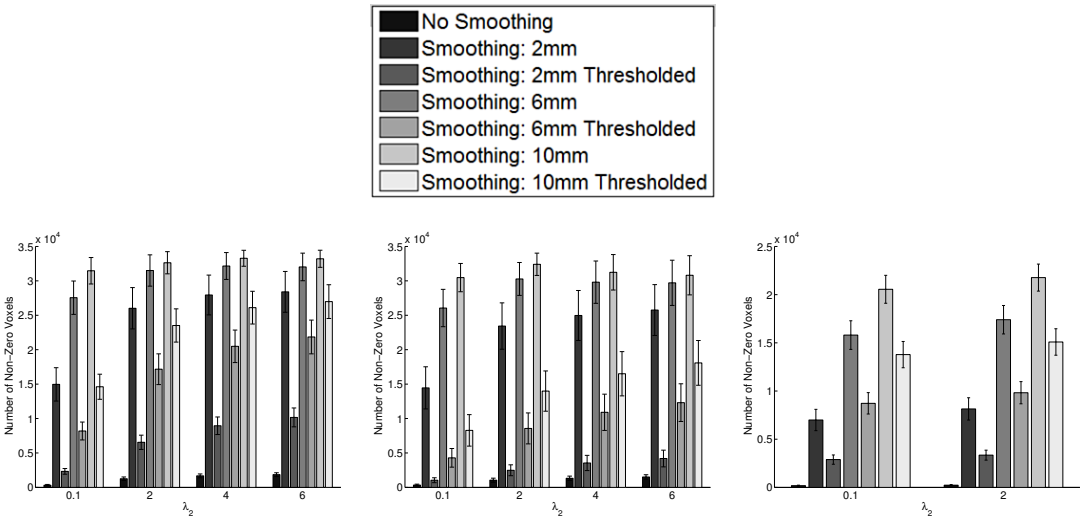
(a) Best-predicted PBAIC tasks (b) Moderately well-predicted PBAIC tasks (c) Pain tasks

Figure 3.11: *Prediction performance degrades slightly as larger smoothing kernels are applied to the map.* Test correlation (cross-validated) by λ_2 and smoothing parameters, for PBAIC Best Predicted (a), PBAIC Moderately Well-Predicted (b), and Pain (c) tasks; averaged over all 3 (PBAIC) or 2 (Pain) tasks, 3 (PBAIC) or 14 (Pain) subjects, and 2 runs. Bars reflect 95% confidence.

3.3.5 Reliability Estimates

As Section 3.3.1 made clear, when evaluating reliability, it is first important to consider the number of non-zero voxels in the maps. As discussed in Chapter 2, as λ_2 is increased, a greater number of voxels are needed to obtain the same prediction performance. Thus, training a model with a higher λ_2 and using cross-validation to select an optimal number of voxels results in maps with a greater number of non-zero voxels. Figure 3.12 shows that, regardless of task, this effect holds and, furthermore, smoothing increases the number of non-zero voxels. In fact, even applying a relatively small spatial kernel results in most voxels being selected. Applying the threshold of 0.01 drastically decreases this number, making the models again sparse, yet the number of voxels is still greater than those yielded prior to smoothing. Note also the very small number of voxels selected by cross-validation for the Pain tasks; as a result, even maps smoothed with a large kernel do not tend to fully saturate.

In Figures 3.13 and 3.14, we apply our 3 reliability metrics and 2 null hypotheses to evaluate the reliability of the PBAIC maps. We first note that the trend observed in Chap-



(a) Best-predicted PBAIC tasks (b) Moderately well-predicted PBAIC tasks (c) Pain tasks

Figure 3.12: *Increasing λ_2 and smoothing, with or without thresholding, increase the number of non-zero voxels.* Number of non-zero weighted voxels by λ_2 and smoothing parameters for PBAIC Best Predicted tasks (a), PBAIC Moderately Well Predicted Tasks (b), and Pain tasks (c); averaged over all 3 (PBAIC) or 2 (Pain) tasks, 3 (PBAIC) or 14 (Pain) subjects, and 4 cross-validation folds. Bars reflect 95% confidence.

ter 2 for reliability, measured as non-zero overlap, to increase with λ_2 is again observed in Figure 3.13(a), but there are diminishing returns, perhaps due to the computational necessity of limiting the maximal number of included voxels. In Figure 3.13(b), we take a first look at the weighted overlap metric in Figure 3.13(b), and find that scores for it are higher than for non-zero overlap. We would expect these scores to be at least as high as non-zero overlap; the fact they are significantly higher shows, encouragingly, that the most significant voxels are most likely to be selected in both runs. Note, though, that regardless of λ_2 value, both overlap and weighted overlap for the raw maps are far from the ideal 100%. As shown in the Appendix, these trends for increased non-zero overlap, and higher weighted overlap, were also found for the other PBAIC tasks.

When examining the uncorrected metrics in Figures 3.13a-c, it is clear that smoothing has a dramatic impact; however the pattern observed in Figures 3.13(a) and (b) looks nearly identical to the number of non-zero voxels plotted in Figure 3.12(a), as we would expect, given the dependence of these metrics on the number of non-zero voxels. Correlation in Figures 3.13(c) presents a more interesting trend, tending to increase as more smoothing is applied, but not nearly as dramatically as the overlap measures, and being

identical whether or not thresholding is applied. This result is consistent with the finding that correlation is not directly dependent on the number of non-zero voxels. Again, the same effect is observed in Figure 3.14(c).

Figures 3.13(d-f) show the z-scores for these metrics assuming the non-spatial null hypothesis, which accounts for the specific value distribution in the data. The non-zero overlap measure is highly dependent on the sparsity level in the data. The non-spatial null hypothesis captures this information and, as a result, all of the smoothed models are sharply penalized, although for smaller λ_2 values, they remain significantly better than the non-smoothed map. Figure 3.13(e), however, reveals a slightly different dependency for the weighted overlap metric that is not apparent in the uncorrected scores. For this metric, the weights of consistently selected voxels relative to the remaining weights is crucial. Smoothing necessarily decreases this contrast, but thresholding recovers some of the lost information. As a result, the purely smoothed maps are harshly penalized relative to their thresholded counterparts. For this null hypothesis, the thresholded maps actually remain significantly more meaningfully reliable than the non-smoothed map, but there is a slight trend toward decreasing reliability as more smoothing is applied. Note that since the highest weighted voxels by definition have the most impact on prediction, the blurring induced by smoothing is also likely responsible for the impaired prediction performance at higher smoothing levels.

In contrast, Figure 3.13(f) provides yet another illustration of the independence of the correlation metric and the value distribution of the data. The pattern in Figure 3.13(f) is virtually identical to that of 3.13(c).

While the non-spatial null hypothesis clearly applies penalization to the uncorrected metrics, Figures 3.13(e-g) show that this null hypothesis does not go far enough. First, notice that across all 3 metrics, the z-scores with the spatial null hypothesis are lower, showing that spatial structure increases the expected reliability regardless of metric. This effect is especially strong for weighted overlap and correlation, which depend on the specific distribution of real weight values. For weighted overlap in 3.13(h), even the thresholded maps are significantly impacted by this penalty, with only the least smoothed map remaining competitive with the original map.

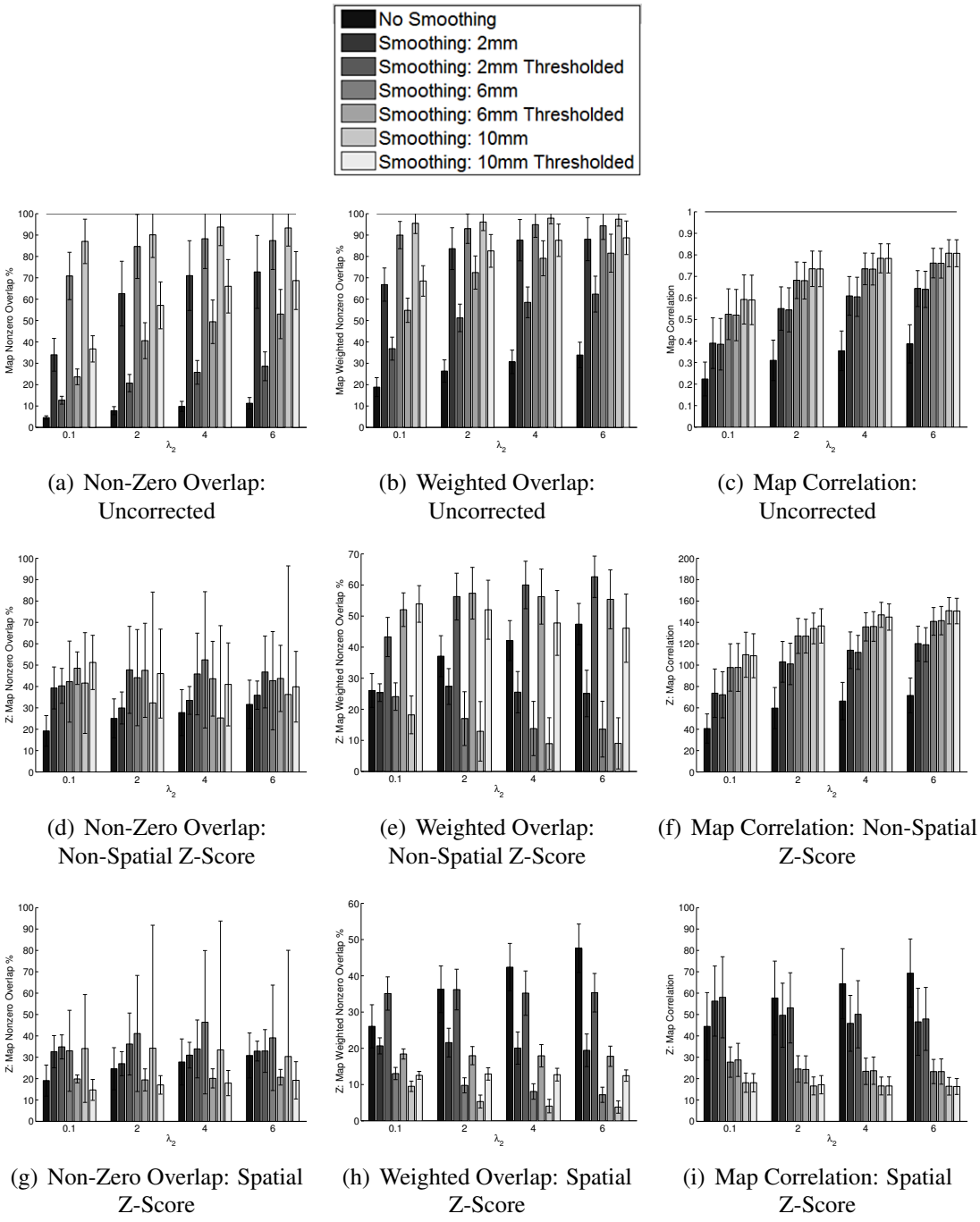
The most dramatic effect of the spatially-based null hypothesis is on correlation in 3.13(i). The trend observed in both 3.13(c) and (f) is almost directly *reversed*; greater amounts of smoothing are shown to produce maps that are much less meaningfully reli-

able. The weighted overlap and correlation metrics are also seen to converge more closely under this null hypothesis: both show large amounts of smoothing to be universally detrimental, and smaller amounts applied to maps trained with low λ_2 values to be less detrimental or perhaps beneficial, especially if thresholding is applied. Note that all 3 metrics show that the reliability gains produced with λ_2 increases alone are significant or significantly trending. For these tasks, increasing λ_2 is the most consistent way to improve reliability.

The results for the 3 moderately well-predicted PBAIC tasks, in Figure 3.14, are similar to those of the best predicted tasks in Figure 3.13 in several ways. First, as with the best-predicted tasks, the non-spatial null hypothesis (Figures d-f) penalizes the smoothed maps, but all maps display significant overlap. The thresholded maps again retain stronger weighted overlap scores relative to the non-smoothed maps, and again all forms of smoothing retain relatively strong correlation scores relative to non-smoothing. Just as with the other tasks, the spatial null hypothesis (g-i) penalizes all maps, but especially the most smoothed maps, reversing the trend for the correlation metric.

There are, however, also some notable differences in the results between the sets of tasks in Figures 3.13 and 3.14. First, all 3 uncorrected metrics in Figures 3.14(a-c) are lower than for the best-predicted tasks. More importantly, the maps produced using a small amount of smoothing and thresholding still remain significantly *more reliable* even when accounting for spatial structure. In fact, for these tasks, regardless of metric, this simple approach of smoothing then applying thresholding produces greater gains in reliability than those achieved solely through increasing λ_2. Smoothing and thresholding a map trained with the highest λ_2 value considered, which is still within a range for which computation is manageable, produces the most meaningfully reliable maps.

In Figure 3.15, we observe for the Pain data the same general trend observed for both sets of PBAIC tasks, in that the overall effect of smoothing reverses when spatial information is incorporated into the null hypothesis so that the greatest amount of smoothing is shown to be the least significantly reliable. However, as with the moderately well-predicted PBAIC tasks, which Figure 3.11 showed to have similar test correlations to the Pain tasks, some smoothing improves reliability even when accounting for spatial structure. The Pain tasks have the lowest scores for the raw (no smoothing) maps and, correspondingly, receive the greatest boost in reliability after some smoothing is applied. In fact, for the Pain tasks, *any* form of smoothing, with or without thresholding, improves



(a) Non-Zero Overlap: Uncorrected

(b) Weighted Overlap: Uncorrected

(c) Map Correlation: Uncorrected

(d) Non-Zero Overlap: Non-Spatial Z-Score

(e) Weighted Overlap: Non-Spatial Z-Score

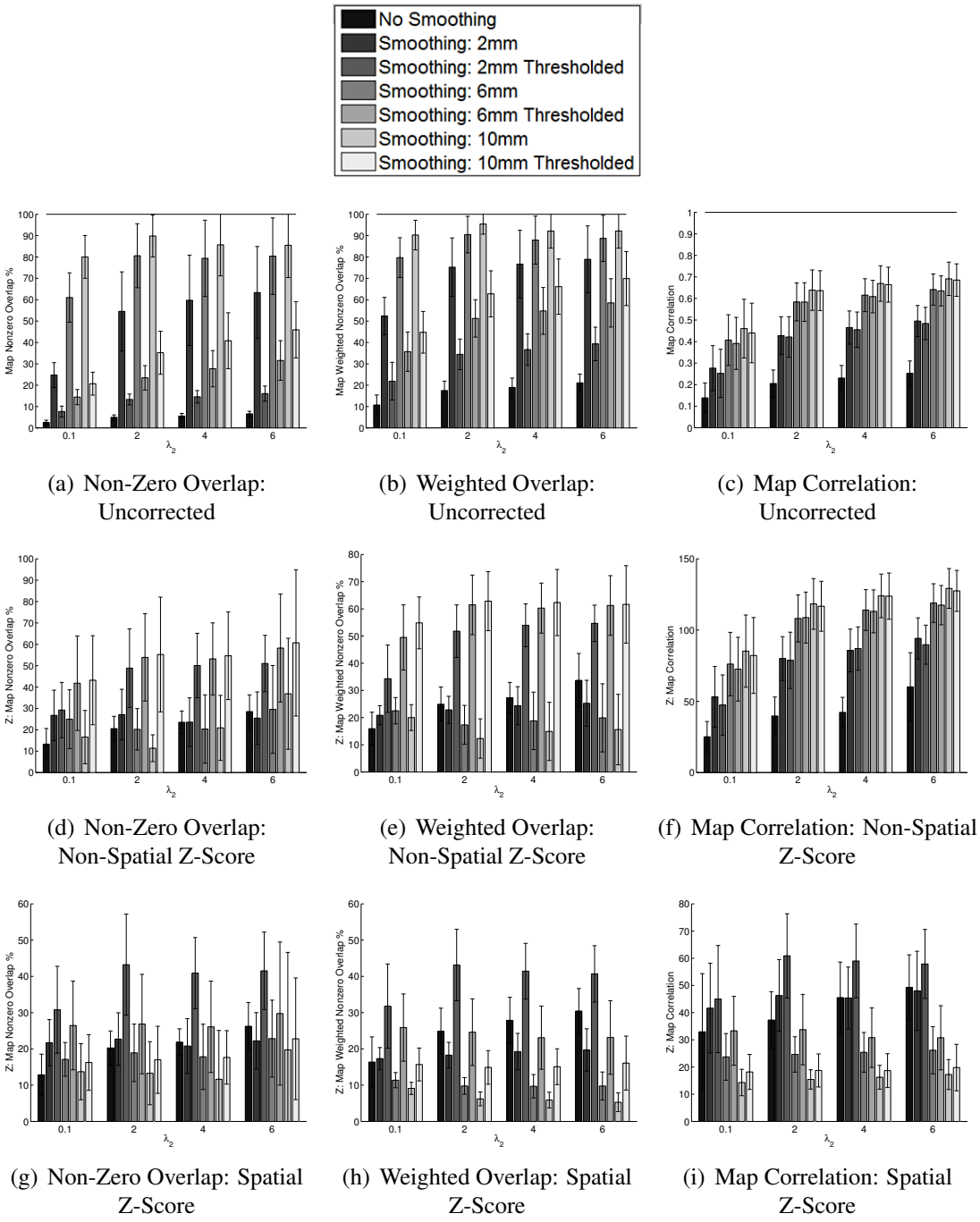
(f) Map Correlation: Non-Spatial Z-Score

(g) Non-Zero Overlap: Spatial Z-Score

(h) Weighted Overlap: Spatial Z-Score

(i) Map Correlation: Spatial Z-Score

Figure 3.13: *For the **best-predicted PBAIC tasks**, z-scores underscore the impact of weight value distribution and spatial structure on all 3 reliability metrics, with λ_2 increases providing the most meaningful reliability improvements.* Uncorrected scores for the 3 metrics for the 3 best predicted tasks, along with z-scores given the non-spatial and spatial null hypotheses. Means are calculated over all 3 tasks, 3 subjects, and 4 cross-validation folds; variance over tasks and subjects. Bars reflect 95% confidence.



(a) Non-Zero Overlap: Uncorrected

(b) Weighted Overlap: Uncorrected

(c) Map Correlation: Uncorrected

(d) Non-Zero Overlap: Non-Spatial Z-Score

(e) Weighted Overlap: Non-Spatial Z-Score

(f) Map Correlation: Non-Spatial Z-Score

(g) Non-Zero Overlap: Spatial Z-Score

(h) Weighted Overlap: Spatial Z-Score

(i) Map Correlation: Spatial Z-Score

Figure 3.14: *For the **moderately well-predicted PBAIC tasks**, z-scores underscore the impact of weight value distribution and spatial structure on all 3 reliability metrics, with smoothing with a small kernel and thresholding resulting in reliability competitive with λ_2 increases.* Uncorrected scores for the 3 metrics for the 3 moderately well-predicted PBAIC tasks, along with z-scores given the non-spatial and spatial null hypotheses. Means are calculated over all 3 tasks, 3 subjects, and 4 cross-validation folds; variance over tasks and subjects. Bars reflect 95% confidence.

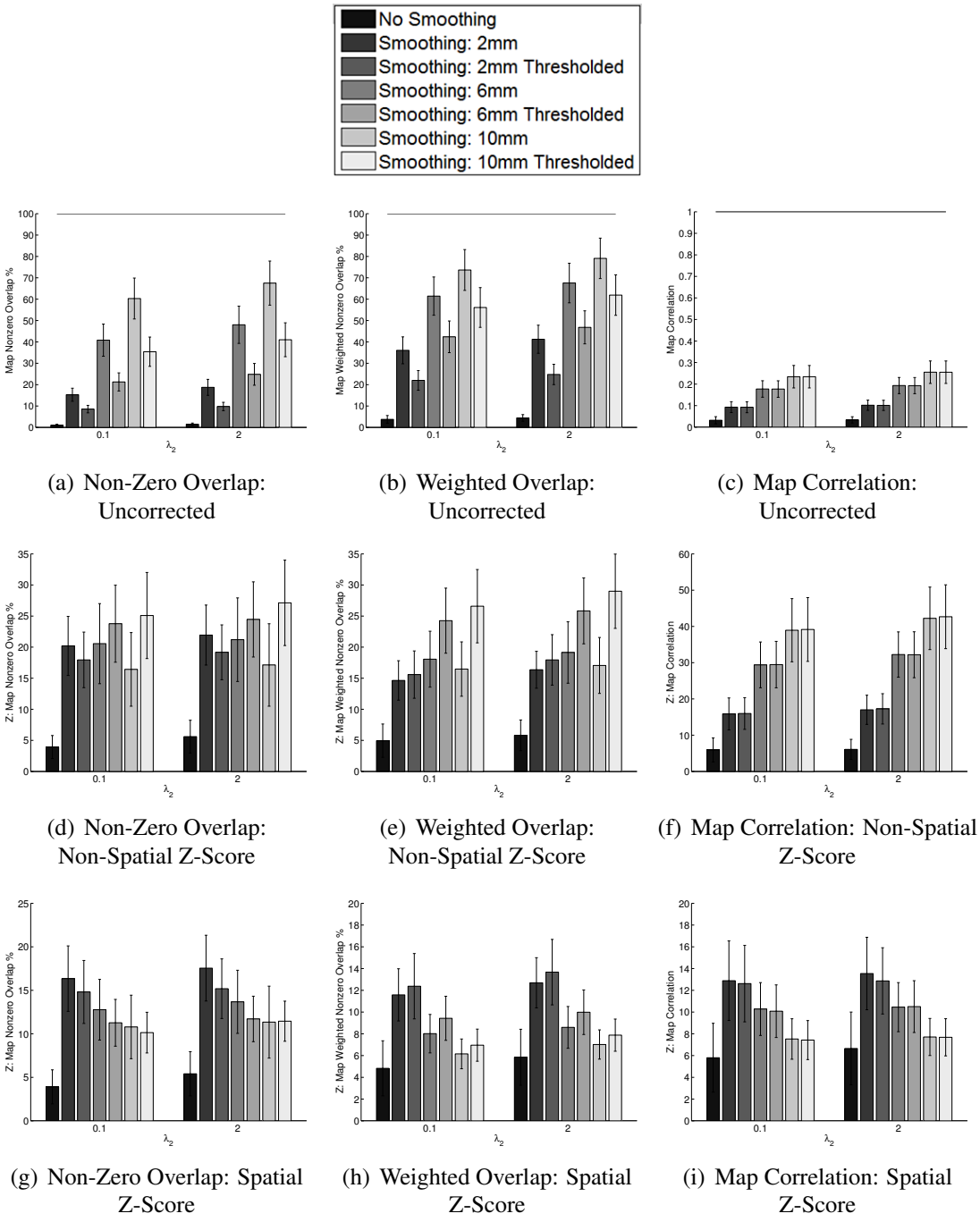
reliability, even when considering both null hypotheses. Even when applying the largest smoothing kernel, the reliability estimate is not significantly different than that of the raw map.

3.3.6 Spatial Structure and Increased Variance

Clearly, both null hypotheses affect the estimate of the reliability score distribution $\mathcal{D}$. In Figure 3.16, we reveal that increased spatial structure increases the *variance* of the estimated scores. We also confirm that decreased sparsity, captured by both null hypotheses, results, as expected, in a higher estimated mean for the overlap scores.

Figure 3.16(a) shows the effect of spatial structure and null hypothesis on the estimate of the map correlation metric. The mean is always estimated to be 0, regardless of spatial structure or H_0, and so only the variances are plotted. For the non-spatial H_0, the estimate variance is the same with or without much spatial structure in the map, and is in turn observed for the spatially-based H_0 if there is little spatial structure in the data (no smoothing). However, when the map exhibits strong spatial structure from smoothing, and the null hypothesis incorporates spatial information, the variance of the estimation skyrockets. While not shown, this finding generalizes across tasks as well.

Figure 3.16(b) shows that the effect observed for map correlation in Figure 3.16(a) is also observed for the weighted overlap metric, although the ceiling effect of the overlap metrics slightly obscures it. This figure shows the means of the expected weighted overlap score, with the standard deviation estimates plotted as error bars. Several phenomena are observed: (1) the estimated mean weighted overlap score, for a given amount of smoothing, is the same regardless of H_0; however, it increases dramatically when smoothing is applied. (2) As sparsity vanishes (with higher λ_2), the estimated mean is always very close to 100%, so the variance is quite low, creating a ceiling effect for variance. (3) Just as for the correlation metric, with the non-spatial null hypothesis, the variance of the estimation is identical regardless of whether there is spatial structure in the data; likewise, the variance estimated using the spatial null hypothesis is the same as for the non-spatial if the map lacks spatial structure. However, when there is spatial structure in the data, especially at lower sparsity levels, the estimated spatially-based variance is higher. Therefore, generating estimates when controlling for the specific value distribution provides the proper mean expected value, but further incorporating second-



(a) Non-Zero Overlap: Uncorrected

(b) Weighted Overlap: Uncorrected

(c) Map Correlation: Uncorrected

(d) Non-Zero Overlap: Non-Spatial Z-Score

(e) Weighted Overlap: Non-Spatial Z-Score

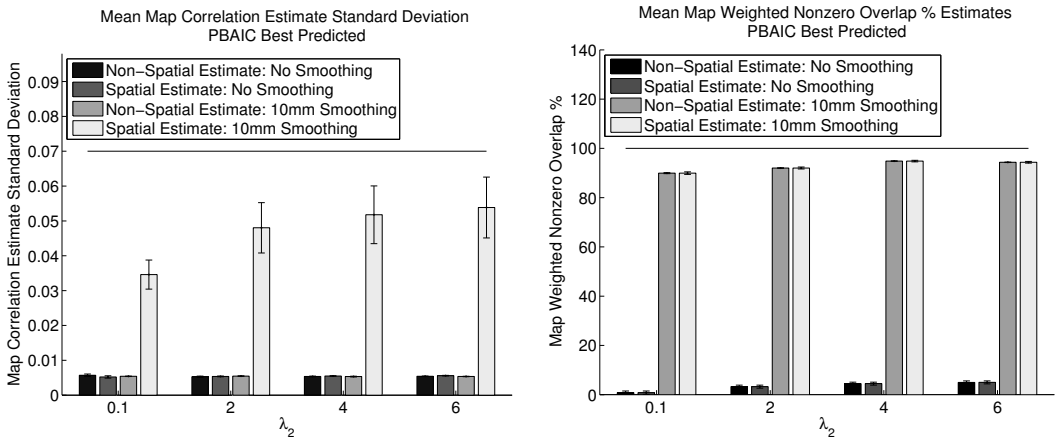
(f) Map Correlation: Non-Spatial Z-Score

(g) Non-Zero Overlap: Spatial Z-Score

(h) Weighted Overlap: Spatial Z-Score

(i) Map Correlation: Spatial Z-Score

Figure 3.15: *For the **Pain tasks**, z-scores underscore the impact of weight value distribution and spatial structure on all 3 reliability metrics, with any form of smoothing producing significantly more meaningfully reliable maps.* Uncorrected scores for the 3 metrics for the 3 Pain tasks, along with z-scores given the non-spatial and spatial null hypotheses. Means are calculated over both tasks, 14 subjects, and 4 cross-validation folds; variance over tasks and subjects. Bars reflect 95% confidence.



(a) Correlation Metric

(b) Weighted Overlap Metric

Figure 3.16: *The estimated variance of the score estimate is significantly higher when the map possesses spatial structure that the null hypothesis captures; estimate mean overlap is higher for less sparse maps regardless of null hypothesis.* (a) Mean estimated standard deviation of map correlation score, given smoothing and λ_2, averaged over all 3 subjects and 3 best-predicted PBAIC tasks. Error bars reflect 95% confidence. (b) Estimates for weighted overlap scores, given smoothing and λ_2. Bar heights reflect estimated mean score, averaged over all 3 subjects and 3 best-predicted PBAIC tasks and error bars reflect standard deviation of estimate, averaged over subjects and tasks.

order statistics, like spatial auto-correlation, leads to proper estimation of the variance, so the z-scores for the smoothed maps are properly penalized.

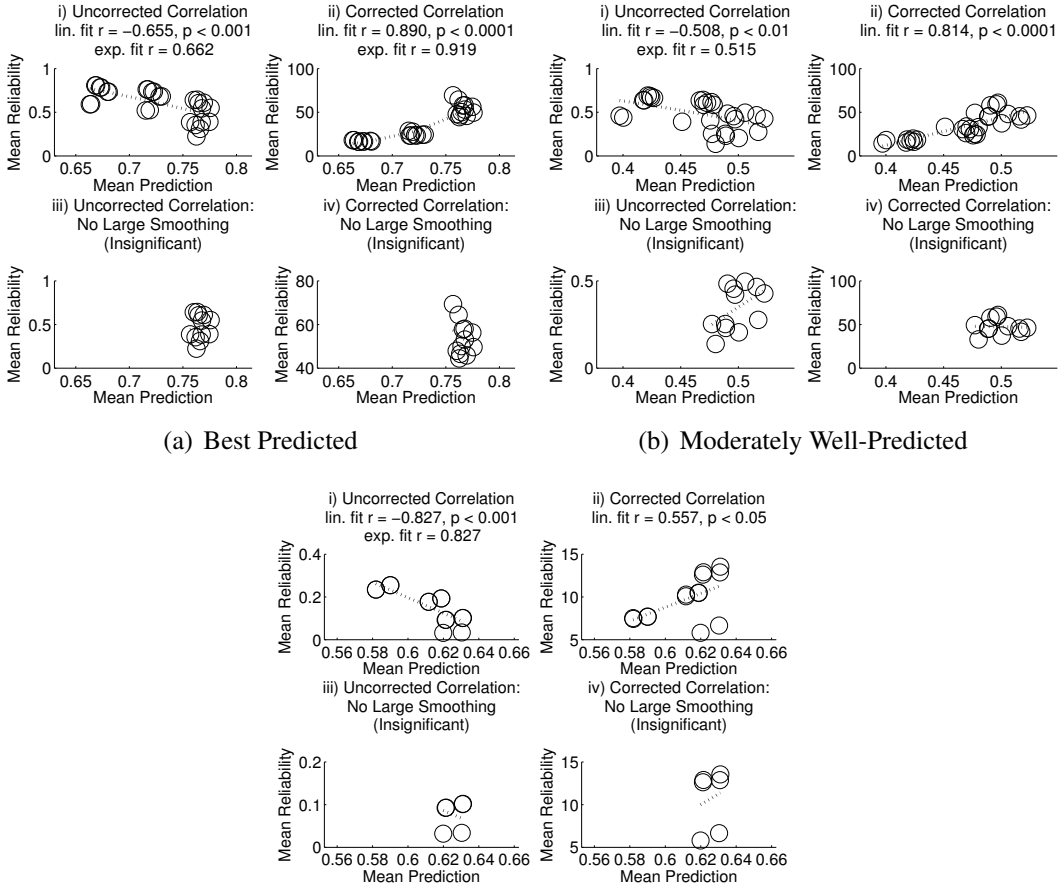
Note that these ceiling effect results reflect the high-mean, low-variance estimate situation described in Section 3.2.5. Despite the high mean of the estimated reliability scores for the smoothed maps, the z-scores for such maps still illustrate high significance, suggesting that even the smoothed maps are "meaningfully" reliable.

3.3.7 Relationship Between Reliability and Prediction

Figure 3.17 summarizes the effect observed in Figures 3.13, 3.14, and 3.15 by highlighting how the interpretation of the relationship between prediction and reliability can differ dramatically if the uncorrected map correlation score is used rather than the spatially-based z-score. In Figures 3.17(a)(i), (b)(i), and (c)(i), prediction and reliability appear to be directly negatively correlated, an effect observed by Strother et al. [86] and others, though in a slightly different context. However, when the corrected score is instead

considered in Figures (a)(ii), (b)(ii), and (c)(ii), the relationship appears exactly opposite. In fact, for the best-predicted PBAIC tasks, this relationship is exponential (given the best fit trend line), rather than linear, suggesting that as prediction is increased, the relationship between prediction and reliability is less direct. Subfigures iii-iv for each set of tasks further illustrate this independence by illustrating the disappearance of this effect when the most heavily smoothed, and hence least well predicted, maps are removed. From the prediction and reliability results, it is clear that most of the variance in prediction and reliability, observed in Figure 3.17(a)(i-ii), is due to manipulation of the smoothing parameters. When the maps that were smoothed by the large filters (6mm and 10mm) and their thresholded maps are omitted, in subfigures (iii-iv) for all tasks, the relationship between prediction and reliability is insignificant: the two properties vary independently, regardless of whether corrected or uncorrected correlation is considered.

Figure 3.18 further demonstrates this relationship between prediction and reliability by plotting points for each PBAIC subject and task by λ_2 for the non-smoothed maps. The trend is exponential: for subjects and/or tasks for which a well-predicting model is very difficult to obtain (prediction performance is within a low range), the maps will be completely unreliable. This finding is perfectly understandable, since poor prediction performance implies that the modeling procedure was unable to find any pattern reliable enough between the two runs to enable decent test prediction. Such difficulties may be due to the modeling procedure, but, given the consistently poor performance of many methods on certain PBAIC tasks as observed in the PBAIC [75] results, it is very likely that some aspect of the tasks themselves, or the subjects' data, make predictive modeling of the form most commonly used quite difficult. For tasks for which any reliable pattern can be detected, however, the overall pattern is somewhat linear: generally, better predicted models will, understandably, exhibit more reliability. Just as when examining the trend within parameterizations, though, the trend over subjects and tasks within a similar predictive range becomes much less direct. Just as variation in reliability can be observed among equally well-predicting models when varying λ_2, great variability in reliability is possible even among equally well-predicted subject-task combinations. Some such combinations lead to more reliable models than others that are equally well-predicted. For PBAIC, given the large number of tasks relative to subjects, most of this variability is among tasks, though there is some variability in overall prediction performance among subjects. For Pain, most of the variability is among subjects rather than tasks.

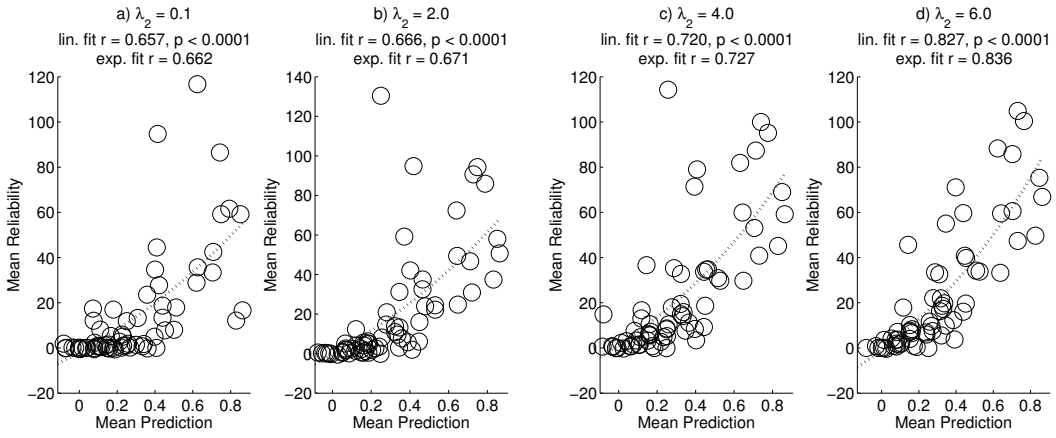


(a) Best Predicted

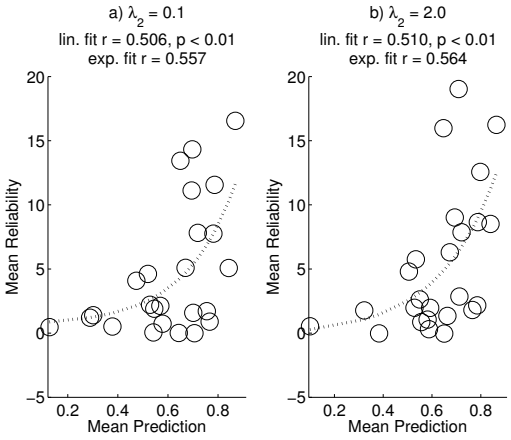
(b) Moderately Well-Predicted

(c) Pain

Figure 3.17: *If uncorrected map correlation is used, prediction and reliability appear negatively correlated across methods, but the opposite is true if corrected correlation is used; when large smoothing results are removed, reliability and prediction are independent.* For the three sets of PBAIC/Pain tasks, (a) and (b) and (c) respectively, prediction (test correlation) versus reliability measured as (i) and (iii) uncorrected map correlation or (ii) and (iv) spatially z-scored map correlation. For (i) and (ii) all smoothing levels are included; for (iii) and (iv), only no smoothing and 2mm smoothing with or without thresholding, are included. Data points correspond to each combination of smoothing and λ_2 parameter, with results averaged over all 3 (PBAIC) or 14 (Pain) subjects, 3 (PBAIC) or 2 (Pain) tasks, 2 runs, and 2 cross-validation folds. Trend lines are shown, along with model fit, if significant.



(a) PBAIC

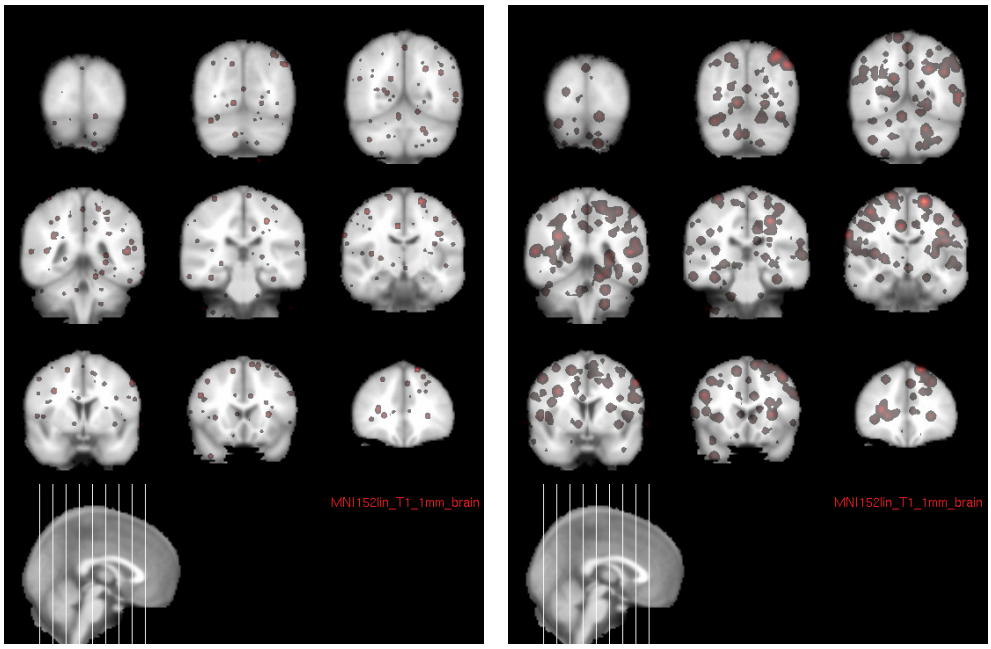


(b) Pain

Figure 3.18: *Reliability increases exponentially with prediction performance.* Prediction (test correlation) versus reliability (spatially-based z-scored map correlation) for all combinations of 3 (PBAIC) or 14 (Pain) subjects and 24 (PBAIC) or 2 (Pain) tasks with no smoothing, by λ_2 value (a(a-d) and b(a-b)). Trend lines are shown, along with model fit, if significant.

3.3.8 Brain Map Visualizations

Figures 3.19 and 3.20 provide visualizations of applications of smoothing, with or without thresholding, that resulted in improved reliability. Since smoothing and thresholding with a 2mm filter improved the meaningful reliability of maps for the PBAIC Hits task, Figure 3.19 displays two maps: the raw Hits map for one subject learned with a λ_2 value of 2.0, and the same map after the 2mm filter and 0.01 threshold have been applied.



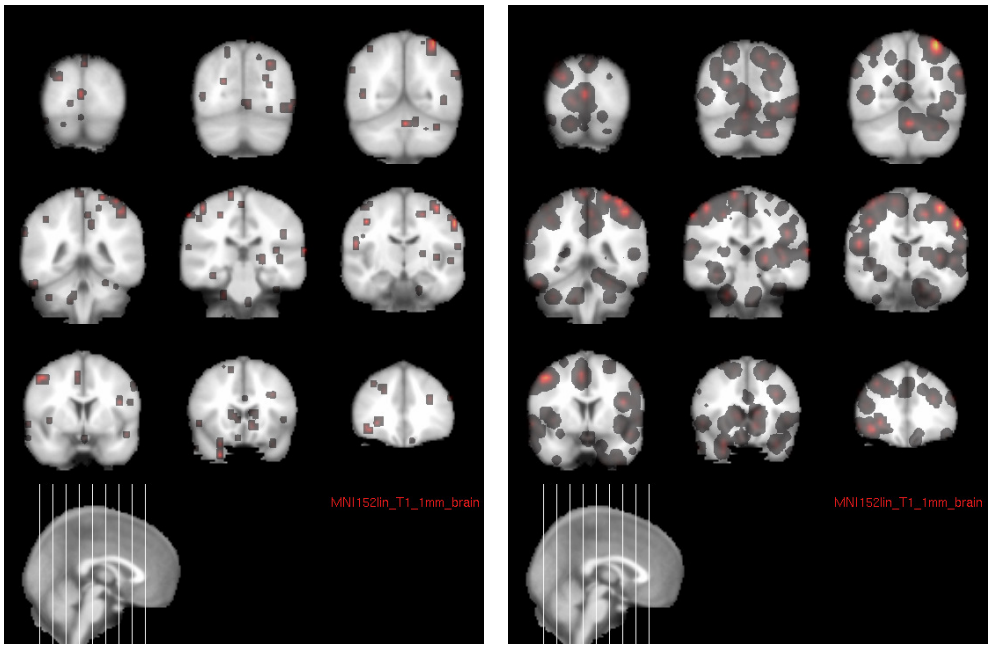
(a) No Smoothing (b) 2mm Smoothing and Thresholding

Figure 3.19: *Application of smoothing and thresholding to Hits map improved meaningful reliability.* (a) Map for PBAIC Hits task for Subject 1 Run 1 Fold 1 trained with $\lambda_2 = 2.0$; (b) same map after smoothing with a 2mm filter and applying 0.01 threshold.

Since smoothing, especially with a 2mm filter, improved the meaningful reliability of maps for the Pain tasks, Figure 3.20 displays two maps: the raw Pain Perception map for one subject learned with a λ_2 value of 2.0, and the same map after the 2mm filter has been applied. The difference in appearance is indeed dramatic, but the smoothed map is much more meaningfully reliable than the raw map.

3.4 Discussion

One of the most striking findings of this work is that a small amount of smoothing does not adversely affect prediction or reliability on these data. In a sense, smoothing, like regularization, has the effect of decreasing model variance across data subsets. Just as with regularization, though, this decreased variance usually comes at the expense of training accuracy because it results in increased model bias; however, regularization increases bias in a controlled manner, while smoothing does so blindly. One might



(a) No Smoothing (b) 2mm Smoothing

Figure 3.20: *Application of smoothing Pain Perception map improved meaningful reliability.* (a) Map for Pain Perception task for Subject 2 Run 1 Fold 1 trained with $\lambda_2 = 2.0$; (b) same map after smoothing with a 2mm filter.

suspect that regularization would therefore yield better prediction performance than blind smoothing. However, the findings in this chapter underscore the difference between training bias and generalization error. The small amounts of smoothing employed here did not impair model generalization; in fact, they sometimes slightly improved it.

In this work, we described a methodology for assessing the *significance* of a reliability estimate. The general trends observed highlight the importance of considering both the distribution of weight values and, crucially, spatial structure when evaluating significance. While this approach revealed general trends, it is intended as a proof of principle, and could be improved or extended in many ways. For instance, the parameterization of 20 surrogates per map, or 400 pairs per evaluation, was arrived at ad hoc, though it did result in very consistent estimates. This parameter choice could be studied in more detail. Also, the sharing of surrogates across the pair-wise evaluations could create some biases. Re-using surrogates is more computationally efficient, but if more accurate estimates are required, it would be best to avoid such practice.

In addition, one major limitation is that, for convenience, a normal distribution of metric scores was assumed. In practice, due to several factors, including the inter-dependence of the sampled scores due to surrogate re-use, this distribution is surely not Gaussian. An alternative pure resampling approach would estimate the p-value directly from the sampled values; however many more surrogate samples would be needed for such an approach, making it computationally infeasible. In addition, this approach would yield only p-values, which, in practice, often suffer from floor effects and lead to numerical complications. A more rigorous variant of the present approach would first use a goodness-of-fit test, e.g., Kolmogorov-Smirnov (K-S) [57], to determine an appropriate parametric form for the sample distribution and then estimate significance assuming the appropriate parametric form. Still, in practice the Gaussian assumption most likely approximates the true distribution, is computationally efficient to calculate, and yields an easily interpreted and compared z-score. Furthermore, if the results are in fact affected by this inaccurate assumption, it is most likely that the parametric approach is overly *conservative*. The fact that such clear trends were observed despite this overly stringent assumption suggests that the effects are indeed real.

The results we observe highlight general trends observed when incorporating spatial information into both modeling and reliability evaluation. The FFT-based approach we used is basic, intended as a proof of principle. As Figure 3.8 showed, this approach to surrogate generation retains some of the spatial information, but loses some of it to retain the actual value distribution. More sophisticated approaches might improve on this approach to retain more of the spatial information as well. In addition, depending on the modeling goals, one might even expand the null hypotheses. For instance, the null hypotheses we used were chosen because they are generic, that is, applicable to brain maps produced by any method, for instance a General Linear Model or human expert knowledge. However, if one wishes to be more specific and, say, focus solely on the reliability of models learned through sparse regression approaches, information about the learning algorithm might be incorporated into the null hypothesis, providing a more stringent significance estimation.

We also discussed how knowledge of the spatial properties of response can, and should, be incorporated when evaluating model reliability. Ultimately, measuring the reliability of a model involves measuring the similarity between two brain maps. We discussed three such similarity metrics (overlap, weighted overlap, and correlation). Finding

distance metrics is, however, very much an open and active research area in general, especially in fields such as Computer Graphics and Vision that deal with 3D images. One similarity metric that directly incorporates spatial information is Earth Movers Distance (EMD) [80], which essentially measures how much "mass," or voxel weight, must be moved to make two maps equal. The metric is independent of the significance estimation procedure and null hypotheses, so EMD could easily be used to evaluate reliability in place of these other metrics.

One caveat about the entire model interpretation approach in this chapter, Chapter 2, [103], and numerous related efforts, is that the magnitudes of model weight coefficients are highly dependent on properties of the predictors and the data transformations, including centering and scaling [36]. In this work, voxels are centered and standardized, so the model weight coefficients should be generally meaningful estimates of predictor importance, and expected to be consistent across experimental runs. However, when model reliability is poor, it can be useful to first consider whether the model coefficients themselves are meaningful and directly comparable across runs.

Both this chapter and Chapter 2 underscore that reliability can be manipulated independently of prediction performance. This previous work found no significant differences in prediction performance when regularization was used, and most efforts employing spatial regularization have not seen major prediction performance differences due solely to regularization. Most striking in the present chapter may be the finding that even a small amount of smoothing applied *after* model learning (as opposed to pre-processing smoothing of the data) does not significantly impair prediction performance, and in some cases slightly improves it. Thus, within a set of well-predicted tasks, drastically different maps are often equally reliable. As discussed, most of the variability in prediction we observed was due to the task (for PBAIC) or subject (for Pain), rather than any aspect of the modeling. This dominance of subject and/or task on prediction performance has been increasingly observed in the literature on predictive modeling for fMRI.

In contrast, we observe considerable variation in reliability across modeling approaches. There is some relationship between reliability and prediction performance, since the most easily predicted tasks tend to exhibit relatively high model reliability as well; however, even among these approaches, we do observe differences in reliability by λ_2 value and smoothing parameters. When factoring in reliability confounds for these tasks, increasing λ_2 alone is shown to produce the most significant increases in reliability, which is under-

standable: if these models are predicting well, they are doing best at finding reliable patterns in the data. Thus we would expect the weights to be more consistent across runs, and neither their prediction nor reliability is likely to be topped using a method that does not consider prediction. For the moderately well-predicted PBAIC tasks, however, smoothing followed by thresholding significantly improved reliability; for the Pain data, which were moderately well-predicted, any form of smoothing significantly improved reliability, with a small amount of smoothing, with or without thresholding, creating the most meaningful improvements. For these tasks, therefore, smoothing and thresholding present a viable alternative to much more computationally demanding approaches, such as training Elastic Net for more iterations, or applying forms of spatial regularization. All of these results suggest that, at the very least, this simple smoothing of maps should be used as a baseline for comparison when evaluating any method that performs direct or indirect spatial regularization or smearing.

Reliability should therefore be considered along with prediction as an important model criterion; however this work highlights a critical point: the method chosen to evaluate reliability can significantly affect the evaluation. The first important choice is the similarity metric used to compare two activation maps. Overlap measures offer intuitive interpretation. We also showed that non-zero overlap between maps is closely related to notions of sensitivity and specificity. Correlation, while lacking some of this interpretability, is a more general metric, applicable regardless of sparsity. We have also demonstrated, however, that all of these metrics are influenced by aspects of the map-generating process and therefore, regardless of chosen metric, there must be an effort to evaluate the significance of the observed similarity, given these map properties. Non-zero overlap is highly sensitive to the sparsity of the data, weighted overlap is sensitive to the contrast between weights, and all 3 metrics, especially correlation, are sensitive to the spatial structure of the data. This spatial sensitivity may not be immediately obvious, but is quite understandable, since spatial structure implies correlation; with higher covariance, there will be fewer degrees of freedom and hence higher variance. We have described a resampling approach that accounts for all 3 confounds, and is simple and general enough to warrant its use whenever the reliability of fMRI maps is evaluated. Note that, even when applying these stringent hypothesis tests, all of the predictive models exhibit highly significant reliability.

One slight confound of these results is that the Pain data did undergo a small amount of smoothing, on the order of $4\text{mm}^3 - 8\text{mm}^3$, prior to analysis, so the additional 2mm smoothing that worked well for these data may be a reflection of this imposed spatial property of the data. However, since smoothing is a standard pre-processing technique used to reduce noise and facilitate cross-subject analyses, our findings remain generally applicable, demonstrating that smoothing map weights is especially useful when image smoothing has been performed. In fact, this finding hints at a higher level perspective on smoothing and suggests future directions. The nearly universal application of image smoothing has its roots in the Matched Filter Theorem [65], which states that signal-to-noise (SNR) is maximized when the images are convolved with a filter that matches the generating process. In a sense, models learned from smoothed images reveal the extent to which a match at a particular location with a particular spatial filter is associated with the task being predicted; therefore, it is logical to correspondingly smooth the learned model weights. Data smoothing offers well-known advantages, such as noise reduction and facilitation of cross-subject analyses. While this work explored only within-subject analyses, a combined approach of image smoothing and model weight smoothing may result in more well-predicting and reliable analyses across subjects, as well as more general sets of tasks. As Penny et al. [71] have discussed, though, the choice of appropriate smoothing parameters is difficult, and likely varies by task, brain region, and subject. We will discuss an expanded approach that learns appropriate filter parameters given these properties in Chapter 4.

Chapter 4

Spatial Filter-Based Modeling

4.1 Introduction

In Chapter 2, we observed that predictive fMRI models exhibited distributed clusters of localized correlated activity. In Chapter 3, we observed that even a small amount of Gaussian spatial smoothing applied to learned beta weights can in some cases improve model reliability without impacting prediction performance. In addition, a primary assumption of the Statistical Parametric Mapping (SPM) approach to General Linear Model-based fMRI modeling [32] is that the activation pattern takes the form of Gaussian spatial "blobs," and multiple comparison adjustments for significance testing are performed under this assumption. Together, these properties and findings suggest that predictive fMRI models will be more successful if they can successfully locate distributed parametric spatial clusters, that is, detect the "blobs."

Spatial smoothing is now a standard, and essentially universal, pre-processing step in the fMRI pipeline. At the broadest level, the purpose of this pre-processing is to remove some of the large amounts of noisy artifacts from processes unrelated to neural functioning, such as interference in the physical signal itself. From a signal processing perspective, however, smoothing serves a specific purpose: according to the Matched Filter Theorem [65], if data had been generated via convolution with a spatial *filter*, for instance a spatial Gaussian function, the Signal-to-Noise Ratio (SNR) in the data can be maximized by re-convolving the data with the generating filter, i.e., the maximally correlated filter. This theory is quite useful when the data are generated from one filter

and the parameters are known or can be approximated. In essence, the assumption behind Gaussian smoothing with kernel size k is that the generating filter was a Gaussian with standard deviation k; by convolving the data with this filter, the SNR can be boosted.

For fMRI data, the approximate kernel size is generally understood to be in the range of $2\text{mm}^3 - 10\text{mm}^3$; however, the specific parameterization is unknown. In most cases, researchers simply choose one filter size found generally to produce desirable results. However, some researchers have found notable variations in results depending on the specific smoothing parameters used [86]. Unfortunately, the actual parameterization also likely depends on several variables, including the task being studied, the subject, and even the region of the brain being smoothed; in fact, it is not certain that the "true" parameterization is even Gaussian.

Despite this knowledge that the true smoothing parameters are variable, standard SPM assumes uniformly sized spatial kernels, the parameters of which must be determined separately from the modeling procedure. Ideally, these parameters would be estimated as part of the modeling step, and the model would adjust its significance estimations accordingly. In a Bayesian variation of SPM, Posterior Probability Maps (PPM) [33], the significance is estimated as the posterior probability of each voxel's activation exceeding a threshold given priors over the regression coefficients, which can be estimated from the data. Penny et al. [71] exploit this flexibility by incorporating both spatial and temporal information into the PPM priors. This approach requires assumptions about the spatial generating process to be made, such as the specific parameterization of the spatial kernel. Given the Random Field assumption underlying SPM, a logical candidate for these SPM-based models is a Gaussian Random Field (GRF) [2] kernel. Penny et al. thus explore "Gaussian blob" parameterizations as well as related Laplacian parameterizations, which penalize the differences between adjacent voxels. They further extend this work in [28] to handle wavelet basis filters, which can account for variation in kernel precision across regions. Wavelet bases also inherently perform shrinkage, in that most coefficients will be small and can be set to 0 using a prior. While this model enables optimization of parameters across subjects, task, and regions, its computational complexity increases when considering more than a few wavelet resolutions (precisions) and incorporating realistic assumptions such as shift- and translation-invariance. Also, this particular model does not take into account the correlation between the coefficients at different resolutions.

Predictive models, in which the behavior or cognitive response of interest is predicted from the fMRI data, offer the advantage of providing an inherent model validity measure. These techniques therefore provide a strategy for selecting optimal spatial perturbations to the data: choose those perturbations that maximize the prediction performance of the model. As discussed in Chapter 3, the most common technique for doing so is to apply a form of *spatial regularization* to the data, which has the effect of essentially smearing model weights in a manner so as to optimize prediction. Doing so often slightly improves prediction performance; however, while prediction optimization is an important goal, the spatial properties of models are also inherently interesting. From a modeling point of view, it would be useful to know something about the optimal spatial parameters so such parameters could be used without the need for computationally intensive spatially-informed predictive modeling. From a neuroscientific point of view, one would like to understand the spatial frequency of response and how it varies across subjects, tasks, and brain regions; however, unlike the Bayesian GLM models, the predictive models produced using spatial regularization can be difficult to *interpret* to explore these properties.

Gaussian Process Models (GPMs) [77] offer some promise for interpretation of the spatial pattern, by learning a multivariate Gaussian function parameterized by mean levels of activity at a theoretically infinite number of spatial locations, and the associated covariance matrix; this function is roughly a linear combination of univariate Gaussian "blobs." This covariance matrix can offer insight into the spatial frequency of response. However, even with such models, it is still difficult to determine the specific properties we seek about each blob: where exactly is the blob localized, and what is the extent of its spread? Furthermore, we ultimately seek to characterize a spatial pattern that is predictive of some response. Since, in a sense, we wish to predict the level of spatial activation from a combination of the response and the spatial location, we might consider the response to be a fourth spatial dimension, but doing so is not exactly the same as learning a model that predicts the response. In addition, GPMs are often computationally demanding to learn, and learning more complicated models would likely be especially burdensome.

Since we seek to make predictions from the data, we would of course like to maximize the SNR of the data. When employing smoothing, we assume some "true" signal exists and has been convolved with one spatial filter uniformly to produce the observed image. In our application, however, we know it is likely that the convolution filters are not uniform, and vary throughout the image. In addition, we suspect that they vary according

to external factors, including the subject, but also particularly the mental task being performed. Therefore, we seek to find the *spatial basis*, or set of specific filters at specific brain locations, in which lies the signal that is most *predictive* of some task, along with the coefficients of the signal itself.

As described in Chapter 2, sparse models can aid interpretability by producing more parsimonious models; however, sparse models trained with voxels as features do not yield insight into spatial properties. To build a predictive model that does facilitate analysis of spatial properties, one can draw inspiration from the Computer Vision literature. The Viola-Jones [93] algorithm is state-of-the-art for detecting faces in a 2D images. It uses spatial filters to produce an over-complete feature set and employs the AdaBoost classification algorithm [31] to select those spatial features that are most relevant to predicting whether the larger image is of a face or not. More specifically, the authors convolve the original image with a large set of spatial filters, in their case simple orthogonal Haar wavelets [17], and combine the mappings to create one very large new feature set used for training. The simple spatial features are easily computed yet enhance prediction performance. Furthermore, the modeling approach offers the advantage that the features selected can be examined, and in fact visualized, to provide clear insight into the spatial properties most associated with being a face, for example two dark boxes where the eyes would be.

This approach works well for faces, in which there is a great degree of localized spatial structure, and since the fMRI models we seek are characterized by distributed clusters of localized activity, we know that these models too exhibit a great deal of localized spatial structure. Indeed, it is these local structures we seek to discover and characterize. Therefore, it seems natural to apply the Viola-Jones approach to discover and understand those spatial structures that optimize prediction from fMRI data. In [14], we directly ported the Viola-Jones algorithm, using wavelets and AdaBoost, to predict the type of object being viewed by an fMRI subject. The results suggested that prediction performance may be slightly improved by exploiting the additional information, thus warranting further study, including the use of other machine learning algorithms. Given the results observed in previous chapters, sparse modeling is an obvious choice.

An approach of finding spatial features by training a sparse model from among a large set of spatially-derived features offers a particular advantage over the spatial regularization approaches and Bayesian methods described above. These approaches are

limited in that they require the assumption of one parametric form for the spatial features, for instance a Gaussian "blob" or a wavelet. To a sparse learning algorithm, however, the method used to generate the features is irrelevant, so the approach can be *generic*, enabling the setting of parameters for any spatial parametric form. For instance, the same method can be used to build Gaussian filter-based models for one task, but wavelets for a different task. Heterogeneous filter sets could even be used within the same model, i.e., one can combine features from different parametric forms finding, for instance, that Gaussian filters best characterize the structure in one region, but wavelets in another.

The approach of "blowing up" the image into a massively over-complete representation to be used as a feature set in sparse modeling is thus warranted; however, this approach presents a major computational challenge. The Viola-Jones algorithm was designed for use on small windows of images, with approximately 400 pixels. In fMRI experiments, however, the number of brain voxels is typically on the order of $30,000$ voxels. As we will quantify, when this feature set is expanded by convolving all voxels with a large set of candidate spatial filters, the resulting training dataset consumes typical maximal memory resources, prohibiting the use of most learning algorithms. Thus, training with a full set of candidate features seems infeasible; however, modern High-Performance Computing (HPC) tools are specifically designed to address problems of scale such as this one. HPC devices allow parallel processing across hundreds or thousands of nodes, enabling brute force approaches for problems that previously required computationally intensive algorithms and/or sacrifices and approximations. We can exploit HPC technology to distribute the problem of learning sparse models from enormous feature sets across nodes, making training from over-complete representations feasible.

We therefore describe an approach for generating feature sets from spatial features in a completely generic manner, and address the computational challenge imposed by the resulting very large feature sets by training a sparse model using a distributed version of the LARS-EN regression algorithm [103]. This approach provides a test bed for optimizing the spatial parameters of predictive models and interpreting the models to gain insight into the spatial structure of fMRI response. Our preliminary exploration also reveals challenges to be tackled when employing such approaches. In particular, we find that in the over-complete representation, correlation among filters of different scales at the same location confounds training with existing sparse methods, suggesting

the need for an algorithm that explicitly constrains the model search space. Likely related to this confound, we find that the use of Gaussian spatial filters on real fMRI data does not provide an advantage in terms of prediction or reliability over models learned with standard voxel-only features with or without smoothing, but the interpretability of the approach allows the revelation of a correlation among prediction performance, model size, and spatial structure that re-affirms the characterization of predictive fMRI response as distributed clusters of localized activity. Thus the test bed reveals insight into both appropriate feature types and the broader properties of the spatial response.

4.2 Methods

4.2.1 Definitions

To specify the procedure for generating features, learning from them, and generating synthetic data, we first define key notation and terms. We assume a dataset of fMRI images $\mathbf{X}$ with N time points (TRs) and M voxels corresponding to actual brain locations; i.e., if the 3D image space is of size $64 \times 64 \times 34$, $M \sim 30,000$. Call the set of these voxels $\mathcal{M}$ such that $|\mathcal{M}| = M$. The terms are as follows:

Filter: A generic template signal $\mathbf{T} \in \mathbb{R}^{a \times b \times c}$, where a, b, and c are arbitrary, which parameterizes a spatial pattern we seek to detect in the real data. The vector form of $\mathbf{T}$ is the coefficient vector $\pi_{\mathbf{T}} \in \mathbb{R}^{a*b*c}$. We seek linear filters with which to correlate the data to maximize the Signal-to-Noise ratio. Filters can take on any parametric form, but we focus on Gaussian filters for now. We will refer to the set of candidate filters we consider as T.

Gaussian spatial filter: A template signal parameterized as a Gaussian distribution over spatial dimensions, with a spread corresponding to the standard deviation of the "spread" of the Gaussian. For instance, Figure 4.1 shows a 2D slice of a filter corresponding to a Gaussian "blob" in space with a diagonal covariance matrix $\Sigma = I$ voxels, where I is the identity matrix. The filter has been $l_\infty-$normalized to have maximum value 1.0 and here is discretized here over a 2D pixel grid of size 7×7. The full spatial extent of a Gaussian is of course infinite, so we have applied a threshold, making the extent of the filter as large as necessary to contain all values of 0.001 or greater; we apply

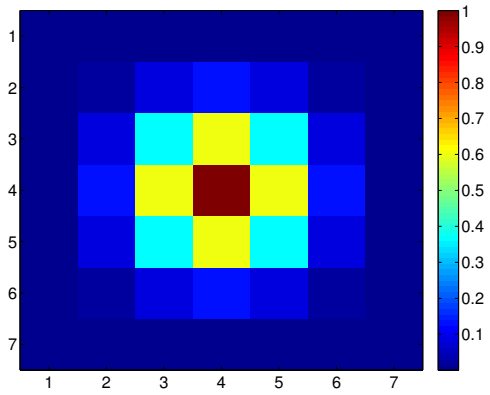


Figure 4.1: An example 2D Gaussian filter with an identity covariance.

this normalization and thresholding to all Gaussian filters that we consider. As a result, this filter has $|\pi_{\mathbf{T}}| = 7 \times 7 = 49$ coefficients, and the corresponding 3D filter has $|\pi_{\mathbf{T}}| = 7 \times 7 \times 7 = 343$ coefficients. Vector $\mathcal{D}_t \in \mathbb{R}^3$ stores the coordinates of the spatial extent of filter $\mathbf{T}$, $\{7,7,7\}$ in this case. The *size* of the filter is the corresponding scalar $|\pi_{\mathbf{T}}|$. In this work, we will consider only symmetric Gaussians (with a diagonal covariance matrix), so each filter will be parameterized by a vector $\sigma \in \mathbb{R}^3$ with 3 parameters corresponding to the non-zero elements of the diagonal covariance matrix.

Field: A filter $\mathbf{T}$ that has been positioned at a given spatial location (voxel in this case), which we will refer to as μ, implying the set of absolute voxel locations $\mathbf{L}_{\mathbf{T}\mu}$ that lie within the field. Since we are considering only symmetric filters, $\mathbf{L}_{\mathbf{T}\mu} = \{\mu_1 - \lfloor \frac{\mathcal{D}_{t,1}}{2} \rfloor, \ldots, \mu_1 + \lfloor \frac{\mathcal{D}_{t,1}}{2} \rfloor\}, \{\mu_2 - \lfloor \frac{\mathcal{D}_{t,2}}{2} \rfloor, \ldots, \mu_2 + \lfloor \frac{\mathcal{D}_{t,2}}{2} \rfloor\}, \{\mu_3 - \lfloor \frac{\mathcal{D}_{t,3}}{2} \rfloor, \ldots, \mu_3 + \lfloor \frac{\mathcal{D}_{t,3}}{2} \rfloor\}$.

We will refer to the set of fields corresponding to a specific filter $\mathbf{T}$ as $\mathbf{F}_{\mathbf{T}}$. Theoretically, $|\mathbf{F}_{\mathbf{T}}| = M$. In practice, however, some fields for a given filter $\mathbf{T}$ will lie mostly outside of valid brain regions and will not be computed, so $|\mathbf{F}_{\mathbf{T}}| \leq M$, but $|\mathbf{F}_{\mathbf{T}}| \approx M$. We will refer to a field as *valid* if it can be computed, and to the full set of valid fields corresponding to all filters T as $\mathcal{F}$. Let F be the total number of fields over all filters T. Thus $F = |\mathcal{F}| = \sum_{\mathbf{T}} |\mathbf{F}_{\mathbf{T}}| \leq M|T|$.

Gaussian field: A field corresponding to a Gaussian filter, in which the field location, μ, corresponds to the Gaussian mean. The spread of the field is determined by the parameters of the corresponding filter.

Convolution Matrix: A convolution matrix $\mathbf{C}_T \in \mathbb{R}^{M \times |\mathbf{F}_{\mathbf{T}}|}$ s.t. for a filter $\mathbf{T}$, multiplying a vectorized image by $\mathbf{C}_T$ yields the convolution of $\mathbf{T}$ with the image. More

specifically, let $G \in \mathbb{R}^{X \times Y \times Z}$ be a 3-D image with b voxels total. Let $\mathbf{I} \in \mathbb{R}^b$ be the vectorized form of $\mathbf{G}$. $\mathbf{C}_T$ is a *convolution matrix* for filter $\mathbf{T}$ if $\mathbf{T} \otimes \mathbf{G} = \mathbf{I}\mathbf{C}_T$ or, equivalently, since convolution commutes, $\mathbf{G} \otimes \mathbf{T} = \mathbf{I}\mathbf{C}_T$.

For a given filter $\mathbf{T}$, we can define a convolution matrix $\mathbf{C}_T$ for the M valid voxels as follows:

1. Initialize $\mathbf{C}_T \in \mathbb{R}^{M \times M}$ to be all 0s. Refer to a column j of this matrix as $\mathbf{C}_{T_j}$ and an element at row i and column j as $\mathbf{C}_{T(i,j)}$.
2. Choose some threshold θ indicating the minimal proportion of locations in $\mathbf{L}_{\mathbf{T}\mu}$ that must lie within the brain, i.e., be within $\mathcal{M}$ for a field to be considered "valid." We set $\theta = 50\%$.
3. For each voxel location $\mu \in \mathcal{M}$:
   (a) If the number of locations that are in both the field ($\mathbf{L}_{\mathbf{T}\mu}$) and the brain ($\mathcal{M}$) does not exceed $\theta|\pi_{\mathbf{T}}|$, the field is not valid, so delete column $\mathbf{C}_{T_\mu}$. Otherwise, proceed.
   (b) Assign the coefficients $\pi_{\mathbf{T}}$ to the corresponding locations $\mathbf{L}_{\mathbf{T}\mu}$ in $\mathbf{C}_{T_\mu}$, i.e., $\mathbf{C}_{T(\mathbf{L}_{\mathbf{T}\mu},\mu)} = \pi_{\mathbf{T}}$.

If $\mathcal{D}_T = \{1,1,1\}$, and therefore $\pi_{\mathbf{T}}$ has only one coefficient equal to 1, $\mathbf{C}_T$ will be an $M \times M$ identity matrix.

Spatial basis: A spatial basis matrix $\mathbf{U} \in \mathbb{R}^{M \times F}$, in which the true signal in the data is postulated to lie, is the column-wise concatenation of all convolution matrices $\mathbf{C}_T, \mathbf{T} \in T$, with one column for each valid field in each filter. Each row vector indicates the coefficient value at that location for every field.

Mapped feature set: Once the basis set $\mathbf{U}$ is defined, it can be used to convolve an image $\mathbf{G}$ with the full set of candidate filters T by performing $\mathbf{I}\mathbf{U}$ using the vectorized form $\mathbf{I}$ of G, resulting in a vector of coefficients $\mathbf{p} \in \mathbb{R}^F$. Each coefficient in $\mathbf{p}$ encodes the strength of the match between the corresponding field and the image $\mathbf{G}$. In a sense, each coefficient is therefore a *feature* of that image and will be used as a *predictor* for our model learning. Thus, we can call $\mathbf{p}$ a *mapped feature set*.

Furthermore, let $\mathbf{z}$ be a subset of filters, i.e., $\mathbf{z} \subseteq T$. We can define a subset $\mathbf{p}_\mathbf{z} \subseteq \mathbf{p}$ as the elements of $\mathbf{p}$ that are coefficients of fields in $\mathbf{z}$. This feature set is produced by

convolving only with the filter subset $\mathbf{z}$. If $\mathcal{F}_\mathbf{z}$ are those fields corresponding to filters $\mathbf{z}$, let $\mathbf{U}_\mathbf{z}$ be shorthand for $\mathbf{U}_{\mathcal{F}_\mathbf{z}}$. Then $\mathbf{p}_\mathbf{z} = \mathbf{I}\mathbf{U}_\mathbf{z}$. This terminology will be useful when we distribute predictors across nodes.

Mapped dataset: If $\mathbf{X}$ is the original dataset, define $\mathbf{X}^{'} \in \mathbb{R}^{N \times F}$ as the *mapped dataset*, where each row $\mathbf{x}_n^{'}$ corresponds to the mapped feature set for corresponding image (row) $\mathbf{x}_n$. Thus $\mathbf{X}^{'} = \mathbf{X}\mathbf{U}$. Likewise, $\mathbf{X}_\mathbf{z}^{'} = \mathbf{X}\mathbf{U}_\mathbf{z}$.

Mapping function: A vector $\mathbf{v} \in \mathbb{R}^F$ of coefficients corresponding to a signal in the spatial basis $\mathbf{U}$ that is used to map from $\mathbf{U}$ *back* to a **map** (see below) in real 3D (brain) space, with each coefficient value representing the strength of the corresponding field in the signal. This function is assumed to be the "true" signal and the real image data the result of this true signal having been convolved with a spatial basis. This function is what we seek to learn, but it can also be known in advance, for instance if the data are synthetic. When describing known mapping functions, we will use the term **template function** and when describing the function we are seeking to learn, we will use the term **learned mapping function** and refer to the individual elements as **beta weights**, since they correspond to weights learned in a linear model. In this application, we assume a sparse learned mapping function, i.e., we assume that most values in $\mathbf{v}$ will be 0.

Recall that we seek to know the true spatial basis in which the predictive signal lies, along with the predictive signal itself. In the present work, we will accomplish these goals by learning a *sparse* predictive mapping function in the *full* spatial basis corresponding to *all* candidate filters. We will also define the "true" spatial basis $\hat{\mathbf{U}}$ as the sub-matrix of $\mathbf{U}$ in which the columns correspond to the predictors with non-zero weights in the learned mapping function $\mathbf{v}$. Likewise, let the set of fields corresponding to these non-zero weights be denoted $\hat{\mathcal{F}}$.

Map: The vector $\mathbf{w} \in \mathbb{R}^M$ produced by mapping from the spatial basis back into voxel space via the mapping function, i.e., $\mathbf{w} = \mathbf{U}\mathbf{v}$. Furthermore, define **Brain map** as the 3-dimensional array produced by re-shaping vector $\mathbf{w}$ into the spatial coordinates of the brain. For conciseness, we often use the shorthand *map* to refer implicitly to a brain map, but the precise denotation should be clear through context.

4.2.2 Problem Formulation

As in the previous chapters, we seek a map, which serves as a linear model of activity over voxels, from which we can predict $\mathbf{y}$ and evaluate model properties, such as reliability. In this work, however, we assume that this map is the result of having convolved some true signal $\mathbf{v}$ with a spatial basis $\mathbf{U}$. Therefore, instead of the map $\mathbf{w}$ of β values as in the previous chapters, we seek to learn $\mathbf{w} = \mathbf{U}\mathbf{v}$. In this work, we fix $\mathbf{U}$ and seek to learn $\mathbf{v}$, with the assumption that learning a model in brain space is more complex because it must capture the spatial convolution as well. We fix the spatial basis and seek to learn a more *parsimonious* representation in that space, with the hope that doing so will yield:

1. better prediction by reducing degrees of freedom for noise.
2. reliability due to the simpler model.
3. interpretation by directly indicating the (small) set of relevant *locations* and *spatial filter parameterizations* at those locations, i.e., the spatial properties of the data.

We would learn $\mathbf{w}$ using the standard fMRI regression formulation for least-squares, but, since $\mathbf{U}$ is fixed, we can instead learn $\mathbf{v}$ as follows:

$$\mathbf{w} = \text{argmin}\,||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 \tag{4.1}$$

$$\mathbf{v} = \text{argmin}\,||\mathbf{y} - \mathbf{X}\mathbf{U}\mathbf{v}||_2^2 \tag{4.2}$$

$$\mathbf{v} = \text{argmin}\,||\mathbf{y} - (\mathbf{X}\mathbf{U})\,\mathbf{v}||_2^2 \tag{4.3}$$

$$\mathbf{v} = \text{argmin}\,||\mathbf{y} - \mathbf{X}^{'}\mathbf{v}||_2^2 \tag{4.4}$$

In other words, we seek to solve the standard regression formulation, but using the *mapped* dataset $\mathbf{X}^{'}$ instead of the original dataset $\mathbf{X}$. Therefore, we can employ the same set of regression tools we used in previous chapters. To do so, consider the vector $\mathbf{v}$ we are seeking to learn. $\mathbf{v}$ is of length F, i.e., there is one coefficient for every possible field. We therefore make two assumptions about $\mathbf{v}$, and the predictor set $\mathbf{p}$:

1. It is sparse, since the premise of this work is to find the small subset of true fields.
2. True fields that are independent may still yield correlated predictors in $\mathbf{p}$ if the fields are spatially proximal.

Therefore, we seek to find a sparse representation with the potential to select correlated predictors, so Elastic Net is a logical choice, and we seek to solve:

$$\mathbf{v} = \text{argmin}\,||\mathbf{y} - \mathbf{X}'\mathbf{v}||_2^2 + \lambda_1||\mathbf{v}||_1 + \lambda_2||\mathbf{v}||_2^2. \tag{4.5}$$

4.2.3 Distributed Modeling Procedure

We can simply run the LARS-EN Elastic Net training algorithm [103, 24] on the problem specified in Eq. (4.5) and obtain $\mathbf{v}$. However, there is a problem: consider the number of predictors, corresponding to the number of columns in $\mathbf{X}'$. $|\mathbf{p}| = F$, i.e., the number of fields. We know that $F \approx M|T|$. We know that $M \approx 30,000$, but let us now consider $|T|$, which is the number of candidate filters we will consider. In this work, we begin our exploration by considering simple symmetric Gaussian filters, with $\sigma = 3$ for all filters; in other words, each filter has 3 variable parameters, corresponding to the spatial spread of the filter. It is certainly conceivable that for fMRI we may want to consider all sizes from 2mm to 10mm, and we will want to consider every possible value for each of the 3 coefficients. Even assuming a coarse discretization of 1mm will lead to $9 \times 9 \times 9 = 729$ candidate filters. In that case, $F \approx M|T| = 21,870,000$ predictor fields. Assuming $N \sim 500$, $\mathbf{X}'$ will therefore be of size $\sim 500\times \sim 22,000,000$, requiring roughly 90GB of RAM to learn a model from one dataset. While RAM capacity is constantly increasing, so acquiring such resources on one machine may be theoretically feasible, running many experiments would become quite burdensome.

Furthermore, this set of parameter values is relatively small; first consider we may wish to examine much finer discretizations over a much larger range, and potentially learn all 9 parameters in a full covariance matrix. Then consider that we may wish to consider *heterogeneous* datasets in which we add other types of filters, such as wavelets, and seek to learn the parametric form as well as the parameters. For wavelets, we must consider various types, sizes, and shapes. Clearly the feature space can explode quite dramatically.

Fortunately, this problem can be distributed across nodes to reduce the memory requirements at any one node, and potentially speed up computation. In Chapter 5, we will describe a parallel implementation of LARS-EN, Parallel LARS-EN, which allows one to distribute across nodes datasets characterized by very large feature sets. In this section,

we detail a distributed procedure we follow to create the dataset $\mathbf{X}'$ to be processed by Parallel LARS-EN, and a distributed procedure we follow to map back from the learned vector $\mathbf{v}$ to the map $\mathbf{w}$ we ultimately wish to obtain.

Generation of the Mapped Feature Dataset

Recall that $\mathbf{X}' = \mathbf{XU}$. Therefore, computation of $\mathbf{X}'$ factorizes across the columns of $\mathbf{U}$. Each column in $\mathbf{U}$ corresponds to a field associated with a particular candidate filter $\mathbf{T}$. Thus, one can compute the sub-matrix $\mathbf{U_z}$ from some subset of candidate filters $\mathbf{z} \in T$, and in turn use it to produce a subset $\mathbf{X}'_\mathbf{z}$ of $\mathbf{X}'$. Therefore, the data generation procedure is the following:

For each node z:

1. Load the original data $\mathbf{X}$.
2. Determine the subset of one or more filters $T_\mathbf{z} \in T$ that the node has been assigned, perhaps by node rank, or all filters if running on a single node. Load pre-computed representations of these filters from some source, such as generic text files.
3. Compute $\mathbf{U_z}$ by following the Convolution Matrix computation procedure in Section 4.2.1 using the filter subset $\mathbf{z}$.
4. Compute $\mathbf{X}'_\mathbf{z} = \mathbf{XU_z}$.

Parallel LARS-EN is then run on the distributed dataset $\mathbf{X}'$.

Figure 4.2 visualizes this process. Assuming one filter per node, $\mathbf{X}'_\mathbf{z}$ will occupy approximately the same amount of space as $\mathbf{X}$, so the mapped data should easily fit in memory.

By virtue of how $\mathbf{U}$ is calculated, this procedure:

1. is generic with regard to the parametric form of the filter. The filter is represented simply by a set of coefficients corresponding to relative 3D locations, so the procedure works equally well for Gaussian filters, wavelets, or any other filter type.
2. exploits the sparsity of the brain data, since the procedure creates columns in $\mathbf{U}$ only for "valid" fields, reducing the computation required.

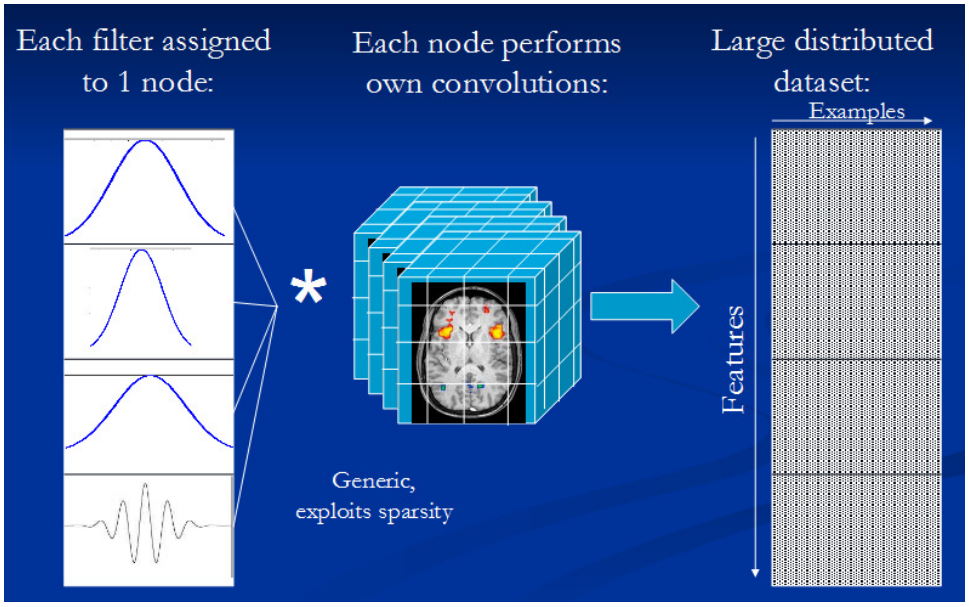


Figure 4.2: *The mapping for each filter is performed separately to produce a subset of a large, distributed feature set.*

Note that this procedure can handle a potentially very large set of candidate filters, even if each node is assigned only 1 filter. One popular modern supercomputing platform, the IBM Blue Gene/L, typically features $1000-16,000$ nodes per machine, and, in some cases, up to nearly $300,000$ nodes. Depending on the dataset size, as the amount of RAM per node increases on such machines, it will be feasible to assign dozens of filters to each node, facilitating a comprehensive search of the candidate filter space.

Predictive Map Generation

The returned result from Parallel LARS-EN is the matrix $\beta \in \mathbb{R}^{F \times i}$ where i is the number of training iterations for which the algorithm was run. For a given iteration i, the column vector β_i is the mapping function learned at i. Therefore, the map $\mathbf{w}_i$ can be computed as $\mathbf{w}_i = \mathbf{U}\beta_i$ and the full matrix $\mathbf{W} = \mathbf{U}\beta$. However, recall that $\mathbf{U}$ itself is distributed, with each node z computing $\mathbf{U_z}$. Therefore, this process must be distributed as well. Note that:

$$\mathbf{W} = \mathbf{U}\beta \tag{4.6}$$

$$\mathbf{W} = \sum_{z} \mathbf{U_z}\beta_\mathbf{z} \tag{4.7}$$

Therefore, we can distribute the computation of $\mathbf{w}$ as follows:

For each node z:

1. Compute a subset of maps $\mathbf{W}_z = \mathbf{U_z}\beta_\mathbf{z}$
2. Send $\mathbf{W}_z$ to Master

Master node computes $\mathbf{W} = \sum \mathbf{W}_z$.

In practice, we can again exploit the sparsity of both β and $\mathbf{U}$ to improve both memory and time efficiency by only computing non-zero values, although the steps are slightly more complicated (not shown). Eq. (4.6) makes clear the fact that, for a given location L, there will be multiple filters $\tilde{T}$ such that $L \in \mathbf{L}_{\tilde{\mathbf{T}}_\mu}$, and thus the location is within the range of multiple fields. Therefore, the value at each voxel represents the sum of the contributions of each field in which it lies. Figure 4.3 visualizes this process, in which β indicates the strength of specific filters at specific locations (fields), most of which are of weight 0, or "un-selected", and the map $\mathbf{w}$ is a linear sum of these selected fields weighted by β.

It is important to note that a map $\mathbf{w}$ is just like any other weighted brain map. It can be used directly to make predictions; that is, for a new dataset χ, the predicted target vector can be computed as $\mathbf{y}_\chi = \chi\mathbf{w}$ *without needing to perform any filter convolutions*. In other words, once we have $\mathbf{w}$, we never need to access the set T again. Furthermore, the properties of this map, such as reliability, spatial distribution, etc., can be evaluated just as they would for any other map, and the map can be directly compared to maps learned by other means, for instance Elastic Net performed without spatial filters. We perform all such evaluations in the sections that follow.

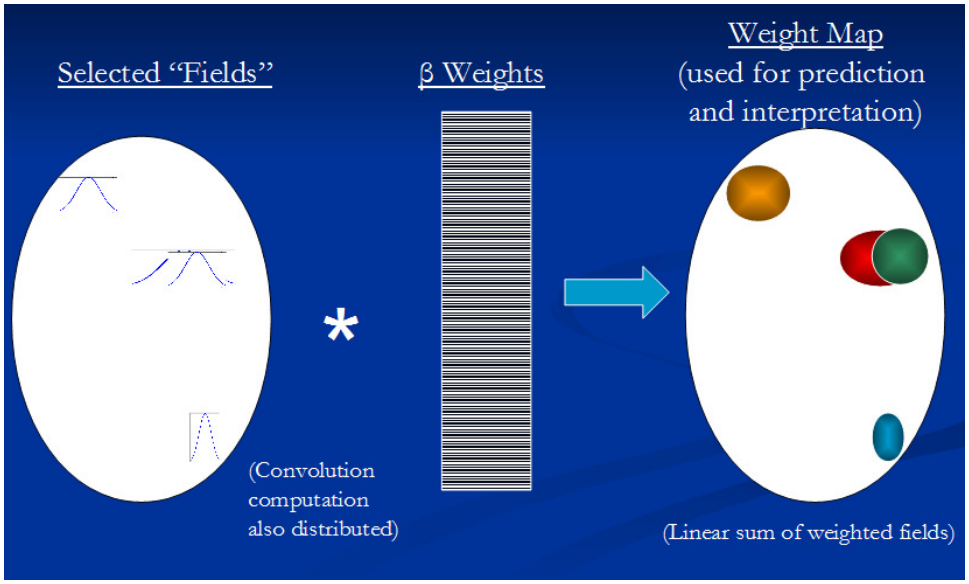


Figure 4.3: *Each node performs a convolution from field space into brain space for its set of filters, and the maps are added together to produce a final map that can be evaluated.*

4.2.4 Data

PBAIC 2007

Experiments were run on the PBAIC Competition data described in Section 2.2.3.

Pain

Experiments were run on the Pain data described in Section 3.2.2.

Synthetic Data

Ideally, our method will be able to recover a mapping function v that correlates strongly enough with the pure data-generating signal as to maximize the Signal-to-Noise ratio. Since the true signal is unknown for real fMRI data, a synthetic dataset was generated using the assumptions of the methodology described herein. Specifically, the data were assumed to lie in a "brain" space of 1200 one-dimensional "voxels" and were assumed to be generated from a basis set consisting of all one-dimensional Gaussian fields of standard deviation $1.0, 2.0$ or 3.0. The data are assumed to be either drawn from a modeled class

(positive examples) or random noise (negative examples), where the modeled class is generated from a template function that maps from the basis set to brain space of size $M = 1200$. In theory, it is possible for the template function to have non-zero values for multiple fields (of different sizes) at the same location, in effect corresponding to one larger Gaussian field at the location. Since we are interested in the ability of the algorithm to detect a true signal in which the fields are assumed to be of size between 1 and 3 standard deviations, we choose to assume at most one "active" (non-zero) field per location. This assumption will allow the revelation of a complication in the use of this algorithm, described in section 4.3.1. With that constraint in mind, the data are generated using the following procedure that relies on the previous definitions; parameter values used for the specific dataset used for evaluation are indicated in parentheses:

1. Determine the dimensionality of the "brain" space ($1200 \times 1 \times 1$) and hence M ($M = 1200$).
2. Determine the set of generating filters $\tilde{T}_s$ (Gaussians with standard deviation $\{1.0, 2.0, 3.0\}$).
3. Create the generating spatial basis $\tilde{\mathbf{U}}_s$ from $\tilde{T}_s$.
4. Determine the number of datasets in the set $\tilde{\mathbf{D}}$ to be generated ($|\tilde{\mathbf{D}}| = 10$).
5. Choose a number of examples N per label per dataset ($N = 110$).
6. Generate a "template beta" $\tilde{\mathbf{v}}$ as follows:
   (a) Choose a set of positions $\tilde{\mathbf{p}}$ at which a blob will be present uniformly at random ($|\tilde{\mathbf{p}}| = 30$ out of $M = 1200$ candidate positions).
   (b) For each position $p \in \tilde{\mathbf{p}}$:
      i. Choose one of the generating filters g_p from the set T uniformly.
      ii. Add the field corresponding to position p and filter g_p to the set of template fields $\tilde{\mathcal{F}}$.
   (c) For each field $f \in \tilde{\mathcal{F}}$, choose a template beta weight $\tilde{\mathbf{v}}_f \sim norm(0, sb)$ ($sb = 1.0$).
7. For each dataset $\mathbf{D} \in \tilde{\mathbf{D}}$, generate N positive examples and N negative examples as follows:
   (a) For each example $n \in \mathbf{D}$ with label $+1$:

i. Create a generating function $\mathbf{v}_n \in \mathbb{R}^F$ of all zeros.

ii. Choose a probability of activation pr uniformly from some range ($0.3 < pr < 0.7$).

iii. For each field $f \in \tilde{\mathcal{F}}$: select f to be active with probability pr.

A. If f is inactive, set $\mathbf{v}_{n_f} = 0$.

B. If f is active, set $\mathbf{v}_{n_f} = \tilde{\mathbf{v}_f}$ and add noise $nf \sim norm(0, sf)$ s.t. $\mathbf{v}_{n_f} = \tilde{\mathbf{v}_f} + nf$ ($sf = 1.0$).

iv. Generate raw example $\mathbf{x}_n = \tilde{\mathbf{U}}_s \mathbf{v}_n$.

v. Add white noise to the example, i.e., $\mathbf{x}_n = \mathbf{x}_n + nn \sim norm(0, sn)$ ($sn = 2.0$).

vi. Add a small amount of additional white noise that has been passed through a band-pass filter to create spatial correlation.

(b) For each example $n \in \mathbf{D}$ with label -1:

i. For each position ("voxel") p, choose $\mathbf{x}_{n_p} \sim norm(0, rn)$ ($rn = 1.0$).

The performance of the method on these data can be evaluated as on the real data; however, the primary additional benefit of the synthetic data is that the *ground truth* representation is known. Specifically, from the "template beta" vector $\tilde{\mathbf{v}}$, we can produce a "template map" $\tilde{\mathbf{w}}$ as:

$$\tilde{\mathbf{w}} = \tilde{\mathbf{U}}_s \tilde{\mathbf{v}}. \tag{4.8}$$

The $\tilde{\mathbf{w}}$ map will be referred to as the *template map* or *ground truth map*.

4.2.5 Model Training and Prediction Evaluation

We evaluated the properties of models trained using this procedure with different sets of spatial filters, or no filters at all, which used the standard individual voxels as features, i.e., "voxel-only" models. For models trained using filters, the filters always consisted of symmetric Gaussians with diagonal covariance matrices, the parameters of which varied as described below. Note that, on real data, the feature generation procedure requires

convolving each brain map with a Gaussian filter. The brain map, however, has an irregular shape, since many voxels do not correspond to actual brain locations. Therefore, a field was only included if at least half of the voxels within it, necessarily including the center voxel, lied in brain space.

Once the feature set has been generated using the above procedure, model training with the parallelized LARS-EN algorithm proceeds exactly as does training with the non-parallel version. We followed the exact same methodology described in Chapter 3. In particular, we ran Parallel LARS-EN for a certain number of iterations, specified below, and used the regularization path computed by the algorithm to select an appropriate number of features using cross-validation, except that our features were fields rather than voxels. Specific details differed for the 3 sets of data, as described in the following sections.

A separate model was learned for each subject, task, and run in a given dataset. For each model, Parallel LARS-EN was applied to the full set of voxels and run until a given number of voxels were selected. Prediction was then evaluated on data from whichever run was not used for training; however, given the potential confound presented by evaluating prediction on the same data used to choose the model size (number of voxels), we followed the same procedure used in Chapter 3, in which the test set was sub-divided into 2 optimization folds. Predictions were made for the fold not used for optimization and predictions from the two folds were concatenated. The prediction score was the Pearson correlation between these predictions and the true test response.

As will become evident in the results, when training models generated with filters that are highly correlated with each other, lower λ_2 values should be used, including the value 0.0, corresponding to pure Lasso. For fairness, all models, whether using filter-based features or not, were evaluated for the same set of λ_2 values. As described in the previous chapters, computational limitations require the choice of a maximum number of LARS-EN training iterations, and models trained with higher λ_2 values require more selected features to obtain the same prediction performance due to the increased selection of relevant but redundant features. In addition, Lasso does not have a valid solution when the number of features M exceeds the number of instances N. Therefore, for real data, models were trained with a relatively small number of iterations for Lasso and a slightly higher number for higher λ_2 values.

PBAIC

For PBAIC model building, we evaluated voxel-only models as well as models generated with filter sets of 1, 27, or 125 filters. The 1 filter set was 1 Gaussian with a diagonal covariance matrix with value 0.5 in all directions. The 27 filter set consisted of 27 filters with diagonal covariances derived from all permutations of $0.0, 0.5$, and 1.0, where the 0.0 value resulted in 1 non-zero voxel in the corresponding dimension. The 125 filter set consisted of filters derived from all permutations of $0.0, 0.5$, and 1.0; it thus included the 27 filter set as a sub-set. Rationale for evaluating the 1 filter set is explained in Section 4.2.6.

The number of training instances and test instances (for prediction) was 704. The number of instances per optimization fold was 352.

The number of features generated from these filter sets depends on the number of non-zero brain voxels in a given subject's brain. The number of brain voxels, and hence number of features for the voxel-only models, ranged from $\sim 33,300-\sim 35,300$. For the 1 filter set, the number of features ranged from $\sim 31,800-\sim 33,600$. The 27 filter set features count ranged from $\sim 867,000-917,000$, and, for the 125 filter set, the range was $\sim 3,940,000-4,170,000$. Therefore, the training set size for 125 filter models was up to $704\times\sim 4,170,000$.

Models were evaluated using λ_2 values of $0.0, 0.01, 0.1$, and 2.0. Models were trained until the number of selected features were $500, 800, 1500$, and 1500 respectively. Note that the actual number of training iterations may have exceeded these values, since some features were dropped by LARS (see [24] for more details).

For simplicity, many results are visualized only for the 3 best predicted tasks, which are Instructions, VRFixation, and Velocity, and 3 moderately-well predicted tasks, viewing of Body, viewing of Face(s), and Hits.

Pain Data

The number of training instances and test instances (for prediction) was 120. The number of instances per optimization fold was 60.

Filter-based models for Pain were trained using the same sets of 1, 27, or 125 filters used for PBAIC analyses, as well as voxel-only feature sets. The brain sizes for the 14 subjects ranged from $\sim 26,500-\sim 28,200$. Feature size ranges for the 1, 27, and 125 filter sets were: $\sim 25,700-\sim 27,300$, $\sim 700,000-\sim 744,000$, and $\sim 3,182,000-\sim 3,385,000$ respectively. Therefore, the training set size for 125 filter models was up to $120\times \sim 3,385,000$, an extremely large imbalance between instances and features.

The same λ_2 values, $0.0, 0.01, 0.1$, and 2.0, were used. Due to the smaller number of instances in Pain relative to PBAIC, fewer training iterations were used, specifically $100, 1000, 1000$, and 1000 respectively.

Synthetic Data

For the synthetic data, the filter set was created so as to add several "decoy" filters, not used for generating the data, to evaluate the ability of the algorithm to use discretion to successfully select only the truly "relevant" features. The filter set consisted of 13 filters, ranging in standard deviation from 0.0 (1 "voxel") to 6.0 at increments of 0.5. The maximum number of non-zero "voxels" in a filter was therefore 37. Since all voxel locations were considered valid, all fields were included. The feature set was therefore of size $1200 * 13 = 15,600$ and the training set size was therefore $220 \times 15,600$.

Note that for the synthetic data, we are seeking a discriminative function, so ideally logistic regression would be used; however, linear regression can still be used in such situations, and in fact we will see that the linear model predicts nearly perfectly. We used these data only to illustrate properties of the methodology, which includes linear regression, thus warranting the approximation.

Prediction was measured as accuracy, i.e., by binarizing the prediction using 0.0 or lower as a threshold (given the $+1/-1$ labels), and calculating the percentage of instances correctly classified. As with the real data, cross-validation was used to select the optimal model size; however, for the synthetic data, we can exploit the fact that a potentially infinite number of datasets can be created, so we created 10 datasets. As with the real data, we did not evaluate prediction on datasets used to optimize model size. Therefore, training was performed for each of the 10 datasets. From the 10 resulting regularization paths, a model was selected and evaluated on each of the 9 remaining datasets, resulting in 90 models and evaluations. The optimization for a given model

was therefore performed using the 8 datasets not used for training or testing, rather than the 1 optimization set used for training with the real data. The existence of multiple optimization datasets afforded us some flexibility, and we chose to exploit it by following a *minimax* approach similar to that used in *robust optimization* [8], which optimizes the worst-case scenario. The goal of robust optimization is to produce more stable models, which is similar to our goal of enhancing model reliability. Interestingly, Xu et al. [98] have shown that Lasso itself solves a robust optimization problem.

We evaluated each model in the full regularization path for each of the 8 datasets, determined the minimum training accuracy, by iteration, for each of these 8 datasets, and selected the model with the maximum minimum accuracy. Precisely, given k iterations, 8 optimization datasets in collection O, and associated accuracy scores A, we chose model x as:

$$x = \underset{k \in K}{\operatorname{argmax}}(\min_{o \in O}(A_{ko})). \tag{4.9}$$

As discussed below, we also evaluated the reliability of these models. Results are reported for the 45 unique pairs of models from the set of 90.

GroupX: Maximum One Field Per Location

When we are computing with Gaussian filters, it is clear that our underlying assumption is that there is at most one "true" field at a given location; in fact, this assumption is specifically encoded as a constraint in the synthetic data generation. While Lasso and Elastic Net will find a sparse representation amid the vast space of candidate fields, the objectives themselves do not incorporate any constraint on the number of fields at a given location. In fact, what we really seek is a method that takes groups of related predictors, for instance fields at the same location, as input, similarly to the Group Lasso [100], but, unlike Group Lasso, assigns a weight to only *one* of the members of the group, rather than assigning a similar weight to all members of the group. Ignoring the desire to select correlated predictors as in Elastic Net, the basic objective might be:

$$\beta = \text{argmin}\,||\mathbf{y} - \mathbf{X}\beta||_2^2 \text{ s.t. } ||\beta||_0 \leq G,$$
$$||\beta_{grp1}||_0 \leq 1, ||\beta_{grp2}||_0 \leq 1, \ldots, ||\beta_{grpK}||_0 \leq 1. \tag{4.10}$$

This objective could be solved directly using a greedy algorithm, such as Orthogonal Matching Pursuit [89], which directly solves an approximate l_0-regularized solution, and thus can easily be adapted to simply remove all members of a group from the set of candidate predictors when one member is added to the active set. LARS-EN is not a greedy algorithm, since it updates a coefficient on each step only to the extent that another predictor becomes equally correlated. However, since Elastic Net has the advantage of facilitating inclusion of relevant predictors that are correlated yet in different groups, ideally we would like to solve the objective in Eq. (4.10) within the LARS-EN framework. In practice, LARS-EN can just as easily be modified to eliminate all members of a group from consideration when one is added to the active set. If $\mathcal{G}$ is the set of all groups, $\text{grp}(j)$ denotes the group of predictor j, and $\text{grp}(\mathbf{V}) = G \subseteq \mathcal{G}, \exists v \in \mathbf{V}$ s.t. $\text{grp(v)} = G$ returns the set of all groups with at least one member in vector $\mathbf{V}$, we modify the selection of $\hat{j}$ in ([24, Eq. 2.9]), to be discussed more thoroughly in Eq. (5.2), to be:

$$\hat{j} = \underset{j}{\text{argmax}}\,\{|\hat{\mathbf{c}}_j|\}, \text{grp}(j) \notin \text{grp}(\mathcal{A}). \tag{4.11}$$

and ([24, Eq. 2.13]), to be discussed more thoroughly in Eq. (5.12), to be:

$$\hat{\gamma} = \min_{\text{grp}(j) \notin \text{grp}(\mathcal{A})}^{+} \left\{ \frac{\hat{C} - \hat{\mathbf{c}}_j}{\mathbf{A}_{\mathcal{A}} - a_j}, \frac{\hat{C} + \hat{\mathbf{c}}_j}{\mathbf{A}_{\mathcal{A}} + a_j} \right\}. \tag{4.12}$$

However, since LARS-EN is not approximating the l_0 solution, LARS-EN modified in this manner is not necessarily solving the objective in Eq. (4.10). In fact, as its behavior will underscore, it is not immediately clear which objective it is optimizing.

We call LARS-EN with the modifications given in Eqs. (4.11) and (4.12) LARS-EN GroupX to reflect the exclusion of an entire group of predictors when one is added to the active set (or re-consideration when a group member is dropped from the active set). We will evaluate GroupX on the synthetic data as a preliminary exploration of the complications of using Lasso and Elastic Net with Gaussian filters, and the potential

benefit of enforcing a one-field-per-location constraint. However, since the approach is still poorly understood, we consider only standard LARS-EN when analyzing the real data. As we will see, even this approach reveals some interesting model properties.

4.2.6 Map Smoothing

In Chapter 3, we found that model properties, specifically prediction and reliability, can sometimes be improved by applying a small amount of spatial smoothing directly to learned models, where smoothing is obtained by convolving the map with a Gaussian spatial kernel of a particular size, using the exact same procedure described above to generate a set of features for a given filter, including setting non-brain voxels to 0.

Therefore, the filter-based approach described in this chapter could be considered a more sophisticated form of smoothing that seeks the optimal set of smoothing parameters based on prediction performance. The main difference between filter-based smoothing and the smoothing approach used in Chapter 3, which we will refer to as *blind smoothing* due to its disregard of predictive information, is that with filter-based approaches, one is applying smoothing to both the *image data* and the *learned map*. More specifically, to generate the feature set used for training filter-based models, one smooths the data with each candidate filter and concatenates the results. After the model is learned, the final stage, which uses the learned mapping function to produce a map in real space, is the same procedure used to produce a smoothed map from a raw map.

In Chapter 3, we argued that the success of blind smoothing warrants comparison to blind smoothing as a baseline when evaluating any procedure that incorporates spatial structure into modeling. Since, like blind smoothing, the filter-based approach is directly imposing additional spatial structure on models, such comparison is especially relevant in this case. Herein, we not only compare raw maps learned using voxel-only feature sets, but for fairness, also evaluate filter-based models that have had additional smoothing directly applied. We evaluate maps learned with both voxel-only and filter-based maps smoothed with the same 2mm $\times$ 2mm $\times$ 2mm (approximately) filter shown in Chapter 3 to sometimes produce more reliable and still well-predicting maps. As in that chapter, we also consider maps that have been smoothed and then *thresholded* by setting values below 0.01 to 0.

The filter used in the 1 filter feature generation set is the same $\sim 2mm$ filter used to evaluate smoothed models, allowing direct comparison between the filter-based approach, which involves a data smoothing step, and blind smoothing, which does not. For the 27 filters set, the maximum filter size is approximately 4mm $\times$ 4mm $\times$ 4mm and, for the 125 filters set, 8mm $\times$ 8mm $\times$ 8mm, placing these filter sets within the range of smoothing kernels considered in Chapter 3 (and within the range of sizes commonly used in the literature for data smoothing).

4.2.7 Model Metrics

Reliability

In Chapter 3, we described 3 metrics for measuring the *reliability*, or consistency across runs, of a model: non-zero overlap, weighted overlap, and correlation. We then discussed the necessity for reporting a significance estimate for these scores, which take into account both the value distribution of model weights (1-point correlations) and the covariance of the weights (2-point correlation), as reflected in spatial correlations. In this work, we report both the uncorrected scores and the z-scores obtained using the spatially-based significance estimation approach in Chapter 3. As in that chapter, we compute reliability over all 4 pairs of optimization folds for a given pair of models and report means over folds and standard deviations over model pairs. Please see Chapter 3 for more details.

Ground Truth Overlap

Prediction is an important measure of validity, but in this work we seek to go beyond prediction by learning the "true" representation of the data. While we do not know the ground truth representation for real data, for the synthetic data we compute the *template beta* and *template map* as described above. As discussed in Section 3.2.6 in Chapter 3, the same procedure used to evaluate the reliability between two learned maps can also be used to evaluate the similarity between a learned map and a *ground truth* map. In fact, the overlap metrics when used to compare learned maps to ground truth were shown to have interesting parallels to notions of sensitivity and specificity. In the present work, we compute the non-zero overlap and weighted overlap between learned maps and template

maps on the synthetic data, and report the significance of these scores using the spatially-based z-score approach described in Chapter 3.

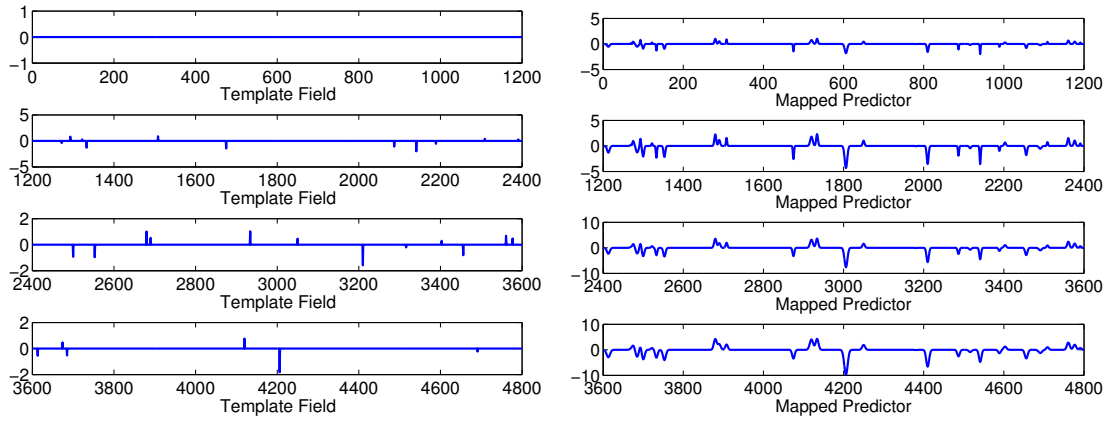
Mean Weighted Filter Size

One major advantage of this technique is that it can allow one to interpret the model to make inferences about the spatial structure and how the structure varies across subjects, tasks, and regions. Specifically, one can examine properties of the fields selected in a particular model; for Gaussian filters, it is natural to explore the size and shape of the fields. One simple property therefore is the spread in voxels of the filter. For one-dimensional filters, this property can be quantified simply as the standard deviation, but in multiple dimensions, a full covariance matrix would be needed. For comparison purposes, a one-valued metric is desirable. A simple representation therefore is the number of voxels affected by the filter, i.e., the number of non-zero voxels. For instance, with our $l_\infty-$regularized filters and threshold of 0.01, a one-dimensional Gaussian filter with a standard deviation of 2 voxels has 13 non-zero voxels, so its "size" can be considered to be 13 voxels. A 3-dimensional diagonal Gaussian filter with a standard deviation of 1.0 in (any) 2 dimensions and 0.5 in the remaining dimension has size 7 voxels in the 1.0 dimensions and 5 in the remaining dimension, giving it a size of 245 voxels. The size of the fields selected by a model can be a useful metric. To arrive at a one-valued score for this metric, therefore, one can simply report the average size of the filters corresponding to the selected fields. However, some filters may be given much higher weights in the model; therefore, we report a *mean weighted filter size*, by scaling each field by its model weight before averaging. Precisely, let sz be the size of a field in voxels and β the beta weights of the learned model. The mean weighted filter size is:

$$mwfs = \frac{\sum_{f \in F} sz_f \beta_f}{|F|}. \tag{4.13}$$

Spatial Distribution

As discussed in Chapter 2, another useful model metric is its Spatial Distribution. We use the metric described in Section 2.2.4, which computes the entropy of weight sums



(a) Template Mapping Function (b) Mapped Feature Vector

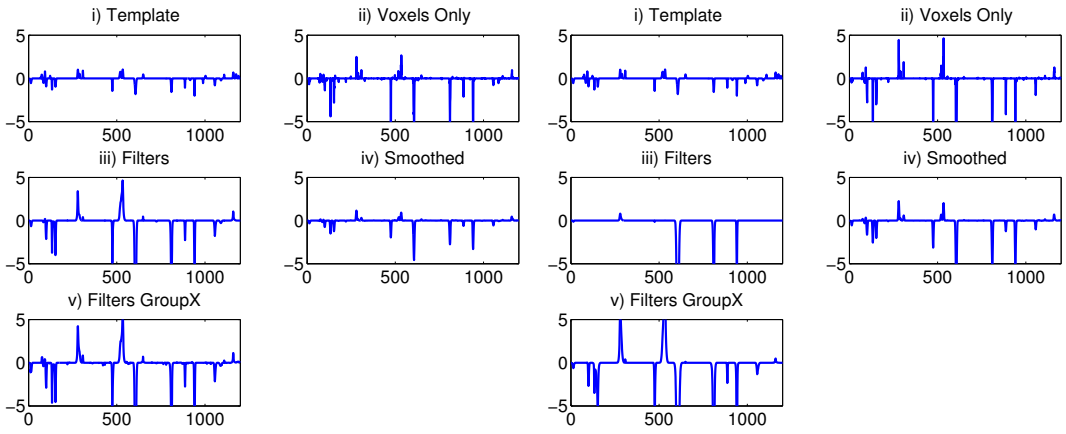
Figure 4.4: *There is a dramatic difference between the true mapping function and the mapped feature vectors used for prediction; features localized at the same position are highly correlated with each other.* (a) True mapping function $\tilde{\mathbf{v}}$ used to generate data; rows correspond to filters with standard deviation 0 (voxels only), $1, 2$, and 3 respectively, x axis corresponds to the 1200 locations, and y axis to the weight assigned to the given field. (b) $\mathbf{w}$ after mapping into feature space; rows again correspond to fields for the filters listed in (a), x axis to locations, and y axis to the weight for the given field.

throughout the brain, approximating the spatial distribution properties of a model. Please refer to that section for further details and rationale.

4.3 Results

4.3.1 Synthetic Data: Correlated Fields

Before examining the prediction and reliability performance of this approach, it is first illuminating to visualize exactly how a feature set created using the described Gaussian filters appears. Figure 4.4(a) shows the template mapping function $\tilde{\mathbf{v}}$ that was used to generate the data, divided into 4 rows, where each row corresponds to the 1 voxel-only filter or one of the 3 filters in $\tilde{T}_s$ (hence filter standard deviations $0, 1, 2, 3$ respectively) and each position corresponds to the field associated with the corresponding location μ. Clearly there is no correlation among fields for different filters, as intended; however, Figure 4.4(b) shows the mapped feature set $\tilde{\mathbf{v}}' = \tilde{\mathbf{w}}\tilde{\mathbf{U}}_s$, just as if the template map $\tilde{\mathbf{w}}$ were a data instance mapped into training feature space. The first row, resulting from



(a) Maps $\lambda_2 = 0.01$ (b) Maps $\lambda_2 = 2.0$

Figure 4.5: *Using filters with higher λ_2 values results in under-selection of relevant fields, but the GroupX modification can mitigate the effect.* (i) template synthetic data map (same regardless of λ_2 value), (ii-v) maps learned using voxels only, filters, smoothing the voxel only map, or filters with GroupX respectively, for $\lambda_2 = 0.01$ (a) and $\lambda_2 = 2.0$ (b); averaged over all 10 datasets.

convolution with a one-voxel filter and hence the identity matrix, is equivalent to $\tilde{\mathbf{w}}$. The other three rows represent convolution of $\tilde{\mathbf{w}}$ with the 3 filters in $\tilde{T}$ and are almost identical to this row, illustrating the extremely high correlation between features generated by filters of different scales. Together, the 4 rows illustrate the mapped feature vector $\tilde{\mathbf{v}}'$. Our procedure seeks to recover, in part, the very sparse orthogonal signal in Figure (a) from vectors similar to this highly correlated transformed signal.

To observe how such a correlated feature set impacts performance, it is illustrative to evaluate performance on the synthetic dataset. Figure 4.5 shows several of the one-dimensional "maps" for the synthetic data, divided into two sections based on the λ_2 value used for training (0.01 or 2.0). For each λ_2 value, map (i) shows the template map $\bar{\mathbf{w}}$, which is a characteristic of the data and hence not dependent on the training parameters. The remaining 4 maps are the learned maps $\mathbf{w}$ yielded by training with a λ_2 of 0.01 or 2.0 using 4 approaches: (ii) training using the voxel-only set T_0, (iii) training using the filter-based set T_s (with decoy filters), (iv) smoothing the voxel-only maps, and (v) training using T_s with the *GroupX* method.

The primary result of this figure is the difference in maps based on λ_2; however, one limitation of the dataset apparent from Figure 4.5(a) alone is that the used filters are quite

small relative to the full size of the map space and, as a result, the voxel-only map in (ii) actually approximates the template map quite well. After applying the small amount of smoothing in (iv), the map becomes somewhat muted, but the widths of the "blobs" even more closely approximate those of the template map. Note, however, that results were similar even for datasets created from slightly larger filters (with larger filters used for testing).

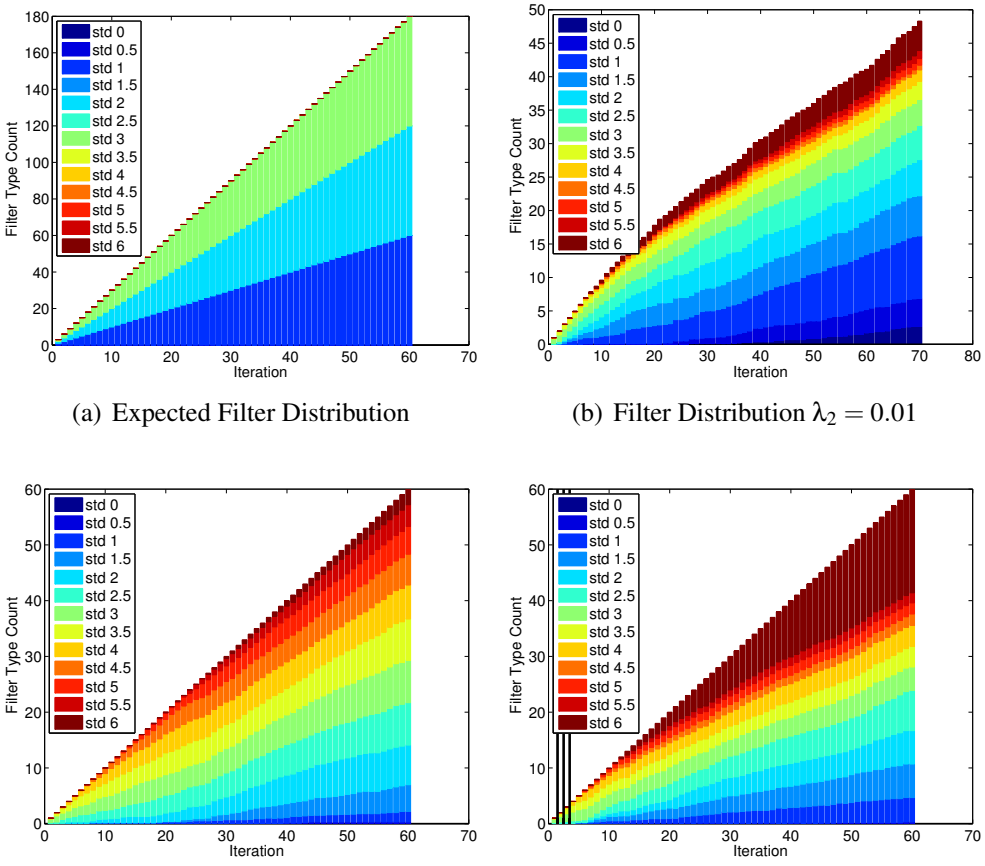
When we consider the results using different λ_2 values, a trend emerges. Comparing figures (a)(iii) and (b)(iii), it is clear that the higher λ_2 value results in a map in which the majority of "blobs" are not recovered. Since all training was performed for the same number of iterations, and cross-validation was used to select an appropriate number of fields, the optimal performance was obtained without selecting many of the relevant fields, suggesting that the remaining fields selected were irrelevant. To understand a possible explanation, recall that when using higher λ_2 values, correlated predictors are more likely to be included. As Figure 4.4 showed, Gaussian features generated with the same mean position will be highly correlated with each other, regardless of the standard deviations of the filters. In effect, fields with different standard deviations centered at the same location create a cluster of highly correlated predictors. Higher λ_2 values will result in a greater number of these correlated fields being selected. Still, to optimize prediction performance, it would seem counter-productive to indiscriminately select correlated but redundant features; however, examining the template map in Figure 4.5(a)(i) and 4.5(b)(i), it is clear that certain positions have higher template weights, due to the proximity of several generating filters. Ideally, only the true filters would be selected; however, since these positions are so much more relevant than the other positions, training performance can actually be maximized by repeatedly adding weight to the positions, which is accomplished by selecting multiple features localized at the position. By the time the 80 training iterations have been completed, these most relevant positions will have been over-selected and the remaining relevant positions under-selected.

Training with a higher λ_2 value may therefore lead to over-inclusion from the clusters of highly correlated predictors resulting from the group of fields centered at the same location. One solution, therefore, may be to simply train using Lasso instead of Elastic Net. However, we expect there will be some predictors that are correlated yet not centered at the exact same location, so ideally we would like to employ l_2-regularization while prohibiting selection of more than one field centered at a particular location. Recall

that the GroupX LARS-EN modification was designed to limit selection to at most one member from each of a set of pre-defined groups. If we define our groups to be those predictors associated with fields centered at the same location μ, GroupX should eliminate such over-selection. The results in Figure 4.5 confirm this hypothesis. When training with the low λ_2 value of 0.01, the results are nearly identical whether or not GroupX is used. However, with the higher λ_2 value of 2.0, the learned map is noticeably different when GroupX is used (Figure 4.5(b)(v) versus 4.5(b)(iii)). More relevant blobs are selected, and the map looks much more similar to the map learned using the lower λ_2 value, although fewer blobs are still selected with the higher λ_2 value. In Section 4.4, we expand on this notion of "relevance" in this setting.

The results in Figure 4.5 illustrate that the learned maps may over-select irrelevant fields, especially as the number of iterations increases. One simple way to summarize how accurately the learned model is capturing the true properties of the data is to compare the distribution of the sizes of the filters used to generate the selected predictors at each iteration to those used to generate the actual data. Figure 4.6 visualizes this comparison. Recall that the set of generating filters, $\tilde{T}_s$ includes only 3 filters, of standard deviations $1.0, 2.0,$ and 3.0 voxels; however, the set of candidate filters T_s used for training included 13 filters, ranging from standard deviation 0.0 to 6.0 at increments of 0.5 voxels.

Figure 4.6(a) shows vertical histograms of the 13 filter sizes at each iteration, with each representing the expected distribution if the distribution of selected filter sizes in the learned model exactly matched the true distribution of generating filter sizes (scaled given the number of selected fields expected at each iteration). Only 3 filter sizes have non-zero counts. Figures 4.6(b-d), however, show the counts of each filter size at each iteration (averaged over the 10 datasets) for 3 methods: Elastic Net (EN) with $\lambda_2 = 0.01$, EN with $\lambda_2 = 2.0$, and EN GroupX with $\lambda_2 = 2.0$. Indeed, for $\lambda_2 = 0.01$ (b) and GroupX with $\lambda_2 = 2.0$ (d), the number of fields corresponding to filter sizes that were too large to have generated the data increases as the number of training iterations increases (note that the total number of selected fields increases non-monotonically over iterations due to fields being dropped from the model). $\lambda_2 = 2.0$ without GroupX (c) shows a more consistent pattern over iterations, but, as suspected, also shows the most dramatic over-selection of irrelevant filter sizes. In fact, the proportions using this approach are so uniform across iterations, that the selection, based on filter size, appears arbitrary. Unfortunately, while Figure 4.5 showed GroupX to suffer less from over-selection of



(a) Expected Filter Distribution

(b) Filter Distribution $\lambda_2 = 0.01$

(c) Filter Distribution $\lambda_2 = 2.0$

(d) Filter Distribution $\lambda_2 = 2.0$ GroupX

Figure 4.6: *Higher λ_2 values result in over-selection of irrelevant fields.* Histogram of filter size counts (a) based on proportions in template mapping function scaled by expected active set size, (b-d) selected at each iteration using filters with $\lambda_2 = 0.01$, $\lambda_2 = 2.0$, and $\lambda_2 = 2.0$ with GroupX respectively; averaged over all 10 datasets.

fields centered at the same location, EN GroupX with $\lambda_2 = 2.0$ still does display some over-selection of larger filters, slightly more so than when using the lower λ_2 value of 0.01. However, the lower λ_2 value of 0.01 is not immune to this effect either, likewise displaying some over-selection of larger filter sizes; in fact, while not shown, even pure Lasso suffered from similar over-selection biases. Interestingly, though, in the GroupX case, the largest filter size is dramatically over-selected at later iterations, as the model size exceeds the "true" size. These over-selection trends suggest that the larger filters are especially likely to be selected when most of the gains in prediction have already occurred, perhaps because the largest filters provide the least sensitivity to noise. Since

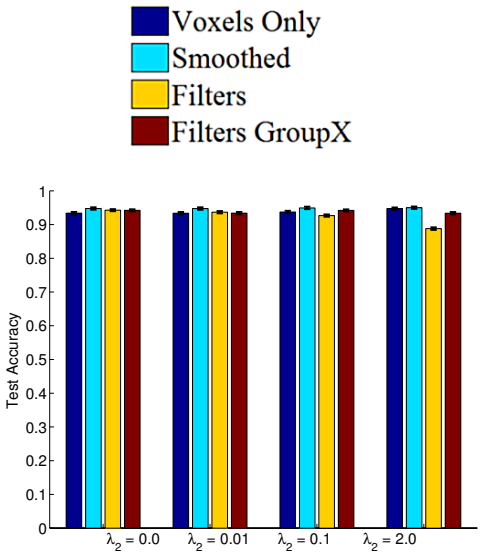
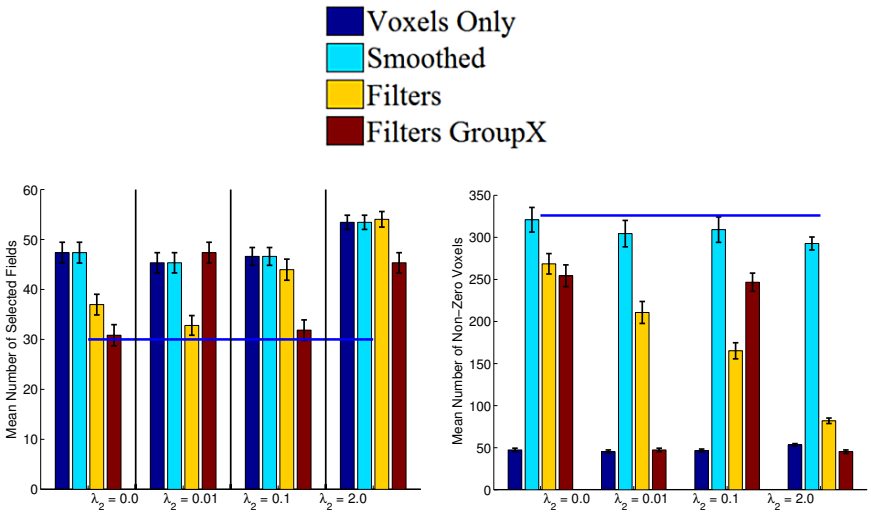


Figure 4.7: *Prediction performance on synthetic data is strong for all methods, but weakens slightly when filters are used with high λ_2 values, especially without GroupX.* Cross-validated prediction performance by method and λ_2 value, averaged over all 90 cross-validated maps. Bars reflect 95% confidence.

these larger filter predictors are likely be more highly correlated, they are less likely to be over-selected when using a lower λ_2 value.

The impact of over-selection at high λ_2 values is evident in Figure 4.7, in which we evaluate the prediction performance for voxel-only models, with or without blind smoothing, and filter-based models. Ultimately, the signal in this synthetic dataset is quite easy to learn, as all methods have nearly perfect prediction performance. Despite the limitations of the Gaussian filters, filter-based models perform as well as blind smoothing and better than raw voxel-only models. However, performance for the filters does begin to degrade when the high λ_2 value of 2.0 is used, especially if the GroupX modification is not employed. It is likely that the over-selection bias leads to this impaired performance.

Figure 4.8(a) summarizes the effect of λ_2 on model size, measured as the number of selected fields. Note that raw voxel-only models and their smoothed counter-parts will always have the same model sizes for a given λ_2. Filter-based models exhibit more variability in model size across λ_2 than voxel-based models, likely due to the increased correlation structure in the mapped data. With lower λ_2 values, the filter-based models tend to achieve their maximal prediction performance with smaller model sizes than the voxel-only counterparts. Thus, as hypothesized, filter-based models tend to be more

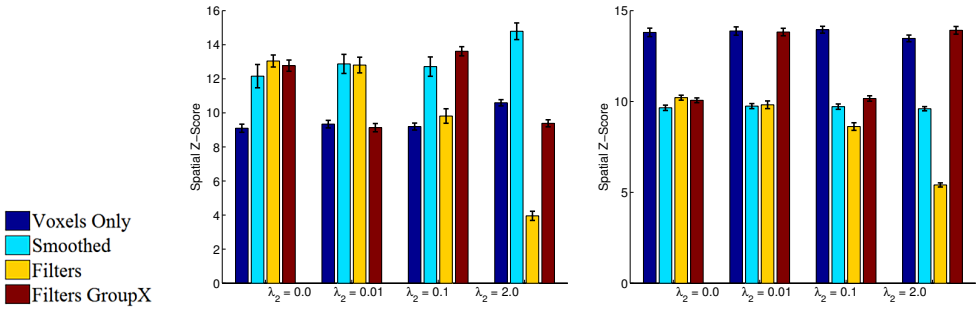


(a) Model Size: Synthetic (b) Map Non-Zero: Synthetic

Figure 4.8: *Filter-based models achieve optimal synthetic data prediction performance with fewer features than voxel-based models; smoothed maps are the least sparse, followed by filter-based maps.* (a) Model size, measured as number of selected fields with cross-validation, and (b) Number of non-zero voxels (non-sparsity) in cross-validated learned maps; both plots by method and λ_2 value, averaged over all 90 cross-validated maps. Bars reflect 95% confidence. Horizontal line indicates "true" number of generating fields (a) or template map non-zero values (b). Note: smoothed maps necessarily have the same model size as corresponding raw maps.

parsimonious than their voxel-only counter-parts, requiring fewer training iterations to achieve similar performance, and facilitating interpretation. Since the running time of LARS and its variants can increase dramatically as the number of iterations is increased, depending on implementation details, fewer iterations can provide a major computational advantage. Interestingly, though, the model size does not increase monotonically for the filter-based models, especially when using the GroupX modification. Anomalous values for GroupX, particularly with $\lambda_2 = 0.01$, were observed across several measures, indicating a behavior inconsistency that likely reflects the as yet poorly understood nature of this ad hoc modification.

Figure 4.8(b) demonstrates an additional property of models involving smoothing, whether blind or filter-based. This figure shows the number of non-zero voxels (non-sparsity) in the maps $\mathbf{w}$, in the original spatial dimensions, produced by the learned models. Not surprisingly, blind smoothing of voxel only maps produces the largest number of non-zero voxels, though most closely approximating the true number, and

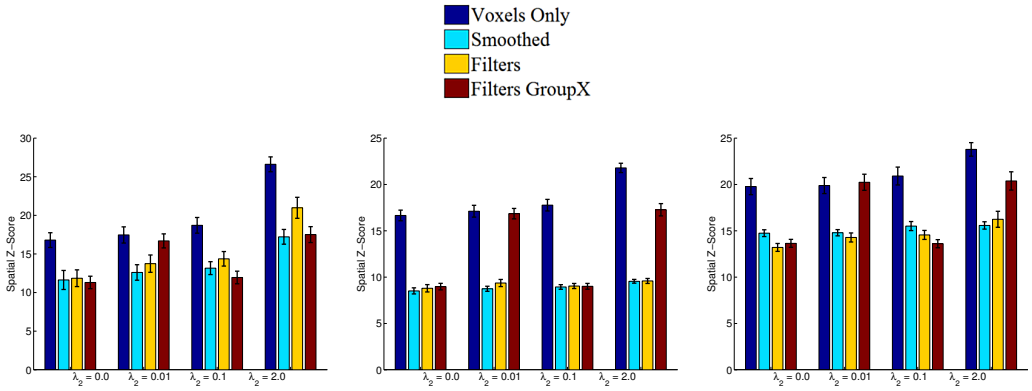


(a) Ground Truth Non-Zero Overlap: Z (b) Ground Truth Weighted Overlap: Z

Figure 4.9: *Map synthetic ground truth overlap scores are small yet significant regardless of method, with filter-based training using GroupX showing promise for achieving both significant sensitivity and specificity.* Spatial z-scores for non-zero overlap and weighted overlap between ground truth (template) map and cross-validated weight maps by method and λ_2 value, averaged over all 90 cross-validated models. Bars reflect 95% confidence.

the raw voxel-only maps the fewest; filter-based maps, which effectively involve an intermediate amount of smoothing, tend to lie between the two. Interestingly, voxel-only maps increase in size as λ_2 is increased, due to the inclusion of a greater number of (correlated) voxels, but blindly smoothed maps and, especially, filter-based maps (with some anomalies for GroupX) tend to *decrease* in size as λ_2 is increased, most likely because of high spatial overlap among the true "blobs."

The preceding figures essentially show many falsely selected or non-selected fields, especially with higher λ_2 values, and thus we expect relatively poor sensitivity and specificity with regard to the true generating set of fields. Using the methods mentioned in Section 4.2.7, Figure 4.9 shows spatially-corrected non-zero overlap (a) and weighted overlap (b) between the learned and true maps. While the significance values for all methods are small, they are all significant. As expected, maps employing some form of smoothing, whether through blind-smoothing or filter-based learning, demonstrate the most significant non-zero overlap with the ground truth (roughly corresponding to sensitivity); however, this significance drops significantly for the filter-based models as λ_2 is increased. The drop is not as precipitous if GroupX is used, but the non-monotonicity of the GroupX scores is again observed. The raw voxel-only maps display the most consistently significant weighted overlap with ground truth (roughly reflecting specificity). However, while filter-based models without GroupX show poor ground truth weighted overlap significance, the GroupX modification tends to be on par with



(a) Non-Zero Overlap: Spatial Z-Score (b) Weighted Overlap: Spatial Z-Score (c) Map Correlation: Spatial Z-Score

Figure 4.10: *Reliability of filter-based synthetic data maps increases monotonically with λ_2 value, whether uncorrected or corrected; uncorrected scores exceed those of voxel-only maps, with or without smoothing, but, when spatially corrected, show no significant benefit over voxel-only maps.* Non-zero overlap, weighted overlap, and map correlation spatial z-scores between cross-validated weight maps by method and λ_2 value, averaged over all 45 pairs of cross-validated models. Bars reflect 95% confidence.

voxel-only models (though again inconsistently) . Thus while the over-selection effect is observed, filter-based models trained using GroupX shows some promise for yielding models that exhibit both strong sensitivity and specificity.

Figure 4.9 illustrated the similarity between the learned maps and the ground truth maps. Figure 4.10 meanwhile illustrates the reliability between the learned maps on the synthetic data. Figure 4.10 shows spatial-z scores for non-zero overlap, weighted overlap, and correlation. Interestingly, the reliability scores for blindly smoothed maps are much lower than the voxel-only maps, but nearly identical to those achieved by filter-based models without the GroupX modification. Filter-based models trained using the GroupX modification, however, usually exhibit significantly improved reliability scores that rival those of the voxel-only models. Recall that the spatial z-scores penalize for spatial structure, such that smoothing uniformly, as in blind smoothing, will be penalized most harshly. The similar reliability observed for such blindly smoothed maps and filter-based models without GroupX is another indication that filter-based models have a tendency to over-select irrelevant fields, bordering on arbitrariness as observed in Figure 4.6(c). In this case, while the GroupX modification again mitigates this effect, the most consistently significantly reliable maps are those obtained with voxel-only training, though this trend may be specific to these data. Since Chapter 3 demonstrated improvement in reliability

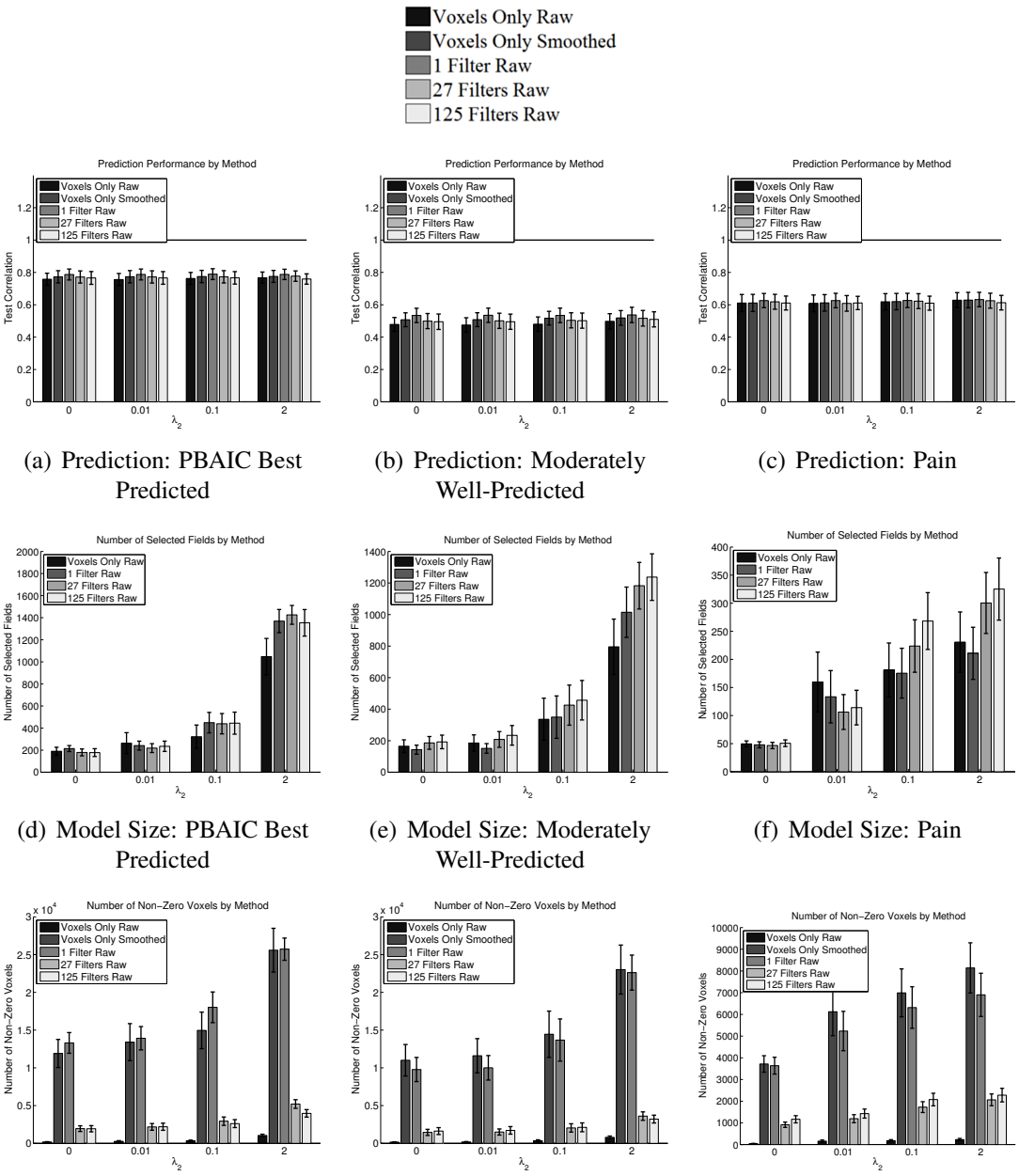
when a small amount of smoothing was used, but impairments when too much smoothing was applied, the strong comparative reliability of voxel-only maps might be attributable to over-smoothing.

The results in Figure 4.10 also underscore one further point about prediction and reliability. The filter-based models trained with GroupX again show a non-monotonic increase in reliability; however, in Figure 4.7, prediction scores for this method were nearly identical across λ_2 values and decreased slightly at the highest λ_2 value. Thus, prediction and reliability scores are distinct measures that vary independently. As Chapter 3 showed, models will often exhibit both strong prediction and reliability; however, it is also possible for a relatively poorly predicting model to achieve strong reliability scores, which is likely what is observed in the case of overly simplistic filter-based models trained with a high λ_2. Thus, reliability is not a measure of model validity in the same way as prediction. Ultimately, there is not a direct relationship between prediction and reliability, yet both measures are informative and thus important to consider when evaluating a model.

4.3.2 Real Data: Prediction, Reliability, and Visualization

Figure 4.11, shows the prediction, model size, and map size of models trained on the real fMRI data, beginning with the 3 task sets of greatest interest: well-predicted PBAIC tasks, moderately well-predicted PBAIC tasks, and Pain tasks. Prediction and map size are plotted for 5 models, including all 4 feature sets (voxels only, 1 uniform filter, 27 filters, and 125 filters) and, for comparison, the results of blindly smoothing the voxel-only maps with the $2mm$ filter; model size is not shown for blind smoothing, since it is necessarily equivalent to the corresponding voxel only map. Figures a-c show that prediction performance does not significantly vary across any of these methods. Also, unlike the synthetic data, prediction performance does not begin to suffer when $\lambda_2 = 2.0$. It is likely that, as with any use of Elastic Net regularization, λ_2 values are not absolute, and the behavior observed with specific values will be dependent on the dataset; relative trends will likely be similar though.

Likewise, Figures d-e shows that, somewhat surprisingly, model size tends not to vary across methods trained with the same λ_2 value. Indeed, the most significant differences result from increasing λ_2, which is consistent with the results in both Chapter 2 and



(a) Prediction: PBAIC Best Predicted

(b) Prediction: Moderately Well-Predicted

(c) Prediction: Pain

(d) Model Size: PBAIC Best Predicted

(e) Model Size: Moderately Well-Predicted

(f) Model Size: Pain

(g) Map Non-Zero: PBAIC Best Predicted

(h) Map Non-Zero: Moderately Well-Predicted

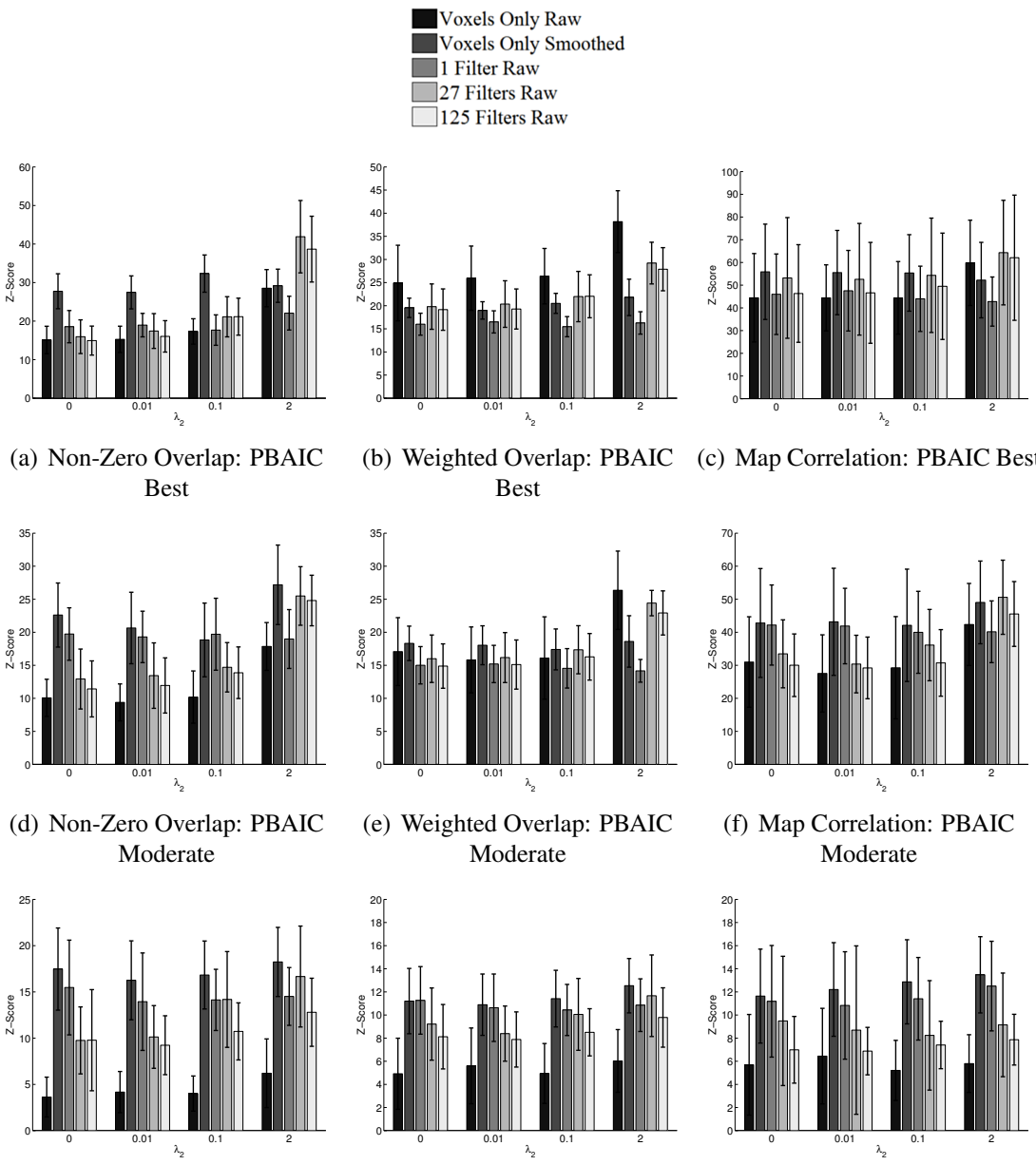
(i) Map Non-Zero: Pain

Figure 4.11: *Prediction and model size, within λ_2 values, are similar regardless of training method, but number of non-zero voxels is significantly higher with universal filters.* Cross-validated prediction performance (a-c), number of selected fields (d-f), and number of non-zero voxels (g-i), by set of PBAIC or Pain tasks, by method and λ_2; averaged over all 3 (PBAIC) or 14 (Pain) subjects, 3 (PBAIC) or 2 (Pain) tasks, and 2 runs (a-c) or 4 cross-validation folds (d-i); model size for smoothed is same as for raw non-zero voxels and therefore not shown; Bars reflect 95% confidence.

Section 4.3.1. The relative map sizes in Figures f-h are expected, though, and are similar to those observed on the synthetic data: voxel-only maps have the smallest number of non-zero voxels, and maps created using one uniform filter have the greatest number, with selective filter-based models between the two extremes. Interestingly, the models trained with one uniform filter have similarly-sized maps, regardless of whether the smoothing was performed blindly on the voxel-only maps, or made use of training data that had been smoothed with a matched filter (in the case of the 1-filter-based models). Likewise, the map sizes are similar for models produced from selective filter sets, regardless of the number of filters in the set (27 or 125).

Figure 4.12 shows spatially-corrected reliability z-scores for these 5 methods over the 3 sets of tasks. Overall, trends observed in Chapter 3 are observed again: voxel-only based maps have the most significant reliability for the best-predicted PBAIC tasks (especially for weighted overlap); voxel-only maps and their directly smoothed counterparts are more significantly reliable for moderately well-predicted PBAIC tasks; and, for Pain tasks, voxel-only maps are not as reliable as smoothed maps. Within maps involving any degree of smoothing, though, blindly smoothed maps tend to be at least as reliable as maps produced using any set of filters. Note also that PBAIC maps are much more significantly reliable than Pain maps, though all are significant. Also note that smoothing alone does not necessarily increase the reliability of any map. As shown in Figure B.5 in the Appendix, even when the filter-based maps are themselves smoothed, they are not more reliable than the smoothed voxel-only maps.

Figure 4.13 provides a visualization of the differences between maps (for the PBAIC Instructions response) produced using voxels only, blind smoothing of voxel-only maps, and selective filters (27 in this case). While prediction and reliability scores tend to be similar between blind smoothing and filter-based smoothing, the resulting maps appear quite different. As supported by the number of non-zero voxels in the maps (approximately shown in Figure 4.11(g)), blindly smoothed maps appear more dense, and the spatial structure is, understandably, more uniform. These differences likely have implications for map interpretation, although the advantages and disadvantages depend on the goals of the modeler, underscoring a point that was emphasized in Chapter 2: map evaluation requires researchers to define their goals in producing and interpreting brain maps. The fact that maps can still appear quite dissimilar despite similar prediction and



(a) Non-Zero Overlap: PBAIC Best

(b) Weighted Overlap: PBAIC Best

(c) Map Correlation: PBAIC Best

(d) Non-Zero Overlap: PBAIC Moderate

(e) Weighted Overlap: PBAIC Moderate

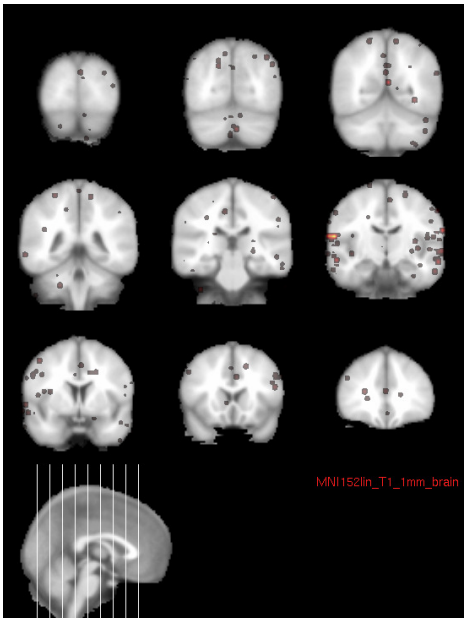
(f) Map Correlation: PBAIC Moderate

(g) Non-Zero Overlap: Pain

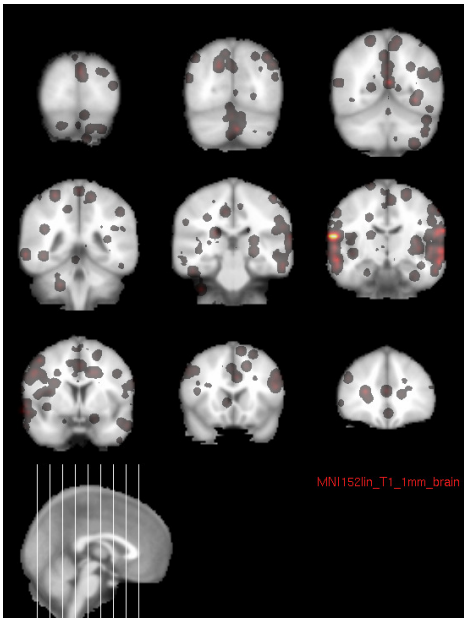
(h) Weighted Overlap: Pain

(i) Map Correlation: Pain

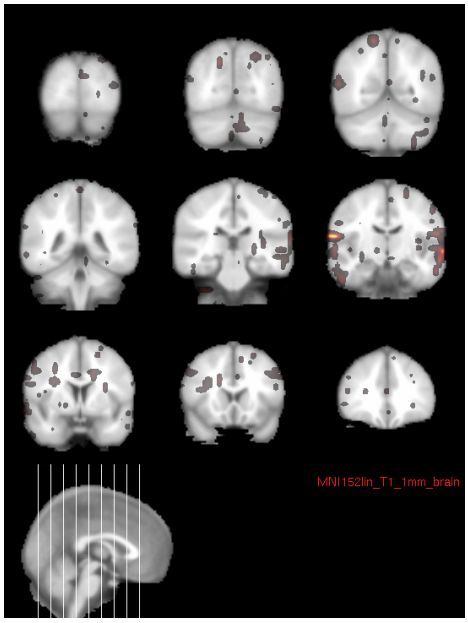
Figure 4.12: *On real data, filters do not significantly improve reliability compared to simple smoothing of voxel-only maps, which often improves reliability compared to raw voxel-only maps.* Spatial z-scores for the non-zero overlap, weighted overlap, and map correlation reliability metrics for voxel-only, voxel-only smoothed, and 1, 27, and 125 filters; voxel-only smoothed is shown for comparison with 1 filter approach. Means are calculated over all 3 (PBAIC) or 14 (Pain) subjects, 3 (PBAIC) or 2 (Pain) tasks, and 4 cross-validation folds; variance over tasks and subjects. Bars reflect 95% confidence.



(a) Instructions Voxels Only



(b) Instructions Smoothed



(c) Instructions 27 Filters

Figure 4.13: *Despite similar prediction and reliability scores, maps produced with uniform smoothing of voxel-only maps appear much less selective than those produced with filter-based smoothing.* Maps learned for subject 1 run 1 cross-validation fold 1 Instructions with λ_2 of 0.1 are shown at indicated slice locations: (b) is the voxel-only map in (a) after 2mm Gaussian smoothing, (c) is the map learned using the 27 filter feature set.

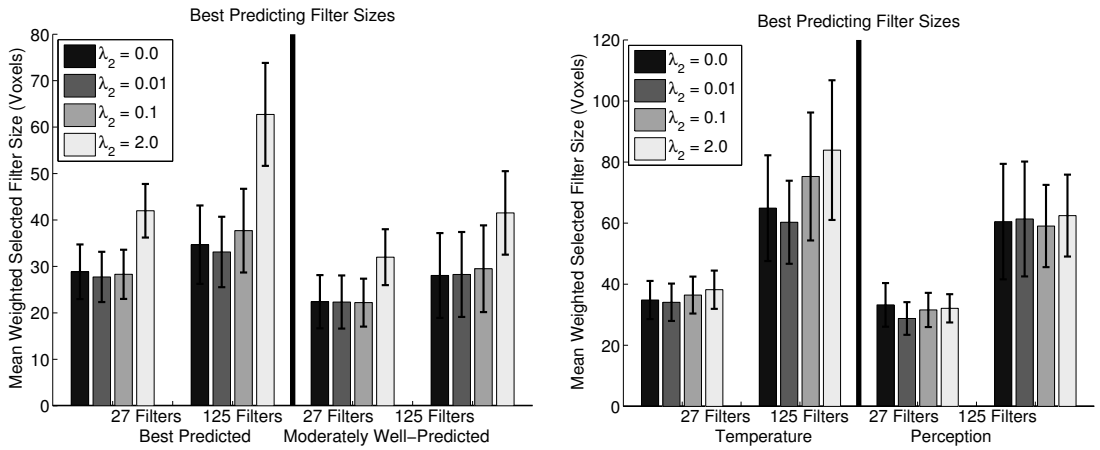
reliability indicates that these two properties alone are not sufficient for evaluating the "goodness" of a map.

4.3.3 Filter Size, Model Size, Spatial Distribution, and Prediction

In this section, the mean weighted selected filter size of filter-based models is considered; in doing so, several interesting properties of predictive fMRI models emerge. Figures 4.14(a) and 4.14(b) show that some relationships between selected filter size, filter set, and λ_2 value exist, but depend on the tasks being predicted. Figure 4.14(a) shows that, for both sets of PBAIC tasks, filter set size is similar regardless of whether 27 or 125 filter feature sets are used, but for both sets, there is a significant spike in mean selected filter size with a high λ_2 value of 2.0. This effect is consistent with the results in Figures 4.5 (b) and (c) on synthetic data, in which higher λ_2 values resulted in a greater selection of larger-sized filters, even though such filters were not truly relevant. Note that the expected filter sizes for the 27- and 125-filter sets for real data, if selection were random, would be 81 and 288 respectively. Thus, selection on these real data is not random, but there is still a bias toward larger filters with larger λ_2 values.

For Pain data, however, the reverse is true: Figure 4.14(b) shows filter size to differ depending on the filter set used, but not depending on λ_2. If all filter sizes in a set were truly relevant, one would expect the mean weighted selected filter size to be higher for the 125 filter set, given its much higher mean weight. For Pain data, this effect is what we observe, suggesting that the larger filter sizes of the 125 filter set are actually relevant for Pain data, while perhaps not for PBAIC data. These results would be consistent with the large improvements in reliability observed when blindly smoothing voxel-only Pain maps, as opposed to PBAIC maps. These higher true filter sizes would also make the mean selected filter sizes slightly less sensitive to the over-selection induced by higher λ_2 values, as observed here. Surprisingly, though, the reliability scores for the Pain data in Figure 4.12 f-g are not any higher for the 125 filter set than the 27 filter set, and prediction in Figure 4.11(c) is also not higher for the larger filter set, suggesting that, on the contrary, the selection of larger filters may not actually reflect recovery of the true underlying signal.

While Figures 4.14(a) and 4.14(b) do not reveal much consistent variability in selected filter size across modeling choices, i.e., methods or λ_2 values, Figure 4.15 shows that the

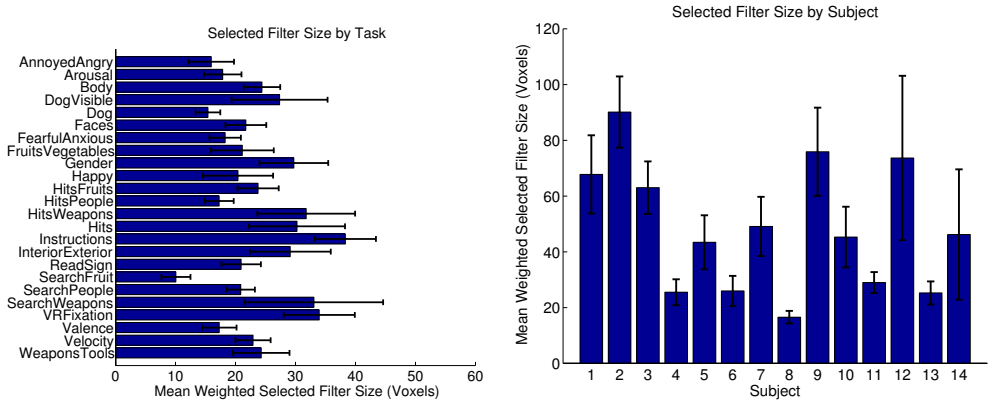


(a) PBAIC (b) Pain

Figure 4.14: *Higher λ_2 values increase the average selected filter size for PBAIC, but including larger filters does not; the reverse is true for Pain.* Mean filter size in number of voxels, weighted by model parameter weight, by filter set and task set, calculated over all 3 (PBAIC) or 14 (Pain) subjects, 3 (PBAIC) or 2 (Pain) tasks, and 4 cross-validation folds. Bars reflect 95% confidence.

variability is quite large across factors inherent to the data: PBAIC task or Pain subject. This result is consistent with results in previous chapters showing that the most variability in model evaluation metrics, prediction and reliability, occurs across PBAIC tasks, rather than being influenced by modeling decisions. Note that, while not shown here, there was not a significant direct relationship between the size of a subject's brain and his or her average weighted selected filter size, for either dataset.

Given the variability across PBAIC tasks or Pain subjects in prediction, reliability, and filter size, Figures 4.16(a) and (b) consider whether prediction is significantly associated with selected filter size across PBAIC tasks or Pain subjects. There is in fact a significant positive correlation between prediction performance and selected filter size for PBAIC tasks, though not for Pain subjects. This effect alone suggests that larger filter sizes are associated with less noise in the data. This result notably differs from the result in Chapter 3, in which, as greater amounts of smoothing were applied to beta weights, prediction began to degrade; however, the filters used here are all smaller than the 10mm largest smoothing filters used in those results, and filter-based methods involve smoothing of not just weights, but the data itself (in a predictively-guided fashion). Given the data smoothing that occurs, the observed correlation between prediction and filter size might seem obvious: greater amounts of data smoothing will reduce noise and enhance

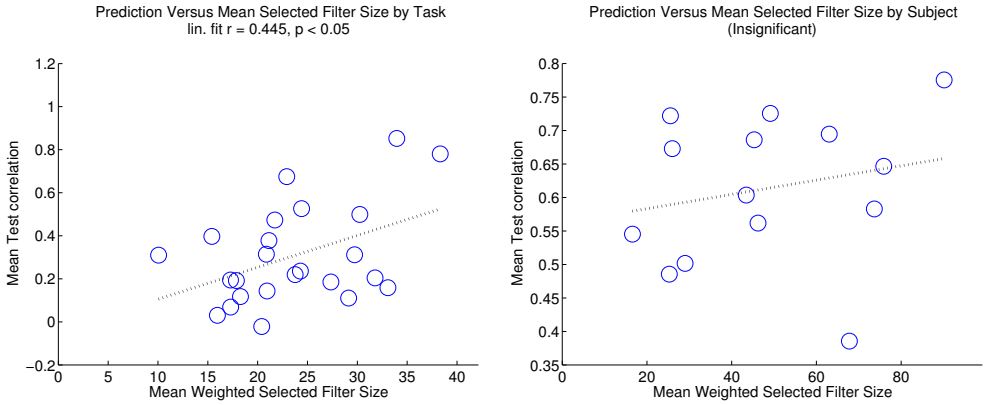


(a) PBAIC (b) Pain

Figure 4.15: *Selected filter sizes are highly variable by task (PBAIC) or subject (Pain).* Mean filter size in number of voxels, weighted by model parameter weight, by (a) task (PBAIC) or (b) subject (Pain), calculated over the 3 subjects (PBAIC) or 2 tasks (Pain), 2 filter sets (27 or 125), λ_2 values of $0.0, 0.01,$ and 0.1, and 4 cross-validation folds. Bars reflect 95% confidence.

prediction. However, recall in Figure 4.11 a-c that prediction performance over groups of similarly well-predicted tasks does not improve as filters are used, including larger filter sets. The results in Figures 4.16(a) and (b) are showing the inverse: better predicted tasks tend to lead to models consisting of larger selected filters, suggesting instead that tasks that are better predicted are associated with data with some properties, perhaps less noise, that allow selection of larger filter sizes, which may reflect the true underlying structure (incidentally, filter size was not found to be significantly associated with reliability, measured as map correlation spatial z-score, for either set of tasks).

Figure 4.16(c) and (d) consider another model property, model size, and its relationship to mean selected filter size. The range of values on the x-axis in this figure illustrate that this property is also highly variable across PBAIC tasks and Pain subjects. One might expect that smaller models would have made more use of larger filters; however, the results in Figure 4.16(c) contradict this expectation: given that the best fit trend is exponential, selected filter size is found to increase exponentially with model size for PBAIC tasks (though again, not for Pain subjects). Better predicted tasks therefore lead to models that incorporate a greater number of features, and make more use of larger filter-derived features.



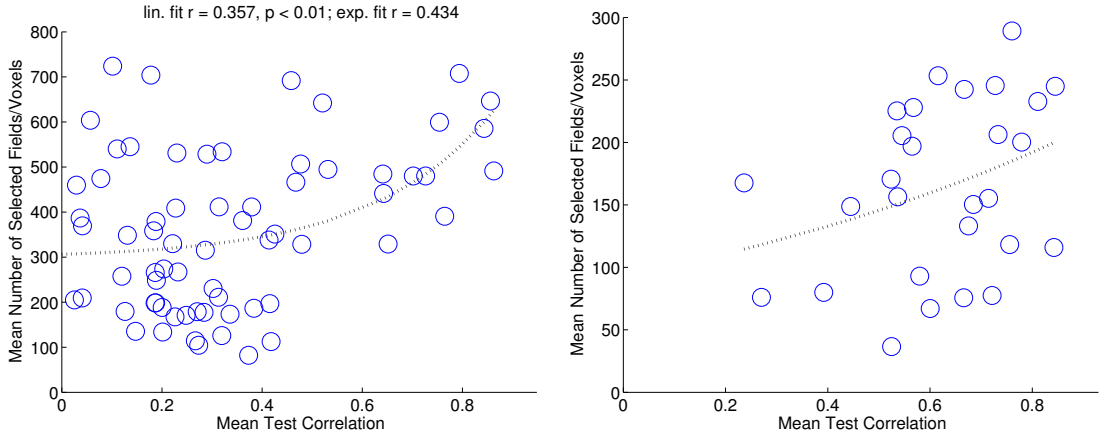
(a) Filter Size and Prediction: PBAIC

(b) Filter Size and Prediction: Pain

(c) Filter Size and Model Size: PBAIC

(d) Filter Size and Model Size: Pain

Figure 4.16: *Selected filter size is significantly associated with prediction performance by PBAIC task, but not Pain subject. Selected filter size is exponentially associated with PBAIC task model size, but not associated with Pain subject model size.* Mean filter size in number of voxels, weighted by model parameter weight, versus mean test correlation (a-b) or number of selected fields (c-d) for (a,c) PBAIC task or (b,d) Pain subject, calculated over the 3 subjects (PBAIC) or 2 tasks (Pain), 2 filter sets (27 or 125), λ_2 values of 0.0, 0.01, and 0.1, and 4 cross-validation folds (model size) or 2 runs (prediction). Trend lines are shown, along with model fit, if significant.



(a) PBAIC (b) Pain

Figure 4.17: *For PBAIC, prediction performance is exponentially associated with model size across tasks and subjects; there is a non-significant trend for Pain.* Mean test correlation versus mean number of selected fields, for all 24 (PBAIC) or 2 (Pain) tasks and 3 (PBAIC) or 14 (Pain) subjects, averaged over all 3 filter set sizes (voxels only, 1 filter, 27 filters, 125 filters) and λ_2 values.

Figure 4.17 in fact expands on the result in Figure 4.16 to show that prediction itself is associated with greater model size: better predicted tasks have models that incorporate a greater number of features, averaged over models produced using all filter sets and λ_2 values. This finding is somewhat unintuitive, if one supposes that more easily predicted tasks would require the incorporation of fewer features; however, recall that Chapter 2 showed that better predicted PBAIC tasks made more use of information distributed throughout the brain. Even if filters are used, incorporating distributed information requires inclusion of a greater number of features.

Given an interest in examining the possibility that filter-based models might also be incorporating more distributed information, along with the manipulation of spatial properties inherent in filter-based model learning, it seems natural to examine this quantification of spatial structure. Figure 4.18 shows how the spatial distribution metric varies by filter set for the PBAIC tasks. Recall that this metric is, to an extent, measuring the inverse clustering properties of the map weights, such that maps characterized by localized clusters will have a lower score. Not surprisingly, we find that filter-based models have a significantly lower score than voxel-only models, though the score is nearly identical between the two filter sets. This similarity across filter sets is consistent with Figure 4.15(a), which showed that the mean selected filter size was not significantly

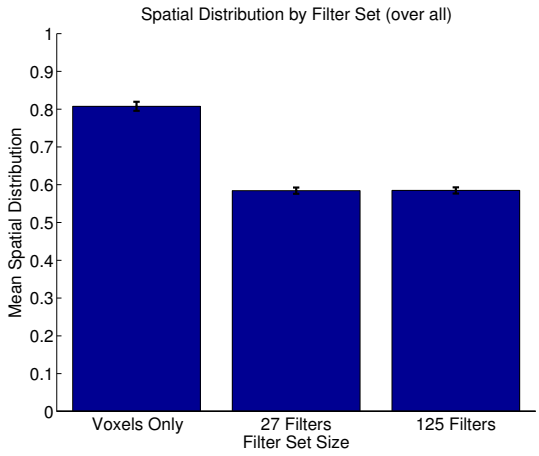
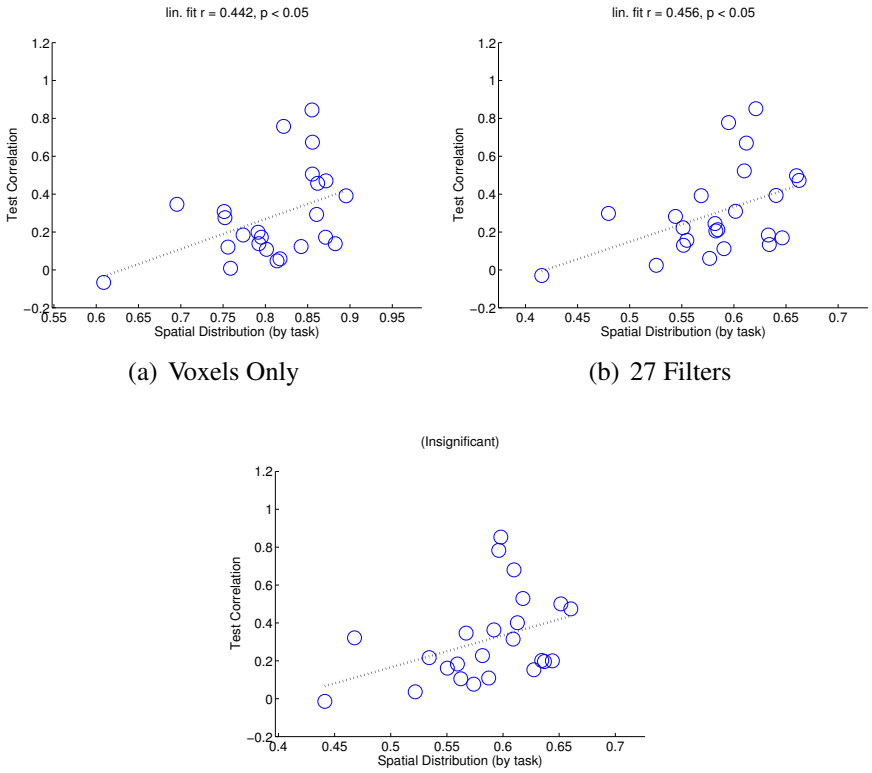


Figure 4.18: *PBAIC Models trained using filters exhibit less spatial distribution than maps learned with voxels only, but spatial distribution is not changed when larger filters are added.* Mean model spatial distribution score for PBAIC maps by filter set, averaged over all 24 tasks, 3 subjects, λ_2 values of 0.0, 0.01, and 0.1, and 4 cross-validation folds. Bars reflect 95% confidence.

higher with the larger filter set, suggesting the two filter sets should have similar spatial structure. The fact the scores are nearly identical, though, is somewhat surprising unless one considers that the spatial distribution metric requires arbitrary division of the brain map into bins of 27 neighboring voxels before entropy is computed. The 125 filter set likely includes slightly larger localized clusters, but clusters larger than 27 voxels are not counted as one cluster.

Recalling the previous association between prediction and spatial distribution, one may hypothesize that filter-based models, being less distributed, should be more poorly predicting, but recall from Figure 4.11 that prediction performance is equivalent across tasks. Instead, one may hypothesize that better predicted tasks yield models that more fully exploit the spatial structure in the filters, and hence exhibit lower spatial distribution; however this relationship was also not observed. In fact, Figure 4.19 shows that the finding in Chapter 2 is consistent across both voxel-only models and models that exhibit the same externally imposed spatial structure. Figure 4.19(a) again shows the positive correlation between spatial distribution and prediction across PBAIC tasks for voxel-only models (though numbers will be slightly different than in the previous work due to the different λ_2 values considered and different cross-validation approaches). Interestingly, Figure 4.19(b) shows that the same result is true *within 27-filter models*: while 27-filter



(a) Voxels Only

(b) 27 Filters

(c) 125 Filters

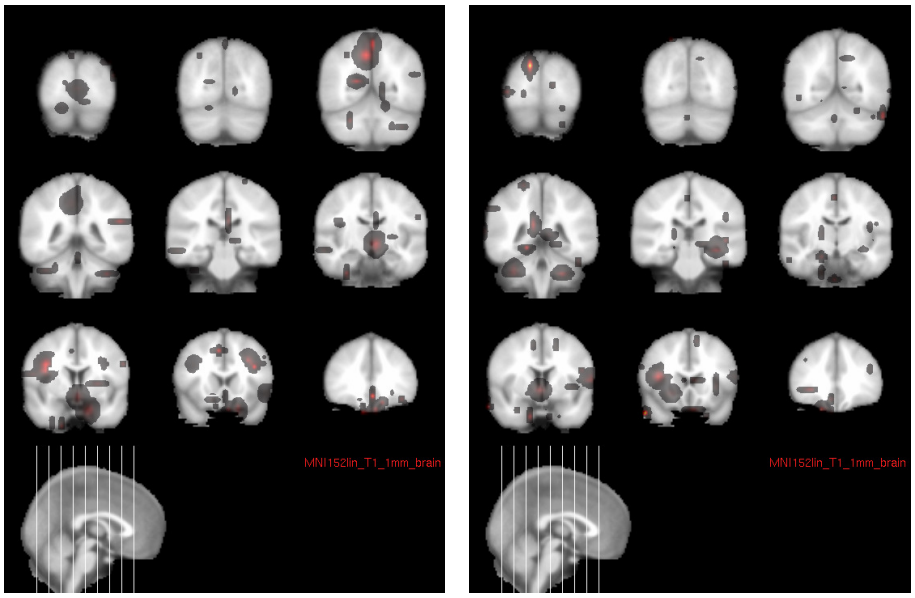
Figure 4.19: *Within each filter set used for training, better predicting models tend to exhibit greater spatial distribution (significant correlation for voxels only and 27 filters, trend for 125 filters).* Mean model spatial distribution score versus test correlation for PBAIC tasks, averaged over all 3 subjects, λ_2 values of 0.0, 0.01, and 0.1, and 4 cross-validation folds (spatial distribution) or 2 runs (test correlation). Trend lines are shown, along with model fit, if significant.

models overall have lower spatial distribution scores than the voxel-only models, more distributed 27-filter models tend to be better predicting. The same trend is evident for 125-filter models in Figure (c), but is not significant, which may be due to the 27-voxel bin confound described above. These findings therefore underscore the primary finding of Chapter 2 with regard to the spatial properties of sparse predictive fMRI models: the best predicting models are characterized by *distributed* clusters of *localized* activity.

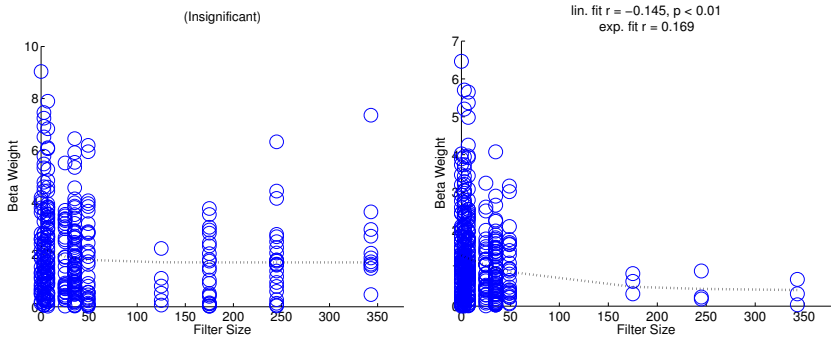
Finally, Figure 4.20 shows how these relationships between prediction, model size, filter size, and spatial distribution come together and may relate to inherent properties of the data, such as noise. Figures (a) and (b) show the maps produced using the 27 filter set for one cross-validation fold for the Pain Perception task for subjects 3 and 6. Figures

(c) and (d) show the distribution of weights for these two maps respectively. Each data point in these maps corresponds to a non-zero voxel weight. It is clear that subject 6 is using a much greater number of small filter features than subject 3. This difference is immediately apparent when quantified as mean weighted filter size in (e), which shows the average selected filter size to be significantly smaller for subject 6 than for subject 3. Figure 4.15(b) showed that these differences in filter size are consistent across all maps for these two subjects. The spatial distribution scores for these 2 maps, however, as shown in Figure (f), are about equal. Recall that the spatial distribution metric in effect measures the spatial entropy of the distribution of non-zero map weights throughout the brain map. These equal scores therefore imply that, while subject 3 is making more use of larger filter sizes, the overall spatial distribution of the selected features is similar between the two subjects, which indeed appears visually true in the two maps. Now consider the prediction and reliability scores for these two subjects, over all filters sets and λ_2 values. Figures 4.20(g) and (h) show that the mean prediction and reliability for these two subjects is highly similar, despite the striking difference in the types of filters selected; however, both prediction and reliability scores for subject 6 are much more *variable*. One reason that models for subject 6 may tend to incorporate smaller filter-based features is that that subject has noisier data, so the learned model can sometimes be over-fit to the training set; in those cases, prediction would be poor, and reliability may be high or low, depending on whether the small number of selected fields overlaps. In fact, while not shown here, subject 6 has a slightly lower average model size, suggesting that many irrelevant features were "pruned" through cross-validation.

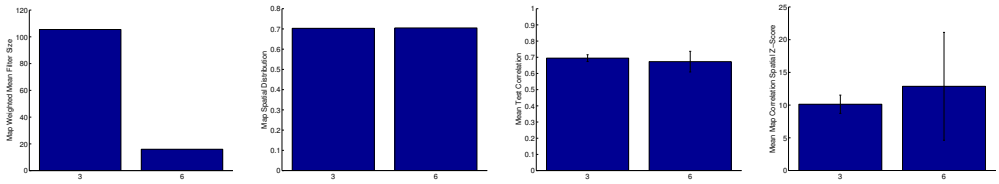
The relationship between these various properties may then be as follows: if there is less noise in the data, fewer irrelevant features will be selected, so optimized model sizes will be larger. Among noisy features, the divergence between training and test error will be magnified for features derived from larger filters, so they will be even less likely to be retained after cross-validation is applied. The noisy features are also more likely to be distributed randomly throughout the brain, so the cross-validated models will become less spatially distributed. Noisy models will tend to result in poorer prediction performance, so the better predicted models will tend to be less sparse and more spatially distributed, even when the features are in the form of large, localized clusters.



(a) Subject 3 27 Filters (b) Subject 6 27 Filters



(c) Weight by Filter Size: 3 (d) Weight by Filter Size: 6



(e) Filter Size (f) Spatial Distribution (g) Prediction (h) Reliability

Figure 4.20: *Inter-subject differences: maps have same spatial distribution, but Subject 3's makes much greater use of larger filters; prediction and reliability scores are similar, but Subject 3's scores are much less variable.* Maps learned for subject 3 (a) or 6 (b) run 1 cross-validation fold 1 Perception task with λ_2 of 0.1 on the 27 filter set; filter size versus absolute map weight for subjects 3 (c) and (d) (data point: 1 non-zero weighted voxel); (e): spatial distribution scores for maps (a) and (b); (f): mean weighted filter size for maps (a) and (b); (g) and (h): mean test correlation and spatially z-scored map correlation for subjects 3 and 6, over all filter sets and λ_2 values.

4.4 Discussion

Elastic Net is designed to facilitate inclusion of predictors that are correlated yet relevant. From the algorithm's point of view, if a feature is correlated with a highly predictive feature, it appears to be relevant; however, at least in the present application, "relevant" refers to only those fields that were present in the true underlying signal. This application therefore poses a challenge to this assumption, since there are many predictors that are correlated with the true "relevant" predictors, but should still be excluded. At the simplest level, all Gaussians centered at the same location will result in predictors that are correlated with each other. More troubling is that not all such correlated fields will truly be "irrelevant," or even incorrect, since "relevant" is ambiguous in this case. The diagonal-covariance Gaussians comprising candidate fields, by definition, are themselves linear sums of univariate Gaussians, which also serve as candidate fields. Therefore, a model that assigns appropriate non-zero linear weights to the 3 component fields of a truly "relevant" 3-dimensional field is equivalent to one that assigns non-zero weight only to the aggregate field; however, since we are seeking the most parsimonious model, we wish to only include the most informative field in our model.

Therefore, one major limitation of this approach is that *prediction* performance is being optimized by the model, but ideally we are seeking to optimize prediction, reliability, and parsimony simultaneously, and should ideally incorporate all of these constraints into our modeling procedure. One reason Gaussian filters may not provide a predictive advantage over other forms of "smoothing" is that, as shown in Chapter 3, prediction performance is affected very little by a small amount of smoothing, since blindly smoothed models still predict quite well. The smoothing parameters therefore are likely close to optimal, with little room for improvement in prediction due to more precise parameter tuning. In addition, the lack of impact from smoothing indicates that prediction performance from these models is not highly sensitive to additional spatial structure. These findings are consistent with those in Chapter 2 showing that prediction itself stayed fairly stable across methods, while other model properties, such as reliability, fluctuated. The small variability in prediction scores we observe also suggest that, while the filter-based approach seeks to find optimally predicting features, prediction performance already appears to be close to optimal. Since more "accurate" representations of the data do not have much effect on training error, there is no need to include the more accurate features.

Given both the strong auto-correlation among fields centered at the same location, and this confound of linear combinations of Gaussians, if we seek parsimony we must impose a harsh penalty on the selection of more than one field at a given location. The GroupX modification attempts to constrain models in exactly this way. The preliminary results using an ad hoc implementation of this approach demonstrate some potential for improving the selectivity and parsimony of the model, warranting further investigation; however, a simpler modification might be to enforce greater sparsity directly. We employed standard l_1-regularization, but it is known that l_q-regularization, when $q < 1$, results in even more sparse models, and, by definition, l_0 penalization would produce the most sparse models, as it directly minimizes the number of non-zero weights. While a tractable algorithm for solving l_0-regularization does not exist, algorithms for l_q-regularization exist for many values of $q < 1$. Such algorithms unfortunately do not usually provide the full regularization path, which is particularly useful in applications such as the present one, in which interpretability is important, yet such sparsity may eliminate some of the observed over-selection, warranting further investigation. A final alternative approach to enforcing such sparsity might involve pre-selecting candidate filters, perhaps by first performing an FFT on the data to determine a set of filters that are most likely to match the data, limiting candidates to at most one field per location. The High-Performance Computing approach made modeling from the full set of features *computationally* feasible, but the modeling itself may not yet be *algorithmically* feasible, perhaps justifying such a pre-selection strategy.

Ultimately, however, Gaussian filters may pose so many confounds that the Gaussian filter-sparse modeling approach may be impractical. The sparse modeling approach described in the present chapter was motivated in large part by the success of the Viola-Jones [93] face detection algorithm. One difference between that application and ours is the choice of learning algorithm (Boosting versus sparse regression), but more importantly, the spatial filters used by Viola and Jones were *wavelets*, rather than Gaussians. Penny et al. [71] also explored the use of wavelet-based features. Wavelets are appealing, since they would provide a *non-linear* transformation of the data and, if imposing the limitation of one field per location, would provide an orthogonal basis, eliminating correlation among predictors. A primary advantage of the feature generating, training, and prediction procedures we describe is that they are generic, so any spatial filter can be tested, including wavelets. Just as in the Viola-Jones work, wavelets may be most useful in a discriminative setting, since they reflect *differences* among adjacent voxels. Future

work should investigate wavelet filters in this framework, especially for classification of discrete mental tasks, such as determining if a subject is viewing a face or a house, or reading a sentence versus viewing a picture.

Despite these limitations, the approach in this form does offer the advantage that, regardless of the filter set used, spatially-based features will always encapsulate more information than raw voxel-only features. The synthetic data results highlighted one potential advantage for spatial features: smaller model sizes, requiring fewer training iterations. The LARS-EN algorithm involves some operations that can be exponential in the number of selected predictors; thus, runtime often increases exponentially as more training iterations are performed. Smaller model sizes therefore present a major advantage in speed, assuming one has the resources needed to process the large feature sets.

In addition, this approach also offers a potentially strong additional advantage still to be explored: facilitation of model-building across subjects. Like most predictive modeling techniques in the fMRI literature, this work trains models solely within subjects, due to the high sensitivity of voxel-scale models to cross-subject differences; however, there is considerable information to be exploited when combining information across subjects [81]. For multi-subject applications, some degree of smoothing or morphing is performed to reduce the inter-subject differences that exist at the scale of voxels. For such applications, the Gaussian filters used in this work present a strong advantage, in that they perform the necessary smoothing, and potentially do so in a way that optimizes prediction performance. Future work should explore whether mapping data from multiple subjects into the very high-dimensional over-complete representation and training models in the mapped space yield models exhibiting better prediction and reliability than models trained on individual subjects.

Chapter 5

Parallel LARS-EN

5.1 Introduction

Sparse regression is now routinely used to learn linear models when only a small subset of predictors are relevant to a problem. Existing algorithms for solving sparse regression problems generally assume that the entire dataset is stored in memory. In a number of applications, however, the feature space can be quite large and, in many such applications, most predictors will be irrelevant, making sparse methods a natural choice. Unfortunately, often such datasets can be too large to be stored in memory.

One of the most popular algorithms used to solve sparse regression problems is Least Angle Regression (LARS) [24], which is appealing because in one execution it returns the full *regularization path*, the solutions for incremental sparsity values. An extension of this algorithm, LARS-EN, [103] solves Elastic Net regression, which addresses some limitations of LARS yet reduces to LARS under one parameterization. These algorithms were designed with single-processor execution in mind; however, in this chapter, we describe a parallel implementation of the algorithm designed to handle datasets that exceed memory resources, especially those in which the number of predictors is very large and exceeds the number of available processors. The implementation utilizes a master-slave paradigm in which the predictors of a dataset are distributed across nodes. We use the Message Passing Interface (MPI), though any parallelization framework can be used.

The primary goal of this implementation is to enable training from datasets with too many predictors to be easily stored in memory; however, we also provide timing tests on synthetic data that demonstrate speed efficiency gains due to the parallelization. We conclude by briefly describing two potential general applications.

5.2 Notation

5.2.1 LARS-EN

This chapter is not a specification of the LARS or LARS-EN algorithm, but rather a summary of a parallel implementation. As such, most of the details of the implementation are in the work of Efron et al. [24], which specifies the LARS algorithm, and Zou and Hastie [103], which describes the LARS-EN modification to LARS. This work implements LARS-EN, but equations are take from both references, and the source is noted.

As in the previous chapters, we suppose a dataset with N observations and M predictors ([24] and [103] use slightly different notation), in which x_{ij} denotes the ith row and jth column of matrix $\mathbf{X}$. The response $\mathbf{y} = (y_1, \ldots, y_N)^T$ and the model matrix $\mathbf{X} = (\mathbf{x}_1 | \ldots | \mathbf{x}_M)$ where $\mathbf{x}_j = (x_{1j}, \ldots, x_{Nj})^T, j = 1, \ldots, M$, in which the response has been centered and the predictors standardized. We seek to learn linear regression coefficients $\beta = (\beta_1, \ldots, \beta_M)^T$.

Let $P = \{1, \ldots, M\}$. Herein, we will assume that the data $\mathbf{X}$ have been distributed across a set of nodes D of size K, such that each node $D_k, k = \{1, \ldots, K\}$ stores the data for predictors $Q_k \subseteq P$; for example, if $M = 1000$ and $K = 10$, a logical distribution will let $|Q_k| = 100, k = 1, \ldots, K$. In this case, node D_2 will store a subset of $\mathbf{X}$, specifically $\{\mathbf{x}_{101} | \ldots | \mathbf{x}_{200}\}$, and $Q_2 = \{101, \ldots, 200\}$. We will refer to the subset of $\mathbf{X}$ stored at node k as $\mathbf{X}_k$. Each node refers to its local predictors by indices $\{1, \ldots, |Q_k|\}$, but also maintains the lookup vector of absolute indices Q_k, which will be used to convert from local to absolute indices as needed. For notational clarity, we will refer to the lookup $Q_k(q)$ as $Q(k, q)$.

Note that the algorithm for distributing predictors is application-dependent and therefore not pertinent to the parallelization of the learning algorithm itself; however, we will

briefly describe example applications and corresponding example predictor distribution schemes.

This implementation uses a *Master-Slave* framework [23]: one node, D_0, is chosen to be the *Master* node; its role is to distribute computations to the remaining nodes (*Slaves*), aggregate results from the slaves, and perform centralized computations where needed. Note that D_0, in this implementation, can also serve as a slave, since the centralized computations require results from all slaves before they can be performed. The notation herein will assume D_0 is also a slave.

λ_2 refers to the Elastic Net grouping effect parameter. LARS-EN returns the same result as LARS when $\lambda_2 = 0.0$.

5.2.2 MPI

This implementation uses the Message Passing Interface (MPI) [60] for communication between nodes. When describing the communication patterns between nodes, some shorthand will be used to describe common MPI functions:

- **BROAD**(x) will refer to Master node D_0 broadcasting information x to all nodes.
- **SEND**(x,k) will refer to D_0 sending information x solely to slave node k.
- **RETURN**(k,x) will refer to node k sending information x to D_0.

5.3 Algorithm

The following implementation was derived from the Sjöstrand Matlab LARS-EN implementation [84].

5.3.1 Initialization

LARS-EN is an iterative algorithm, which essentially builds up a set of "active" (non-zero weighted) predictors in the model and their corresponding weights $\beta_{\mathcal{A}}$, where, at any given iteration, $\mathcal{A}$ is the set of indices corresponding to the active predictors, which are

those covariates with the greatest absolute current correlations with the residual vector. Increasing this set of active predictors essentially corresponds to decreasing the λ_1 Lasso sparsity penalty. On the first iteration, $\mathcal{A}$ is the empty set.

As will be seen, each node must also maintain a list of its own active predictors, i.e., the subset $A_k \subseteq \{1,\ldots,|Q_k|\}$ s.t. $Q(k,a) \in \mathcal{A}, \forall a \in A_k$. Only one copy of $\mathbf{y}$ is needed, so it will be stored on D_0, The vector $\hat{\mu} \in \mathbb{R}^N$ stores the prediction vector at the current iteration ([24, Eq. 1.2]). It is initialized to all zeros. The coefficient vector $\beta \in \mathbb{R}^M$ is initialized to all zeros.

5.3.2 Cholesky Factorization Design Choice

In LARS-EN, $\mathbf{X}$ is theoretically substituted with $\mathbf{X}^*$, an augmented dataset with $N+M$ observations and M predictors, which we will describe further. The matrix $\mathbf{X}^*_{\mathcal{A}} \in \mathbb{R}^{N+M\times M}$ is stored and the temporary result $\mathcal{G}_{\mathcal{A}} = \mathbf{X}^{*T}_{\mathcal{A}}\mathbf{X}^*_{\mathcal{A}}$ is inverted. As Zou and Hastie [103] describe, however, the sparsity of $\mathbf{X}^*$ can be exploited so that the full $\mathbf{X}^*$ (and $\mathbf{X}^*_{\mathcal{A}}$) never actually need to be computed or stored. This inversion step in particular can be accomplished by maintaining the *Cholesky Factorization* [39] $\mathbf{R}$ of $\mathcal{G}_{\mathcal{A}}$ (a real-valued triangular matrix) and updating or downdating as predictors are added to or removed from $\mathcal{A}$, respectively. Zou and Hastie note that, while a slight modification to the factorization is required for LARS-EN, the standard procedure detailed by Golub and Van Loan [39] can still be followed.

Regardless of the specific algorithm, the matrix $\mathbf{X}_{\mathcal{A}}$ is needed to compute the factorization. In a single-processor implementation, $\mathbf{X}_{\mathcal{A}}$ is easily obtained; however, recall that, in our implementation, $\mathbf{X}$ is distributed across all nodes. Thus, to obtain $\mathbf{X}_{\mathcal{A}}$, columns corresponding to $\mathcal{A}$ need to be pulled from their assigned nodes, $\mathcal{G}_{\mathcal{A}}$ must be computed, and $\mathbf{R}$ must be updated or downdated. There are at least two ways of implementing these steps in a distributed environment:

1. Make use of parallelism by distributing the Cholesky update and downdate steps, such that each node k performs calculations only for the sub-matrix $\mathbf{X}_{\mathbf{k}_{\mathcal{A}}} \in \mathbf{X}_k$, consisting of the concatenation of those columns of $\mathbf{X}_k$ corresponding to active predictors, and sends the intermediate results to D_0, which completes the steps and stores them in $\mathbf{R}$.

2. Maintain a copy of $\mathbf{X}_{\mathcal{A}}$ on D_0 by having the appropriate node send D_0 the data vector $\mathbf{x}_j$ as j is added to $\mathcal{A}$. Have D_0 perform the updates and downdates just as would be done on a single processor.

The advantages of the first approach are:

1. Not as much data is duplicated between D_0 and the other nodes, freeing space on D_0.

2. There is no need for the potentially time-consuming task of sending the full data vector $\mathbf{x}_j$ from a slave to D_0 on each iteration.

3. There are many parallel implementations of Cholesky Factorization described in the literature that could potentially be modified for use in this setting, in which the data are distributed, for example [41]. We have observed in practice that this factorization begins to dominate running time as the algorithm is run for more iterations and $|\mathcal{A}|$ thus increases, so parallelizing parts of the computation might dramatically impact performance.

The second approach, however, is simpler and exploits the fact that computation of $\mathbf{R}$ requires both $\mathbf{X}_{\mathcal{A}}$ and the new column $\mathbf{x}_j$ to be inserted. The centralized Cholesky Factorization can exploit this natural sub-division by having D_0 maintain $\mathbf{X}_{\mathcal{A}}$ and receive only the new $\mathbf{x}_j$ on each iteration. In addition, Eq. (5.10) shows that $\mathbf{X}_{\mathcal{A}}$ will also be required on a subsequent step, which, as we will see, is most naturally performed on D_0, thus also requiring access to $\mathbf{X}_{\mathcal{A}}$ on D_0.

In addition, the performance benefits of the first approach are not necessarily observed in practice. To test the first approach, a simple, non-optimized formulation for the distributed data in this setting was developed and implemented from scratch; however, even one update or downdate step in this implementation required multiple iterations of communication between a slave and the master, including the sending of large vectors of data. Furthermore, D_0 still needed to perform many centralized operations that resulted in bottlenecks. As such, the increases in time efficiency achieved through parallelization and not sending $\mathbf{x}_j$ were offset by the increased communication overhead; however, it is certainly possible that with a more efficient implementation, the communication overhead could be drastically reduced.

Exploration of the impact of parallelization of the Cholesky Factorization on this implementation could consume an entire paper. For simplicity, we describe the second, centralized, approach to computing this factorization. We will omit the specification of the Cholesky updates and downdates, since many implementations exist, which are well-described in the literature. Other than the update and downdate procedures, the two approaches are identical except where noted.

The Cholesky Factorization $\mathbf{R}$ is initialized to the 1×1 matrix 0.

5.3.3 Each Iteration: Overview

The LARS algorithm is an iterative piecewise coefficient update algorithm. Each iteration consists of a series of updates, defined by Efron et al. in [24], that are actually quite straightforward. We list them here in full so the motivation for certain design decisions can be established in advance, but we will elaborate on the updates as the implementation is discussed.

1. Compute the correlation of each predictor with the residual:

$$\hat{\mathbf{c}} = \mathbf{X}^T(\mathbf{y} - \hat{\mu}_{\mathcal{A}}) \tag{5.1}$$

2. Determine the maximally correlated predictor:

$$\hat{j} = \underset{j}{\operatorname{argmax}} \{|\hat{\mathbf{c}}_j|\}, j \notin \mathcal{A} \tag{5.2}$$

3. Determine the correlation with the residual of the maximally correlated predictor:

$$\hat{C} = \hat{\mathbf{c}}_{\hat{j}} \tag{5.3}$$

4. Add the maximally correlated predictor to the "active set:" [1]

$$\mathcal{A} = \mathcal{A} \cup \hat{j} \tag{5.4}$$

[1] Note that Eqs. (5.2) and (5.4) are essentially subsumed into one step in ([24, Eq. 2.9]), since the premise of LARS is that $\mathcal{A}$ automatically consists of those predictors that are (equally) maximally correlated with the current residual. For logistical reasons, however, we will maintain a data structure for $\mathcal{A}$ to which we add and, when necessary, remove predictors.

5. Determine the signs of the correlations of the predictors with the residual

$$s_j = \text{sign}\left(\hat{\mathbf{c}}_j\right), j \in \mathcal{A} \tag{5.5}$$

6. Add the data column for the maximally correlated predictor to the dataset of active predictors:

$$\mathbf{X}_{\mathcal{A}} = \left(\ldots s_j \mathbf{x}_j \ldots\right), j \in \mathcal{A} \tag{5.6}$$

7. Compute the covariance matrix of the dataset of active predictors:

$$\mathcal{G}_{\mathcal{A}} = \mathbf{X}_{\mathcal{A}}{}^T \mathbf{X}_{\mathcal{A}} \tag{5.7}$$

8. Compute the equiangular vector, the unit vector making equal angles with the dataset of active predictors:

$$\mathbf{A}_{\mathcal{A}} = \left(1_{\mathcal{A}}^T \mathcal{G}_{\mathcal{A}}{}^{-1} 1_{\mathcal{A}}\right)^{-1/2} \tag{5.8}$$

where $1_{\mathcal{A}}$ is a vector of 1's of length equaling $|\mathcal{A}|$, the size of $\mathcal{A}$.

$$\mathbf{w}_{\mathcal{A}} = \mathbf{A}_{\mathcal{A}} \mathcal{G}_{\mathcal{A}}{}^{-1} 1_{\mathcal{A}} \tag{5.9}$$

$$\mathbf{u}_{\mathcal{A}} = \mathbf{X}_{\mathcal{A}} \mathbf{w}_{\mathcal{A}} \tag{5.10}$$

9. Compute the correlation of the predictors with the equiangular vector

$$\mathbf{a} = \mathbf{X}^T \mathbf{u}_{\mathcal{A}} \tag{5.11}$$

10. Determine the magnitude of coefficient update required to make a new predictor equally correlated with the residual [2]:

$$\hat{\gamma} = \min_{j \notin \mathcal{A}} + \left\{ \frac{\hat{C} - \hat{\mathbf{c}}_j}{\mathbf{A}_{\mathcal{A}} - a_j}, \frac{\hat{C} + \hat{\mathbf{c}}_j}{\mathbf{A}_{\mathcal{A}} + a_j} \right\} \tag{5.12}$$

[2] Note that our definition again differs slightly from that of Efron et al., specifically ([24, Eq. 2.13]), due to our maintenance of a data structure for $\mathcal{A}$. The minimizing $\hat{j}$ in this case will be the predictor from among the *inactive* predictors that will be added to the active set in the next iteration. The original equation assumes $\hat{j}$ is already in $\mathcal{A}$.

where min+ indicates the minimum is taken over only positive components within each choice of j.

11. Update the residual:
$$\hat{\mu}_{\mathcal{A}} = \hat{\mu}_{\mathcal{A}} + \hat{\gamma}\mathbf{u}_{\mathcal{A}} \tag{5.13}$$

12. Update the coefficients:
$$\beta_{\mathcal{A}} = \beta_{\mathcal{A}} + \hat{\gamma}\mathbf{w}_{\mathcal{A}}^T \tag{5.14}$$

5.3.4 Each Iteration: Elastic Net Modifications

Zou and Hastie [103] discuss how LARS is easily modified to perform Elastic Net regression, by substituting the $\mathbf{X}$ with an augmented matrix $\mathbf{X}^*$ and $\mathbf{y}$ with an augmented vector $\mathbf{y}^*$. $\mathbf{X}^*$ is of size $(N+M)\times M$ and $\mathbf{y}^*$ is of size $(N+M)\times 1$. The augmented matrices are defined as follows:

$$\mathbf{X}^* = \frac{1}{\sqrt{1+\lambda_2}}\begin{pmatrix}\mathbf{X}\\ \sqrt{\lambda_2}\mathbf{I}\end{pmatrix} \tag{5.15}$$

$$\mathbf{y}^* = \begin{pmatrix}\mathbf{y}\\ \mathbf{0}\end{pmatrix} \tag{5.16}$$

However, creation of these full augmented matrices is not required. The properties of these augmented matrices allow the LARS steps to be computed nearly identically, with just a few modifications, two of which we describe as follows. We indicate the modified variables with an asterisk. First, Eq. (5.10) is modified:

$$\mathbf{u}_{\mathcal{A}}{}^* = \mathbf{X}_{\mathcal{A}}{}^*\mathbf{w}_{\mathcal{A}} \tag{5.17}$$

$$= \frac{1}{\sqrt{1+\lambda_2}}\begin{pmatrix}\mathbf{X}_{\mathcal{A}}\\ \sqrt{\lambda_2}\mathbf{I}\end{pmatrix}\mathbf{w}_{\mathcal{A}} \tag{5.18}$$

$$= \frac{1}{\sqrt{1+\lambda_2}}\begin{pmatrix}\mathbf{u}_{\mathcal{A}}\\ \sqrt{\lambda_2}\mathbf{w}_{\mathcal{A}}\end{pmatrix} \tag{5.19}$$

As will be evident in the next modification, this full vector $\mathbf{u}_{\mathcal{A}}{}^*$ is not required either. Instead we will define:

$$\tilde{\mathbf{u}_{\mathcal{A}}{}^*} = \frac{1}{\sqrt{1+\lambda_2}}\mathbf{u}_{\mathcal{A}}. \tag{5.20}$$

Eq. (5.11) is now modified as follows:

$$\mathbf{a}^* = \mathbf{X}^{*T}\mathbf{u}_{\mathcal{A}}{}^* \tag{5.21}$$

$$= \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda_2}\mathbf{I} \end{pmatrix}^T \frac{1}{\sqrt{1+\lambda_2}} \frac{1}{\sqrt{1+\lambda_2}} \begin{pmatrix} \mathbf{u}_{\mathcal{A}} \\ \sqrt{\lambda_2}\mathbf{w}_{\mathcal{A}} \end{pmatrix} \tag{5.22}$$

$$= \left(\frac{1}{1+\lambda_2}\right)\left(\mathbf{X}^T\mathbf{u}_{\mathcal{A}} + \lambda_2\mathbf{w}_{\mathcal{A}}\right) \tag{5.23}$$

$$= \left(\frac{1}{\sqrt{1+\lambda_2}}\right)\left(\mathbf{X}^T\tilde{\mathbf{u}_{\mathcal{A}}{}^*}\right) + \left(\frac{\lambda_2}{1+\lambda_2}\right)\mathbf{w}_{\mathcal{A}} \tag{5.24}$$

We will follow the LARS steps with the above LARS-EN modifications, making adjustments for our distributed implementation, but for simplicity, we will not use the asterisk. $\mathbf{u}_{\mathcal{A}}$ will refer to $\tilde{\mathbf{u}_{\mathcal{A}}{}^*}$ and $\mathbf{a}$ will refer to $\mathbf{a}^*$.

5.3.5 Each Iteration: Update Active Set

The first steps, essentially summarized in ([24, Eq. 2.9]), are to calculate the current residual from $\hat{\mu}$, determine the predictor $\hat{j}$, over all predictors $j \in P$, that is maximally correlated with the current residual, add $\hat{j}$ to $\mathcal{A}$, and store the correlation $\hat{\mathbf{c}}_{\hat{j}}$ as $\hat{C}$.

The residual $\mathbf{y} - \hat{\mu}_A$ can be easily computed by D_0, which stores the target vector $\mathbf{y}$. Let z store this residual:

$$\mathbf{z} = \mathbf{y} - \hat{\mu}_{\mathcal{A}} \tag{5.25}$$

$\hat{\mathbf{c}}$, however, results from a matrix vector operation involving $\mathbf{X}$ that uses the entire set of predictors, which are distributed across nodes. Fortunately, this operation factorizes, so the step can be easily distributed by having each node k compute a partial result $\hat{\mathbf{c}}_k$ for only its nodes. To do so, D_0 must first provide $\mathbf{z}$ to the slave nodes:

$$BROAD(\mathbf{z}) \tag{5.26}$$

Then, for each node k:

$$\hat{\mathbf{c}}_k = \mathbf{X}_k^T\mathbf{z}. \tag{5.27}$$

Since the step in Eq. (5.2) is the last time the full vector $\hat{\mathbf{c}}$ is used during the iteration, it suffices for each node k to compute its own $\hat{j}$ and $\hat{C}$ over its predictors $q \in Q_k, q \notin A_k$, and send the results to D_0. Given that we iterate here over q predictors, rather than the full M, we will call the local maximally correlated predictor $\hat{q}_k$. Each node computes these values and sends $\hat{C}_k$ to D_0.

For each node k:

$$\hat{q}_k = \underset{q}{\operatorname{argmax}} \{|\hat{\mathbf{c}}_q|\} \tag{5.28}$$

$$\hat{C}_k = \hat{\mathbf{c}}_{k\hat{q}_k} \tag{5.29}$$

$$RETURN\left(k, \hat{C}_k\right) \tag{5.30}$$

D_0 will then compute the overall maximum correlation $\hat{C}_0$ over all correlations $\hat{C}_k$. D_0 also needs to obtain the corresponding index $\hat{j}$, requiring the absolute index from the node $\hat{k}$ that stores $\hat{j}$. D_0 will determine and notify $\hat{k}$, which will then send $Q\left(\hat{k}, \hat{q}_{\hat{k}}\right)$ back to D_0 to be stored as $\hat{j}$.

$$\hat{k} = \underset{k}{\operatorname{argmax}} \{\hat{C}_k\} \tag{5.31}$$

$$\hat{C}_0 = \hat{C}_{\hat{k}} \tag{5.32}$$

$$SEND\left(1, \hat{k}\right) \tag{5.33}$$

$$RETURN\left(\hat{k}, Q\left(\hat{k}, \hat{q}_{\hat{k}}\right)\right) \tag{5.34}$$

D_0 will then add $\hat{j}$ to $\mathcal{A}$:

$$\hat{j} = Q\left(\hat{k}, \hat{q}_{\hat{k}}\right) \tag{5.35}$$

$$\mathcal{A} = \mathcal{A} \cup \hat{j} \tag{5.36}$$

$\hat{k}$ must also update its local list $A_{\hat{k}}$:

$$A_{\hat{k}} = A_{\hat{k}} \cup \hat{q}_{\hat{k}} \tag{5.37}$$

5.3.6 Each Iteration: Cholesky Factorization

We discuss here the steps needed to compute $\mathcal{G}_{\mathcal{A}}$, which requires $\mathbf{X}_{\mathcal{A}}$. As mentioned above, in the LARS-EN framework:

$$\mathcal{G}_{\mathcal{A}} = \mathbf{X}_{\mathcal{A}}^{*T}\mathbf{X}_{\mathcal{A}}^{*}. \tag{5.38}$$

For LARS-EN, this matrix would be of size $(N+M)\times(N+M)$, and, in subsequent steps, would need to be inverted, requiring significant computation; however Zou and Hastie [103] note that:

$$\mathcal{G}_{\mathcal{A}} = \frac{1}{1+\lambda_2}\left(\mathbf{X}_{\mathcal{A}}{}^{T}\mathbf{X}_{\mathcal{A}} + \lambda_2 I\right). \tag{5.39}$$

thus requiring storage only of $\mathbf{X}_{\mathcal{A}}$ rather than the full $\mathbf{X}_{\mathcal{A}}^{*}$ (our other LARS-EN modifications will achieve the scaling).

Furthermore, we can also avoid storing the large square matrix $\mathcal{G}_{\mathcal{A}}$ in Eq. (5.38), and computing its inverse, by instead storing its *Cholesky Factorization*. Specifically, from the definition of Cholesky Factorization [39]:

$$\mathbf{X}_{\mathcal{A}}{}^{T}\mathbf{X}_{\mathcal{A}} = \mathbf{R}\mathbf{R}^{T}. \tag{5.40}$$

The Cholesky Factorization $\mathbf{R}$ of $\mathcal{G}_{\mathcal{A}}$ can be computed by updating the factorization matrix on each iteration as new predictors are added (or downdating when predictors are removed). To update $\mathbf{R}$, D_0 needs matrix $\mathbf{X}_{\mathcal{A}}$ and vector $\mathbf{x}_{\hat{j}}$. Therefore, $\hat{k}$ will send data vector $\mathbf{x}_{\hat{j}}$ to D_0, which will then add it as a new column in $\mathbf{X}_{\mathcal{A}}$:

$$RETURN\left(\hat{k}, \mathbf{x}_{\hat{j}}\right) \tag{5.41}$$

D_0 performs:

$$\mathbf{X}_{\mathcal{A}} = \mathbf{X}_{\mathcal{A}} \cup \mathbf{x}_{\hat{j}} \tag{5.42}$$

A Cholesky update is then performed on $\mathbf{R}$ using $\mathbf{X}_{\mathcal{A}}$. Note that the signs in Eqs. (5.5) and (5.6) can be temporarily ignored when factorizing, and reincorporated in a subsequent step, which we will note. The details of the Cholesky Factorization are not trivial, yet are

beyond the scope of this chapter; however they are well-described in the literature. Any implementation will suffice, but minor additional scaling terms are needed for LARS-EN.

5.3.7 Each Iteration: Signs of the Correlations

Eq. (5.5) and subsequent updates require the signs $\mathbf{s}_{\mathcal{A}}$ of the correlations $\hat{\mathbf{c}}_{\mathcal{A}}$. As Efron et al. explain, these signs can change from iteration to iteration, which, in fact, differentiates the Lasso solution from stepwise linear regression. Thus, these signs must be recomputed on each iteration. Since each node k computed its own $\hat{\mathbf{c}}_k$ in Eq. (5.1), each node can and must on each iteration re-compute the signs for its active predictors A_k as follows:

For each node k:

$$\tilde{\mathbf{s}_k} = \text{sign}\left(\hat{\mathbf{c}}_{A_k}\right). \tag{5.43}$$

Each slave node now has its own local signs computed, but ultimately D_0 must obtain the full vector $s_{\mathcal{A}}$. Depending on the implementation, however, aggregating these signs into the full vector may not be trivial. In our implementation, $\mathcal{A}$ is stored on D_0, and $\mathcal{A}_{1\ldots A}$ indicate the absolute indices of $\mathbf{X}_{1\ldots A}$, where $A = |\mathcal{A}|$. Each slave k, meanwhile, maintains its own local indices A_k and the lookup table Q_k. This implementation reflects a particular trade-off among computation time and storage on the master and slave nodes. The slaves must send their signs $\tilde{\mathbf{s}_k}$ to D_0, but, depending on implementation details, the order in which they arrive may be completely different than the order in $\mathcal{A}$. Therefore, the signs received by D_0 must be re-ordered. To do so, we have each node k first send the vector of signs, and then send the absolute indices of the signs, and have D_0 sort the signs to match $\mathcal{A}$. The steps are as follows:

For each node k:

$$RETURN\left(k, \tilde{\mathbf{s}_k}\right) \tag{5.44}$$

$$\mathbf{Q}_{\mathbf{A_k}} = Q\left(k, A_k\right) \tag{5.45}$$

$$RETURN\left(k, \mathbf{Q}_{\mathbf{A_k}}\right) \tag{5.46}$$

D_0 then:

1. Aggregates vectors $\tilde{\mathbf{s}_k}$ into $\mathbf{s}_A$.
2. Aggregates vectors $\mathbf{Q}_{\mathbf{A_k}}$ into concatenated vector $\tilde{A}$.

3. Sort $\tilde{A}$ s.t. $\tilde{A} = \mathcal{A}$ and save the vector indices used to produce the sorted vector as IND.

4. Let $\mathbf{s}(\mathcal{A}) = \mathbf{s}_A(\text{IND})$.

5.3.8 Each Iteration: Coefficient Updates

At this point, D_0 can compute ${\mathcal{G}_\mathcal{A}}^{-1}$ from Eq. (5.7), using $\mathbf{R}$. We compute ${\mathcal{G}_\mathcal{A}}^{-1}$, taking into account the signs in Eq. (5.6) that we omitted when computing the Cholesky Factorization. Keep in mind that ${\mathbf{X}_\mathcal{A}}^T\mathbf{X}_\mathcal{A} = \mathbf{R}\mathbf{R}^T$, while Eq. (5.7) requires the signs $\mathbf{s}$, so $\mathbf{s}$ needs to be re-incorporated:

$$ {\mathcal{G}_\mathcal{A}}^{-1} = \mathbf{R}^{-1}\left((\mathbf{R}^T)^{-1}\mathbf{s}\right). \tag{5.47} $$

$\mathbf{A}_\mathcal{A}$ can now be computed by slightly modifying Eq. (5.8) to incorporate the signs. $\mathbf{w}_\mathcal{A}$ can be computed exactly as in Eq. (5.9):

$$ \mathbf{A}_\mathcal{A} = \left(1_\mathcal{A}^T {\mathcal{G}_\mathcal{A}}^{-1}\mathbf{s}\right)^{-1/2} \tag{5.48} $$

$$ \mathbf{w}_\mathcal{A} = \mathbf{A}_\mathcal{A}{\mathcal{G}_\mathcal{A}}^{-1}1_\mathcal{A} \tag{5.49} $$

At this point, we must compute $\tilde{\mathbf{u}_\mathcal{A}}^*$ and $\mathbf{a}^*$ as indicated in Eqs. (5.19), (5.20), and (5.24). Note again that for simplicity of notation, we will refer to the LARS-EN modified variables by the original LARS names of $\mathbf{u}_\mathcal{A}$ and $\mathbf{a}$ despite their modified calculation.

This section requires computation of several intermediate variables and we are faced with several choices regarding whether to centralize or distribute their computation. Our choice requires us to look ahead to Eq. (5.13). Since $\hat{\mu}$ is of size $N \times 1$ and thus does not factorize by predictors, and since its computation requires $\hat{\gamma}$, which is based on a maximum over all predictors, D_0 should compute $\hat{\mu}$. This computation requires $\mathbf{u}_\mathcal{A}$, so D_0 must either compute or receive $\mathbf{u}_\mathcal{A}$. Likewise, however, computation of $\mathbf{a}$ also requires $\mathbf{u}_\mathcal{A}$. Computing $\mathbf{a}$ necessitates computing the inner product of the full distributed vector $\mathbf{X}^T$ with $\mathbf{u}_\mathcal{A}$. Fortunately, like Eq. (5.27), this computation factorizes over predictors and thus can be distributed; however, each slave therefore also requires access to $\mathbf{u}_\mathcal{A}$, which Eq. (5.19) shows requires $\mathbf{X}_\mathcal{A}$. Therefore, there are two choices: compute $\mathbf{u}_\mathcal{A}$ on D_0

and broadcast the $N \times 1$ vector to each node, or have each node compute a partial $\mathbf{u}_{\mathcal{A}}$ using a partial matrix $\mathbf{X}_{\mathbf{k}_{\mathcal{A}}}$, and send each result back to D_0, which will sum the partial vectors into the final $\mathbf{u}_{\mathcal{A}}$. Both scenarios require similar amounts of communication and computation. The latter approach would be necessary if the Cholesky Factorization is distributed, and thus a centralized $\mathbf{X}_{\mathcal{A}}$ never computed. However, since in our approach $\mathbf{X}_{\mathcal{A}}$ is stored but $\mathbf{X}_{\mathbf{k}_{\mathcal{A}}}$ are not, the centralized $\mathbf{u}_{\mathcal{A}}$ computation is justified.

Regardless of how $\mathbf{u}_{\mathcal{A}}$ is computed, computation of the vector $\mathbf{a}$ must be distributed, which, from Eq. (5.24), requires each node to receive a copy of $\mathbf{w}_{\mathcal{A}}$, which is a vector of size $|\mathcal{A}| \times 1$. Fortunately, however, the computation in Eq. (5.24) factorizes, so each node k needs access only to a vector ω that stores the subset of $\mathbf{w}_{\mathcal{A}}$ corresponding to the active predictors in $\mathbf{Q}_{\mathbf{A}_\mathbf{k}}$. Thus, each node will receive a vector of size $|\mathbf{Q}_{\mathbf{A}_\mathbf{k}}| \times 1$ and $\sum_k |\mathbf{Q}_{\mathbf{A}_\mathbf{k}}| \leq \mathcal{A}$. Each node computes its subset of $\mathbf{a}$. These partial results could be sent back to D_0 to compute $\hat{\gamma}$; however, the next step, which is the only step that makes use of $\mathbf{a}$, is Eq. (5.12). In addition to the scalars $\hat{C}$ and $\mathbf{A}_{\mathcal{A}}$, it requires the full vector of correlations $\hat{\mathbf{c}}$. Recall from Eq. (5.31) that this vector is also stored only in a distributed form. Eq. (5.12) involves computing a minimum over all (inactive) predictors, and thus factorizes over predictors. Each node, therefore, has access to $\mathbf{a}_k$ and $\hat{\mathbf{c}}_k$, which are needed to compute the "score" for its predictors, while D_0 does not, so it is natural that computation of Eq. (5.12) be distributed across nodes, achieving perhaps a slight speedup in computation as well. Therefore, in addition to sending $\mathbf{w}_{\mathcal{A}}$ to each node, we must also send $\hat{C}$ and $\mathbf{A}_{\mathcal{A}}$, which are both scalars, adding a small amount of computational overhead. The return data, however, is only the partial minimum $\hat{\gamma}_k$, which is also a scalar. Since $k \ll M$, the total amount of data to be sent to D_0 is much less than if each node were to send back its full vector $\mathbf{a}$ to D_0.

We summarize the procedure as follows. First, D_0 computes $\mathbf{u}_{\mathcal{A}}$ as in Eq. (5.17). Then D_0 broadcasts, $\hat{C}$, $\mathbf{A}_{\mathcal{A}}$, and $\mathbf{u}_{\mathcal{A}}$ and sends each node k with active predictors the subset ω of $\mathbf{w}_{\mathcal{A}}$ corresponding to the active predictors in k:

$$BROAD\left(\hat{C}\right) \tag{5.50}$$

$$BROAD\left(\mathbf{A}_{\mathcal{A}}\right) \tag{5.51}$$

$$BROAD\left(\mathbf{u}_{\mathcal{A}}\right) \tag{5.52}$$

$$\tag{5.53}$$

For each node k where $|\mathbf{Q}_{\mathbf{A_k}}| > 0$:

$$\omega_k = \mathbf{w}_{\mathcal{A}}(j), j \in \mathbf{Q}_{\mathbf{A_k}} \tag{5.54}$$

$$SEND\left(\omega_k, k\right) \tag{5.55}$$

Each node now computes $\mathbf{a}$ for its data subset based on Eq. (5.21), along with its partial minimum $\hat{\gamma}_k$, computed as in Eq. (5.12). For each node k:

$$\mathbf{a} = \left(\frac{1}{\sqrt{1+\lambda_2}}\right)\left(\mathbf{X}_k^T \mathbf{u}_{\mathcal{A}}\right) + \left(\frac{\lambda_2}{1+\lambda_2}\right)\omega_k \tag{5.56}$$

$$\hat{\gamma}_k = \min_{j \notin \mathbf{Q}_{\mathbf{A_k}}}^{+} \left\{ \frac{\hat{C} - \hat{\mathbf{c}_{k,j}}}{\mathbf{A}_{\mathcal{A}} - a_k(j)}, \frac{\hat{C} + \hat{\mathbf{c}_{k,j}}}{\mathbf{A}_{\mathcal{A}} + a_k(j)} \right\} \tag{5.57}$$

where, as in ([24, Eq. 2.13]), "min+" indicates a minimum over only positive elements. Each node now sends its $\hat{\gamma}_k$ to D_0. For each node k:

$$RETURN\left(k, \hat{\gamma}_k\right) \tag{5.58}$$

Now, D_0 can determine the overall minimum $\hat{\gamma}$; however, for reasons that will become clear in the next section, we will store the result in temporary variable $\tilde{\gamma}$:

$$\tilde{\gamma} = \min_k \hat{\gamma}_k \tag{5.59}$$

5.3.9 Each Iteration: Dropping Predictors

As Efron et al. describe, to use LARS to solve for the Lasso (or LARS-EN for Elastic Net), one modification is needed. ([24, Eq. 3.1]) states the sign of a new β estimate

for a predictor must match the sign of its correlation with the residual. To enforce this constraint, ([24, Eq. 3.5]) states that the new estimate $\tilde{\gamma}$ must be less than the existing estimate $\hat{\gamma}$. If this condition is not met, the Lasso modification must be made, which requires that the predictor added during the current iteration be removed from the active set and $\hat{\mu}$ not be updated. Thus, in our distributed implementation, if the Lasso condition does not hold, we must update both $\mathcal{A}$ and $A_{\hat{k}}$, the latter of which requires notification of node $\hat{k}$, which can be accomplished by sending a yes or no signal:

$$(\tilde{\gamma} < \hat{\gamma}) \rightarrow \mathcal{A} = \mathcal{A} - \hat{j} \tag{5.60}$$

$$SIG = 0; (\tilde{\gamma} < \hat{\gamma}) \rightarrow SIG = -1 \tag{5.61}$$

$$SEND\left(SIG, \hat{k}\right) \tag{5.62}$$

where $\rightarrow$ denotes conditional execution. At node $\hat{k}$:

$$(SIG == -1) \rightarrow A_{\hat{k}} = A_{\hat{k}} - \hat{j} \tag{5.63}$$

where $==$ denotes equality evaluation. In addition, the Cholesky Factorization $\mathbf{R}$ must be downdated. We omit the details of this non-trivial but standard procedure.

Finally, if the Lasso condition was not met, we can update the vector of coefficients β and $\hat{\gamma}$:

$$(\tilde{\gamma} \geq \hat{\gamma}) \rightarrow \beta_{\mathcal{A}} = \beta_{\mathcal{A}} + \hat{\gamma}\mathbf{w}_{\mathcal{A}}^{T} \tag{5.64}$$

$$(\tilde{\gamma} \geq \hat{\gamma}) \rightarrow \hat{\gamma} = \tilde{\gamma} \tag{5.65}$$

5.3.10 After M Iterations

Per ([24, Eq. 2.21]), if the algorithm is run for M iterations, such that all M predictors are in $\mathcal{A}$, the least-squares solution is chosen:

$$\hat{\gamma} = \frac{\hat{C}}{\mathbf{A}_{\mathcal{A}}} \tag{5.66}$$

5.3.11 Values Returned

D_0 will return β, which may also be stored as a sparse vector $\beta_{\mathcal{A}}$, by also returning $\mathcal{A}$ as the set of associated indices. If the full regularization path is desired, copies of β and $\mathcal{A}$ should be stored at each iteration.

5.4 Timing Tests: Synthetic Data

5.4.1 Timing Tests: Introduction

Rather than computational speed-up, the primary purpose of this distributed implementation is to enable use of LARS-EN when the number of predictors is so large that the entire dataset cannot be stored in memory. Still, ideally the parallel implementation would also scale well, so that, as more predictors are added, the running time would remain manageable. Ideally, in fact, the running time would be within some constant factor of the time needed to process a proportionally smaller dataset on one node. Within the High-Performance Computing literature, the goal in such cases is good *weak scaling* [70], in which the problem size, M in our case, grows as the number of nodes D increases, with the sub-problem size on each node, Q_k in our case, remaining fixed as the problem size increases. An optimally efficient algorithm exhibits the same running time as the problem size increases, making it a useful measure of whether scaling larger problem sizes offsets the benefits of increased memory resources.

While weak scaling is arguably our goal, the second type of scaling often considered is *strong scaling*, in which the overall problem size is fixed, but the number of processors is increased, such that the sub-problem on each node becomes smaller. Strong scaling is often the desired goal for systems in which running time can be significantly increased through parallelism; the goal is to decrease the sub-problem size while keeping communication and related overhead minimal. Generally, according to Amdahl's Law [4], there is a limit at which communication overhead and code that must be run serially begin to dominate, and speedup gains diminish. It is generally more difficult to achieve good strong scaling than good weak scaling.

We find that our algorithm achieves good weak scaling, and decent strong scaling, up to a limit as expected, on a small toy dataset, suggesting it is useful both for the intended goal of increasing the feasible problem size, and potentially useful for achieving speedup through parallelization.

These results also underscore another key aspect of LARS-EN. LARS-EN is an iterative algorithm, in which the number of non-zero voxels generally grows with the number of iterations. The main benefit of this approach is that the model at each iteration can be evaluated using a cross-validation procedure to determine the appropriate level of sparsity. Theoretically, it would seem ideal to simply run the algorithm until all predictors are active, so all possible solutions can be considered when cross-validating. In practice, however, the running time increases significantly (likely approximately exponentially by observation) as the number of iterations increases, for two reasons:

1. The Cholesky Factorization running time is dependent on the size of $\mathbf{R}$, which is in turn dependent on the number of active predictors.

2. As the number of active predictors begins to far exceed some un-defined "optimal," the Lasso condition is met more often, resulting in frequent predictor dropping, and therefore high instability, requiring a much greater number of iterations to achieve the same number of non-zero coefficients.

Recall that LARS-EN can be used to find Elastic Net solutions, as well as the Lasso solution, when $\lambda_2 = 0.0$. We have observed that the second above phenomenon tends to be an issue mainly when solving for a Lasso solution on data with many correlated predictors, since the optimal solution will often have very few non-zero coefficients. Setting the target maximal number of active predictors much higher than the optimal number can significantly increase the required number of training iterations. In our timing tests, we perform runs solving for both Lasso and Elastic Net, and the results show that such an increase in the number of required iterations becomes the dominant factor in the increased running time.

5.4.2 Timing Tests: Methods

Data

The data were generated to have two characteristics for which Elastic Net was designed to be beneficial: $M \gg N$ and many predictors that are associated with the target variable yet correlated. Specifically, $N = 1000$, $M = 32,000$ and there were 500 pairs of correlated, relevant predictors. The generating procedure was the following:

1. Generate 500 normally-distributed time series τ of size 1000×1 with mean 0 and standard deviation 1.
2. Generate target variable y as a linear sum of τ and add to it Gaussian noise with mean 0 and standard deviation .01.
3. Create 1000 predictor time-series from 2 copies of τ and add Gaussian noise of mean 0 and standard deviation .01 to each
4. Generate $30,000$ additional normally-distributed predictor time-series with mean 0 and standard deviation 1.

Therefore, 1000 of the $32,000$ predictors are relevant and consist of 500 correlated pairs.

Testing

The algorithm was tested on these data by running on the IBM Blue Gene/L System with 512 MB per 700 Mhz node. Four testing parameters were varied:

1. Weak or strong scaling
2. Number of processors (D): $1,2,4,8,16,32$
3. $\lambda_2 = \{0.0, 2.0\}$ (Lasso or Elastic Net (EN), respectively)
4. Maximal active set size = $\{500, 1000\}$

For the maximal active set size, the algorithm was run until the number of active predictors $|\mathcal{A}|$ reached the target (500 or 1000). The number of iterations is thus proportional to this value, although, as we will see, not exactly the same.

Note that this data generation scheme may slightly confound the results, since the distribution of relevant predictors is not random, although the computation/communication trade-offs may provide balance.

Weak Scaling

For weak scaling, the objective is to increase M along with the number of processors. Thus, for the single processor case, a subset of the data was used for training, consisting only of the 1000 predictors that are correlated with the target variable. As more processors were added, subsets of the remaining (irrelevant) predictors were added while keeping the sub-problem size Q_k fixed as follows:

1. 1 processor: $M = 1000$ relevant predictors
2. 2 processors: $M = 2000$ (1000 irrelevant predictors)
3. 4 processors: $M = 4000$ (3000 irrelevant predictors)
4. 8 processors: $M = 8000$ (7000 irrelevant predictors)
5. 16 processors: $M = 16,000$ (15,000 irrelevant predictors)
6. 32 processors: $M = 32,000$ (31,000 irrelevant predictors)

Strong Scaling

For strong scaling, M is fixed at the full set of $32,000$ predictors, but the predictors are distributed evenly across the nodes. Thus, when the number of nodes $D = 1$, one processor will be "assigned" all $32,000$ predictors, whereas when $D = 32$, each node has only 1000 of the predictors; thus that case is identical to the $D = 32$ case for weak scaling. Regardless of the number of processors, the strong scaling problem is always the same, and thus the solutions and the number of required iterations will always be the same.

The runtime was measured as the wallclock time to complete one run of the LARS-EN algorithm, from initialization to return of the full regularization path given the maximal active set size. Only one run was performed for each set of running parameters.

5.4.3 Timing Tests: Results

Figure 5.1 shows how the runtime varies as the number of processors increases for weak scaling. Immediately evident is the difference in trends between Lasso trained until the active set size is 1000 and the other results. As Figure 5.1(b) shows, timing for EN, or for Lasso trained up to 500 active predictors, stays constant as the problem set size is increased except for minor fluctuations that would likely wash out with repeated runs. This is the desired weak scaling result, since the goal is to be able to increase the problem size without significantly impacting runtime. Clearly, however, Figure 5.1(a) shows that Lasso with 1000 active predictors behaves differently; in fact, the runtime is actually highest for the smallest problem size. For weak scaling, though, recall that the problem itself changes as the number of processors changes. While the target maximal set size remains fixed at 500 or 1000, the number of iterations required to achieve that set size may vary dramatically depending on the predictors. Figure 5.1(c), which plots number of iterations rather than runtime, demonstrates that, in fact, the observed behavior is almost entirely explained by the number of required iterations. The number of iterations is the dominant factor in runtime; controlling for the number of iterations, the algorithm essentially scales perfectly, within the range of parameters considered, as the problem size is increased.

The observed number of iterations is itself somewhat counter-intuitive, since it is highest when the problem size is smallest, and is only observed for Lasso, for the higher target active set size. The reason, however, becomes more clear when considering the structure of the data. Recall that there are 500 pairs of relevant but correlated predictors. For both Lasso and Elastic Net, we would expect the learned model to incorporate 500 of these 1000 predictors; indeed, the number of iterations when the active set size is 500 is approximately 500. When the target active set is 1000, however, the behavior between the algorithms will differ. Elastic Net is still likely to incorporate all 1000 relevant predictors, even though the additional 500 predictors are correlated with the predictors already included. Indeed, the number of iterations for Elastic Net with a target active set size of 1000 is approximately 1000. Lasso, however, will avoid including the additional 500 predictors that are correlated with predictors already in the model. The observed behavior is that Lasso will "struggle" to find a model of that size, and will become unstable, adding and dropping predictors until the final model size is achieved. Interestingly, the behavior is even more pronounced when only the 1000 predictors are

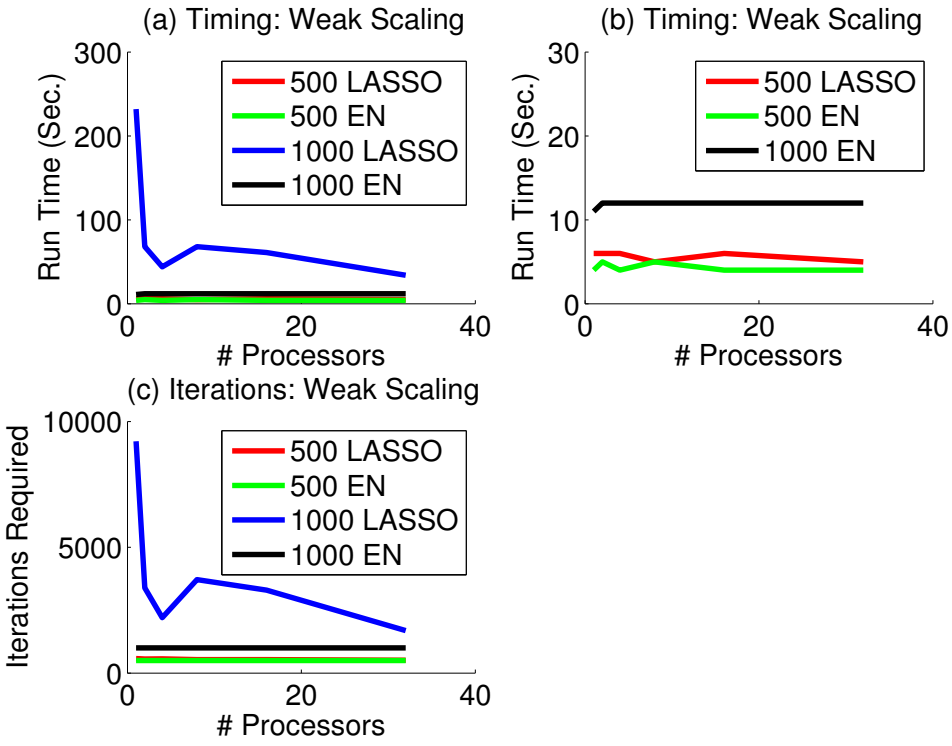


Figure 5.1: *Weak scaling results illustrate distribution with manageable overhead, with the number of iterations being the dominant runtime factor.* Processors always store 1000 predictors. (a) Number of processors versus wallclock running time, by maximal active set size and Lasso/EN. (b) Same graph as (a) but only for Lasso with active set size of 500, or EN. (c) Number of processors versus number of iterations, by maximal active set size and Lasso/EN.

included in the training set. The reason is that, most likely, when the other up to $31,000$ predictors are included, the algorithm finds some predictors that are spuriously correlated with the test signal, which it incorporates. With the smaller training set size, however, the algorithm is forced to try to include only correlated predictors, which it tries to avoid, creating even more instability. These results therefore also reveal properties of the algorithm itself, independent of the parallel implementation.

For strong scaling, the problem size remains fixed, in this case on the full $M = 32,000$ case; thus, the number of iterations for the 1000 active predictor Lasso run is relatively small. Figure 5.2 shows that strong scaling is close to optimal when the maximal active set size, and hence number of required iterations, is smaller, regardless of whether the target solution is Lasso or Elastic Net. As the number of processors doubles, the running

time is approximately halved, even up to 32 processors ($Q_k = 1000$). When the algorithm is run for Elastic Net until a larger active set size is achieved, the running time for each sub-division increases by an approximately constant factor until overhead begins appearing around 32 processors. When the algorithm is trained to a larger Elastic Net active set size, the running time increases by a linear factor, consistent with a fixed linear increase in the time to compute the Cholesky Factorization, along with other factors that scale with the size of the active set. The performance for Lasso with a larger active set size, however, illustrates both of the observed LARS-EN phenomena. First, again, the number of required iterations overall is greater for Lasso than for Elastic Net run up to the same active set size, leading to an observed linear factor increase compared to Elastic Net, for most values of D. As the number of processors approaches 32, however, overhead begins to mount and the optimal speedup is no longer achieved. Since the number of iterations is fixed, these results suggest that there is communication overhead limiting efficiency, which becomes more evident for problem sizes requiring more training iterations. Thus, experiments using larger maximal active set sizes for all methods might exhibit greater diminishing returns as the number of processors is increased; this limit imposed by overhead is expected though. In general, algorithms becomes less useful as the problem size per node becomes small relative to the number of processes.

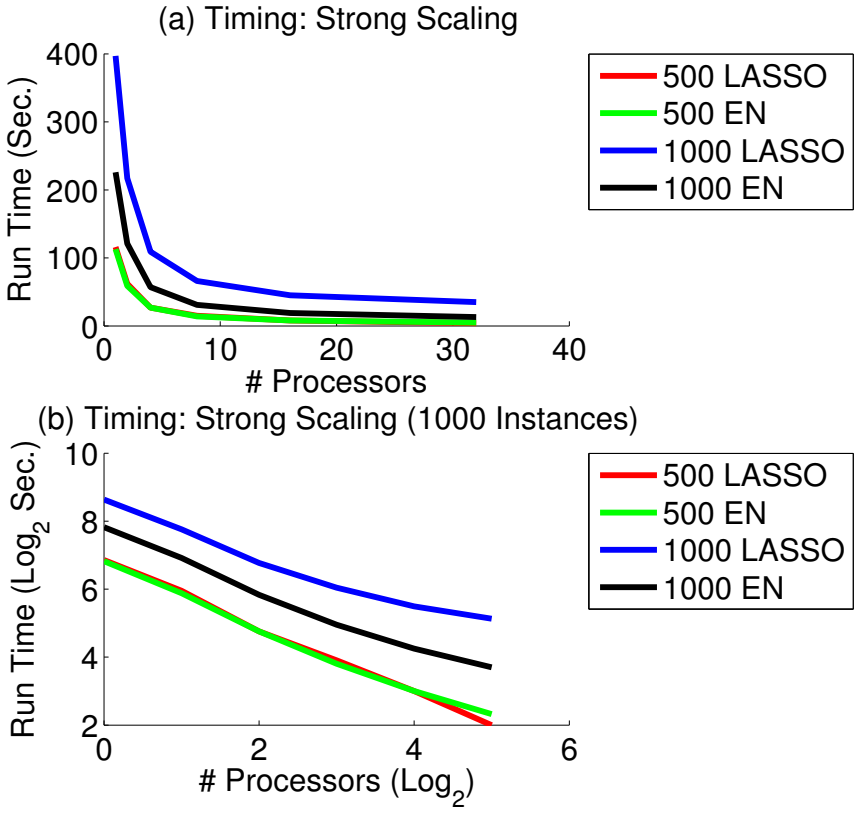


Figure 5.2: *Strong Scaling results illustrate potential for speedup through parallelization.* Total number of predictors is always 32,000. (a) Number of processors versus wallclock running time, by maximal active set size and Lasso/EN; (b) same graph as in (a) but with number of processors and runtime displayed in $\log_2$ scale.

5.5 Example Applications

In an increasing number of applications, especially those involving scientific data, the data contain a tremendously large set of predictors. Furthermore, modelers are increasingly finding that "raw" features and linear models do not suffice. One of the first approaches in such cases is to transform the features before building a linear model, which may "blow up" the feature space even further. In such cases, a distributed algorithm like LARS-EN may become essential. Two example scenarios are listed below, but countless other possibilities exist.

1. The relationship between predictors, which may be heterogeneous, and a target variable is known to be non-linear, but, perhaps for computational reasons, a linear model is desired. The original dataset might be augmented with predictors capturing interactions between predictors.

2. The raw data are known to reflect some true process that has been altered by a process or through the measurement method itself; for instance, the raw data correspond to an image in which a spatial point process has been convolved with a Gaussian filter, and measured at uniform, discrete intervals. If the parameterization of the spatial process is unknown, the true, predictive "features" might be recovered by convolving the raw image with a large set of candidate spatial filters. This framework was explored in Chapter 4, which highlighted the utility of Parallel LARS-EN in an applied setting such as fMRI analysis.

Chapter 6

Conclusion

6.1 A Summarizing Haiku

brain scan mind readers
capture the spatial structure
more robust models

6.2 Overview

Befitting its title, this work has explored predictive fMRI modeling with special focus on three related issues: sparsity, spatial structure, and reliability. To some extent, sparsity and spatial structure together constitute the issue of *data representation*. In this conclusion, we provide a high-level review of the findings in the general areas of data representation and reliability, along with suggestions for future work beyond those discussed in the preceding chapters. Finally, we take a broad look at the field of predictive fMRI modeling and speculate on future directions.

6.3 Data Representation

In our modeling, we are seeking the most parsimonious well-predicting model. A key prerequisite to model parsimony is to represent the fMRI data in such a way that the true structure of the brain activity signal is captured in the simplest form. In this work, we have emphasized the *spatial structure* of response. All of the methods discussed have converged on a spatial structure of the data characterized by distributed patterns of localized activity. It is well known that neural activity undergoes biological, physical, and computational transformations before being represented as voxel time-courses. Intuition alone suggests that, if we know that such transformation exists, we should attempt to incorporate any knowledge about the transformation into our model.

One approach that we used to address this spatial structure was to employ a domain-independent sparse modeling algorithm, Elastic Net, that addresses an observable aspect of the transformation, the auto-correlation among voxels. Elastic Net is in fact just one of seemingly countless Lasso variants that have emerged in recent years. As mentioned in Chapter 4, Group Lasso [100] takes a more direct approach to smoothing weights across clusters of predictors by accepting pre-defined groups of predictors as input and assigning the same, or similar, weight to all group members. A key difference between Elastic Net and Group Lasso is that in the Group Lasso, the clusters are hard and pre-defined. Both algorithms can be useful for modeling fMRI data, but the algorithm choice depends on the modeling assumptions. Our primary use of the algorithm is to capture the local spatial structure, which we know varies across subjects, tasks, and regions, but the parameterization of which we do not know. Therefore, clusters are likely to overlap and are not known in advance, both limiting the applicability of Group Lasso; however, if the clustering properties we sought were functional, perhaps among larger regions of related activity, the Group Lasso could be more applicable. For instance, some models built solely from *Regions of Interest (ROIs)* or more localized *Brodmann* functional areas have been shown to be useful for predicting mental states or subject groups [53]. Usually, such approaches involve averaging activity over all voxels in a region, but Group Lasso allows one to specify to the model which voxels are part of an ROI, while still learning from the full set of raw voxel time courses.

Regardless of the learning algorithm employed, it is clear that voxels themselves are arbitrary and therefore not the ideal features to be using. When domain knowledge about

"true" features is available, it is generally desirable to model within the space of the actual features. The filter-based approach attempts to do just that. Like Elastic Net, the filter-based approach allows spatial "groups" to overlap, and, ultimately, voxels in the resulting maps to have different weights even from neighboring voxels that are part of the same group; however, the filter-based approach also captures the grouping nature of the predictors by returning information about how voxels group together, specifically where clusters are centered and how large they are.

Even the approach of transforming the feature set spatially and building a sparse model is only one of countless approaches that can be used to address the spatial structure of the data in the model building, while ideally revealing insight into the nature of the spatial properties themselves. An approach that could build off the popular technique of employing spatial regularization is to incorporate spatial regularization into a *Supervised Dimensionality Reduction* approach [78, 72] which would learn predictive components so that the components are spatially localized. More closely related to the filter-based approach, rather than pre-computing a large feature set in which to build models, one might "grow" spatial features as part of the modeling. Our work in [13] is a preliminary exploration of such an approach.

Alternatively, another approach involves representing the data completely differently; instead of building a model in the space of spatial components, the model is built in the space of *spatial coordinates*. When we use Gaussian features in the filter-based approach, we are seeking to find Gaussian "blobs" in space that are associated with a particular task. These blobs correspond to a spatial spread parameterized as a Gaussian with a mean at a particular voxel, and a spread over voxels, with some variance. The distribution therefore is really over 3D spatial coordinates, and Xu et al. [99] in fact seek to learn a graphical model of such distributions.

An approach along these lines would be to build a mixed membership model of a distribution of activity in space, with functional "blobs" shared across mental tasks. If the model is viewed as a distribution over spatial coordinates, the distribution must correspond to some set of events occurring at those locations. In this case, the events would be abstract "units of activity" occurring at particular voxels. These "counts" of events are of course real-valued in this domain, complicating the use of inference procedures that count actual samples drawn from the distribution; however, in variational inference procedures, the real-valued counts can be used directly. Variational inference

procedures are also often more efficient for performing inference on large datasets, as with fMRI. As a mixed membership model, the model is analogous to Latent Dirichlet Allocation (LDA) [10] models for text, in which topics are shared across documents, with mental states corresponding to documents. The parallels with the LDA framework can also be exploited to incorporate developments in Supervised LDA [9], so that the learned topics are those that lead to the best prediction of real- or discrete-valued mental tasks. Such an approach would be predictive, interpretable, and allow great flexibility for modeling across hierarchical levels, such as subjects, experiments, or hierarchical task structures.

The issue of data representation goes well beyond capturing spatial features though. For instance, the field in general will likely make greater use of dynamic and non-linear representations. The present work did not consider temporal dynamics. At the simplest level, TRs, like voxels, are meaningless from a neuro-scientific point of view, since they too are an artifact of the scanning parameters, rather than corresponding to anything neurological or psychological; however, most MVPA techniques treat TRs as independent and identically distributed. At the most fundamental level, it is well-known that there is an *adaptation effect* in the BOLD response during intervals of repeated presentation of related stimuli, especially visual stimuli [47]. For instance, in some experiments, such as those of Haxby et al. [43], the classes of mental tasks correspond to categories of stimuli that are presented as blocks of images over a certain span of time. Usually, there are many TRs within one of these blocks. Single TR-based classifiers, typically used in the MVPA community, treat each TR as an example. We have experimented with building classifiers only on sub-blocks of TRs within each block, for instance using only the first or second 5 TRs in a block, and have observed prediction accuracy degrade as latency in the block increases. Some effort has been made to incorporate knowledge about the adaptation effect into GLM-based models [69], but MVPA models typically do not account for the effect. Doing so might improve prediction or reliability.

Not only might incorporation of temporal information boost the signal-to-noise ratio of the data, but it is very likely that meaningful, predictive patterns exist within the temporal domain. While fMRI does have much poorer temporal resolution than other neuro-imaging techniques, such as EEG, researchers have been able to learn some dynamic information from the data. For instance, Hidden Process Models [48] learn a dynamic pattern associated with latent tasks. One might imagine extending the spatial filter ap-

proach to learn non-linear feature transformations in both space and time simultaneously, perhaps using 4-dimensional wavelets. Temporal information can also be used to learn a completely different representation for the data as a network of activity. Cecchi et al. [15] represented data as a network reflecting patterns of correlation among voxels, and were able to discriminate schizophrenic from healthy subjects using the properties of the network. It seems quite plausible that new insights could be revealed by modeling in the space of non-linear features representing, for instance, relationships between voxels. The Parallel LARS-EN algorithm described herein could be used to perform searches in the resulting large feature spaces.

6.4 Reliability

As discussed in Chapter 1, prediction performance serves as an easily quantified measure of model validity, making it an appealing model evaluation metric; however, as discussed in this work and previous work such as that of Strother et al. [86], prediction may fail to tell the entire story, and another common metric, reliability, should be considered as well. In the context of learning a model from training data, there are rough parallels between these two metrics and bias and variance, respectively. Bias and variance are usually viewed as antagonistic and, indeed, previous examinations of prediction and reliability of fMRI models have generally considered the trade-offs between these metrics. Bias, however, is viewed relative to training data, whereas prediction captures generalization to test data; as a result, prediction and reliability are not necessarily antagonistic.

Indeed, the present work has emphasized that prediction and reliability are separate properties that are related but can be manipulated independently. Roughly, based on best fit trend lines, we have found that there is an overall exponential trend between the two properties, for understandable reasons. If a model exhibits poor prediction performance on held-out data, it is failing to capture a reliable pattern between the training and test sets; therefore, it is unlikely that it will be similar to an identically trained model on the other dataset, unless the model is overly simplistic. Indeed, Shalev-Shwartz et al. [83] recently built off the work of Mukherjee et al. [63] to demonstrate that the existence of a *stable* model of a problem, which exhibits a defined robustness in the face of removal of training instances, is both a necessary and sufficient condition for the *learnability*, or

existence of a consistent learning rule, for the problem. Clearly, therefore, one is unlikely to observe poor reliability but strong prediction, assuming a sufficiently complex model.

In this work, we described two factors that were shown to improve reliability without sacrificing prediction performance: the grouping effect of Elastic Net, and a small amount of spatial smoothing. In both these approaches, the improvement in reliability was an indirect consequence of a modification to the model weights; however, weights could also be learned specifically to optimize reliability more directly. Within the optimization community, the Robust Optimization [8] sub-discipline focuses on building models that are more robust to data perturbations, especially outliers or deviation from modeling assumptions. In contrast, within the statistical learning community, little attention has been devoted to enhancing robustness, or specifically reliability across data subsets, despite the vast space of potential approaches and promising utility. Recently, however, some algorithms have been introduced, most notably Stability Selection, which applies resampling methodology [40] with the specific goal of selecting those predictors that are most consistently predictive across data subsets. Stability Selection [58] and similarly motivated methods represent a logical next step in optimizing both prediction and reliability.

As emphasized in this work, however, reliability is not always as straightforwardly evaluated as prediction. The fundamental notion of *similarity* between brain maps is equivocally defined and, at an even higher level, the notion of reliability itself is not precisely defined, since its meaning is dependent on modeling goals. We focused in this work entirely on the question of reliability across repeated runs of the same experiment by the same subject on the same task in the same experimental session. There are, however, numerous other data subsets over which one might expect to find and measure model consistency, including subjects, sessions, and tasks. Slightly different approaches are likely needed to evaluate and improve reliability in these contexts.

Another complication is the lack of consensus regarding which patterns can actually be reliably recovered with fMRI. Inside and outside the community, there is debate over the true correspondence between the fMRI signal and underlying neural activity. Broadly, many experts feel the spatial and temporal resolution of fMRI limits the signal that can be recovered. Kamitani and Tong [50] demonstrated that multivariate methods might be capable of recovering signals beyond those assumed; however, the theoretical limits of fMRI are still poorly understood. Further complicating the relationship between brain

function and fMRI may be the highly redundant nature of brain information processing, revealed especially in Chapter 2 through the modest success of random feature selection and the importance of distributed information in models. Regardless of the data subsets considered, if it is unknown whether there is truly a recoverable reliable signal in the data, it can be difficult to determine whether poor reliability is due to limitations of the modeling technique or the data itself.

Beyond these difficulties in assessing reliability lies the challenge of fMRI model evaluation in general. The goal of this work, and the work upon which it builds, is to produce "better" brain maps from fMRI data. The first step in building such maps must be to precisely define characteristics brain maps should have. Prediction and reliability are appealing metrics because they are well-established criteria for evaluating data models in general, but, within the field, there is still great ambiguity about what makes a "good" brain map, or even what are the general goals of brain mapping. Do we seek a starting point for future explorations, or do we seek a comprehensive model? In some fields, such as physics, models are precise enough that the natural system can be replicated artificially. Do we seek to "reverse engineer" the brain? The field in general does not seem to have converged on a set of goals and questions, and often individual researchers are unclear about their own goals. In Section 6.5, we also discuss the need for greater input from scientists to clarify modeling goals. Without such guidance from the field and individual researchers, it will be impossible to determine appropriate model evaluation metrics.

6.5 Future Directions

fMRI as a technology is still relatively new, and only fairly recently have sophisticated statistical learning techniques been applied to the data. As with any new application area, the space of potential approaches has been vast. Much insight has been gleaned from the initial efforts; however, we are beginning to enter a stage of further refinement that may eventually lead to convergence on a unified paradigm for fMRI analysis. This section proposes that the greatest progress in the field will come through efforts directed in the following areas.

6.5.1 Generalization and Modeling Space

Modeling specific tasks engaged in by specific subjects is useful for gaining a preliminary sense of the broad issues involved in fMRI modeling, but ideally our models should say something general about how the brain processes information. One immediate goal is to build models that generalize more readily across individual subjects, requiring more sophisticated handling of the diverse sizes and shapes of brains. More broadly, models should also capture both similarities and differences among *groups* of subjects, perhaps across demographic or cultural boundaries. Most importantly, before true progress can be made in understanding brain function, models should generalize across mental tasks, to mental states not observed during model training. The studies by Mitchell et al. [62] and Kay et al. [52] mentioned in Chapter 1 accomplish such generalization, to an extent, by seeking the underlying representation basis for information, such as object relations or visual features.

Furthermore, just as we discussed the importance above of choosing an appropriate representation for the fMRI data, these pioneering studies suggest that in order to build sufficiently general models, it will be necessary to first converge on the appropriate representation for the input space. More broadly, the field may have reached the limits of what can be discovered through linear modeling of spatial maps, since the ideal representation of the brain data itself will surely depend on how the brain represents information. While these early studies suggest the promise of finding underlying "basis sets," it is likely that the true representation will be even more complex. The problem of finding this true representation is akin to that of finding a "genetic code" for the brain. Just as the cracking of the genetic code revealed a unifying paradigm for genetics research, the discovery of the "neural code" will precipitate the most significant advances in the field, making its discovery the "holy grail" of neuroscience.

Better understanding of the neural code underlying information processing in general should also resolve some of the ambiguity, mentioned above, regarding the limitations of predictive modeling. Researchers frequently observe that certain mental tasks are more difficult to predict than others. In fact, our PBAIC results and findings of related work suggest that the task being predicted is itself often a greater determinant of predictive performance than the modeling approach taken. Should we simply assume poorly predicted tasks to be "unmodelable" with fMRI? Such dismissiveness seems unsatisfying; ideally, we would instead adjust our modeling strategies. Doing so, however, requires a greater

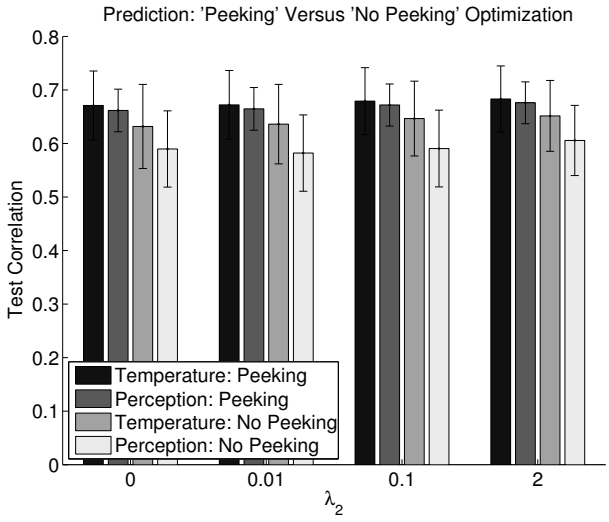


Figure 6.1: *Prediction performance can be over-estimated if evaluated on the same data used for model optimization.* Test correlation on the Pain Temperature and Perception tasks by Elastic Net λ_2 value, when the test set is used to optimize the number of selected voxels ("peeking") or when held-out optimization sets are used ("no peeking"); averaged over all 14 subjects and 2 runs.

understanding of the basis for such differences across tasks, which implies the need for a firmer understanding of the neural code.

6.5.2 Collaboration Between Modelers and Experimentalists

Any statistician will agree that applied statistics hinges on communication between the domain experts and the statistical experts. In this work, some specific issues have emerged in which greater coordination between the researchers designing the experiments and collecting the data, and those building models from the data (often the same person), would be beneficial. At the simplest level, such coordination can impact low-level design choices, such as having three runs in an experiment rather than two to facilitate cross-validation. While the general prediction trends in Chapter 2 were not significantly affected by the use of "peeking" by using only two runs, the specific numbers can often be inflated if peeking is used. For instance, consider Figure 6.1, which shows the difference in prediction performance for the two Pain tasks, depending on whether prediction is evaluated on the same dataset used to select the model size.

At the highest level, dialogue between experimentalists and modelers is necessary to clarify the modeling goals. When seeking "relevant" voxels, does the researcher wish to err on the side of inclusiveness or exclusivity? How specific should the model be? Is the researcher only seeking to determine which larger brain structures are most strongly associated with particular tasks? If so, multivariate voxel-level models may not provide a definite advantage over simpler GLM-based techniques. Experimentation and modeling must inform each other. Some labs have begun tightly integrating the two aspects, and facilitating cross-disciplinary communication, but considerable improvement is still warranted.

6.5.3 Real-Time Interface between Data Collection and Modeling

In addition to increased dialogue between the people involved in data collection and those involved in modeling, technology is beginning to facilitate a more interactive coordination between these two processes in general. Leading the way in this regard will likely be the emerging field of *real-time fMRI*, in which a model is used to make predictions, or perhaps even trained, online while the subject is being scanned. This output can then be used to provide feedback to the subject. Interest in this field is growing rapidly, not only because it presents challenging computational requirements, but because of its potential to enhance the modeling procedure.

One immediate benefit of such an interactive approach is that the subject can be informed about the "strength" of his neural signal and perhaps learn to modify it to improve prediction [54]. This feedback can engage the subject, and hence increase his or her motivation and attention. Researchers frequently find that images scanned using more attentive subjects exhibit a stronger signal, and we have personally observed instances in which the best predicted models were trained from data from especially motivated subjects, so improving motivation is itself a worthwhile goal.

Beyond this simple effect, there are almost limitless opportunities for creatively exploiting the interactive nature of real-time fMRI to provide a greater link between data collection and modeling. For instance, one might modify the experimental tasks based on which aspects of the task space are being learned poorly and require more data, drawing on advances in the *online learning* machine learning field. As computational hurdles

are overcome, real-time fMRI will continue to gain in popularity, and may well lead to fruitful new research paradigms.

6.5.4 Gaining Insight from the Models

Ultimately, the purpose of fMRI modeling, predictive or not, is to learn about how the brain works; therefore, the most interesting developments in the field will result from the application of these models to reveal insight into functioning, rather than enhancement of the techniques themselves. The Mitchell et al. [62] and Kay et al. [52] papers both exploited prediction as a measure of model validity to reveal likely brain encoding schemes. Another elegant application of predictive models is that of Polyn et al. [76] to demonstrate recurrence of patterns as information is recalled.

Other applications might reveal correspondences between internal and external processes. As an example, one dataset used for model evaluation in the present work captured subjects' response to pain. Recall that two aspects of the task were predicted: the actual temperature of the stimulus applied, and the subjects' rating of perceived pain. We can, for each subject, create a score that determines how accurately their perception corresponds to the real pain inflicted. Figure 6.2 shows that such a score is positively associated with how well the fMRI model for that subject was able to predict the actual temperature. In other words, the degree to which a signal in the brain is correlated with the strength of the pain stimulus is associated with how accurately the subject perceives the strength of the stimulus. Perhaps this brain responsiveness is directly leading to the more accurate perception, or perhaps some other hidden factors, such as overly high or low pain thresholds, or even attentional issues, might be influencing both the internal and external response. Also, it is interesting to note in Figure 6.1 that prediction performance for the actual temperature was not significantly different than that of the subjects' perception, but was very slightly better, likely reflecting the greater directness of the task.

In both this Pain example and the Polyn et al. [76] application, the ease with which a model is learned from the fMRI data is used as a proxy variable for the degree to which some state is reflected in the brain. The most interesting developments in the field will likely involve clever exploration within this paradigm.

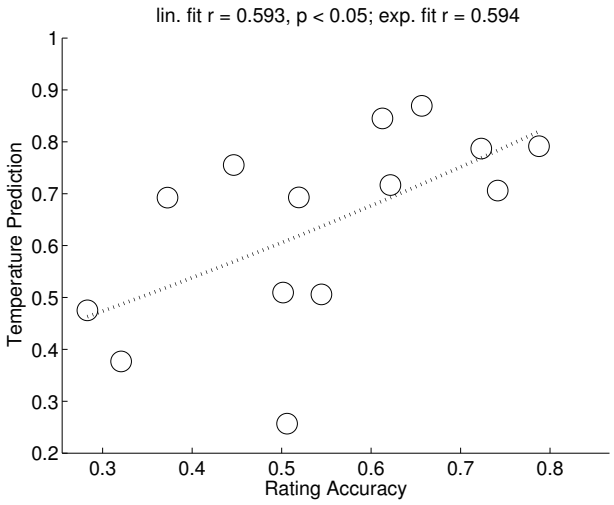


Figure 6.2: *Accuracy of subjects' pain rating is associated with accuracy of temperature prediction from fMRI.* Rating accuracy (correlation between perceived stimulus temperature and actual temperature) by 14 Pain subjects, versus accuracy of corresponding learned model predicting stimulus temperature from fMRI. Linear and exponential correlation and p-values are shown.

6.6 Final Thoughts

Building off these hypothesized directions, the key question going forward will likely be: What do we do with these tools? We began this thesis by speculating about a future in which "mind reading" techniques are employed practically, which is likely closer than many think. Psychology is a field concerned with understanding people. Through fMRI modeling, we are seeking a better understanding of the brain, and predictive techniques offer a very tangible way of doing so; however, history is filled with attempts to read minds, often for dubious or nefarious purposes. As with any technology, greater understanding and capability open doors to a wealth of possible uses. The prospect of understanding the brain and the mind naturally inspires awe and curiosity in people, yet better understanding of the mind implies better understanding of ourselves and each other, a prospect that can be as frightening as it is captivating. The developments in this field, no matter how oriented toward low level details, are all relevant to this high level objective, making it a pertinent, challenging, and ultimately exciting field in which to engage.

Appendix A

Supplementary Reliability Figures

A.1 Prediction, Reliability, and λ_2: All PBAIC Tasks

Using the cross-validation approach described in Chapter 3, we show in Figure A.1 that, for all 24 PBAIC tasks, prediction performance is not significantly affected by increasing λ_2 from 0.0 to 6.0. Chapter 2 did not show results for each individual task, and considered only a maximum λ_2 of 2.0.

In Figure A.2, we also expand the results in Chapter 2 by showing that reliability, measured as non-zero overlap, also increases across all tasks when λ_2 is increased from 0.0 to 6.0.

We show in Figure A.3 that although overlap scores are nowhere near the ideal score of 100%, weighted overlap scores for a given λ_2 value, in this case 2.0, are significantly higher, illustrating that the voxels deemed most relevant by the model are most likely to be consistently selected.

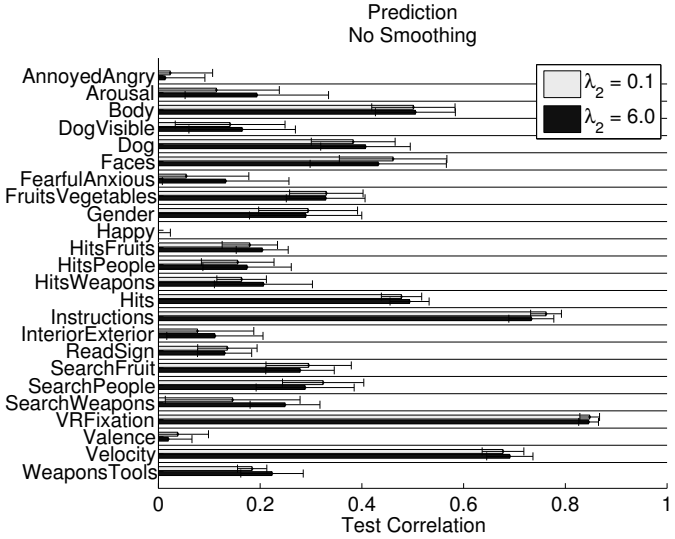


Figure A.1: *Prediction is not significantly affected by increasing* λ_2. Test correlation (cross-validated) for all PBAIC 2007 tasks for λ_2 values of 0.1 and 6.0, averaged over all 3 subjects and 2 runs. Bars reflect 95% confidence.

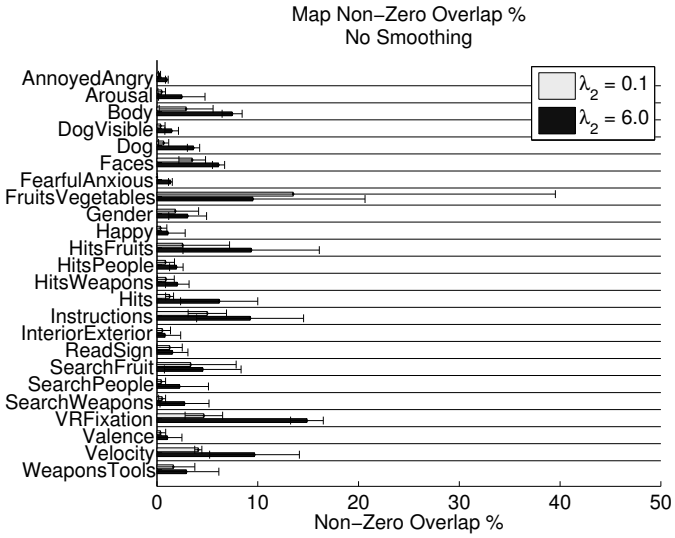


Figure A.2: *Reliability tends to increase significantly as* λ_2 *is increased.* Non-Zero Overlap % for all PBAIC 2007 tasks for λ_2 values of 0.1 and 6.0, averaged over all 3 subjects and 4 cross-validation folds. Bars reflect 95% confidence.

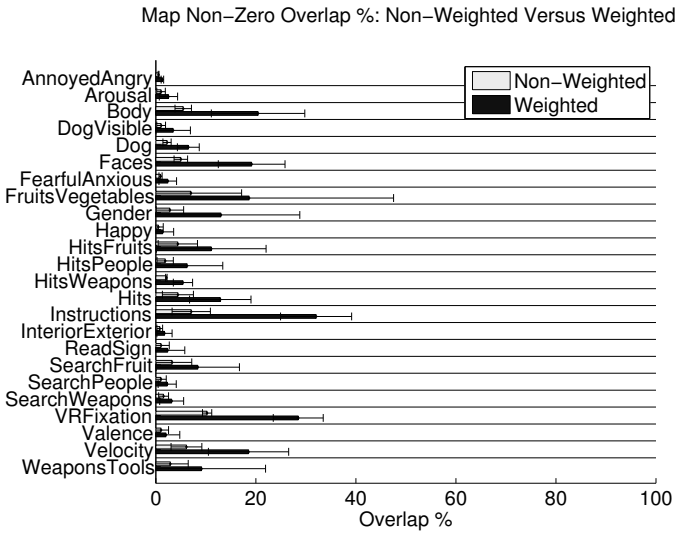
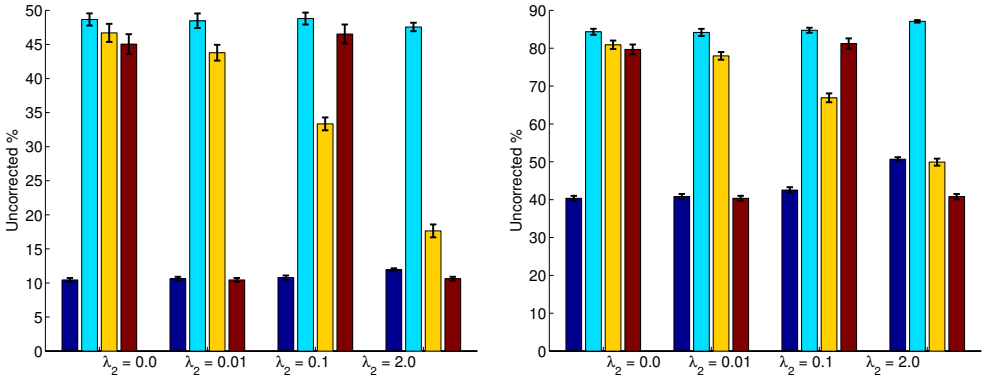


Figure A.3: *Weighted overlap is much higher than non-zero overlap; the most significant voxels are the most consistently selected.* Non-Zero and Weighted Overlap % for all PBAIC 2007 tasks for λ_2 value of 2.0, averaged over all 3 subjects and 4 cross-validation folds. Bars reflect 95% confidence.

Appendix B

Supplementary Filter-Based Figures

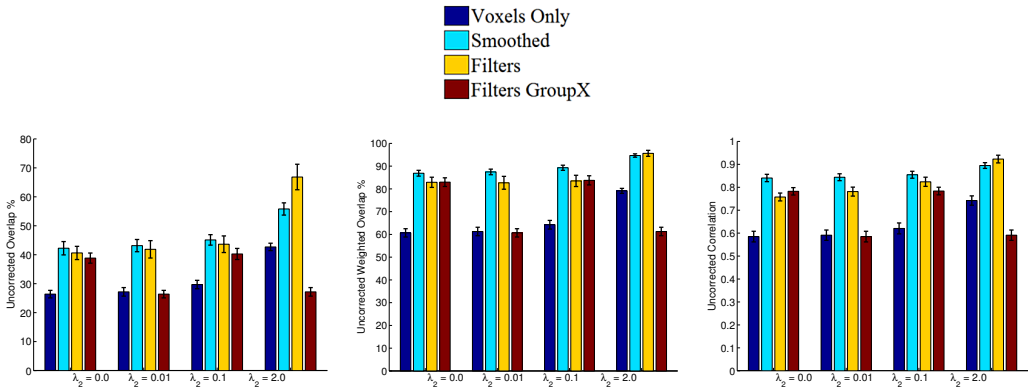
B.1 Filters: Uncorrected Scores and Reliability with Smoothing



(a) Ground Truth Non-Zero Overlap: Uncorrected

(b) Ground Truth Weighted Overlap: Uncorrected

Figure B.1: *Uncorrected ground truth overlap scores for synthetic data are shown.* Uncorrected non-zero overlap and weighted overlap between ground truth (template) map and cross-validated weight maps learned by method and λ_2 value, averaged over all 90 cross-validated datasets. Bars reflect 95% confidence.

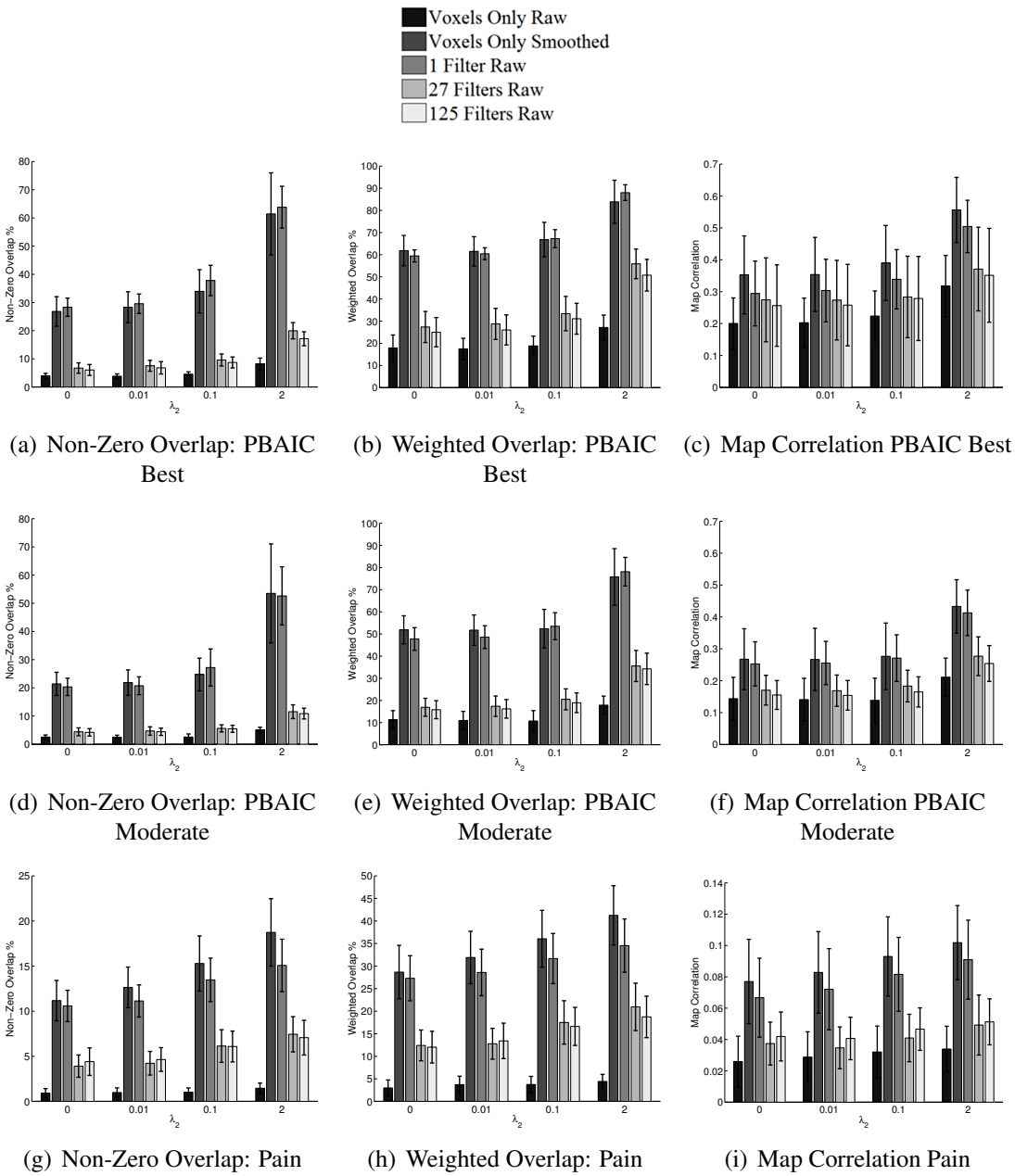


(a) Non-Zero Overlap: Uncorrected

(b) Weighted Overlap: Uncorrected

(c) Map Correlation: Uncorrected

Figure B.2: *Uncorrected reliability scores for synthetic data are shown.* Uncorrected non-zero overlap, weighted overlap, and map correlation, between cross-validated weight maps by method and λ_2 value, averaged over all 45 pairs of cross-validated models. Bars reflect 95% confidence.



(a) Non-Zero Overlap: PBAIC Best

(b) Weighted Overlap: PBAIC Best

(c) Map Correlation PBAIC Best

(d) Non-Zero Overlap: PBAIC Moderate

(e) Weighted Overlap: PBAIC Moderate

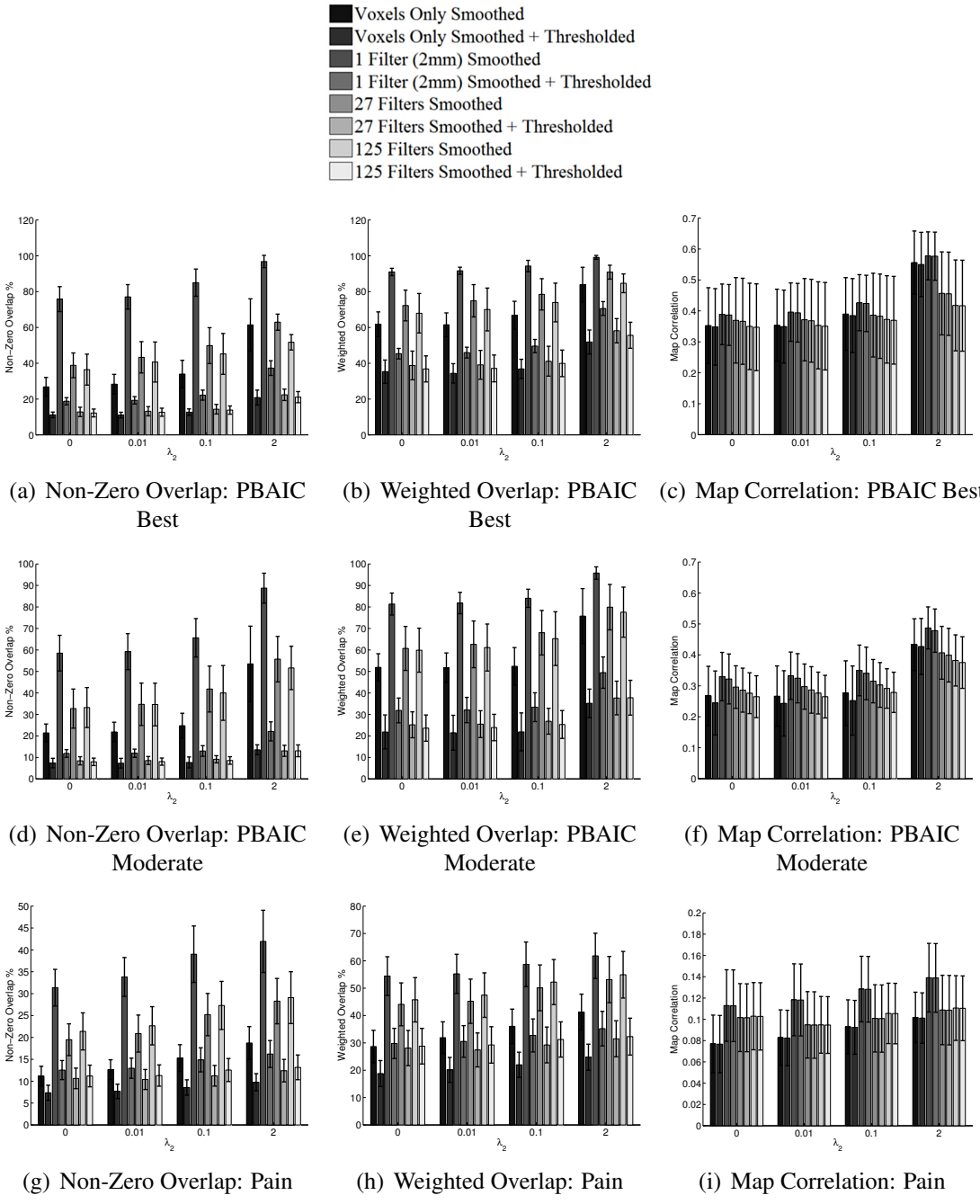
(f) Map Correlation PBAIC Moderate

(g) Non-Zero Overlap: Pain

(h) Weighted Overlap: Pain

(i) Map Correlation Pain

Figure B.3: ***Uncorrected*** *reliability scores are shown.* Non-zero overlap, Weighted overlap, and Map correlation for voxel-only, voxel-only smoothed, and 1, 27, and 125 filters; voxel-only smoothed is shown for comparison with 1 filter approach. Means are calculated over all 3 (PBAIC) or 14 (Pain) subjects, 3 (PBAIC) or 2 (Pain) tasks, and 4 cross-validation folds; variance over tasks and subjects. Bars reflect 95% confidence.



(a) Non-Zero Overlap: PBAIC Best

(b) Weighted Overlap: PBAIC Best

(c) Map Correlation: PBAIC Best

(d) Non-Zero Overlap: PBAIC Moderate

(e) Weighted Overlap: PBAIC Moderate

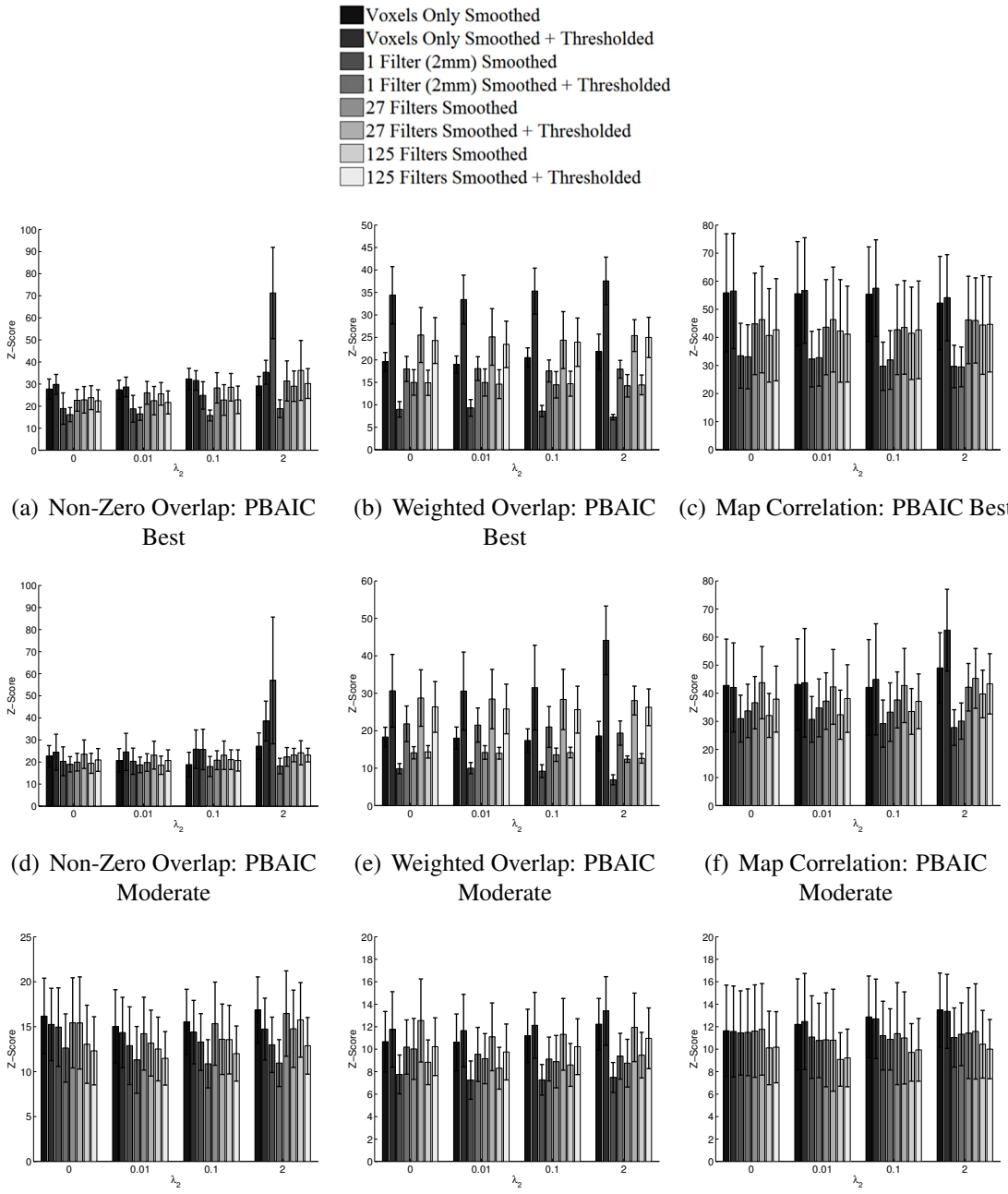
(f) Map Correlation: PBAIC Moderate

(g) Non-Zero Overlap: Pain

(h) Weighted Overlap: Pain

(i) Map Correlation: Pain

Figure B.4: ***Uncorrected*** *reliability scores are shown for* ***smoothed*** *maps.* Non-zero overlap, Weighted overlap, and Map Correlation reliability metrics for maps trained using voxels only, 1, 27, and 125 filters, smoothed and thresholded. Means are calculated over all 3 (PBAIC) or 14 (Pain) subjects, 3 (PBAIC) or 2 (Pain) tasks, and 4 cross-validation folds; variance over tasks and subjects. Bars reflect 95% confidence.



(a) Non-Zero Overlap: PBAIC Best

(b) Weighted Overlap: PBAIC Best

(c) Map Correlation: PBAIC Best

(d) Non-Zero Overlap: PBAIC Moderate

(e) Weighted Overlap: PBAIC Moderate

(f) Map Correlation: PBAIC Moderate

(g) Non-Zero Overlap: Pain

(h) Weighted Overlap: Pain

(i) Map Correlation: Pain

Figure B.5: *Smoothing filter-based maps does not significantly improve reliability compared to smoothing voxel-only maps.* Spatial z-scores for the non-zero overlap, weighted overlap, and map correlation reliability metrics for maps learned using voxels only, 1, 27, and 125 filters, smoothed and thresholded. Means are calculated over all 3 (PBAIC) or 14 (Pain) subjects, 3 (PBAIC) or 2 (Pain) tasks, and 4 cross-validation folds; variance over tasks and subjects. Bars reflect 95% confidence.

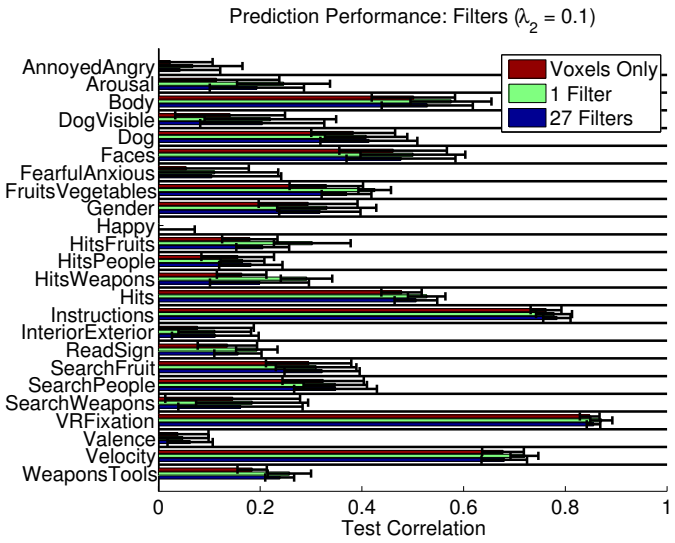


Figure B.6: *PBAIC prediction performance tends to be the same regardless of whether any filters are used.* Cross-validated test correlation by PBAIC task, for $\lambda_2 = 0.1$, averaged over all 3 subjects and 2 runs. Bars reflect 95% confidence.

B.2 Filters: All PBAIC Tasks

Figures B.6-B.8 show how prediction performance, model size, and reliability properties compare across PBAIC tasks, by examining these properties for voxel-only, 1-filter smoothed, and 27-filter-based models. Overall, the finding is similar to that of Chapter 3: the greatest differences are between tasks, rather than methods. Figure B.6 shows very similar prediction scores across methods within tasks, and Figure B.7 shows that differences in model sizes between methods tend not to be significant.

Figure B.8 shows that the observed trend for filter-based maps (whether uniformly or selectively smoothed) to be at least as reliable as voxel-only maps holds for all PBAIC tasks. In most cases, 1-filter and 27-filter reliability is similar; the exception is the 3 fruit-related tasks. These tasks were difficult to predict and, as shown in Figure B.7, the resulting cross-validated models were very small (likely due to high noise). It is likely that the 1-filter smoothing applied to these maps inflated the spatial structure of the maps to such an extent that they were over-penalized. It is interesting, though, that all fruit task maps exhibited the same relative properties.

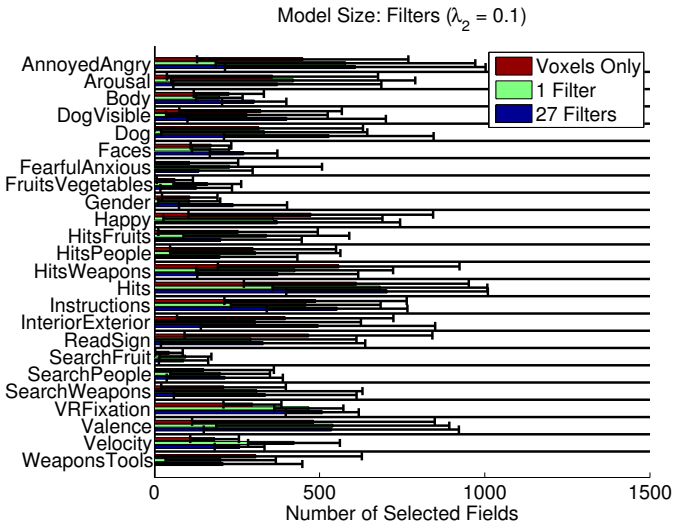


Figure B.7: *PBAIC model size tends to be the same regardless of whether any filters are used.* Number of fields selected using cross-validation, by PBAIC task, for $\lambda_2 = 0.1$, averaged over all 3 subjects and 4 cross-validation folds; Bars reflect 95% confidence.

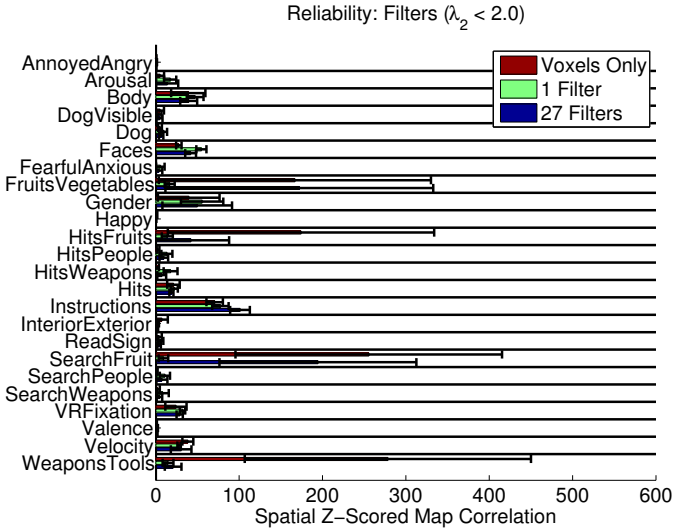
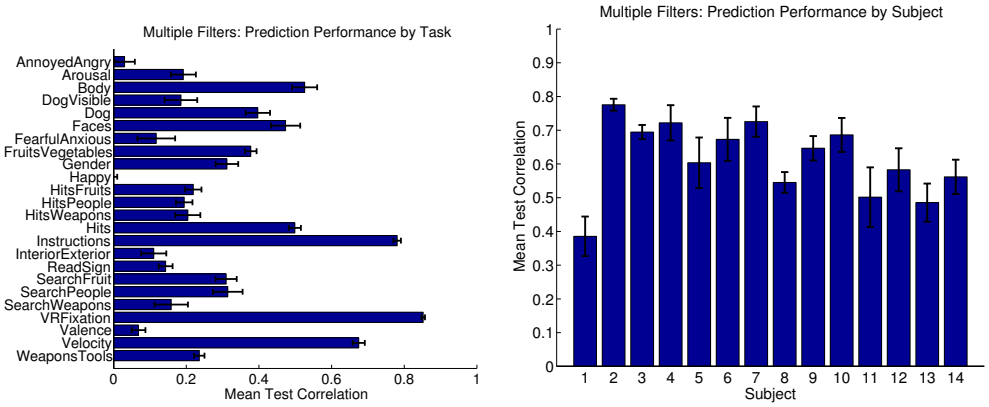


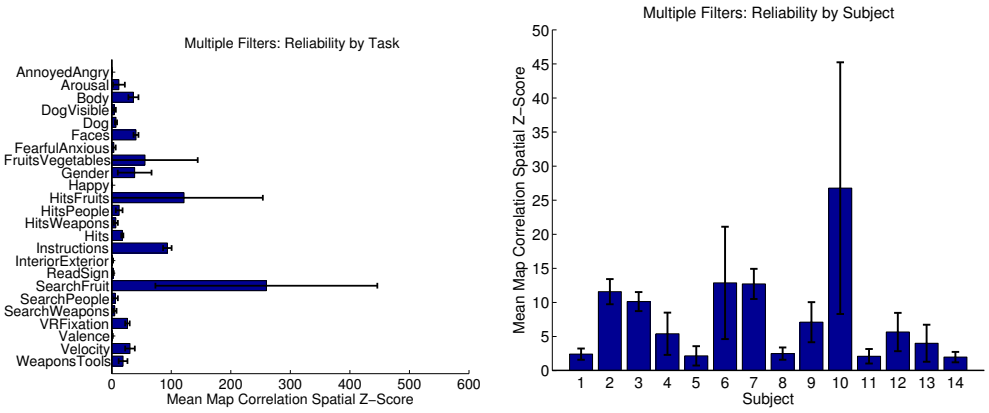
Figure B.8: *PBAIC maps learned with smoothing, whether uniform or selective, tend to be at least as reliable as maps trained without filters (with aberrant exceptions).* Spatial z-scored map correlation of cross-validated maps, by PBAIC task, for λ_2 values of 0.0, 0.01, and 0.1, averaged over all 3 subjects and 3 λ_2 values; Bars reflect 95% confidence.



(a) PBAIC (b) Pain

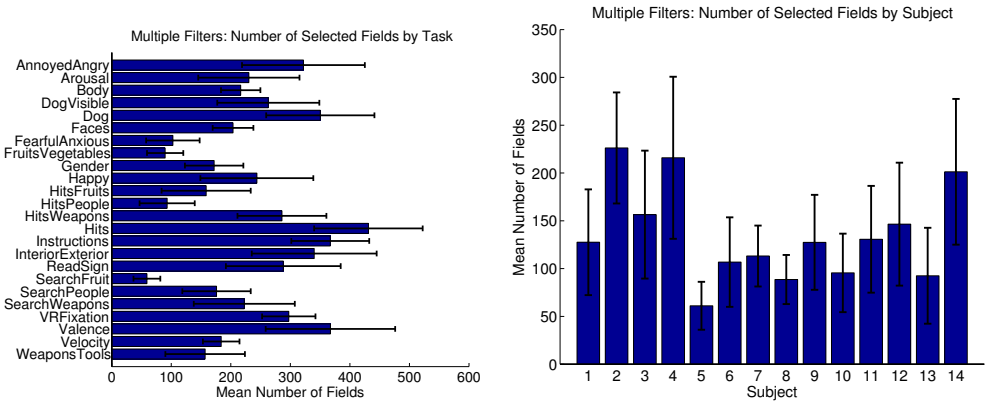
Figure B.9: *Average prediction performance by task (PBAIC) or subject (Pain) is shown.* Mean test correlation by (a) task (PBAIC) or (b) subject (Pain), calculated over the 2 filter sets (27 or 125 filters), λ_2 values of 0.0, 0.01, and 0.1, and 2 runs. Bars reflect 95% confidence.

B.3 Metrics for All Tasks or Subjects



(a) PBAIC (b) Pain

Figure B.10: *Average reliability by task (PBAIC) or subject (Pain) is shown.* Mean map correlation spatial z-score by (a) task (PBAIC) or (b) subject (Pain). Means are calculated over the 2 filter sets (27 or 125 filters) and λ_2 values of 0.0, 0.01, and 0.1, and 4 cross-validation folds; variance over filter sets and λ_2 values. Bars reflect 95% confidence.



(a) PBAIC (b) Pain

Figure B.11: *Average model size by task (PBAIC) or subject (Pain) is shown.* Mean number of selected fields by (a) task (PBAIC) or (b) subject (Pain), calculated over the 2 filter sets (27 or 125 filters), λ_2 values of 0.0, 0.01, and 0.1, and 4 cross-validation folds. Bars reflect 95% confidence.

Bibliography

[1] H. Abdi. Bonferroni and sidak corrections for multiple comparisons. In N. J. Salkind, editor, *Encyclopedia of Measurement and Statistics*. Sage, 2007. 6

[2] R. J. Adler. *The Geometry of Random Fields*. SIAM, 2009. 7, 98

[3] AFNI. AFNI Homepage: http://afni.nimh.nih.gov/afni/. 4

[4] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *AFIPS '67 (Spring): Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485, New York, NY, USA, 1967. ACM. 164

[5] M. N. Baliki, P. Y. Geha, and A. V. Apkarian. Parsing pain perception between nociceptive representation and magnitude estimation. *J Neurophysiology*, 101:875–887, 2009. 56

[6] R. G. Baraniuk. Compressive sensing [lecture notes]. *Signal Processing Magazine, IEEE*, 24(4):118–121, August 2007. 16

[7] A. Battle, G. Chechik, and D. Koller. Temporal and cross-subject probabilistic models for fmri prediction tasks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 121–128. MIT Press, Cambridge, MA, 2007. 52

[8] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, October 2009. 117, 178

[9] D. Blei and J. McAuliffe. Supervised topic models. *Advances in Neural Information Processing Systems (NIPS)*, 2007. 176

[10] D. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003, 2001. 176

[11] V. D. Calhoun, T. Adali, L. K. Hansen, J. Larsen, and J. J. Pekar. ICA of Functional MRI Data: An Overview. In *4th international symposium on Independent Component Analysis and Blind Signal Separation (ICA2003)*, 2003. 24

[12] M. Carroll, G. Cecchi, I. Rish, R. Garg, and A. Rao. Prediction and interpretation of distributed neural activity with sparse models. *Neuroimage*, 44(1):112–122, 2009. 26, 32

[13] M. Carroll and M. Dudík. Feature induction on fmri images using regularized logistic regression. Presented at 13th Annual Meeting of the Organization for Human Brain Mapping (OHBM), 2007. 175

[14] M. Carroll, K. Norman, J. Haxby, and R. Schapire. Exploiting spatial information to improve fmri pattern classificatio. Presented at 12th Annual Meeting of the Organization for Human Brain Mapping (OHBM), 2006. 100

[15] G. Cecchi, I. Rish, B. Thyreau, B. Thirion, M. Plaze, M.-L. Paillere-Martinot, C. Martelli, J.-L. Martinot, and J.-B. Poline. Discriminative network models of schizophrenia. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 252–260. Curran Associates, Inc., 2009. 177

[16] D. Chigirev, G. Stephens, and The-Princeton-EBC-Team. Predicting base features with supervoxels. Abstract presented, 12th HBM meeting, Florence, Italy, 2006. 36

[17] C. K. Chui. *An introduction to wavelets*. Academic Press Professional, Inc., San Diego, CA, USA, 1992. 100

[18] J. Cohen and K. Blum. Overview: Reward and decision. *Neuron*, 36(2):193–198, 2002. 5

[19] D. Cox and R. Savoy. Functional magnetic resonance imaging (fmri) ”brain reading”: detecting and classifying distributed patterns of fmri activity in human visual cortex. *Neuroimage*, 19:261–270, 2003. 11, 23

[20] C. Davatzikos, K. Ruparel, Y. Fan, D. Shen, M. Acharyya, J. Loughead, R. Gur, and D. Langleben. Classifying spatial patterns of brain activity with machine learning methods: Application to lie detection. *NeuroImage*, 28(3):663 – 668, 2005. 11

[21] C. De Mol, E. De Vito, and L. Rosasco. Elastic-net regularization in learning theory. *J. Complex.*, 25(2):201–230, 2009. 28

[22] N. Dunford and J. Schwartz. *Linear Operators, General Theory*. Wiley-Interscience, 1958. 15

[23] L. M. e Silva and R. Buyya. Parallel programming models and paradigms. In *High Performance Cluster Computing: Programming and Applications*. Prentice Hall PTR, NJ, USA, 1999. 150

[24] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann. Statist.*, 32(1):407–499, 2004. 17, 19, 28, 29, 37, 52, 58, 107, 115, 118, 148, 149, 151, 153, 154, 156, 162, 163

[25] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993. 54

[26] J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *J. Am. Statist. Ass.*, 96(2):1348–1360, 2001. 27

[27] H. E. Fisher, A. Aron, and L. L. Brown. Romantic love: a mammalian brain system for mate choice. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 361(1476):2173–2186, December 2006. 5

[28] G. Flandin and W. D. Penny. Bayesian fmri data analysis with sparse spatial basis function priors. *NeuroImage*, 34(3):1108 – 1125, 2007. 6, 98

[29] S. Forman, J. Cohen, M. Fitzgerald, W. Eddy, M. Mintun, and D. Noll. Improved Assessment of Significant Activation in Functional Magnetic Resonance Imaging (fMRI): Use of a Cluster-Size Threshold. *Magnetic Resonance in Medicine*, 33:636–647, 1995. 49

[30] I. Frank and J. Friedman. A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–148, 1993. 27

[31] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. *Proc. 13th Int'l Conf. Machine Learning*, pages 148–156, 1996. 100

[32] K. Friston, A. Holmes, K. Worsley, J. Poline, C. Frith, R. Frackowiak, et al. Statistical parametric maps in functional imaging: a general linear approach. *Human Brain Mapping*, 2(4):189–210, 1994. 7, 52, 97

[33] K. Friston and W. Penny. Posterior probability maps and SPMs. *Neuroimage*, 19(3):1240–1249, 2003. 98

[34] W. Fu. Penalized regression: the bridge versus the lasso. *J. Comput. Graph. Statist.*, 7(2):397–416, 1998. 27

[35] W. Fury, F. Batliwalla, P. Gregersen, and W. Li. Overlapping Probabilities of Top Ranking Gene Lists, Hypergeometric Distribution, and Stringency of Gene Selection Criterion. In *Proc. of the 28th IEEE EMBS Annual International Conference*, pages 5531–5534, 2006. 53, 66, 70

[36] A. Gelman. Scaling regression inputs by dividing by two standard deviations. *Statistics in Medicine*, 27(15):2865–2873, July 2008. 94

[37] C. Genovese, D. Noll, J. Cohen, and W. Eddy. Estimating test-retest reliability in functional mr imaging i: Statistical methodology. *Magn Reson Med*, 38:497–507, 1997. 18

[38] G. Glover. Deconvolution of impulse response in event-related bold fmri. *NeuroImage*, 9:416 – 429, 1999. 2, 3

[39] G. H. Golub and C. F. Van Loan. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)(3rd Edition)*. The Johns Hopkins University Press, 3rd edition, October 1996. 151, 158

[40] P. I. Good. *Resampling Methods: A Practical Guide to Data Analysis*. Birkhauser, 2005. 54, 178

[41] F. G. Gustavson, L. Karlsson, and B. Kagstrom. Three algorithms for cholesky factorization on distributed memory using packed storage. In *Applied Parallel Computing. State of the Art in Scientific Computing. 8th International Workshop, PARA 2006*, pages 550–559, 2007. 152

[42] S. J. Hanson, T. Matsuka, and J. V. Haxby. Combinatorial codes in ventral temporal lobe for object recognition: Haxby (2001) revisited: is there a "face" area? *NeuroImage*, 23:156–166, Sep 2004. 11

[43] J. Haxby, M. Gobbini, M. Furey, A. Ishai, J. Schouten, and P. Pietrini. Distributed and Overlapping Representations of Faces and Objects in Ventral Temporal Cortex. *Science*, 293(5539):2425–2430, 2001. 8, 11, 23, 176

[44] J. D. Haynes and G. Rees. Decoding mental states from brain activity in humans. *Nat Rev Neurosci*, 7:523–524, 2006. 10

[45] A. Hoerl and R. Kennard. Ridge regression. *Encyclopedia of Statistical Sciences*, 8(2):129–136, 1988. 15, 27, 36

[46] A. Holmes, J. Poline, and K. Friston. Characterizing brain images with the general linear model. In R. Frackowiak, K. Friston, C. Frith, R. Dolan, and J. Mazziotta, editors, *Human Brain Function*, pages 59–84. Academic Press USA, 1997. 5

[47] S. Huettel and G. McCarthy. Evidence for a refractory period in the hemodynamic response to visual stimuli as measured by mri. *Neuroimage*, 11(5):547–553, 2000. 176

[48] R. A. Hutchinson, T. M. Mitchell, and I. Rustandi. Hidden process models. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 433–440, New York, NY, USA, 2006. ACM. 176

[49] J. Jia and B. Yu. On model selection consistency of the elastic net when $p \gg n$. Technical Report 756, Department of Statistics, U.C. Berkeley, 2008. 28

[50] Y. Kamitani and F. Tong. Decoding the visual and subjective contents of the human brain. *Nature Neuroscience*, 8(5):679–685, April 2005. 8, 178

[51] N. Kanwisher, J. Mcdermott, and M. M. Chun. The fusiform face area: A module in human extrastriate cortex specialized for face perception. *Journal of Neuroscience*, 17:4302–4311, 1997. 7

[52] K. N. Kay, T. Naselaris, R. J. Prenger, and J. L. Gallant. Identifying natural images from human brain activity. *Nature*, 452(7185):352–355, March 2008. 10, 180, 183

[53] V. Koltchinskii, M. Martínez-Ramón, and S. Posse. Optimal aggregation of classifiers and boosting maps in functional magnetic resonance imaging. In L. K.

Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 705–712. MIT Press, Cambridge, MA, 2005. 174

[54] S. M. M. Laconte, S. J. J. Peltier, and X. P. P. Hu. Real-time fmri using brain-state classification. *Hum Brain Mapp*, November 2006. 182

[55] N. Logothetis, J. Pauls, M. Augath, T. Trinath, and A. Oeltermann. Neurophysiological investigation of the basis of the fmri signal. *Nature*, 412(6843):150 – 157, 2001. 3

[56] R. Maitra, S. Roys, and R. Gullapalli. Test-retest reliability estimation of functional mri data. *Magn Reson Med*, 48(1):62–70, 2002. 18

[57] F. J. Massey, Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951. 93

[58] N. Meinshausen and P. Buehlmann. Stability selection. *ArXiv e-prints*, Sept. 2008. 178

[59] N. Meinshausen and B. Yu. Lasso-type recovery of sparse representations for high-dimensional data. *Annals of Statistics*, 37(1):246–270, 2009. 28

[60] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard, Version 2.2*. High Performance Computing Center Stuttgart (HLRS), September 2009. 150

[61] T. Mitchell, R. Hutchinson, R. Niculescu, F. Pereira, X. Wang, M. Just, and S. Newman. Learning to Decode Cognitive States from Brain Images. *Machine Learning*, 57:145–175, 2004. 11, 24

[62] T. M. Mitchell, S. V. Shinkareva, A. Carlson, K.-M. Chang, V. L. Malave, R. A. Mason, and M. A. Just. Predicting Human Brain Activity Associated with the Meanings of Nouns. *Science*, 320(5880):1191–1195, 2008. 10, 180, 183

[63] S. Mukherjee, P. Niyogi, T. Poggio, and R. M. Rifkin. Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Adv. Comput. Math.*, 25(1-3):161–193, 2006. 177

[64] K. Norman, S. Polyn, G. Detre, and J. Haxby. Beyond mind-reading: multi-voxel pattern analysis of fMRI data. *Trends in Cognitive Science*, 10(3):424–430, 2006. 9, 11, 12, 23

[65] D. North. An analysis of the factors which determine signal/noise discrimination in pulsed-carrier systems. *Proceedings of the IEEE*, 51(7):1016 – 1027, 1963. 96, 97

[66] S. Ogawa, T. Lee, A. Nayak, and P. Glynn. Oxygenation-sensitive contrast in magnetic resonance image of rodent brain at high magnetic fields. *Magnetic Resonance in Medicine*, 14:68 – 78, 1990. 2

[67] M. Ojala and G. C. Garriga. Permutation tests for studying classifier performance. *Journal of Machine Learning Research*, 11:1833–1863, 2010. 54

[68] A. Okabe, B. Boots, K. Sugihara, and S. Chiu. *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams, 2nd Ed.* Wiley, 2000. 31

[69] W. Ou, T. Raij, F.-H. Lin, P. Golland, and M. Hämäläinen. Modeling adaptation effects in fmri analysis. In *MICCAI '09: Proceedings of the 12th International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 1009–1017, Berlin, Heidelberg, 2009. Springer-Verlag. 176

[70] D. A. Patterson and J. L. Hennessy. *Computer Organization and Design, Fourth Edition, Fourth Edition: The Hardware/Software Interface (The Morgan Kaufmann Series in Computer Architecture and Design)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008. 164

[71] W. Penny, N. J. Trujillo-Barreto, and K. J. Friston. Bayesian fMRI time series analysis with spatial priors. *NeuroImage*, 24(2):350–362, 2005. 96, 98, 146

[72] F. Pereira and G. J. Gordon. The support vector decomposition machine. In *ICML*, pages 689–696, 2006. 175

[73] F. Pereira, T. Mitchell, and M. Botvinick. Machine learning classifiers and fmri: A tutorial overview. *NeuroImage*, 45(1):S199–S209, March 2009. 14, 18

[74] R. R. Picard and R. D. Cook. Cross-validation of regression models. *Journal of the American Statistical Association*, 79(387):575–583, 1984. 17

[75] Pittsburgh-EBC-Group. Pbaic homepage: http://www.ebc.pitt.edu/2007/competition.html, 2007. 23, 30, 88

[76] S. M. Polyn, V. S. Natu, J. D. Cohen, and K. A. Norman. Category-specific cortical activity precedes retrieval during memory search. *Science (New York, N.Y.)*, 310(5756):1963–6, 2005. 9, 183

[77] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, December 2005. 99

[78] I. Rish, G. Grabarnik, G. Cecchi, F. Pereira, and G. J. Gordon. Closed-form supervised dimensionality reduction with generalized linear models. In *ICML*, pages 832–839, 2008. 175

[79] Roy, C.S. and Sherrington, C.S. On the Regulation of the Blood-supply of the Brain. *Journal of Physiology*, 11(1-2):85–158.17, 1890. 2

[80] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 59, Washington, DC, USA, 1998. IEEE Computer Society. 94

[81] I. Rustandi, M. Just, and T. Mitchell. Integrating multiple-study multiple-subject fmri datasets using canonical correlation analysis. Proceedings of the MICCAI 2009 Workshop: Statistical modeling and detection issues in intra- and inter-subject functional MRI data analysis, 2009. 147

[82] R. Saxe. Why and how to study theory of mind with fmri. *Brain Research*, 1079(1):57 – 65, 2006. Multiple Perspectives on the Psychological and Neural Bases of Understanding Other People's Behavior. 5

[83] S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan. Learnability, stability and uniform convergence. *Journal of Machine Learning Research*, 11(10):2635–2670, 2010. 177

[84] K. Sjotrand. Matlab implementation of LASSO, LARS, the elastic net and SPCA, 2005. Version 2.0. 150

[85] S. Smith, M. Jenkinson, M. Woolrich, C. Beckmann, T. Behrens, H. Johansen-Berg, P. Bannister, M. D. Luca, I. Drobnjak, D. Flitney, R. Niazy, J. Saunders, J. Vickers, Y. Zhang, N. D. Stefano, J. Brady, , and P. Matthews. Advances

in functional and structural mr image analysis and implementation as fsl. *NeuroImage*, 23(S1):208–219, 2004. 4, 57

[86] S. Strother, S. LaConte, L. Hansen, J. Anderson, J. Zhang, S. Pulapura, and D. Rottenberg. Optimizing the fmri data-processing pipeline using prediction and reproducibility performance metrics. *NeuroImage*, 23(S1):196–207, 2004. 18, 51, 52, 87, 98, 177

[87] R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996. 16, 24, 27

[88] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B*, 67(1):91–108, 2005. 52

[89] J. A. Tropp, A. C. Gilbert, and M. J. Strauss. Algorithms for simultaneous sparse approximation: part i: Greedy pursuit. *Signal Process.*, 86(3):572–588, 2006. 118

[90] M. Ulfarsson and V. Solo. Sparse variable principal component analysis with application to fMRI. In *Proc. IEEE International Symposium on Biomedical Imaging (ISBI07)*, 2007. 24

[91] M. A. J. van Gerven, B. Cseke, F. P. de Lange, and T. Heskes. Efficient Bayesian multivariate fMRI analysis using a sparsifying spatio-temporal prior. *NeuroImage*, 50(1):150–161, 2010. 52

[92] J. Van Wyhe. *Phrenology and the Origins of Victorian Scientific Naturalism (Science, Technology and Culture 1700-1945)*. Ashgate Publishing, March 2004. 7

[93] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:511, 2001. 100, 146

[94] E. Vul, C. Harris, P. Winkielman, and H. Pashler. Puzzlingly high correlations in fmri studies of emotion, personality, and social cognition. *Perspectives on Psychological Science*, 4(3):274–290, May 2009. 6

[95] S. Weisberg. *Applied Linear Regression*. Wiley, New York, 1980. 28

[96] E. W. Weisstein. Fisher's exact test. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/FishersExactTest.html, 2006. 53

[97] Z. Xiang, Y. Xi, U. Hasson, and P. Ramadge. Boosting with spatial regularization. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 2107–2115. Curran Associates, Inc., 2009. 52

[98] H. Xu, C. Caramanis, and S. Mannor. Robust regression and lasso. *CoRR*, abs/0811.1790, 2008. 117

[99] L. Xu, T. Johnson, and T. Nichols. Bayesian spatial modeling of fmri data: A multiple-subject analysis. *The University of Michigan Department of Biostatistics Working Paper Series*, 71, 2007. 175

[100] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006. 117, 174

[101] M. Yuan and Y. Lin. On the non-negative garrotte estimator. *Journal Of The Royal Statistical Society Series B*, 69(2):143–161, 2007. 28

[102] P. Zhao and B. Yu. On model selection consistency of lasso. *J. Mach. Learn. Res.*, 7:2541–2563, 2006. 28

[103] H. Zou and T. Hastie. Regularization and variable selection via the Elastic Net. *Journal of the Royal Statistical Society, Series B*, 67(2):301–320, 2005. 19, 24, 27, 28, 29, 51, 58, 94, 101, 107, 148, 149, 151, 155, 158