Beyond Brain Blobs: Machine Learning Classifiers as Instruments for Analyzing Functional Magnetic Resonance Imaging Data

Francisco Pereira

CMU-CS-07-175

December 2007

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Tom Mitchell, Chair
Geoff Gordon
Christos Faloutsos
Stephen Strother
Department of Medical Biophysics, University of Toronto
Rotman Research Institute of Baycrest, Toronto

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by a grant from NSF's CNCRS program, the Keck Foundation and DARPA under grant no. HR0011-04-1-0041. Francisco Pereira was supported by a PRAXIS XXI scholarship from Fundacão para a Ciência e Tecnologia, Portugal, (III Quadro Comunitário de Apoio, compariticpado pelo Fundo Social Europeu), a PhD scholarship from Fundacão Calouste Gulbenkian, Portugal, and NIH training grant R25 MH62011 and travel support from the graduate training program of the Center for the Neural Basis of Cognition.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity or person.

Keywords: Machine Learning, fMRI, Classifiers, Dimensionality Reduction, Support Vector Decomposition Machine

Esta tese é dedicada aos meus pais Paula e José, avós Clementina e Sidónio e à minha irmã Mariana, por terem sempre confiado em mim de todas as formas possiveis, mesmo demorando tanto tempo…

Abstract

The thesis put forth in this dissertation is that machine learning classifiers can be used as instruments for decoding variables of interest from functional magnetic resonance imaging (fMRI) data. There are two main goals in decoding:

- Showing that the variable of interest can be predicted from the data in a statistically reliable manner (i.e. there's enough information present).
- Shedding light on how the data encode the information needed to predict, taking into account what the classifier used can learn and any criteria by which the data are filtered (e.g. how voxels and time points used are chosen).

Chapter 2 considers the issues that arise when using traditional linear classifiers and several different voxel selection techniques to strive towards these goals. It examines questions such as whether there is redundant, as well as different kinds of, information in fMRI data and how those facts complicate the task of determining whether voxel subsets encode the desired information or whether certain choices of selection method or classifier are universally preferrable. All the results presented were obtained through a comparative study of five fMRI datasets.

Chapter 3 and Chapter 4 introduce the Support Vector Decomposition Machine, a new algorithm that attempts to sidestep the use of voxel selection by learning a low dimensional representation of the data that is also suitable for decoding variables of interest and has the potential to allow incorporation of domain information directly, rather than through the proxy of voxel selection criteria.

Acknowledgments

There are many people that helped me survive and get to this point, sometimes through something as simple as a brief conversation in the lounge late during some desperate night or as complicated as discussing whether any of this was worth it. That would be my first acknowledgement: the very friendly environment among the graduate students in SCS.

I'd like to thank Roy Maxion for, somehow, deciding it was worthwhile to hire me as a research assistant a long time ago and helping me decide what I wanted to do, even when it was clear the path was not through his lab.

I'd also like to thank Roni Rosenfeld and Manuel Blum for taking me under their wing early on during Tom's leave of absence. I owe a lot to Manuel, in particular, for teaching me much about research that is rarely said or written anywhere and always having an open office for a confused graduate student. I will never forget knocking on people's doors with him and asking, very socratically, what went through their heads when trying to find an algorithm.

Throughout the whole process, Sharon Burks was an invaluable source of help, be it through advice, commiseration, helping us run the coffee machine or many, many, many other things. More noticeable were perhaps the administrative mini-miracles worked on a weekly basis for distraught and distracted grad students, a torch which the formidable Deborah Cavlovich has already taken with gusto. May the force be with you, Deb!

I'd like to thank David McWherter and William Lovas for having taken on the sysiphean task of running the espresso machine. It was hard to let go and, at the same time, it was a blessed relief...

Getting closer to the end, I'd like to thank Jeff Baird, K.C. Marshall and Max Novelli for help above and beyond anything which was their duty. Literally hundreds of thousands of cluster processes later, I had enough results to write up a dissertation.

Throughout the whole thing, Zephyr was a constant presence. Note that I use a neutral adjective, not out of indifference but more to testify that it helped as well as hindered. But it was such pleasant hindering! Thank you for being there at all times and in the long dark nights, and for teaching me many things about American culture that I would never ever have learnt otherwise (though now that I think of it "tentacle porn" is Japanese).

Also throughout the whole thing were my brave officemates, the thugs of Wean Hall 5130: Stavros Harizopoulos, Weng-Keen Wong and Paul Bennett. You've heard me shout, insult people, beat my head against the table with a soothing rythm. And yet, you are my friends. Thank you, and may we have a chance to play more foosball in the future!

Paul, in particular, ought to be thanked further for endless hours of conversation about everything under the sun that do not let me forget that he studied (and learnt) much more than Computer Science. Most of it, alas, is too interesting for an acknowledgement... You are one of my best friends and you know why.

Getting back to work, I'd like to thank in particular Geoff Gordon and Stephen Strother for being much much more than committee members (as I realized over time and over protests of fellow graduate students about just how lucky I was when I told them what was going on). A problem was never a problem with Geoff, just the starting point for a dizzying hour on the white board and hope to last me another week. And another graduate student would come after me and it would happen again, as they would tell me later. I learnt most of what I needed to do this dissertation from you, and much more besides.

Starting with our first conversations, Stephen constantly reminded of what scientific integrity was and demonstrated quite clearly that one *could* read every relevant article (and give copies to disorganized graduate students who didn't). Beyond that, he revealed a level of understanding of my circumstances and decisions that would be hard to surpass. Stephen, I owe you and will not forget it!

Finally, Tom. I'm not sure whether to take the road of explaining just how patient he was on a macro scale (the graduation date will attest to that), on a micro scale (taking every eight-baked idea and dissecting it with me) or just how trusting of one's judgement or supportive of one's research ideas he can be. It's terrifying early on (what did I know?), exhilarating when settling on a thesis and rather rewarding when, with his help, one finally gets there. I'll settle for the fact that I was able to tell him most of this in more than one occasion and just thank him yet again...

It's harder to thank family and loved ones properly, but one can try (and fail). April Murphy gave me her love, devotion and food throughout the thick and thin of writing this dissertation, and had the patience to live in a *menage a trois* with it. Research, alas, has proven to be more of a conjoined twin than an interloper, I can only hope she can love us both.

This dissertation is dedicated to my parents, my grandparents and my sister. They always understood, always asked what they could do to help (and would do *anything*, I suspect) and never ever threatened to disown me or asked about fleshly offspring. I don't deserve such a family, but I will do my best to make them proud.

Contents

1 Introduction 1

2 Decoding Information From fMRI Data With Linear Classifiers 11

2.1 Classifiers and Feature Spaces 12

2.1.1 Setting 12

2.1.2 Classifiers 12

2.1.3 What can a linear classifier learn? 15

2.1.4 Voxel scoring methods 17

2.2 Establishing the significance of decoding results 19

2.2.1 Classification, significance and decoding 19

2.2.2 Practical issues 19

2.2.3 What is the “best” number of features? 22

2.2.4 Which results are significant? 23

2.2.5 Are there better classifiers or feature ranking methods? 26

2.3 Experimental Results 30

2.3.1 Datasets in this chapter 30

2.3.2 Result Summaries 32

2.3.3 Null model comparisons 39

2.3.4 Results across classifiers and significance 50

2.3.5 Effects of classifier and voxel selection method choice 56

2.3.6 Effects of training dataset size 64

2.3.7 Issues with cross-validation and feature selection 69
2.4 An application: identifying informative locations and characterizing voxel activity in them 88
2.4.1 Producing information maps 88
2.4.2 Information map results 89
2.4.3 Characterization of informative voxel activity 93

3 The Support Vector Decomposition Machine 99
3.0.4 Singular Value Decomposition 100
3.0.5 Classification 100
3.0.6 Chapter contents 101
3.1 SVDM Algorithm A 102
3.1.1 Optimization Problem A 102
3.1.2 Solving Optimization Problem A 103
3.2 SVDM Algorithm B 105
3.2.1 Optimization Problem B 105
3.2.2 Solving Optimization Problem B 106
3.3 Low-dimensional and feature-space discriminants 109
3.4 Related Work 110
3.5 Experimental Results 112
3.5.1 Datasets 112
3.5.2 Classification experiments (algorithm B) 113
3.5.3 Comparison with classifiers 114
3.5.4 Algorithm A versus algorithm B 116
3.5.5 Algorithm B versus SVD and ICA 124
3.5.6 Constraining the number of informative components 129

4 SVDM extensions 147
4.1 Adding regularization through constraints and penalties 147
4.1.1 Implementation of a smoothness penalty 148

4.1.2 Results 149
4.1.3 Automatically constraining the number of components used . . . 157
4.1.4 Other extensions 168
4.2 Multiclass and multitask learning 169
4.2.1 Setting 169
4.2.2 Algorithm 169
4.2.3 Experiments 170
4.3 Multisubject models 178

5 Conclusions **185**

Bibliography **191**

Chapter 1

Introduction

The thesis put forth in this dissertation is that machine learning classifiers can be used as instruments for decoding variables of interest from functional magnetic resonance imaging (fMRI) data. There are two main goals in decoding:

- Showing that the variable of interest can be predicted from the data in a statistically reliable manner (i.e. there's enough information present).
- Shedding light on how the data encode the information needed to predict, taking into account what the classifier used can learn and any criteria by which the data are filtered (e.g. how voxels and time points used are chosen).

Chapter 2 is concerned with both of these goals when conventional classifiers are used; Chapters 3 and 4 explore the use of a new algorithm for addressing both of them simultaneously, rather than training a classifier and examining it a posteriori.

Functional Magnetic Resonance Imaging

Technique and Data

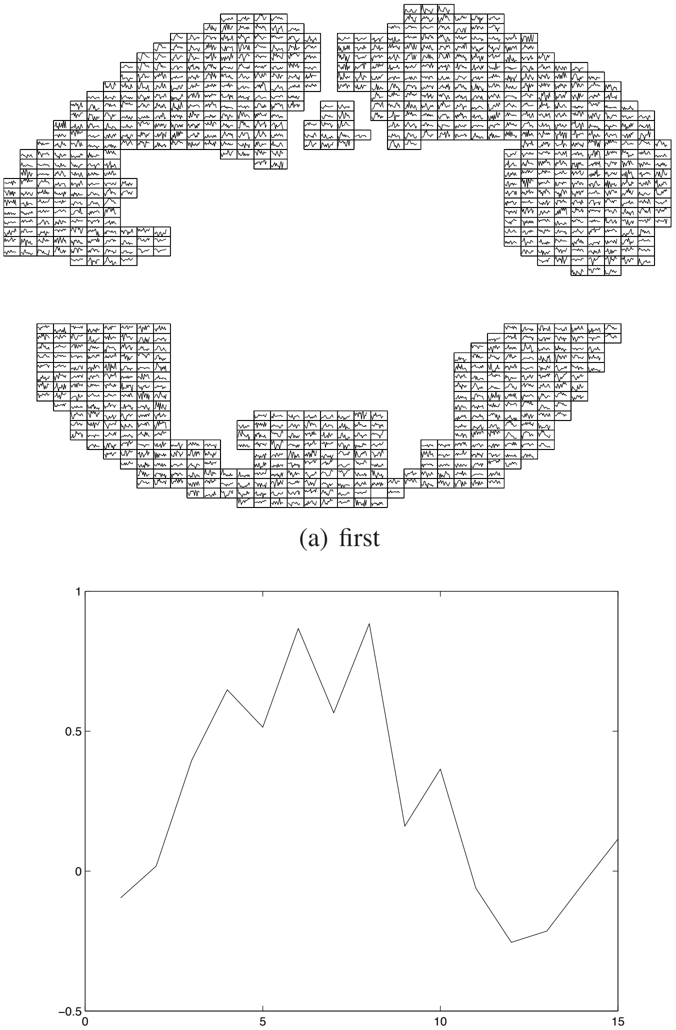
Functional Magnetic Resonance Imaging (fMRI) is a technique for obtaining three-dimensional images related to activity in the brain through time. This will necessarily be a very brief introduction, but we direct the interested reader to [30] for a comprehensive and accessible introduction to fMRI. More precisely, fMRI measures the ratio of oxygenated hemoglobin to deoxygenated hemoglobin in the blood with respect to a control baseline, at many individual locations within the brain. Blood oxygen level is influenced by local neural activity, and hence this blood oxygen level dependent (BOLD) response is generally taken as an indicator of neural activity.

An fMRI scanner measures the value of the fMRI signal (BOLD response) at all the points in a three dimensional grid, or *image*, covering part of the brain. In the studies used in this dissertation, a three dimensional image is captured every second. We refer to the cells within an image as *voxels* (volume elements). The voxels in a typical fMRI study have a volume of a few tens of cubic millimeters, and a three dimensional image typically contains tens of thousands of voxels, a large fraction of which contain cortical matter and are thus of interest. While the spatial resolution of fMRI is good, each voxel nevertheless contains on the order of hundreds of thousands of neurons to millions of neurons.

The temporal response of the fMRI BOLD signal is smeared over several seconds. Given a brief stimulus such as a flashing checkerboard, for instance, the fMRI BOLD response in the visual cortex increases to a maximum after approximately four to six seconds, typically returning to baseline levels after another five to ten seconds. Despite this prolonged temporal response, researchers have found that the relative timing of events can be resolved to within a few tens of milliseconds (e.g. to distinguish the relative timing of two flashes of light - one in the left eye and one in the right eye - as in [50]). The work in this dissertation does not use information in the temporal domain except insofar as consecutive groups of images are averaged into single images that are used as examples for classifiers, with each voxel being a feature. An alternative would be to use the same voxel at different time points in a trial as features instead, as proposed in [52].

A small portion of fMRI data is illustrated in Figure 1.1. This figure shows data collected over a fifteen second interval during which the subject was read a word, decided whether it was a noun or verb (in this case, it was a verb), then waited for another word. This data was sampled once per second for fifteen seconds, over sixteen planar slices, one of which is shown in the figure.

Finally, note that, even though the data for different subjects might reside on grids with the same dimensions, these are generally not directly comparable. Matching datasets from different subjects to each other or to an (annotated) template is called *spatial normalization* ([16]).



(a) first

(b) time (seconds,x axis)

Figure 1.1: **Top:** fMRI data for a selected set of voxels in the cortex, from a two-dimensional image plane through the brain. A fifteen second interval of fMRI data is plotted at each voxel location. The anterior portion of the brain is at the top of the figure, posterior at bottom. The left side of the brain is shown on the right, according to standard radiological convention. The full three-dimensional brain image consists of sixteen such image planes. **Bottom:** The time course for a single voxel from the slice. During this interval the subject was presented a word, answered whether the word was a noun or verb, then waited for another word. The signal scale is the percentage of change relative to a baseline condition.

Analysis

The vast majority of fMRI studies are analyzed using an approach called Statistical Parametric Mapping ([16]), which looks at the extent to which a voxel's time series is explained by a number of regressors in a General Linear Model.

These regressors reflect contrasts of interest (e.g. control condition versus task) as well as other confounds and the analysis is univariate in that each voxel is fit separately. The goal of the analysis is to generate a map that shows, for a particular contrast, the voxels where the coefficient for that contrast is significantly different from 0 (e.g. a voxel that is more active in the task condition than in the control condition).

This fit is often done with the pooled data from several subjects, after spatial smoothing and normalization to fit a template brain. Therefore, if a given voxel is deemed active, it's both likely that it is so in most or all of the subjects and that, furthermore, several neighbours will be as well (given the spatial smoothing). This cluster of activation is colloquially known as a "brain blob".

The typical result of this analysis is a set of centroid coordinates for these clusters on the template brain. In essence, it is the answer to the question "are there brain locations that act differently between two conditions". Note that it is possible to devise more elaborate contrasts in the presence of more conditions, though one would still have to specify the relationship between them and the analysis would still be univariate. In general, we may have many conditions whose relationship is not known a priori (would be necessary to design contrasts) or no voxels showing significant sensitivity to a contrast or presence of interactions between voxels, even as they all have a slight difference in value between conditions that, pooled together, would have enough information [34].

All these issues can be addressed from a different viewpoint, that of asking whether a machine learning classifier can be trained to predict a variable of interest (e.g. experimental condition) using *all* the voxels available. This can be specialized by restricting the voxels considered by location or behaviour, or the relationships between them, but the underlying idea is the same: if the prediction can be made, the information exists and can be found by dissecting the successful learning mechanism. The next section introduces this idea in more detail.

Decoding Information From fMRI Data With Linear Classifiers (Chapter 2)

As said earlier, there are two main aspects of concern in decoding variables of interest from fMRI data. The first is showing that the variable of interest can be predicted from the data in a statistically reliable manner, i.e. there's enough information present, and also revealing how the data encode the information needed to predict.

There is a growing amount of evidence that variables of interest can be predicted from data in many different brain imaging domains. Part of this work was done by us [52] and several other studies are reviewed in two excellent papers [27],[55], thus we will only cite specific studies where required. The second aspect of decoding has mostly been considered in terms of judging whether voxels with certain properties such as location or selectivity of response to stimuli contained enough information to drive the learning of a successful classifier. It will feature in this chapter through questions regarding voxel selection and the hypothesis space of the algorithms being used to learn classifiers.

Both aspects require a notion of *significance* of a classification result, i.e. being able to say that the result would be very unlikely if there were no relationship between the data and the variable being predicted; i.e. that the observed accuracy of the classifier over a particular set of test data was thus being obtained by chance.

While most studies rely on having a significant result in order to declare that decoding took place, the manner in which this is generally done is often insufficient. One way in which this can happen is not to take into account the fact that the classification result on a test set is only an estimate of the true accuracy, the probability that the classifier will label a test example correctly. Similarly, any derived measures (such as sets of selected voxels or classifier weights on voxels) are a function of a random training set and thus have uncertainty attached to them that needs to be quantified. [65] proposes a resampling-based methodology to address this issue systematically, but this is not commonly used. The second and more subtle way in which significance testing may be incorrect or insufficient stems from the fact that in any analysis several choices have to be made, e.g. preprocessing pipeline, classifier (and choice of regularization parameters), voxel selection method, number of voxels used, cross-validation regime, among others. We do not consider the interaction of preprocessing pipeline with classifier performance using different voxel selection methods, for the sake of both brevity and simplicity, but refer the concerned reader to [64] for a comprehensive overview of what is known regarding this issue.

This raises both the question of whether any choices are better all round – across classifiers, subjects, studies – and also that of whether choosing from many possible feature selection results changes how one ought to test for significance (it does). These questions have been posed in isolation in the machine learning literature but not, to our knowledge, as a whole.

This chapter introduces a methodology for addressing both the significance and the comparison questions, for a subset of the choice possibilities above, and applies it to five fMRI datasets. We also compare principled and expensive solutions versus cheap analytic approximations and show that the latter can provide reasonable results. We show that the voxel selection choice matters and particular methods are better than others across datasets; classifier choice matters much less if voxel selection is being used. Finally, we examine the questions of whether voxel selection can be relied upon to identify informative voxels and whether the various methods for selection isolate different – rather than redundant – sources of information. We will show that both things occur. For the former, this is evidenced by the lack of overlap in sets of voxels picked in different cross-validation foldsand, for the latter, by discrepancies in errors made by the same classifier using voxels selected by the different methods.

The Support Vector Decomposition Machine (Chapter 3)

Learning problems in fMRI are often characterized by having tens of thousands of features and only a small number of labelled training examples. This situation arises because of the difficulty of collecting and labelling independent training examples; temporal autocorrelation in sequences of fMRI images is very high and this means one often needs to average many images to obtain a single example. Moreover, it is complicated to obtain long sequences, both in terms of cost and also of having a subject in the scanner for a large span of time.

In order to obtain good learning performance in this domain (and others with similar characteristics, such as DNA microarrays) researchers often perform dimensionality reduction before learning a classifier ([52],[27],[55]. Two typical approaches to dimensionality reduction are feature selection (*e.g.*, forward stepwise selection after scoring features by some criterion, or regularizations that favour the use of few features) and feature synthesis (*e.g.*, singular value decomposition). The former is discussed in detail in Chapter 2 and thus this chapter focuses on the latter (as done in [6], [66] or [39], for instance).

In a typical feature synthesis approach we build new features based on the distribution of the independent variables in the training data, using algorithms like singular value decomposition (SVD,[25]) or independent component analysis (ICA,[31]), to reach a situation with more examples than features. Unfortunately, feature spaces produced in this manner may not necessarily be good for classification, since they are derived without reference to the quantity we are trying to predict.

Consider, for example, examples (images) obtained while a subject was thinking about items belonging to different semantic categories, and suppose that our task is to decide which semantic category was present in each example. If we perform an SVD of this data, the components extracted will capture image variability due to awareness, task control, language use, the visual form of the cues given and many other factors. Most of these dimensions of variability will have little information about the semantic category and, if their variance is too high they may prevent the SVD from noticing the directions of variation which would be useful for classification.

On the other hand, the basic idea of the SVD – trying to find a small set of features which accurately describe the test data – seems sound. The problem is only that the SVD performs its data reduction without paying attention to the classification problem at hand. Thus we pose the question of whether we can combine dimensionality reduction and classification into a single learning algorithm.

In this chapter we introduce one such learning algorithm, the Support Vector Singular Value Decomposition Machine (SVDM). To design the SVDM, we combine the goals of dimensionality reduction and classification into a single objective function, and present an efficient alternating-minimization algorithm for optimizing this objective. Like a Support Vector Machine (SVM, [33]), the SVDM can be viewed as trading off between a classifier's hinge loss and the norm of a learned weight vector; however, instead of regularizing with the 2-norm like the SVM, the SVDM regularizes with a norm derived from the goal of reconstructing the design matrix. We present experiments showing that we can achieve better learning performance and find lower-dimensional representations than approaches that separate dimensionality reduction and classification into independent steps.

Support Vector Decomposition Machine Extensions (Chapter 4)

The SVDM algorithm introduced in Chapter 3 produces a solution that is good for both classifying and reconstructing the data. The linear classifier learnt places weights on each dimension of the low-dimensional representation when classifying an example. In theory these weights should suffice to indicate which dimensions relate to the variable being predicted and thus allow interpretation, by looking at the changes in values across classes in the low-dimensional representation or the pattern of activation in the corresponding basis images. In practice this turns out to be hard to do, in that most dimensions show some kind of class-influenced structure; if there were fewer it would be easier to understand their collective relationship to the variable of interest.

In this chapter we show how the SVDM algorithm can be modified to produce solutions where very few dimensions are used in prediction, while maintaining classification and reconstruction performance at the same level of the original algorithm. This is done by penalizing the weights of the linear classifier learnt by SVDM using a L_1 rather than a L_2 norm, which causes most of them to be 0 and the corresponding dimensions to be ignored in classification. This penalty is just one instance of a general principle; additional penalties or constraints can be added to the optimization subproblem for each solution component, as long as it remains convex. We show how this can be done for the basis vector set component, by deriving a modification that favours spatially smooth fMRI data basis eigenimages, and show that it produces the desired result on two fMRI datasets.

Chapter 3 considers only fMRI datasets with binary classification tasks. In Chapter 4 we show how the algorithm can be adapted to work with more than two classes, apply SVDM to relevant fMRI datases and show that it compares favourably with linear SVMs in terms of classification accuracy. Furthermore, we show that the low-dimensional representations learnt can capture relationships between classes.

Finally, we introduce two different ways of extending the SVDM algorithm to use fMRI data from multiple subjects. The main intuition behind this is the assumption that the relationship between brain activation and the variable being predicted is similar across various subjects. Therefore, both approaches described rely on building a model with subject specific and subject independent parts, with the latter being used to capture the relationship. We show that it is possible to learn multisubject models that perform as well but not better than single subject models, using either approach, at least for the binary classification problem datasets also used in Chapter 3.

Machine Learning Classifiers and Neuroscience

The goal of this thesis is to show that machine learning classifiers allow us to examine fMRI data in ways that go beyond looking for individual voxels matching a specified pattern of behaviour. This will range from showing that there is enough information in a multivariate pattern of activation to predict a variable of interest to revealing how that information is organized; in the limit, we would like to show how the classes being predicted are related to each other and how those relationships are underpinned by parts of the pattern of activation.

That said, the process of doing this also yielded benefits that are more general in terms of machine learning applicability. The lessons learned in terms of training, applying and evaluating classification results generalize to other domains with sparse, high-dimensional datasets (such as DNA microarrays) where feature selection is necessary; furthermore, it's unclear that there is *a* better feature selection method. Rather, there may be many different feature behaviours all of which can have some - but not all - relevant information.

SVDM is an attempt to deal with the fact that the sparse high-dimensional data may, in addition, be more complex than usual due to the presence of many sources of activation. Most of these are unrelated, or only partially related, to what we would like to predict and thus act as a complex, structured noise. Furthermore, it has both a spatial and a temporal dimension (especially in fast event-related designs) that may both provide additional leverage (e.g. spatial smoothness prior for classifier parameters) and hinder us through violated assumptions (e.g. examples no longer being drawn independently, or appearing only as mixtures). Finally, that same spatial structure also manifests in relationships between features: we can expect correlation, mutual inhibition or more complex combinatorial codes.

The issues described in the previous paragraphs all go against the "traditional" setting for machine learning: examples drawn i.i.d. from an example distribution, with more examples than features and uncorrelated features. Our hope is that the work in this domain shows that something can be learnt even when reality is far from this setting, in addition to it being useful to domain experts.

Chapter 2

Decoding Information From fMRI Data With Linear Classifiers

The feasibility of predicting a variable of interest from fMRI data has been shown repeatedly, by us and several others, using a variety of classifiers and of methods for selecting voxels to be used as features. In all of this work the approach is generally to show that a prediction can be made and the accuracy level is "significant" under a simple null hypothesis that the classifier used is operating at chance level.

In this chapter we show that this approach can be too optimistic in what it deems significant. Moreover, it does not take into account a situation where several classifiers, voxel selection methods and numbers of voxels selected are used on the same dataset. We introduce a methodology that allows us to:

- Find the "best" number of voxels to use with a given combination of classifier and voxel selection method
- Report whether results are significant given that many classifiers and methods were used, taking into account the choice of a "best" number of voxels and multiple subjects
- Determine whether a given classifier or voxel selection method choice works better than others

using analytical solutions where possible and comparing them with exact solutions obtained using more expensive nonparametric statistics.

We apply this methodology to the study of five different datasets from semantic categorization experiments, with binary and multiclass problems of varying degrees of difficulty, using an assortment of classifiers and voxel selection methods and showing that there are indeed better choices among these. Furthermore, we describe the results of experiments on the reliability of relating voxels to the label being predicted by considering linear classifier voxel weights and also the properties of voxels selected by different methods.

2.1 Classifiers and Feature Spaces

2.1.1 Setting

All the studies in this chapter use voxels as features, i.e. a classifier will see example vectors $\mathbf{x}$ where each entry corresponds to one of m brain voxels, $\mathbf{x} =< x_1, \ldots, x_m >$. Conversely, each example corresponds to an image which, in these studies, is obtained by averaging a series of consecutive brain images (either from a trial or from a block) which can be labelled in the same way (e.g. "subject is thinking of Tools"). The studies are described in Section 2.3.1.

It would also be possible to use the temporal dimension of task performance by making features be "voxels at a particular time point", as done in our work [52] and [59],[40] among several others. In order to ease comparison between studies and to further brevity we refrained from doing so in this chapter. Using the temporal dimension can also introduce some dependency between examples and violate the assumption of independence and identical distribution, which we resort to in the analytic approaches described later.

2.1.2 Classifiers

Bayes Classifiers and Gaussian Naive Bayes

Bayes Rule tells us how to predict one of k classes from an example $\mathbf{x}$

$$P(\mathtt{class_k}|\mathbf{x}) = \frac{\mathbf{P}(\mathbf{x}|\mathtt{class_k})\mathbf{P}(\mathtt{class_k})}{\mathbf{P}(\mathbf{x})}$$

in terms of the class conditional distribution $P(\mathbf{x}|\mathtt{class_k})$.

The classifier obtained by assuming that that distribution is multivariate gaussian is the rule in Linear Discriminant Analysis (LDA), if we assume all classes share the same covariance matrix, or Quadratic Discriminant Analysis (QDA), if not [25].

If, furthermore, we assume that the class conditional distribution of each feature is independent of the others, i.e. $P(\mathbf{x}|\mathtt{class_k}) = \prod_{\mathbf{j=1}}^{\mathbf{m}} \mathbf{P}(\mathbf{x_j}|\mathtt{class_k})$, then we have a Gaussian Naive Bayes (GNB) classifier [51]. In addition, we will assume that the variance of each feature is the same for all class conditional distributions. This assumption seems reasonable in fMRI data (if the noise in each voxel is independent of its mean condition activation) and appears to lead to more robust estimates and better classifier accuracy by letting us pool data from multiple classes. Given this assumption, GNB is equivalent to LDA with a diagonal covariance matrix. The parameters to learn are the mean and variance (μ and σ^2) in the normal distribution density function for each class (e.g. class A) $f(x|\mu_A, \sigma_A^2) = \frac{1}{\sqrt{2\pi\sigma_A^2}} exp(-\frac{1}{2}(\frac{x-\mu_A}{\sigma_A})^2)$, with $\mu_A = \frac{\sum_{i=1}^{n_A} x_i}{n}$ (index ranges over x_i with class A only) and $\sigma_A^2 = \sigma^2 = \frac{\sum_{i=1}^{n}(x_i-\mu_{x_i})^2}{n}$ (index ranges over x_i in all classes, subtracting the respective class mean).

LDA/QDA have been used in fMRI before [65],[6], though always preceded by a dimensionality reduction process (generally a singular value decomposition) in order to train the classifier having fewer dimensions than examples and be able to estimate a covariance matrix. An interesting alternative that also works by expressing an image in terms of a basis set is Penalized Discriminant Analysis (see [39]), which uses a basis of smooth images to represent the examples and trains an LDA model with ridge terms on its covariance matrix on coefficient space (equivalent to a L2 norm penalty on the classifier weights, see below).

Logistic Regression

Logistic Regression (LR,[25]) models $P(\texttt{class}_\mathrm{k}|\mathbf{x})$ as

$$P(\texttt{class}_\mathrm{k}|\mathbf{x}) = \frac{\texttt{exp}(\mathbf{x}'\mathbf{w_k} + \mathbf{w_{0,k}})}{1 + \sum_{j=1}^{K-1} \texttt{exp}(\mathbf{x}'\mathbf{w_j} + \mathbf{w_{0,j}})}$$

for classes $1, \ldots, K-1$ and

$$P(\texttt{class}_\mathrm{k}|\mathbf{x}) = \frac{1}{1 + \sum_{j=1}^{k-1} \texttt{exp}(\mathbf{x}'\mathbf{w_j} + \mathbf{w_{0,j}})}$$

for class K and fits parameters $\mathbf{w_j}$ to maximize the log conditional likelihood $\sum_{i=1}^{n} log(P(y_i|\mathbf{x_i}))$, where $\mathbf{x_i}$ is the i^{th} example and y_i is its label.

Unless the features are linearly independent - clearly not the case, since at the very least we can expect many neighbouring voxels to be correlated - it's generally necessary for stability reasons to regularize the solutions $\mathbf{w_k}$ by adding a term such as $\lambda\|\mathbf{w_k}\|_\mathbf{2}$ (L2 regularization) or $\lambda\|\mathbf{w_k}\|_\mathbf{1}$ (L1 regularization) to the log conditional likelihood to be maximized, for each class k. We use both of these possibilities in the experiments in this chapter.

At classification time, the predicted class k for example $\mathbf{x}$ is $\texttt{max}_\mathrm{k} P(\texttt{class}_\mathrm{k}|\mathbf{x})$. Given that the expressions for $P(\texttt{class}_\mathrm{k}|\mathbf{x})$ above all have the same denominator, we can consider just the numerators for the purpose of determining the maximum. From that perspective, the problem can be viewed as the learning m-dimensional discriminants $\mathbf{w_k}$ for each class such that $\texttt{max}_\mathrm{k}$ $(\mathbf{x}'\mathbf{w_k} + \mathbf{w_{0,k}})$ (plus the regularization term) is the predicted class label.

Linear Support Vector Machine

A linear Support Vector Machine (SVM,[33]) learns a m-dimensional discriminant $\mathbf{w}$ to minimize

$$\|\mathbf{w}\|_2^2 + \lambda \sum_{i=1:n} h(y_i \mathbf{x_i}'\mathbf{w})$$

where h is the hinge loss function

$$h(\rho) = \begin{cases} 0 & \rho \geq 1 \\ 1-\rho & \text{otherwise} \end{cases}$$

and where y_i is either -1 or 1. If the classification problem has more than two classes it generally has to be converted into several binary problems. One way of doing this is to use a *one versus rest encoding*, where one binary problem is created for each class. That problem is created by assigning one label to all the examples in class k and another to examples in all other classes; this is done for all K classes, yielding K binary classification problems. An alternative would be to use the binary problems created via an error-correcting output code [14], but the results would be both more computationally expensive to obtain and harder to interpret for our purposes.

When classifying a new example, we apply each of the binary problem discriminants $\mathbf{w_k}$ to it and predict the label k such that $\mathtt{max}_\mathtt{k}\ \mathbf{x}'\mathbf{w_k} + \mathbf{w_{0,k}}$. This discriminant differs from the one learnt by LR in that it maximizes the margin of the discriminating hyperplane, but is similar in that it is regularized using the L2 norm of the weight vector.

Linear Discriminant as a common view, regularization

Both LR and SVM are generally expressed as linear discriminants with particular regularizations, but so can GNB with the class-shared variance estimates. To see how, consider a situation with two classes, A and B (this can be generalized to k classes by considering each of $k-1$ against the k^{th}).

Classification of a new example $\mathbf{x}$ is done by comparing the posterior probabilities for each class, $P(A|\mathbf{x})$ and $P(B|\mathbf{x})$, and picking the class for which that posterior probability is highest. Equivalently, one could also say we are considering the *odds ratio* $R = \frac{P(A|\mathbf{x})}{P(B|\mathbf{x})}$ and deciding A if the ratio is greater than 1 or B otherwise. By taking the logarithm of this ratio and assuming equal prior probabilities over both classes ($P(A) = P(B)$), we can rewrite it as follows:

$$R = log\frac{P(A|\mathbf{x})}{P(B|\mathbf{x})} = log\frac{P(\mathbf{x}|A)P(A)}{P(\mathbf{x}|B)P(B)} = log\frac{P(\mathbf{x}|A)}{P(\mathbf{x}|B)}$$

and the decision rule becomes “select class A if $R > 0$ or class B if $R < 0$”.

Applying the naive bayes assumption, R is turned into a sum of per-feature ratios

$$R = \Sigma_{j=1}^{n} log \frac{P(x_j|Y=A)}{P(x_j|Y=B)} = \Sigma_{j=1}^{n} R_j(x_j, \mu_{Aj}, \sigma_{Aj}, \mu_{Bj}, \sigma_{Bj})$$

each of which is a function of the feature value x_j and the means and standard deviations for the two class conditional feature densities.

Each ratio R_j is a fraction of two normal distribution probability density functions, and so can be rewritten as:

$$R_j(x_j, \mu_{Aj}, \sigma_{Aj}, \mu_{Bj}, \sigma_{Bj}) = log \frac{\frac{1}{\sqrt{2\pi\sigma_{Aj}^2}} exp(-\frac{1}{2}(\frac{x_j - \mu_{Aj}}{\sigma_{Aj}})^2)}{\frac{1}{\sqrt{2\pi\sigma_{Bj}^2}} exp(-\frac{1}{2}(\frac{x_j - \mu_{Bj}}{\sigma_{Bj}})^2)}$$

Given that we also assume that the class conditional distributions for each feature have the same variance, the ratio R_j simplifies to the expression

$$R_j(x_j, \mu_{Aj}, \mu_{Bj}, \sigma_j) = \frac{1}{2\sigma_j^2}(2(\mu_{Aj} - \mu_{Bj})x_j + \mu_{Bj}^2 - \mu_{Aj}^2) = \frac{\mu_{Aj} - \mu_{Bj}}{\sigma_j^2} x_j + \frac{\mu_{Bj}^2 - \mu_{Aj}^2}{2\sigma_j^2}$$

where x_j stands by itself. The decision can thus be expressed in terms of a linear discriminant *w* such that $w_j = \frac{\mu_{Aj} - \mu_{Bj}}{\sigma_j^2}$ and $w_0 = \sum_{j=1}^{m} \frac{\mu_{Bj}^2 - \mu_{Aj}^2}{2\sigma_j^2}$.

2.1.3 What can a linear classifier learn?

The hypothesis space for linear classifiers in sparse, high-dimensional datasets

As shown in the previous section, the four classifiers being considered - GNB, LRL1, LRL2 and SVM - can be expressed as linear discriminants, either learning them directly or in a form that is convertible. Given that the number of examples n is typically much larger than the number of features m (tens-hundreds versus tens of thousands), it will always be possible to learn a linear classifier that performs very well on the training set. A more formal justification for this is that the VC dimension of a linear discriminant in m-dimensional space is $m + 1$ (see [51]), but we will not pursue this further here.

If there are many possible linear classifiers that work equally well on the training set, and we cannot obtain more training examples, how should one look for a classifier that does not overfit and thus performs well on the test set? The two possibilities are to add regularization - either directly on the discriminant or by assumptions about the distribution of features - or to reduce the dimensionality using feature selection or a mapping to a low-dimensional space (e.g. using Singular Value Decomposition,), though the latter is considered in Chapter 3 and Chapter 4. In the experiments in this chapter we use both regularization and feature selection, though resorting only to the most generic approach for the former by using norms of discriminant voxel weight vectors. A different direction would be to regularize using domain information, such as the fact that the activations of neighbouring voxels can be related and thus the weights on those voxels should not vary independently (used in [63] or [39], as well as in in our work [57] as described in the subsequent chapters).

Voxel relationships and nonlinear classifiers

The classification decision using a linear classifier can be viewed as voting process over all features, where feature j votes for a class via the sign of its respective weight and the confidence in that vote is the magnitude of the weight. Hence none of the four classifiers above can incorporate the interaction of two voxels into the decision, e.g. "class 1 if voxel A and voxel B both turn on or off but not if only either one is on".

A way of dealing with this particular example would be to add new synthetic features, such as the product $x_A x_B$ of every pair of voxels, either explicitly or implicitly (e.g. through the use of a quadratic polynomial kernel in a SVM). If one desired more latitude in what functions of several voxels could be considered, other SVM kernels or a neural network would be appropriate.

So why not use these classifiers instead of the linear classifiers above? The most serious issue is that it would require training even more complicated models with a small dataset, increasing the chance of overfitting. This could be alleviated by increasing the bias of these models via regularization, if we knew what the appropriate regularization was. We could also use feature selection, although the fact that this is typically done voxel by voxel might mean that the criteria used would not take interesting relationships between the voxels into account to start with.

Existing studies that used nonlinear classifiers in addition to linear ones do not in general report an improvement in results (see, for instance, [12] with SVMs or [24] with neural networks, where the results are not better than ours using linear classifiers on the same dataset, D3 in Section 2.3).

Why could this be the case? The bleakest possibility is that, at the voxel scale currently used (several mm^2 per slice), there are no nonlinear relationships between voxels given how many neurons are being subsumed into one measure and also that hemodynamic response is only a proxy for neural activity. A different possibility, and assuming that feature selection did not remove most voxels involved, is that we simply do not have the numbers of examples to support learning these relationships. One way to ascertain this would be to perform experiments with synthetic data containing such relationships in datasets of various sizes. A different direction would be to look for them first at a local scale, since there are indications that there is interesting local structure that can be leveraged ([38],[58]) and there would be many fewer features to consider (e.g. voxel neighbourhoods of $3 \times 3 \times 3$ or $5 \times 5 \times 5$ voxels). Ideally, this would be done with the same subject performing the same tasks and being scanned at various resolutions.

2.1.4 Voxel scoring methods

In domains with many features (tens to hundreds of thousands or even higher) it is generally the case that feature selection is a necessary step prior to training classifiers capable of attaining high accuracy (e.g. text mining [73] and [15], DNA microarrays [70], fMRI [52]). Almost all the fMRI decoding results reported in the literature used either feature selection or some other form of dimensionality reduction ([55],[27],[6],[65]). Feature selection is too vast a topic to broach in great detail here, but a useful review of both terminology and the general thrust of the various approaches is given in ([20]). One distinction often made is between *scoring/filtering* and *wrapper* methods. The former involves ranking the features by a given criterion and selecting the best in the ranking, while the latter consists broadly in picking new features by how much impact they have on the classifier given the features already selected. This can be done by repeatedly training and applying the classifier in cross-validation within the training set. This can be advantageous in that it can eliminate redundancy and also take advantage of the classifier ability to find relationships between features, instead of having to relegate that to the selection method. Given the number of features to choose from, however, it is expensive to do this withouth some simplifying heuristics. For that reason, this chapter will be concerned solely with feature scoring methods.

A second issue has to do with which methods are suitable for this domain. In this chapter all the datasets are such that voxels and features are one and the same, and the same is the case for most fMRI decoding studies to date. Moreover, there is a desire to be able to specify desired voxel behaviour (e.g. is a voxel very selective for one particular condition or discriminating between two groups of conditions), location or other properties. The method choices in extant papers are driven by this reason as much as by the wish for high accuracy. The scoring methods considered in this chapter and listed below are a sample of the methods that have been proposed in those papers, with some modifications:

- Activity - This method scores a voxel by the difference in its mean activity level during a task condition versus a baseline condition. A voxel undergoes a t-test of its mean value during each task condition, which can be of whether the mean is greater than 0 or of whether it is different from 0. The former makes sense in datasets normalized so that 0 is the mean value for each voxel during the baseline condition, whereas the latter would be more appropriate if the normalization is to units of percent change from mean level across all conditions. For practical reasons we apply both methods to all our datasets, expecting that one will be more appropriate than the other, and refer to both as *activity* voxel scoring. The t-test for each condition yields one ranking of voxels, the overall score of a voxel is its best position in any ranking (ties are broken at random).

- Accuracy - This method scores a voxel by how accurately a classifier can predict the condition of each example in the training set, based only on the voxel. We use a Gaussian Naive Bayes classifier (Section 2.1.2), as we perform cross-validation within the training set for each voxel and thus need a very efficient classifier. The classifier thus learns the parameters of a gaussian distribution for the values the voxel takes, in each class, and uses that to classify. The cross-validation accuracy is the voxel's score.

- Searchlight Accuracy - The same as Accuracy, but instead of using the data from a single voxel (classifier with one feature) we use the data from the voxel and its immediately adjacent neighbours in three dimensions (classifier with up to 27 features). This is a variant of a a strategy proposed by [38].

- ANOVA - This method looks for voxels where there are reliable differences in mean value across conditions (e.g. A is different from B C D, or AB from CD, etc). This is done using an Analysis of Variance (ANOVA, [44]) to test the null hypothesis that the mean voxel value for the various task conditions is the same. The score is the p-value (the lower, the better).

- Replicability - This method picks voxels that react to the various stimuli *consistently* across runs/epochs (even if by being mostly inactive across time). In all our experiments the datasets are divided into epochs or runs. Within each epoch/run there might be one or more examples of each condition (e.g. in dataset D1 there are two categories of stimuli, and for each category seven different exemplars, with each exemplar appearing once in an epoch). This criterion aligns the values each voxel takes across exemplars/categories in each given epoch/run, and considers the average correlation of this vector across all pairs of epochs/runs.

- Range - This set of methods look only at the mean value a voxel takes across different task conditions:

  - max/min difference - the distance between the highest and lowest mean values
  - max/second max difference - the distance between the highest and second highest mean values
  - max/median difference - the distance between the highest mean value and the median of the others
  - spread - the standard deviation of the sample of mean values
  - max - the maximum value the voxel takes

- Logistic Regression Weights - This method trains a logistic regression classifier on the training set (with a penalty on the L2 norm of the weights vector, see Section 2.1.2) and scores each voxel by the range between the largest and smallest weight across conditions. It is the only one where the score of a voxel depends on the scores of others. The rationale for this method is that it should find informative voxels while downplaying the weights of correlated ones and thus lessening their score.

The most common feature selection methods used in fMRI are the ANOVA, some variation of Activity and voxelwise or searchlight accuracy.

2.2 Establishing the significance of decoding results

2.2.1 Classification, significance and decoding

Consider an ideal situation with a single classifier, trained on a large training set and applied to a large test set. The accuracy on the test set is an estimate of the true accuracy of this classifier: the probability that it will correctly label a new example drawn at random from the same distribution the others came from. How good an estimate depends on the size of the test set; intuitively, the larger it is the more confident we should be that the estimate is close to the true probability.

The person training a classifier on fMRI data is concerned with establishing that a variable of interest can be decoded from it, i.e. a classifier can be trained whose true accuracy is better than that of a classifier deciding at random. A *significant* result is thus one where one can reject the null hypothesis that there is no information about a variable of interest. Establishing significance is generally done by determining how improbable a classification results would be were the null hypothesis to be true (this probability is called a p-value).

In this situation the probability of obtaining the result under this null hypothesis is easy to calculate. It can be viewed as the outcome of tossing a coin for each of the examples to be classified, with the coin's bias reflecting the probability of labelling an example correctly by chance (50% in a two class problem, 25% in a four class problem, etc). Thus, the p-value under this null hypothesis of k correctly labelled test examples is simply $P(X \geq k)$ under a binomial distribution with n trials and probability of success 0.5 (two class), 0.25 (four class), etc; if the p-value is below a certain threshold (typically 0.05 or 0.01) the result will be declared significant. We will refer to the process that leads to the result under the null hypothesis as the *null model* and, in particular, to this one as the "coin toss null model". This is essentially a one-sided hypothesis test on the parameter of a binomial distribution.

2.2.2 Practical issues

The previous section describes a procedure for determining whether a single classification result is significant. In practice, however, there isn't generally a single accuracy number to consider. To make this more precise, assume we have a dataset with n examples (hundreds) and m features (tens of thousands). We are using cross-validation for training a series of classifiers $C_1, \ldots, C_j$, all of which can be expressed as linear discriminants (logistic regression, linear SVM, etc). Assume also that there is a need for voxel selection, with a choice of method $M_1, \ldots, M_i$ (see Section 2.1.4 for details on the methods we will consider) to rank the features and also a choice of the number of features (top 10, top 20, etc) to pick from that ranking. Each combination of classifier, ranking method and number of voxels gives rise to an accuracy result $\hat{a}$. More precisely, as we are doing cross-validation we will obtain one such result for each train/test fold, but we will assume that the same classifier was used to label all the test examples in the cross-validation, a reasonable assumption for reasons discussed in Section 2.2.3, and report the overall result combining all the fold test sets.

The problem is how to go from having many possible results for each classifier and method combination (one per number of voxels taken from the ranking) to a single "best" result $\hat{b}_i$. There are three typical courses of action for doing this:

1. Run all combinations of voxel selection method and number of voxels selected and produce a $\#\texttt{methods} \times \#\texttt{voxels}$ table of results, deciding *a posteriori* on a "best" result $\hat{b}_i$ for each method (usually the number of voxels at which the highest accuracy is attained, as used in [52]).

FR methods — #voxels	m_1	$\ldots$	m_k	max accuracy
method M_1	$\hat{a}_{11}$	$\ldots$	$\hat{a}_{1k}$	$\hat{b}_1 = \texttt{max}_j \hat{a}_{1j}$
$\ldots$	$\ldots$		$\ldots$	$\ldots$
method M_i	$\hat{a}_{i1}$	$\ldots$	$\hat{a}_{ik}$	$\hat{b}_i = \texttt{max}_j \hat{a}_{ij}$

2. Use a method-specific answer for how many voxels to keep, if available. One example of this would be testing whether each voxels's mean is greater than 0 and keeping voxels whose p-value under the 0 mean null hypothesis is significant, under a threshold taking into account multiple comparisons (see [12] for a use of this approach). This gives us a set of s_i significant voxels to use and produce the "best" result $\hat{b}_i$.

FR methods — #voxels	significant #	accuracy
method M_1	s_1	$\hat{b}_1 = \hat{a}_{s_1}$
$\ldots$	$\ldots$	$\ldots$
method M_i	s_i	$\hat{b}_i = \hat{a}_{s_i}$

3. Run a cross-validation within the training set (nested cross-validation, NCV) and essentially produce a table of results similar to that in 1. This can be used to pick the number of voxels at which the highest accuracy is attained for each method and use those voxels and the whole training set to produce the classifier used on the test set (as suggested in [21]). This gives us a set of n_i voxels to use and produce the best result $\hat{b}_i$.

FR methods — #voxels	NCV #	accuracy
method M_1	n_1	$\hat{b}_1 = \hat{a}_{n_1}$
$\ldots$	$\ldots$	$\ldots$
method M_i	n_i	$\hat{b}_i = \hat{a}_{n_i}$

The first approach is the one most commonly seen in most comparisons of several feature selection methods. The maximum score for each method is picked and taken as "the" result for that method (e.g. [73],[15],[60] among many others) for comparison purposes. This can lead to this final result being optimistically biased, as explained in Section 2.2.4. Alternatively, one could test each entry of the result table and correct for multiple comparisons by tightening the significance level for each test. This would be excessively conservative for two reasons. The first is that the results in each row are not independent; as the features are being ranked and increasingly larger sets of features are selected, each classifier uses the same features the previous one did and decisions can thus be correlated. The second is that the same dataset is being used to test. Therefore, paired t-tests would be the correct technique to use (see Section 2.2.5).

The second approach is followed in some of the papers doing voxel selection in fMRI datasets and is feasible as long as the method in question has a natural criterion for determining which voxels significantly fulfill it; this generally entails setting a p-value threshold (e.g. 0.05 or 0.01) after correcting for multiple comparisons, as many as there are voxels to select from. A possible issue is that the number of voxels selected is not the one at which the highest accuracy could be attained. Another is that no voxels are deemed significant, even though there is information present (as seen, for instance, in [34]). While this could be dealt with by raising the p-value threshold, it would make the method less principled in that it would become quite similar to using the first approach with a single choice for the number of voxels. Finally, it's possible that there will be no clear way of testing the scores assigned by a given method to each voxel for statistical significance.

The third approach is extremely principled, in that it only uses the training set to make decisions on how many voxels to use with each method and yields a single result per method to be tested. Its main problems are that it is expensive to run a nested cross-validation and, as in the second approach, there is no guarantee that the same number of features is used in every cross-validation fold.

The rest of this chapter reviews the various issues arising from using the first approach in obtaining answers to three questions:

1. Section 2.2.3 - For each classifier and feature ranking method, can we determine a "best" number of features, i.e. a number of features at which the true accuracy higher, in a less biased way?

2. Section 2.2.4 - How to determine whether such a result is significant, given that many numbers of voxels, methods and classifiers are being tried, for several subjects?

3. Section 2.2.5 - How to compare the "best" result for different classifiers and voxel ranking methods? Do the choices interact?

Section 2.3 applies the techniques described to answer those questions over several fMRI datasets, while also comparing those results to those that would be obtained using the third approach where possible instead. The references for the statistical concepts used in these sections are

[67] (all sections), [44] (ANOVA) [36] (correction for multiple comparisons) and [19] (permutation tests).

2.2.3 What is the "best" number of features?

For a fixed training set with a given number of features, a classifier learnt from it can be applied to a test set to produce an accuracy value. A different test set would likely give rise to a different accuracy value. What is the relationship between these values and the classifier's true accuracy (the probability of correctly labelling a test example drawn at random from the example distribution)?

Assuming there are n examples in a test set, let E_l be 1 if the l^{th} example was classified correctly and 0 otherwise. Then $C = \sum_{l=1}^{n}$ is the number of correctly classified examples and $C \sim \texttt{Binomial}(n, p)$, with p being the accuracy, np the mean and $np(1-p)$ the variance. The test set accuracy estimator is thus $A = \frac{C}{n}$, which has mean p and variance $\frac{1}{n^2}np(1-p) = \frac{p(1-p)}{n}$. For a *particular* test set, we get an outcome of the random variable A.

This can be used to produce a confidence interval for the true accuracy of the classifier, as the parameter of a binomial distribution. This can be done exactly (numerically) or via approximations ([42]). The most commonly used approximation is to assume that the binomial can be approximated by a normal distribution and resort to the customary two-sided confidence interval for the mean parameter ([67]). This can be inappropriate in situations where n is small or p is at either extreme, for instance, but not exclusively in those ([9]). We will therefore work with the exact binomial distribution whenever possible (with the exception of the use of ANOVA for comparing many classification results in Section 2.2.5). A different possibility is to use resampling of dataset train/test splits (e.g. [46] describes various schemes, [65] uses a 2-fold version in fMRI data), though the estimates can be biased in some cases ([37]) and expensive to compute; this can be justified if considering the distribution of other quantities besides the accuracy estimate, such as features of the learnt model as considered in ([65]).

To further complicate things, we do not have a single training and test set but rather a sequence of k-fold stratified cross-validation separate test sets (e.g. the proportion of class labels in the overall dataset is maintained in each training fold, see Section 2.3.1 for more details). There are many empirical indications and some theoretical justifications that the k-fold cross-validation estimate of accuracy is unbiased and binomial if the learning algorithm that produces the classifier is stable across folds ([37],[47]), so we will make the assumption that the test labels were produced by the same classifier for all examples).

Finally, we have not one but several classifier accuracy estimates. This is because we are in the situation described in the previous section where, for a given training set and classifier, using feature ranking method i and picking a certain number of features m_k, we get a cross-validation accuracy $\hat{a}_{ik}$ which is an estimate of the true accuracy a_{ik}. As the same classifier is trained using the top $m_1, \ldots m_k$ features of the training set, we end up with a sequence of such accuracy estimates

$\hat{a}_{i1}, \ldots, \hat{a}ik$. The problem is to determine the number of features m_k for which the true accuracy a_{ik} is highest, given the sequence of accuracy estimates. Given that the accuracy estimate obtained is a random variable, it's not necessarily the case that this coincides with the number of features for which the estimate is highest.

If successive estimates were independent (e.g. obtained from independently drawn test sets) it would be no problem to produce correct, simultaneous confidence intervals for all of them by tightening the confidence level as a function of the number of tests. The problem arises from the fact that the features are ranked prior to selection and thus the set of m_{k+1} features contains the set of m_k (e.g. the top 10 are in the top 20, top 40, etc). This, combined with the fact that all the classifiers considered are linear discriminants and hence compute a weighted combination of feature, will virtually ensure that the predictions of the classifier using m_{k+1} features will correlated to some degree with those of the classifier using m_k. We try to avoid this in the experiments described in Section 2.3 by making m_{k+1} be substantially larger than m_k, but it cannot be entirely ruled out. Furthermore, people generally are not interested in the whole curve but rather in its maximum value, hence the focus on finding the number of features with the maximum true accuracy.

Each of the accuracy estimates can be viewed as a result of corrupting the accuracy with noise, i.e.

$$\hat{a}_{ik} = a_{ik} + \epsilon_{ik}$$

where ϵ_{ik} has mean 0 and variance $\frac{a_{ik}(1-a_{ik})}{n}$ (dependent on a_{ik} and following the distribution of test set accuracy described above).

The problem can thus be viewed as one of nonparametric regression, where the unknown curve to be estimated is the sequence of true accuracies $a_{i1}, \ldots, a_{ik}$ and the accuracy estimates $\hat{a}_{i1}, \ldots, \hat{a}ik$ are the data points. Once the unknown curve is estimated its maximum gives us the desired number of features.

Any of several nonparametric regression methods (see [68] for an overview) would work in this case and the bandwidth parameter which controls the amount of smoothing introduced when learning the estimate can be set using cross-validation ([68]). The combination of this and correlation between successive accuracy estimates can, at worse, lead to undersmoothing ([56]) but the precaution of bringing in lots of new features for every new estimate seems to address this to some extent in our experimental datasets.

2.2.4 Which results are significant?

Using the method outlined in the previous section we know a "best" number of features and corresponding accuracy $\hat{b}_{ij}$ for each combination of feature ranking method M_i and classifier C_j, which can be arranged as in Table 2.1.

The second question is whether any of the numbers in Table 2.1 allow us to conclude that the

Classifiers	C_1	$\dots$	C_j
FR methods			
method M_1	$\hat{b}_{11}$	$\dots$	$\hat{b}_{1j}$
$\dots$	$\dots$		$\dots$
method M_i	$\hat{b}_{i1}$	$\dots$	$\hat{b}_{ij}$

Table 2.1: For each combination of classifier C_j and feature ranking method M_i there is a cross-validation accuracy using the "best" number of features.

corresponding classifier+method managed to decode the information we are interested in. This is generally translated into a statement of the form "under the null hypothesis that the classifier learnt nothing and predicts at random, the result obtained is significant". As described earlier, the probability of obtaining the result under this null hypothesis is easy to calculate, as it can be viewed as the outcome of tossing a coin for each of the examples to be classified, with the coin's bias reflecting the probability of labelling an example correctly by chance (50% in a two class problem, 25% in a four class problem, etc). Thus, the p-value under this null hypothesis of k correctly labelled examples is simply $P(X \geq k)$ under a binomial distribution with n trials and probability of success 0.5 (two class), 0.25 (four class), etc. f below a certain threshold α, the result will be declared significant. This means that the probability of falsely rejecting the null hypothesis (Type I error) is α.

This approach raises two issues. The first is that the threshold must be picked to compensate for the fact that there are many entries in Table 2.1, as the chance of a false rejection becomes much higher. This can be done using a criterion such as Bonferroni correction [67], which will give us a threshold such that the overall chance of a false rejection is α, or the false discovery rate [67], which gives us a threshold such that the proportion of false rejections to rejections of the null hypothesis is α or less. The second issue is that what is really being tested is not simply the accuracy of a classifier, but the accuracy using the "best" number of features as selected from Table **??** in the previous section. This introduces an optimistic bias that could make some results appear significant when they are not. To see why, consider a situation where the classifiers is using different numbers of features and, in each case, is doing no better than chance. It's not impossible that, at least for one number of features, the accuracy will be fairly high; it becomes easier the smaller the dataset is. If we are just deeming the "best" number of features to be the one at which the maximum accuracy is attained, we would pick this score and possibly deem it significant. To reiterate this in the terms used before: the "coin toss" null model is not appropriate to test the significant of the accuracy at the "best" number of features. There are two possible ways of dealing with this, described in the two following subsections.

Permutation Test

Assuming there is no class information in the data, the labels can be permuted without altering the expected accuracy using a given classifier and number of features (chance level), a classical permutation test ([19] for a general introduction and also [18], which both reviews the use of permutation tests in classification and also one neuroimaging application).

Given a dataset with permuted labels, we redo the entire process of using each feature ranking method to order the features, training classifiers with varying numbers of them, smoothing the table as described in Section 2.2.3 and picking the best number in each row, ending with the production of something like Table 2.1. Over many permutations this yields a sample of accuracy results for each cell in the table under the null hypothesis. The p-value under the null hypothesis for each entry of Table 2.1 is the fraction of the corresponding sample that is greater than or equal to it.

The main problems of resorting to this approach are its computational cost and, related to that, the possibility that the test is conservative. This stems from the fact that the lowest fraction possible that is greater than 0 varies inversely with the number of permutations, e.g. the lowest nonzero p-value with 1000 permutations is only 0.001. A possible way of getting around this is to assume that the distribution of each cell in the table under the null hypothesis has a known form (e.g. normal) and estimate the parameters from many fewer permutations (say, 100). We examine the feasibility of doing this in Section 2.3.3.

Analytic Approximations

The alternative to using a permutation test is to try to devise a more appropriate null model. Given the smoothing step incorporated in Section 2.2.3 it's unclear whether the same result can be obtained analytically, but there are a number of possible approximations.

The simplest such possibility is the model where, instead of having a single coin tossing classifier labelling each example, we have as many of them as there are numbers of features selected. Intuitively, the idea is to capture the picking of a maximum score out of $< numbers\ of\ features >$ draws from the null hypothesis distribution. As in Section 2.2.3, this yields one accuracy number per coin toss. The output of the process is the maximum of these accuracies. More formally, if $P(X \leq x)$ is the c.d.f. for the accuracy count of a single coin toss classifier and Y is the maximum of accuracy counts out of k coin tosses we have that

$$P(Y \leq y) = P(X_1 \leq y \bigcap X_2 \leq y \bigcap \ldots \bigcap X_k \leq y) = P(X_1 \leq y) \ldots P(X_k \leq y)$$

is the c.d.f. for the maximum of k coin tosses, and can be adapted to produce a p-value under the null hypothesis ($P(Y \geq \texttt{best result} | \texttt{null is true})$, so we need to compute $P(Y < y)$ instead). We will call this model "max row" and contrast it with "coin toss" and also the results of permutation tests in the experiments described in Section 2.3. As we will see there, the model is simple but gives significance results closer to those of permutation tests also to those produced with nested cross-validation; at the same time, it's still completely analytical and thus cheap to compute.

The "max row" null model assumes a series of independent coin tosses; this clearly does not hold, as the classifier using the top n_{i+1} features is also using the top n_i features, and hence its decisions will be correlated to some extent with those of the classifier using just the latter.

One possible way of addressing this while using the "max row" model is to attempt to introduce many new features every time we select a new group, e.g. make n_{i+1} be substantially larger than n_i (say half the features are new). This is the approach we follow in the experiments described in Section 2.3.

A more principled way of tackling the issue is to model the dependency between the decisions of successive classifiers. To see how consider a given feature ranking method and the decisions of a classifier trained using various numbers of features, $n_1, \ldots, n_k$. Let $E_l, n_i = 1$ if the l^{th} example was classified correctly by the classifier trained using the top n_i features selected by the method, and 0 if the example was classified incorrectly. The true accuracy of this classifier is $P(E_l, n_i) = 1$.

The dependency between decisions of successive classifiers in this sequence can be modelled as a Markov Chain. In this chain $P(E_{l,n_1} = 1)$ being is the true accuracy of the first classifier in the chain. Similar probabilities for the other classifiers can be computed from the previous one, e.g. for the n_i, n_{i+1} pair

$$P(E_{l,n_{i+1}} = 1) = P(E_{l,n_{i+1}} = 1|E_{l,n_i} = 0)P(E_{l,n_i} = 0)+P(E_{l,n_{i+1}} = 1|E_{l,n_i} = 1)P(E_{l,n_i} = 1)$$

These marginal probabilities could then be plugged in instead of those of variables assumed independent in the "max row" null model described above. It is possible to estimate the transition probabilities under the null hypothesis for each n_i, n_{i+1} pair by permuting the example labels, training and applying the classifiers and tallying the frequencies of the four transitions $0, 0$, $0, 1$, $1, 0$ and $1, 1$ across examples for the pair. However, it's not clear how to combine the probabilities estimated across many permutations and, as we shall see later, the simpler model that assumes independent coin tosses is generally good enough.

2.2.5 Are there better classifiers or feature ranking methods?

The previous sections were concerned with the use of a single classifier and one or more feature ranking methods for establishing the significance of classification results.

We would like to extend that comparison to multiple classifiers and feature ranking methods in order to answer the following questions:

1. Is there a better (higher accuracy) classifier?
2. Is there a better feature ranking method?

3. Is any combination superior?

for various datasets, in order to determine whether certain choices in either factor are preferrable in general. A different reason for asking this kind of question would be to ascertain the extent to which factors such as choices in the preprocessing pipeline affect classifiers and feature ranking methods differently, for instance (as suggested and documented in [64]).

If one were simply comparing two classification results on independent test sets, one could use a t-test to compare the means of two samples, where each sample is comprised of the outcomes of variables E_l, 1 if the l^{th} example was classifier correctly and 0 if not. If using the same test set, the appropriate technique would be a paired t-test, i.e. a test of the mean of a sample constructed by subtracting the E_l outcome in one sample from that in the other, for each example l. This and other tests are explored extensively throughout the machine learning literature, but [47] is an excellent introduction to the issues and both it and [13] contrast the theoretical expectations for some of the tests with practical evaluations. Given that we are testing whether the mean of the constructed sample is 0 we are essentially relying on the Central Limit Theorem (CLT,[9]) for the use of the test to be valid, where the sample size is the number of examples tested. An alternate possibility is to use a nonparametric version of the paired t-test, the Wilcoxon signed-rank test [44]. The only assumptions required are that the sample points be independent draws (we are also assuming the test examples are drawn independently, in any case) and that the sample be symmetric about a median (under the null hypothesis that the two samples have the same median this would hold and be 0). The drawback of using this approach is that it will be less powerful than the t-test, should the number of examples be large enough to justify using the CLT.

This can be extended to comparisons of more outcome samples by using an ANOVA (Analysis of Variance [44]) to test whether one can reject the null hypotheses that all the samples have the same mean (or do so for the samples where a given classifier was used versus all others, and analogously for feature ranking methods). As in the case of the t-test, obtaining all samples from the same test set requires using the correlated samples version of ANOVA. Note that in that case the two sample, one factor ANOVA is essentially the same as a two-sided paired t-test.

If the global null hypothesis that all classifiers considered have the same true accuracy is rejected, one can test any particular differences using Tukey's HSD (Honestly Significantly Different,[44], [36]) procedure, which will give a conservative threshold for considering any mean difference between two samples significant. Alternatively, we can use a paired t-test for each difference and control for the many possibly correlated multiple comparisons using False Discovery Rate ([67]), the approach we follow in Section 2.3.5.

In this case, we will consider a two-factor ANOVA: classifier and feature ranking method. In this setting we can test the effect of each factor on mean difference (e.g. pool all the feature rankings into a single sample for each classifier to consider the classifier effect) as well as their interaction, by considering each cell of the table against the others.

There have been many uses of either ANOVA or paired t-tests for this purpose (the earliest we are considering here are [29], [47] and [13]), but we will consider a few recent studies to give perspective on how a large comparison of classifiers can use these ideas in different ways.

The first study ([43]) compares the performance of classifiers learnt with various algorithms over a range of datasets, using three different approaches. The first is exploratory and consists in determining, for every dataset, the algorithm with the best accuracy and those whose accuracy is within one standard deviation of it (assuming the accuracy p is normally distributed and the standard deviation is thus $\sqrt{\frac{p(1-p)}{n}}$, this assumption may not be reasonable for certain values of n and p, see [9]). This gives a heuristic for determining whether an algorithm is better than others. A second approach is to treat the accuracy on multiple datasets as sample points of classifier performance and perform a one-factor ANOVA testing the difference in sample mean between classifiers, using Tukey's HSD criterion to determine which classifiers perform worse than the best one. The final approach is to produce one ranking of classifiers in each dataset and consider the mean ranking of each classifier across datasets, using an appropriate nonparametric test of the null hypothesis that the algorithms are equally accurate on average.

The second study ([8]) also compares several classifier learning algorithms on a number of datasets, but also evaluates them by a variety of performance metrics (rather than just using accuracy). The best classifier is found for each metric and paired t-tests comparing performance in cross-validation are used to determine which other classifiers are close enough to be deemed best as well (a fixed p-value is used with no direct correction for multiple comparisons, though arguments are given as to why it shouldn't much change results). This is analogous to a certain extent to a two factor ANOVA, with classifier and metric as the two factors. A second approach is to bootstrap over both datasets and performance metrices, obtaining a distribution over rankings and, in particular, over being the best classifier.

There are older studies cited in both of the above but the problem to be solved and methodology used remain quite similar, though the classifiers considered change over time. One common thread to these studies is that the datasets considered tend to have tens to hundreds of features and generally at least as many examples as features, if not many more, so feature selection is not considered in the classifier comparison. Conversely, there are several studies comparing several feature selection methods on high-dimensional data ([73],[71],[60], [15], among many others), the other dimension of our comparison. Generally, these contrasts are done by considering performance curves of the classifier using varying numbers of features, and seeing whether any curves dominate the others at any point (where "dominate" is defined by a criterion involving an informal confidence interval around each point, as typically no correction for multiple comparisons is done, though paired t-tests are sometimes used). If the dataset sizes are large this generally does not matter, as confidence intervals are extremely narrow. In our case, it does matter: the datasets are small, many numbers of features may be tried and the results at successive numbers may be correlated. Whereas we don't compare at all numbers of features simultaneously (we are interested in peak accuracy, with the smoothing correction described earlier, rather than one curve dominating others at all numbers of features), we do use the ANOVA counterpart of paired t-tests as well as a methodology that takes into account how many comparisons are being done and that feature selection takes place.

dataset	stimulus	details	prediction task
D1	word	2 categories (buildings/tools), 84 examples	2 categories
D2	word	2 categories (buildings/tools), 60 examples	2 categories
D3	picture	8 categories, 96 examples	8 categories
D4	word	5 buildings/5 tools, 60 examples	10 items
D5	word+picture	12 categories, 360 examples	12 categories

Table 2.2: Dataset summary, please see Section 2.3.1 for more details.

2.3 Experimental Results

2.3.1 Datasets in this chapter

We considered five datasets drawn from four semantic categorization studies, with a particular classification task for each. The following sections describe the datasets and how examples are obtained from them, as well as how they are used for classification. The information is also summarized in Table 2.2.

Datasets D1, D2, D4 and D5

In the experiment performed to obtain these datasets the subject is shown a series of words. Each word is the name of an item which is either a kind of tool or a kind of building, and the subject performs the following task: think about the item and its properties while the word is displayed (3 seconds) and clear her mind afterwards (8 seconds of blank screen).

In all four datasets a trial is converted into an example by taking the average image during a 4 second span while the subject is thinking about the stimulus shown a few seconds earlier (these 4 seconds contain the expected peak of the signal during the trial, TR=1000msec).

- Dataset D1

  There are 7 different items belonging to each of the two categories and 6 experimental epochs. In each epoch all 14 items are named, without repetition; all 6 epochs have the same items. Using each trial to obtain an example, this yields a total of 42 examples for each of the two categories. The classification task is to predict the category of a example. This dataset has three subjects.

- Dataset D2

  This dataset is very similar to dataset D1 except for the fact that both word and picture stimuli were used, in alternation. We excised the picture stimulus portion of the dataset (the category classification task is far too easy) and retained the text stimulus portion as dataset

D2. The classification task is to predict the category of a example. There are 5 items of each category and 6 experimental epochs, which gives us a total of 30 examples for each of the two categories. This dataset has six subjects.

- Dataset D4

  This dataset was created from the same source as dataset D2, excising the text portion instead and performing the classification task over *items* rather than categories. There are 5 items of each category and 6 experimental epochs, which gives us a total of 6 classification task examples for each of 10 item classes. This dataset has five subjects.

- Dataset D5

  The task performed is the same as in the previous datasets, except that the subject is shown a series of combined word and picture stimuli corresponding to an item (e.g. both the word “train” and a drawing of a train), rather than just a picture of a word. Each item shown belongs to one of 12 semantic categories: animals, body parts, buildings, parts of buildings, clothing, furniture, insects, kitchen objects, manmade objects, tools, vegetables and vehicles. The classification task is to predict the category of a example. There are 5 different items for each category. There was one trial for each item in each of 6 experimental epochs, giving us 30 examples for each category. This dataset has one subject.

Dataset D3

In the experiment performed to obtain this dataset subjects were shown a series of pictures in blocks, where each block contained images belonging to one of 8 categories: faces, houses, cats, scissors, shoes, chairs, bottles and scrambled images of all the previous categories. There was one block of each category per run, and a total of 12 runs. Images in each block were used to produce an example, yielding a total of 12 classification task examples of each of the 8 categories. Examples are generated by averaging all the images in a block (excluding transitions and one after the beginning and one before the end). This dataset has four subjects.

Using the datasets for classification

All the datasets have a natural division into parts, corresponding to either separate scanner runs (12 runs in D3) or epochs (6 epochs in all other datasets). This division can be exploited to perform a leave-one-epoch/run-out cross-validation, with the additional desirable consequence that the training set in each fold has the same number of examples of each class.

The datasets have undergone a typical fMRI preprocessing stream from DICOM files using either SPM ([61],D1,D2,D4,D5) or AFNI ([2],D3). The steps followed are volume registration, slice acquisition timing correction and some outlier removal, followed by a voxelwise linear detrending in the datasets acquired in a single run (D1,D2,D4,D5) or in each run (D3). D3 voxels were normalized to be zero mean and unit standard deviation across time, within each run.

In all datasets examples are generated by averaging a series of images, be it those at the peak of a trial (D1,D2,D4,D5) or most of the images in a block (D3). Examples are thus images and voxels are the features. In each average image only between 10000 and 20000 of voxels those contain cortex in a typical subject; hence, the voxels used as features were those that were contained in a brain mask as determined automatically by SPM or AFNI.

Each example is further processed by subtracting its spatially pooled mean and dividing by its spatially pooled standard deviation.

2.3.2 Result Summaries

Result measures

In each crossvalidation, and for every set of voxels selected from the training set, a classifier is trained and labels are predicted for the examples in the fold test set. The final performance measure is accuracy, simply the fraction of the examples for which the label was predicted correctly when the example was on the test set, both for binary and multiclass classification problems.

In the problems with more than two classes this may be a relatively harsh measure, in that the classifier receives no credit for almost-right predictions (e.g. if prediction is a ranking of labels from which we pick the top, the classifier gets no credit for making the correct label be the second highest). An alternative measure that addresses this is the normalized rank accuracy:

$$\texttt{rank accuracy} = 1 - \frac{\texttt{average rank of correct label} - 1}{\# \texttt{ classes} - 1}$$

In the two class case this collapses to the accuracy measure. Whereas this measure may be preferrable as the number of classes increases, it is necessary to examine the classification decisions in detail; for instance, is a high rank accuracy the product of always getting the correct label close to but not at the top of the ranking, or a mix of correct and incorrect placements? Another possibility is that classes have a similarity structure that can be distinguished, such as natural object versus artificial object, but within which it's much harder to distinguish individual classes.

Without this careful analysis this type of result can be misleading and, for that reason and to avoid detracting from the main point of introducing a classification comparison methodology we do not use this measure further in this chapter.

method number	method name
1	accuracy
2	accuracy (searchlight)
3	activity (different from 0)
4	activity (greater than 0)
5	logistic regression weights
6	class mean ANOVA
7	replicability
8	range (highest-lowest)
9	range (highest-second highest)
10	range (highest-median)
11	range (sample standard deviation)
12	range (maximum value)

Table 2.3: Numbers used to indicate each voxel ranking method throughout tables and figures in this chapter.

Result tables

Throughout the rest of the chapter the results reported will pertain to a series of voxel ranking methods, numbered 1 through 12. The key relating those numbers to the methods is in Table 2.3; a more complete description of each method can be found in Section 2.1.4.

Table 2.4, Table 2.5, Table 2.6 and Table 2.7 contain the set of results for one subject in each dataset, using a GNB classifier, for each combination of voxel ranking method (row) and number of voxels used (column). This level of detail is shown to give an idea of how the result change as the number of voxels changes. At the right of each table there are four additional columns: *all*, *max*, *smooth* and *CV*. These contain summary results for each table row, respectively the accuracy using all available voxels (all), maximum accuracy (max), the maximum accuracy after smoothing of the table row was done (smooth, as described in Section 2.2.3) and the accuracy obtained if nested cross-validation was done to select the number of voxels from the ranking produced by each method (CV, as described in Section 2.2.2).

Each of these tables shows that voxel selection can greatly improve the maximum accuracy relative to a classifier using all voxels; at the same time, it highlights how much the accuracy can change depending on how many selected voxels are used, let alone on what method is used to do so. In order to examine trends in the three accuracy summary values, Table 2.9, Table 2.10, Table 2.11, Table 2.12 and Table 2.13 contrast those summary values for all the subjects in each of the five studies.

Across studies and subjects it is clear that the nested cross-validation accuracy is consistently

FS	10	20	40	80	150	300	600	900	1200	1800	all	max	max	CV #
											voxels		(smooth)	
1	0.77	0.83	0.81	0.81	0.80	0.79	0.73	0.73	0.74	0.71	0.65	0.83	0.81	0.77
2	0.81	0.80	0.80	0.82	0.83	0.82	0.80	0.79	0.75	0.75	0.65	0.83	0.83	0.81
3	0.56	0.68	0.74	0.79	0.80	0.76	0.71	0.74	0.73	0.71	0.65	0.80	0.80	0.75
4	0.56	0.68	0.75	0.80	0.79	0.81	0.77	0.76	0.75	0.74	0.65	0.81	0.81	0.77
5	0.74	0.73	0.74	0.75	0.79	0.74	0.77	0.76	0.79	0.77	0.65	0.79	0.77	0.77
6	0.81	0.80	0.80	0.76	0.76	0.76	0.74	0.71	0.71	0.70	0.65	0.81	0.81	0.80
7	0.71	0.79	0.81	0.81	0.81	0.76	0.71	0.70	0.71	0.71	0.65	0.81	0.81	0.77
8	0.60	0.67	0.65	0.70	0.70	0.70	0.69	0.70	0.71	0.68	0.65	0.71	0.70	0.65
9	0.60	0.67	0.65	0.70	0.70	0.70	0.69	0.70	0.71	0.68	0.65	0.71	0.70	0.65
10	0.60	0.67	0.65	0.70	0.70	0.70	0.69	0.70	0.71	0.68	0.65	0.71	0.70	0.65
11	0.60	0.67	0.65	0.70	0.70	0.70	0.69	0.70	0.71	0.68	0.65	0.71	0.70	0.65
12	0.63	0.63	0.71	0.69	0.69	0.69	0.71	0.71	0.73	0.74	0.65	0.74	0.74	0.73

Table 2.4: Classification accuracy using GNB on datasets with varying numbers of voxels (columns), selected with each of 12 methods (rows), for subject 01481B from dataset D1. Each row has four summary results: the accuracy using all available voxels (all), maximum accuracy (max), the maximum accuracy after smoothing of the row (max smooth) accuracy using nested cross-validation to select the number of voxels to use.

FS	10	20	40	80	150	300	600	900	1200	1800	all	max	max	CV #
											voxels		(smooth)	
1	0.57	0.62	0.73	0.70	0.67	0.70	0.78	0.70	0.70	0.72	0.73	0.78	0.78	0.65
2	0.50	0.67	0.65	0.72	0.77	0.77	0.78	0.78	0.77	0.75	0.73	0.78	0.77	0.67
3	0.53	0.60	0.58	0.70	0.70	0.68	0.68	0.68	0.63	0.72	0.73	0.72	0.68	0.60
4	0.53	0.58	0.60	0.63	0.65	0.72	0.70	0.75	0.75	0.72	0.73	0.75	0.74	0.63
5	0.62	0.60	0.63	0.62	0.63	0.67	0.73	0.75	0.75	0.75	0.73	0.75	0.75	0.67
6	0.52	0.50	0.63	0.60	0.67	0.73	0.73	0.77	0.72	0.73	0.73	0.77	0.74	0.67
7	0.55	0.60	0.58	0.63	0.60	0.67	0.67	0.70	0.68	0.72	0.73	0.72	0.70	0.62
8	0.63	0.60	0.58	0.60	0.62	0.67	0.65	0.70	0.68	0.67	0.73	0.70	0.68	0.67
9	0.63	0.60	0.58	0.60	0.62	0.67	0.65	0.70	0.68	0.67	0.73	0.70	0.68	0.67
10	0.63	0.60	0.58	0.60	0.62	0.67	0.65	0.70	0.68	0.67	0.73	0.70	0.68	0.67
11	0.63	0.60	0.58	0.60	0.62	0.67	0.65	0.70	0.68	0.67	0.73	0.70	0.68	0.67
12	0.52	0.55	0.60	0.50	0.53	0.58	0.57	0.63	0.63	0.70	0.73	0.70	0.66	0.67

Table 2.5: Same as Table 2.4 for subject 02607B from dataset D2

FS	10	20	40	80	150	300	600	900	1200	1800	all	max	max	CV #
											voxels		(smooth)	
1	0.67	0.69	0.70	0.75	0.75	0.75	0.68	0.66	0.64	0.57	0.27	0.75	0.75	0.70
2	0.42	0.61	0.67	0.72	0.78	0.73	0.70	0.70	0.71	0.70	0.27	0.78	0.78	0.71
3	0.52	0.69	0.74	0.80	0.77	0.78	0.75	0.70	0.69	0.64	0.27	0.80	0.80	0.79
4	0.53	0.70	0.78	0.80	0.79	0.79	0.77	0.74	0.67	0.67	0.27	0.80	0.80	0.77
5	0.71	0.75	0.79	0.80	0.80	0.82	0.77	0.75	0.73	0.71	0.27	0.82	0.82	0.76
6	0.68	0.67	0.71	0.73	0.78	0.80	0.75	0.71	0.70	0.67	0.27	0.80	0.80	0.80
7	0.48	0.61	0.67	0.73	0.74	0.79	0.73	0.71	0.69	0.67	0.27	0.79	0.79	0.78
8	0.58	0.68	0.66	0.75	0.71	0.76	0.71	0.72	0.68	0.64	0.27	0.76	0.73	0.71
9	0.44	0.45	0.53	0.62	0.69	0.72	0.68	0.61	0.59	0.55	0.27	0.72	0.72	0.69
10	0.46	0.44	0.66	0.68	0.73	0.71	0.71	0.68	0.67	0.60	0.27	0.73	0.71	0.71
11	0.53	0.66	0.69	0.73	0.71	0.73	0.71	0.72	0.69	0.64	0.27	0.73	0.73	0.67
12	0.46	0.57	0.66	0.71	0.73	0.76	0.77	0.76	0.73	0.68	0.27	0.77	0.77	0.76

Table 2.6: Same as Table 2.4 for subject subj6_wp from dataset D3

FS	10	20	40	80	150	300	600	900	1200	1800	all	max	max	CV #
											voxels		(smooth)	
1	0.37	0.48	0.52	0.58	0.68	0.72	0.67	0.60	0.53	0.47	0.23	0.72	0.72	0.67
2	0.50	0.55	0.73	0.80	0.85	0.85	0.87	0.85	0.87	0.82	0.23	0.87	0.87	0.83
3	0.30	0.37	0.52	0.53	0.55	0.72	0.68	0.63	0.60	0.55	0.23	0.72	0.72	0.53
4	0.33	0.42	0.47	0.67	0.73	0.73	0.73	0.70	0.65	0.65	0.23	0.73	0.73	0.65
5	0.30	0.55	0.65	0.67	0.68	0.73	0.75	0.65	0.58	0.48	0.23	0.75	0.75	0.57
6	0.67	0.77	0.83	0.90	0.83	0.85	0.73	0.68	0.62	0.58	0.23	0.90	0.90	0.82
7	0.67	0.77	0.80	0.88	0.88	0.85	0.80	0.68	0.63	0.65	0.23	0.88	0.88	0.87
8	0.20	0.45	0.52	0.53	0.52	0.58	0.60	0.52	0.53	0.47	0.23	0.60	0.60	0.52
9	0.25	0.27	0.37	0.52	0.37	0.47	0.43	0.45	0.42	0.40	0.23	0.52	0.44	0.30
10	0.15	0.27	0.37	0.37	0.53	0.58	0.57	0.53	0.55	0.43	0.23	0.58	0.58	0.43
11	0.12	0.35	0.58	0.57	0.57	0.58	0.62	0.57	0.57	0.45	0.23	0.62	0.62	0.57
12	0.30	0.38	0.42	0.58	0.63	0.68	0.67	0.68	0.63	0.55	0.23	0.68	0.68	0.63

Table 2.7: Same as Table 2.4 for subject 02553B from dataset D4

FS	10	20	40	80	150	300	600	900	1200	1800	all	max	max	CV #
											voxels		(smooth)	
1	0.31	0.41	0.42	0.47	0.47	0.50	0.51	0.46	0.46	0.43	0.22	0.51	0.49	0.46
2	0.19	0.21	0.31	0.38	0.46	0.51	0.53	0.52	0.51	0.49	0.22	0.53	0.53	0.52
3	0.18	0.22	0.27	0.36	0.40	0.43	0.45	0.45	0.43	0.41	0.22	0.45	0.45	0.42
4	0.18	0.21	0.28	0.37	0.41	0.41	0.45	0.46	0.45	0.42	0.22	0.46	0.46	0.44
5	0.32	0.34	0.46	0.49	0.47	0.50	0.47	0.44	0.44	0.43	0.22	0.50	0.50	0.45
6	0.29	0.36	0.42	0.47	0.51	0.52	0.53	0.52	0.49	0.45	0.22	0.53	0.53	0.52
7	0.31	0.36	0.44	0.50	0.50	0.53	0.53	0.51	0.49	0.46	0.22	0.53	0.53	0.54
8	0.29	0.36	0.36	0.41	0.40	0.43	0.42	0.41	0.39	0.36	0.22	0.43	0.42	0.40
9	0.20	0.30	0.32	0.36	0.40	0.41	0.37	0.35	0.32	0.31	0.22	0.41	0.41	0.41
10	0.26	0.27	0.34	0.43	0.44	0.46	0.45	0.43	0.39	0.36	0.22	0.46	0.46	0.43
11	0.25	0.27	0.33	0.38	0.42	0.40	0.42	0.40	0.39	0.37	0.22	0.42	0.42	0.41
12	0.19	0.25	0.28	0.35	0.39	0.39	0.43	0.43	0.42	0.41	0.22	0.43	0.43	0.41

Table 2.8: Same as Table 2.4 for subject 03616B from dataset D5

lower than the maximum accuracy. It's also interesting to note that the smoothed maximum is almost always less than or equal to the maximum and many times in between the other two estimates. Note that we are both comparing two different classification procedures (select a number of voxels to use for each cross-alidation fold versus select a fixed number) and two different measures of accuracy. The nested cross-validation accuracy is an unbiased estimate of the true accuracy of a classifier trained with that procedure, whereas the maximum is a function of many unbiased (and possibly correlated) estimates. The two values are shown to suggest the extent of the bias introduced by computing the maximum. In Section 2.3.3 we consider a more appropriate comparison: does either procedure lead to declaring that the same classifier/method can decode with significant accuracy, under the appropriate null model for each procedure?

FS	01481B			01482B			01897B		
method	max	smooth	CV	max	smooth	CV	max	smooth	CV
1	0.83	0.81	0.77	0.79	0.77	0.75	0.81	0.81	0.77
2	0.83	0.83	0.81	0.80	0.78	0.75	0.82	0.82	0.80
3	0.80	0.80	0.75	0.71	0.70	0.69	0.81	0.81	0.76
4	0.81	0.81	0.77	0.73	0.73	0.71	0.86	0.86	0.85
5	0.79	0.77	0.77	0.77	0.77	0.74	0.83	0.83	0.81
6	0.81	0.81	0.80	0.79	0.79	0.74	0.83	0.83	0.80
7	0.81	0.81	0.77	0.77	0.76	0.71	0.83	0.83	0.80
8	0.71	0.70	0.65	0.75	0.75	0.74	0.81	0.80	0.77
9	0.71	0.70	0.65	0.75	0.75	0.74	0.81	0.80	0.77
10	0.71	0.70	0.65	0.75	0.75	0.74	0.81	0.80	0.77
11	0.71	0.70	0.65	0.75	0.75	0.74	0.81	0.80	0.77
12	0.74	0.74	0.73	0.73	0.73	0.68	0.82	0.82	0.81

Table 2.9: maximum row accuracies in D1 (chance 0.5, 84 examples)

FS	02452B			02497B			02509B			02553B			02607B			02714B		
method	max	smooth	CV	max	smooth	CV	max	smooth	CV	max	smooth	CV	max	smooth	CV	max	smooth	CV
1	0.80	0.80	0.65	0.90	0.90	0.83	0.88	0.88	0.78	0.78	0.78	0.53	0.78	0.78	0.65	0.98	0.98	0.95
2	0.88	0.88	0.78	0.93	0.93	0.90	0.90	0.89	0.87	0.88	0.87	0.77	0.78	0.77	0.67	0.97	0.97	0.97
3	0.70	0.66	0.57	0.75	0.75	0.70	0.83	0.83	0.78	0.88	0.88	0.75	0.72	0.68	0.60	0.97	0.97	0.97
4	0.68	0.68	0.62	0.78	0.77	0.72	0.83	0.80	0.75	0.90	0.90	0.80	0.75	0.74	0.63	0.95	0.95	0.95
5	0.82	0.82	0.75	0.93	0.93	0.83	0.88	0.86	0.78	0.87	0.87	0.78	0.75	0.75	0.67	0.98	0.98	0.97
6	0.82	0.79	0.78	0.92	0.91	0.92	0.88	0.86	0.80	0.83	0.83	0.75	0.77	0.74	0.67	0.98	0.98	0.97
7	0.77	0.72	0.77	0.95	0.95	0.93	0.90	0.87	0.80	0.78	0.78	0.68	0.72	0.70	0.62	0.98	0.98	0.95
8	0.68	0.68	0.65	0.88	0.86	0.87	0.82	0.79	0.67	0.80	0.80	0.63	0.70	0.68	0.67	0.98	0.98	0.95
9	0.68	0.68	0.65	0.88	0.86	0.87	0.82	0.79	0.67	0.80	0.80	0.63	0.70	0.68	0.67	0.98	0.98	0.95
10	0.68	0.68	0.65	0.88	0.86	0.87	0.82	0.79	0.67	0.80	0.80	0.63	0.70	0.68	0.67	0.98	0.98	0.95
11	0.68	0.68	0.65	0.88	0.86	0.87	0.82	0.79	0.67	0.80	0.80	0.63	0.70	0.68	0.67	0.98	0.98	0.95
12	0.63	0.63	0.60	0.82	0.82	0.80	0.78	0.77	0.77	0.75	0.74	0.72	0.70	0.66	0.67	0.93	0.93	0.93

Table 2.10: maximum row accuracies in D2 (chance 0.5, 60 examples)

FS	subj1$_ii$			subj2$_ss$			subj4$_jj$			subj6$_wp$		
method	max	smooth	CV	max	smooth	CV	max	smooth	CV	max	smooth	CV
1	0.85	0.85	0.88	0.67	0.67	0.64	0.65	0.65	0.65	0.75	0.75	0.70
2	0.84	0.84	0.83	0.74	0.74	0.71	0.65	0.65	0.55	0.78	0.78	0.71
3	0.86	0.86	0.84	0.61	0.61	0.60	0.61	0.61	0.56	0.80	0.80	0.79
4	0.86	0.86	0.81	0.64	0.64	0.62	0.64	0.64	0.59	0.80	0.80	0.77
5	0.94	0.94	0.92	0.91	0.91	0.78	0.77	0.77	0.77	0.82	0.82	0.76
6	0.92	0.92	0.92	0.89	0.89	0.81	0.73	0.71	0.64	0.80	0.80	0.80
7	0.91	0.91	0.91	0.83	0.83	0.77	0.73	0.73	0.69	0.79	0.79	0.78
8	0.85	0.85	0.85	0.70	0.70	0.64	0.70	0.70	0.58	0.76	0.73	0.71
9	0.76	0.76	0.77	0.53	0.51	0.49	0.59	0.59	0.54	0.72	0.72	0.69
10	0.81	0.80	0.83	0.70	0.70	0.68	0.64	0.64	0.59	0.73	0.71	0.71
11	0.85	0.85	0.85	0.75	0.75	0.68	0.69	0.69	0.69	0.73	0.73	0.67
12	0.91	0.91	0.90	0.78	0.78	0.76	0.65	0.65	0.58	0.77	0.77	0.76

Table 2.11: maximum row accuracies in D3 (chance 0.125, 96 examples)

FS	02553B			02565B			02607B			02714B			02745B		
method	max	smooth	CV	max	smooth	CV	max	smooth	CV	max	smooth	CV	max	smooth	CV
1	0.72	0.72	0.67	0.35	0.35	0.30	0.62	0.58	0.45	0.33	0.31	0.20	0.62	0.62	0.40
2	0.87	0.87	0.83	0.65	0.61	0.62	0.82	0.82	0.63	0.65	0.64	0.45	0.82	0.82	0.68
3	0.72	0.72	0.53	0.38	0.38	0.28	0.60	0.60	0.40	0.35	0.32	0.22	0.58	0.58	0.48
4	0.73	0.73	0.65	0.42	0.42	0.30	0.60	0.60	0.53	0.42	0.42	0.33	0.68	0.68	0.55
5	0.75	0.75	0.57	0.42	0.37	0.28	0.48	0.48	0.42	0.38	0.38	0.33	0.50	0.50	0.37
6	0.90	0.90	0.82	0.62	0.62	0.55	0.68	0.68	0.58	0.55	0.55	0.33	0.82	0.82	0.57
7	0.88	0.88	0.87	0.62	0.60	0.52	0.73	0.73	0.63	0.57	0.56	0.40	0.80	0.79	0.65
8	0.60	0.60	0.52	0.28	0.27	0.23	0.45	0.45	0.38	0.30	0.29	0.27	0.33	0.33	0.28
9	0.52	0.44	0.30	0.25	0.25	0.18	0.30	0.29	0.28	0.33	0.30	0.28	0.27	0.21	0.20
10	0.58	0.58	0.43	0.27	0.27	0.18	0.42	0.39	0.30	0.37	0.37	0.20	0.27	0.24	0.22
11	0.62	0.62	0.57	0.28	0.27	0.25	0.45	0.45	0.33	0.32	0.32	0.28	0.35	0.35	0.33
12	0.68	0.68	0.63	0.33	0.33	0.28	0.53	0.49	0.45	0.52	0.52	0.48	0.58	0.58	0.50

Table 2.12: maximum row accuracies in D4 (chance 0.1, 60 examples)

FS	03616B		
method	max	smooth	CV
1	0.51	0.49	0.46
2	0.53	0.53	0.52
3	0.45	0.45	0.42
4	0.46	0.46	0.44
5	0.50	0.50	0.45
6	0.53	0.53	0.52
7	0.53	0.53	0.54
8	0.43	0.42	0.40
9	0.41	0.41	0.41
10	0.46	0.46	0.43
11	0.42	0.42	0.41
12	0.43	0.43	0.41

Table 2.13: maximum row accuracies in D5 (chance 0.083, 360 examples

2.3.3 Null model comparisons

The overarching question answered by this section is whether the “max row” null model described in Section 2.2.4 is appropriate for judging the significance of results obtained with the procedure where we pick the maximum value of a smoothed row of the accuracy table (MSR).

Significance under various null models

In Section 2.2.4 we discussed the appropriateness of various null models for determining the significance of MSR results. In this section we compare four of them

- coin toss - assume that the classifier is predicting the label randomly
- max of row of coins - assume that the classifier is predicting at random for each entry in a table row and that the maximum accuracy in the row is being selected
- permutation test - permute labels, create a result table, smooth each row, pick the maximum accuracy in the row; over many permutations (1000), this generates a distribution of values of the maximum number in each row
- permutation test (normal) - same as permutation test, but assume the distribution of values is normal and estimate its parameters from a smaller set (100)

on the same results to determine how much difference the null model choice makes in terms of what results are deemed significant. The result table is smoothed using the procedure described in Section 2.2.3 and the maximum of each row is the value tested for significance.

The comparison is done only for results obtained with the GNB classifier, the fastest, given the expense of computing result tables for all subjects and studies after 1000 different permutations of the labels (several weeks of computation time on a 12 node cluster).

Each null model is used to obtain p-values for the values being tested. These are then thresholded with the values for all methods and subjects pooled together and using a bonferroni correction with significance level 0.01.

Figure 2.1, Figure 2.2, Figure 2.3, Figure 2.4 and Figure 2.5 show the values being tested (leftmost) and their significance under each of the four null models. Although we have no ground truth as to which results really are significant, we will treat the permutation test results as a surrogate for it. The determination of significance in that case may be conservative for the reasons indicated in Section 2.2.4 but, in particular, the fact that the number of permutations we could afford to do, 1000, may limit how small the p-value of a result under the permutation distribution is. Given that a few results are still deemed significant even then, we think it may not be unacceptably so.

Given that, it's clear that coin toss *is* optimistic in terms of what is declared significant. The max row model lies somewhere in between coin toss and the permutation test results in terms of how many things are declared significant. Curiously, it's closer to the permutation test in binary classification problems and to the coin toss in the multiple class problems. This might happen for two reasons. On one hand, it's easier for a fairly accurate result in the latter to be deemed significant as the number of classes increases and the expected accuracy under the null hypothesis becomes lower and lower. On the other, in a binary classification problem it's possible that, even after permutation, many labels end up in examples belonging to the same class as the example they originally belonged to.

Finally, the results indicate that it is possible to shortcut the permutation test by assuming that the distribution of each entry in the result table over permutations is normal, though the results might be even more conservative. This suggests running permutation tests for 10% or less of the number of runs one would usually run them for and finding whether any results are already significant prior to proceeding with more runs.

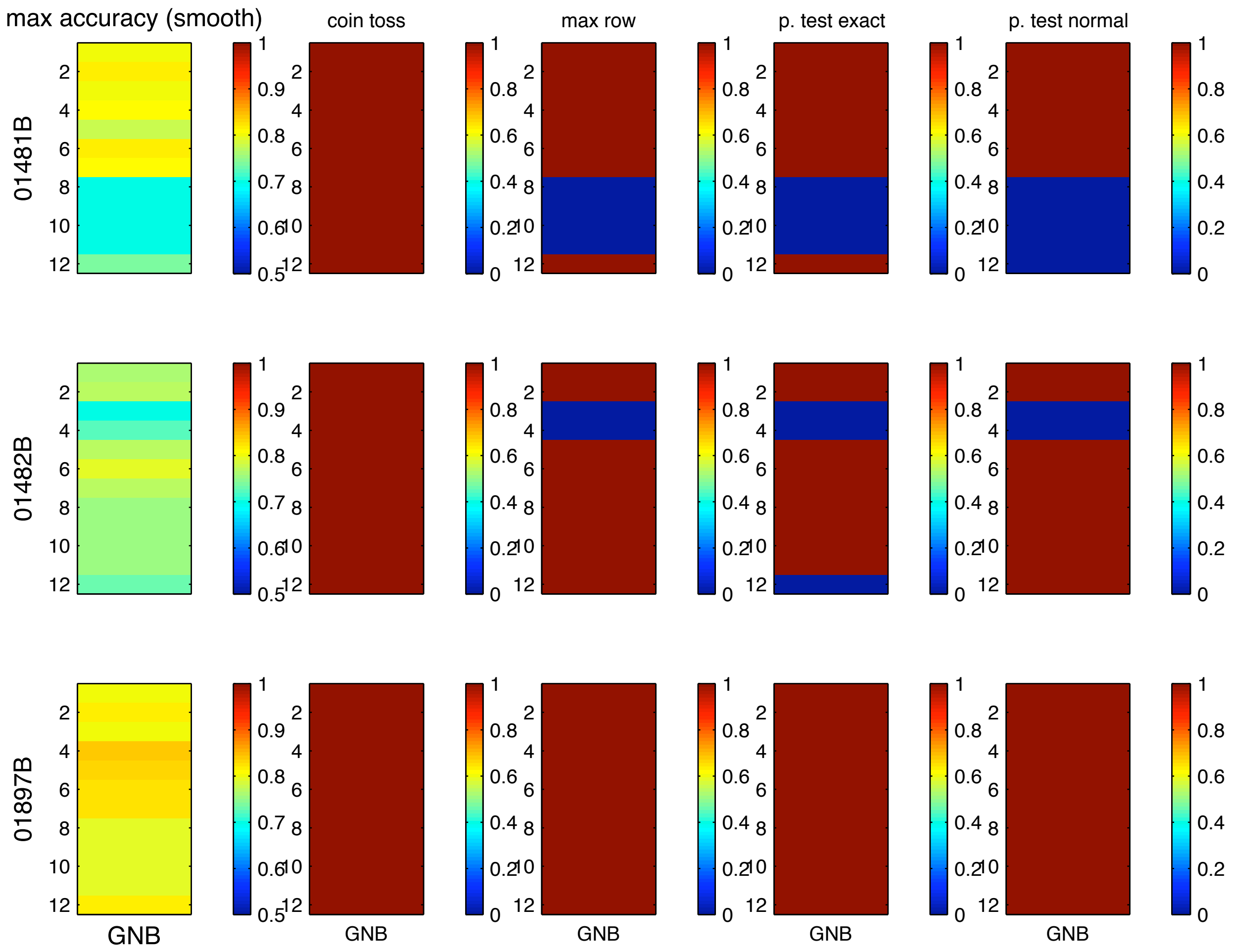


Figure 2.1: Comparison of result significance under various null models, for subjects in dataset D1 (rows). The first column contains the accuracy results using the best number of voxels selected with each of the 12 methods. The remaining columns contain the significance of the results in the first column under each of the null models (red is significant, blue is not).

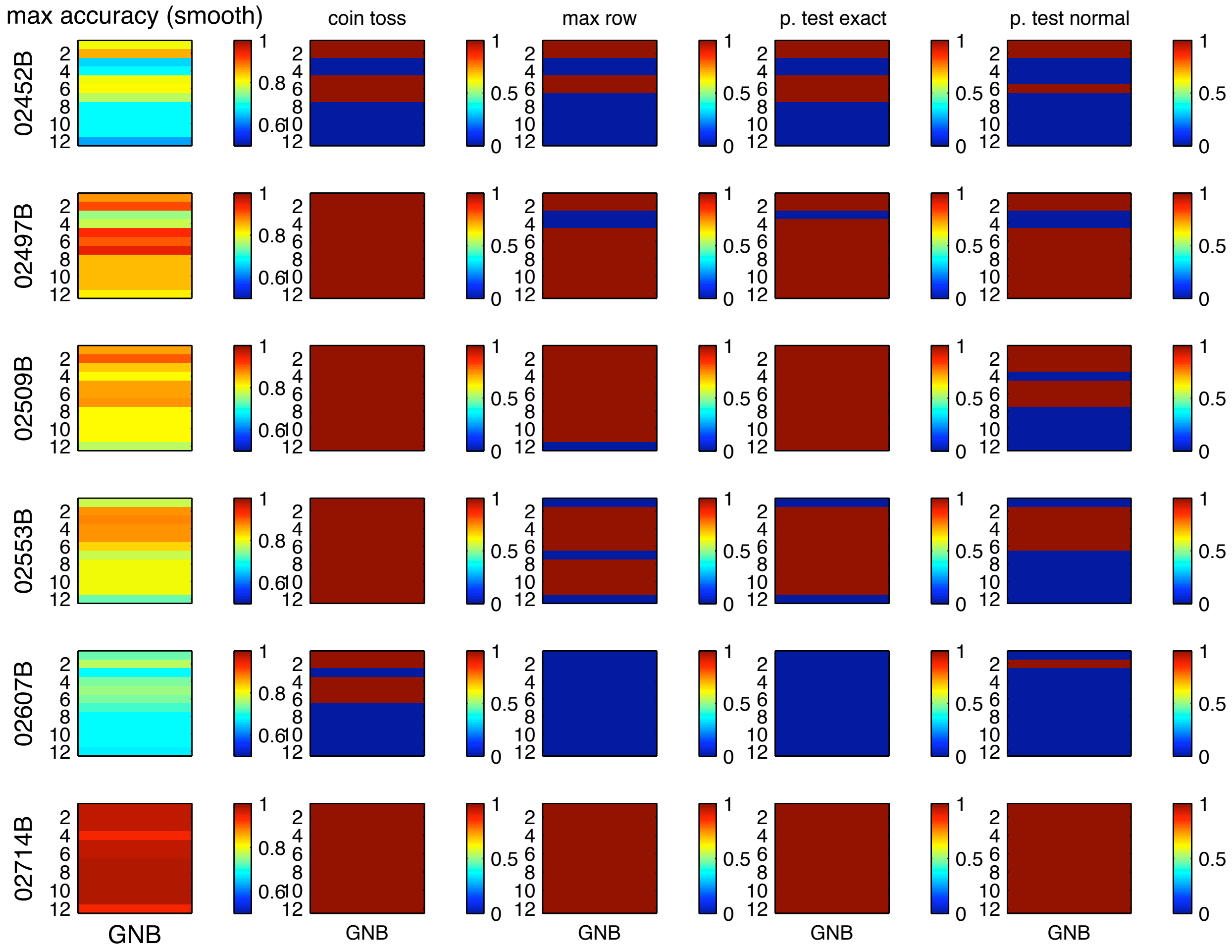


Figure 2.2: Same as Figure 2.1, for dataset D2

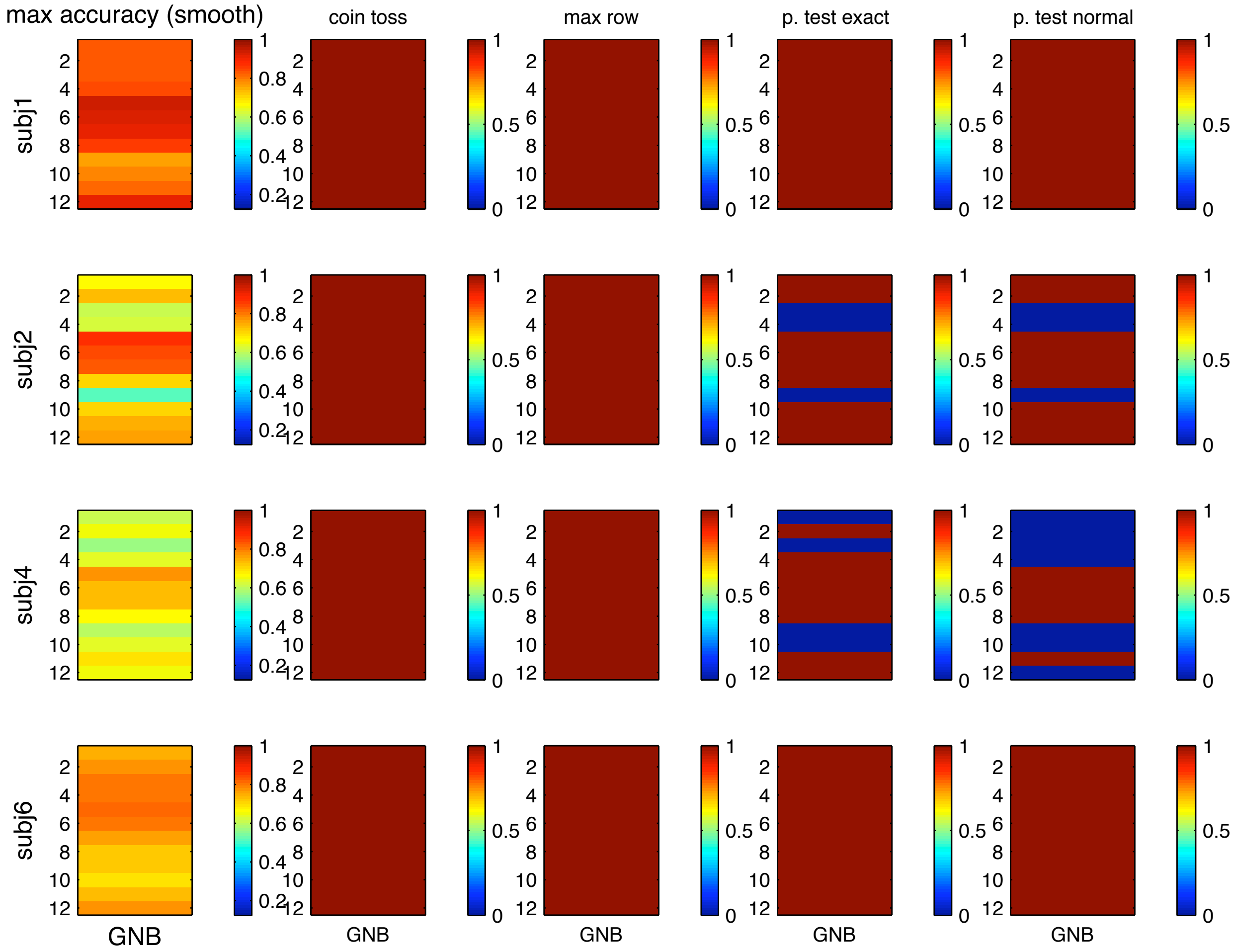


Figure 2.3: Same as Figure 2.1, for dataset D3

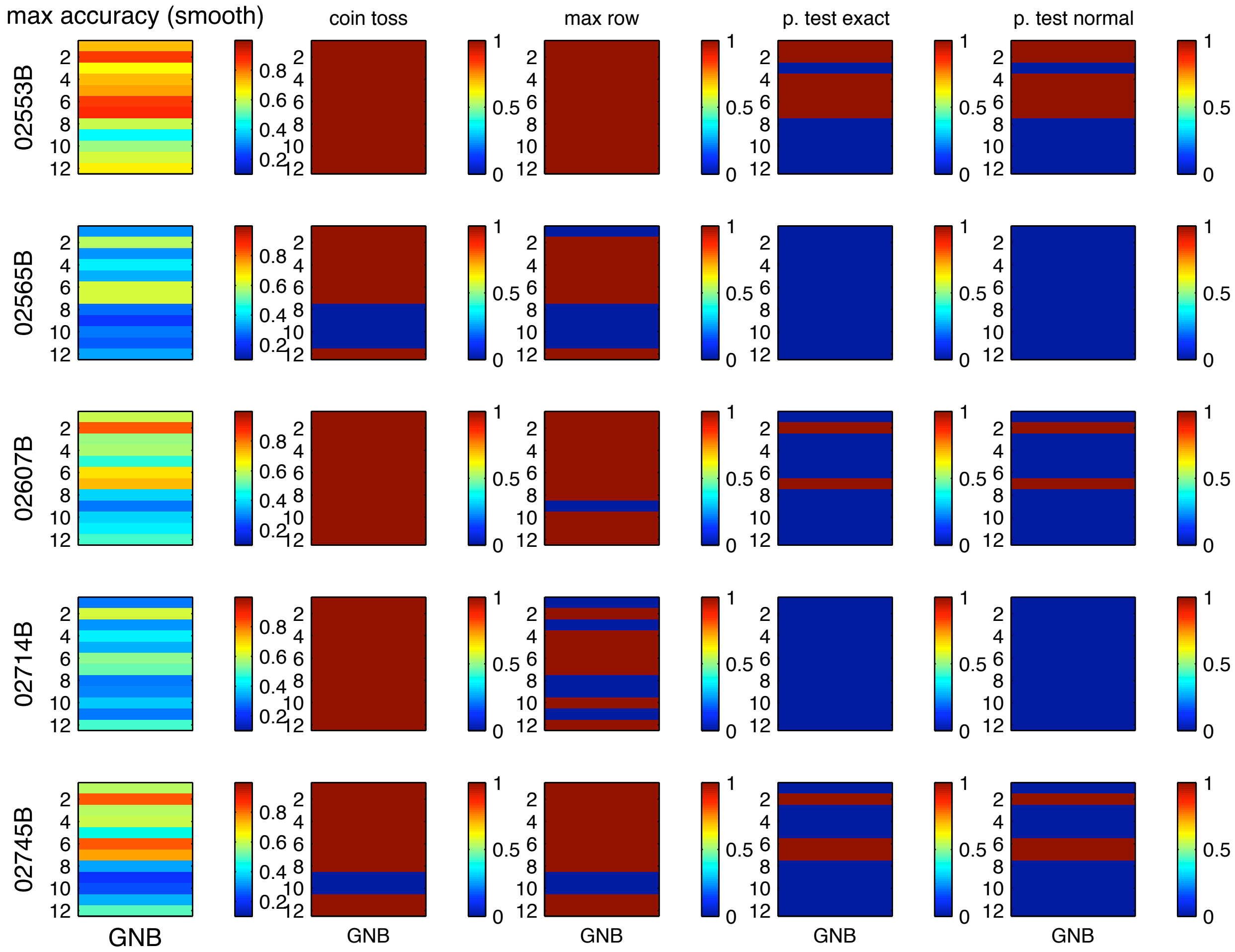


Figure 2.4: Same as Figure 2.1, for dataset D4

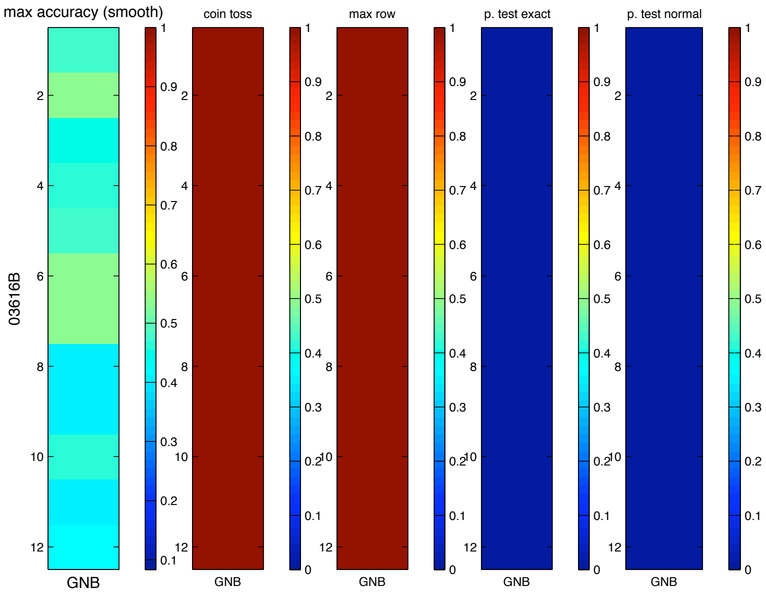


Figure 2.5: Same as Figure 2.1, for dataset D5

Significance of smoothed table versus nested crossvalidation

In Section 2.2.4 we concluded that the NCV results should not be directly compared with those obtained by picking the maximum of the rows of the smoothed table (MSR), since the former are unbiased accuracy estimates while the latter are the maxima of several correlated estimates.

On the other hand, the NCV procedure is one for which "coin toss" is the correct null model under the null hypothesis that there is no information to be decoded. Therefore, another way of validating null models for the MSR procedure is to compare what they deem significant with what was deemed significant for NCV.

Figure 2.6,Figure 2.7,Figure 2.8,Figure 2.9 and Figure 2.10 compare the accuracy and significance for MSR (first two columns) with those of NCV (following two columns), across all the subjects in each dataset (rows), for results obtained using the GNB classifier. From the figures it's clear that, roughly, the results coming from the same voxel selection methods are significant or not significant in both NCV and MSR.

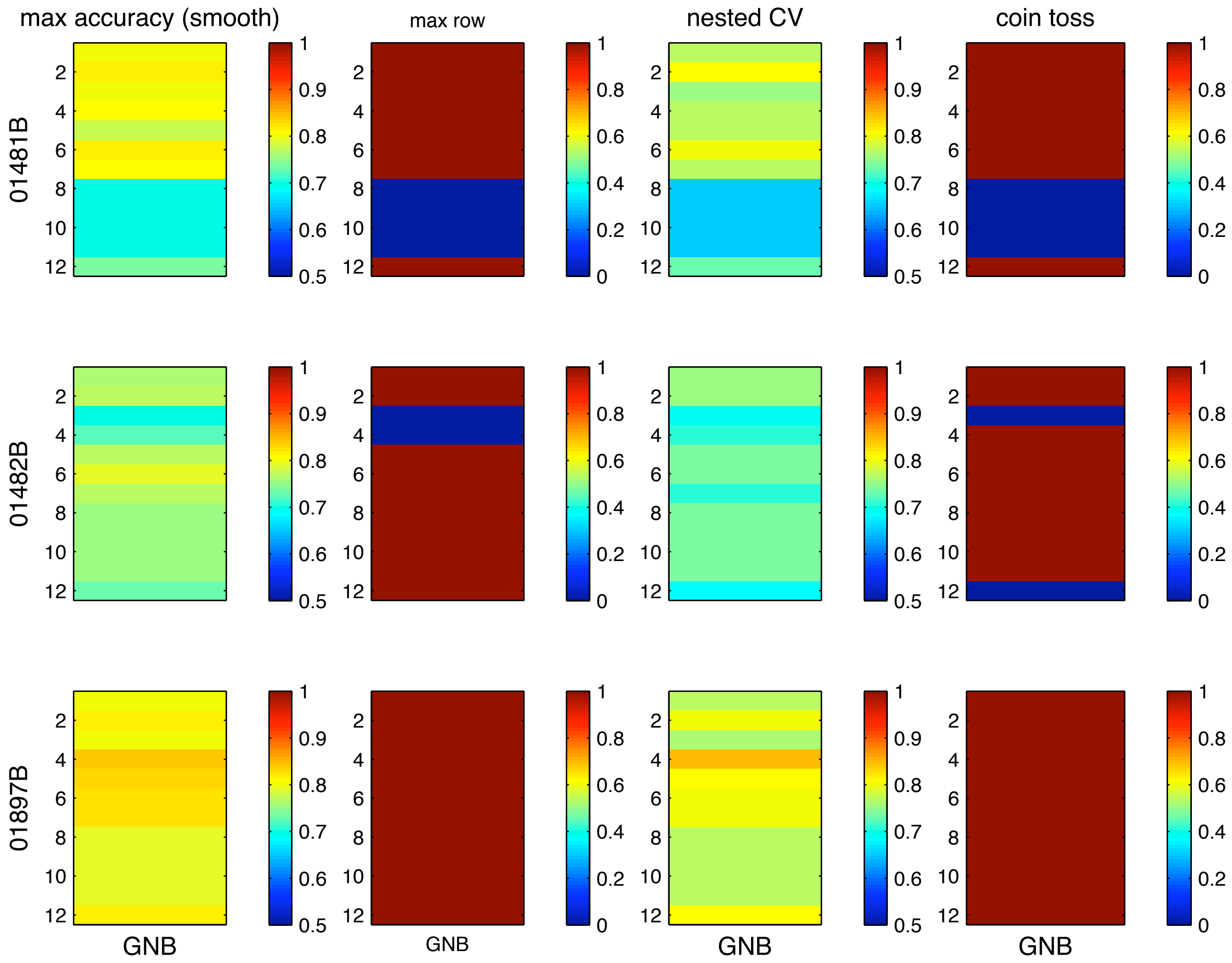


Figure 2.6: Comparison of significance of smoothed table max with nested CV, for all subjects (rows) in dataset D1. The first column contains the accuracy results using the best number of voxels selected with each of the 12 methods, and the second column contains the significance of those results under the max row null model (red is significant, blue is not). The third column contains the accuracy results using nested cross-validation to select the number of voxels to use. The fourth column contains the significance of those results under the appropriate null model in that case, coin toss.

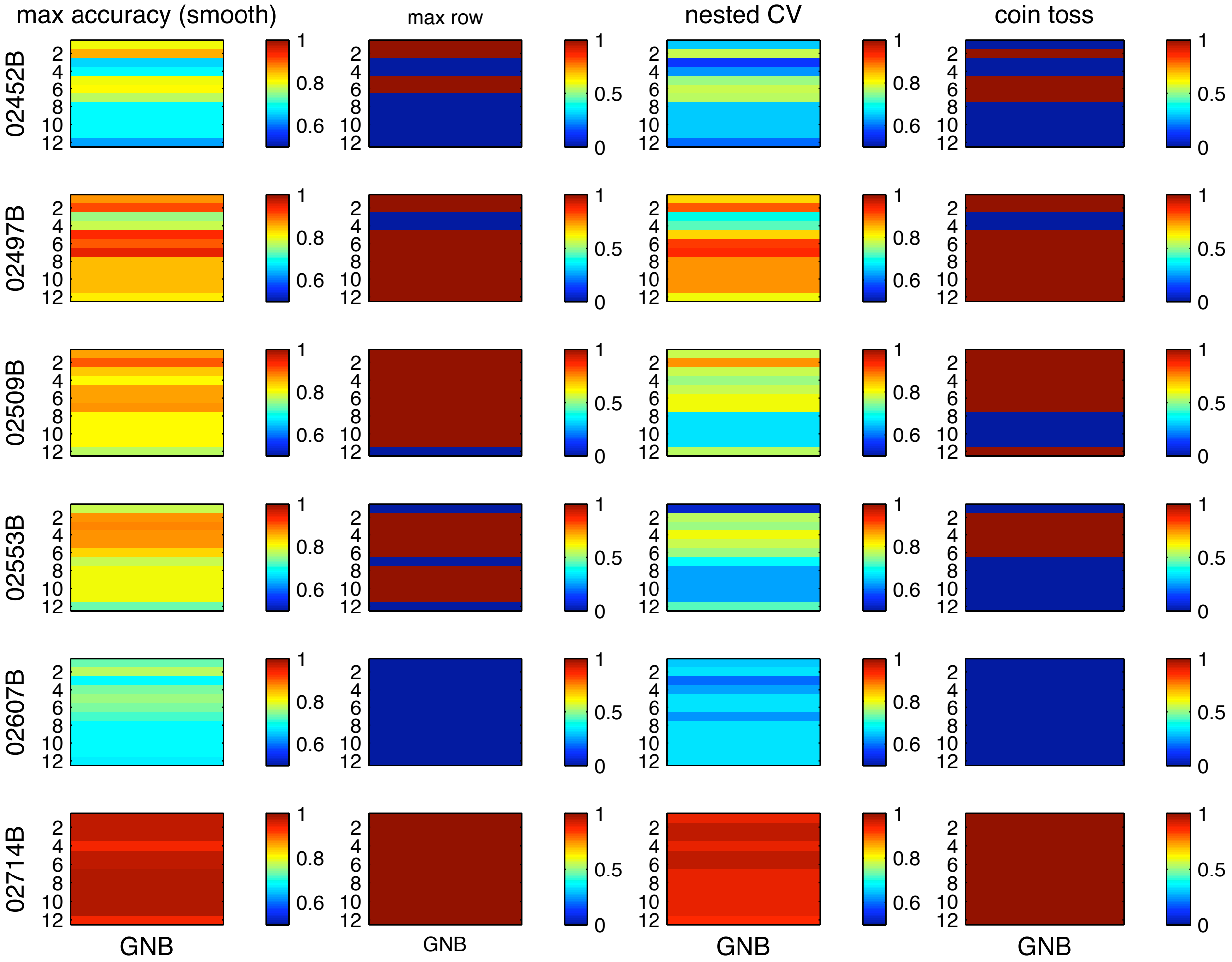


Figure 2.7: Same as Figure 2.6, for dataset D2

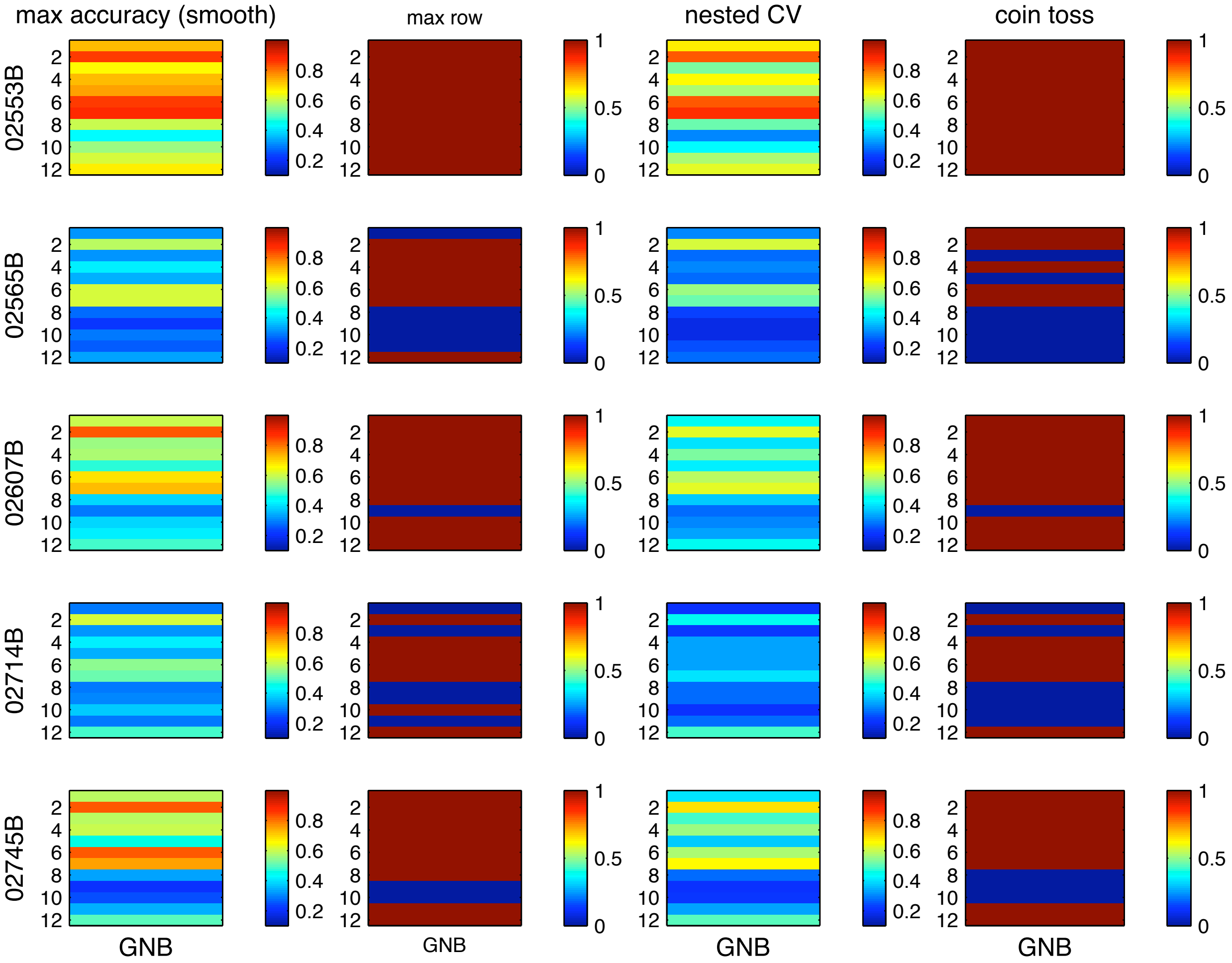


Figure 2.8: Same as Figure 2.6, for dataset D3

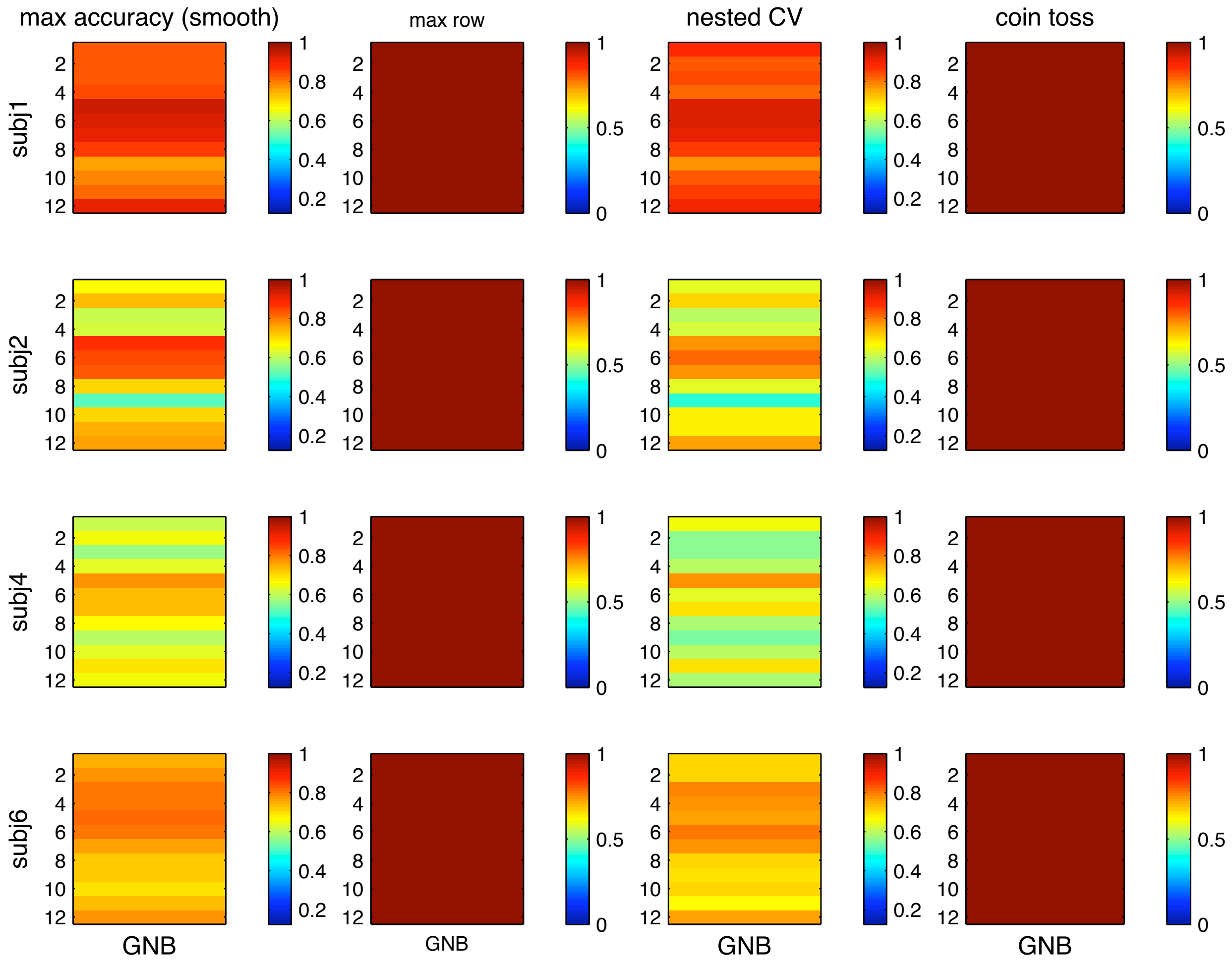


Figure 2.9: Same as Figure 2.6, for dataset D4

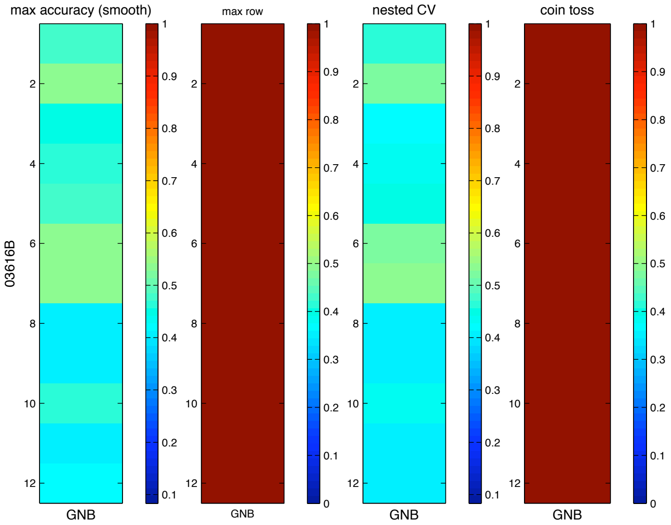


Figure 2.10: Same as Figure 2.6, for dataset D5

2.3.4 Results across classifiers and significance

Section 2.3.3 showed us that what the “max row” null model deems significant lies between what coin toss null model does (optimistic) and what a permutation test null model does (pessimistic). In tandem with that, Section 2.3.3 shows that significance of the maximum accuracy of each voxel selection method result table row under “max row” is quite close to the significance of the nested cross-validation accuracy for each voxel selection method under its *correct* null model. Therefore, it seems reasonable to use “max row” rather than coin toss (for reasons of principle) or the permutation test (for reasons of cost), unless we are computing the significance of relatively few things, in which case the latter should be preferred.

This allows us to answer the question of significance not just when we have one set of results (one per voxel ranking method) but several for each subject (best using a number of voxels for each ranking method and classifier). Additionally we can compare this with what would have been declared significant under the “coin toss” null model.

Figure 2.11, Figure 2.12, Figure 2.13, Figure 2.14 and Figure 2.15 test the accuracy for MSR (first column) according to the two null models (second and third column) across all the subjects in each dataset (rows). The set of results for each subject is a matrix with as many rows as voxel ranking methods and as many columns as classifiers. The classifiers being considered are, in order, Gaussian Naive Bayes, Logistic Regression with L1 regularization, Logistic Regression with L2 regularization and a linear Support Vector Machine.

Each null model is used to obtain p-values for the values being tested. These are then thresholded with the p-values for all methods and subjects considered together while using a bonferroni correction with significance level 0.01.

The top results are significant regardless of which null model is used and generally significant across classifiers, but there are interesting differences for lower results which would usually have been deemed above chance.

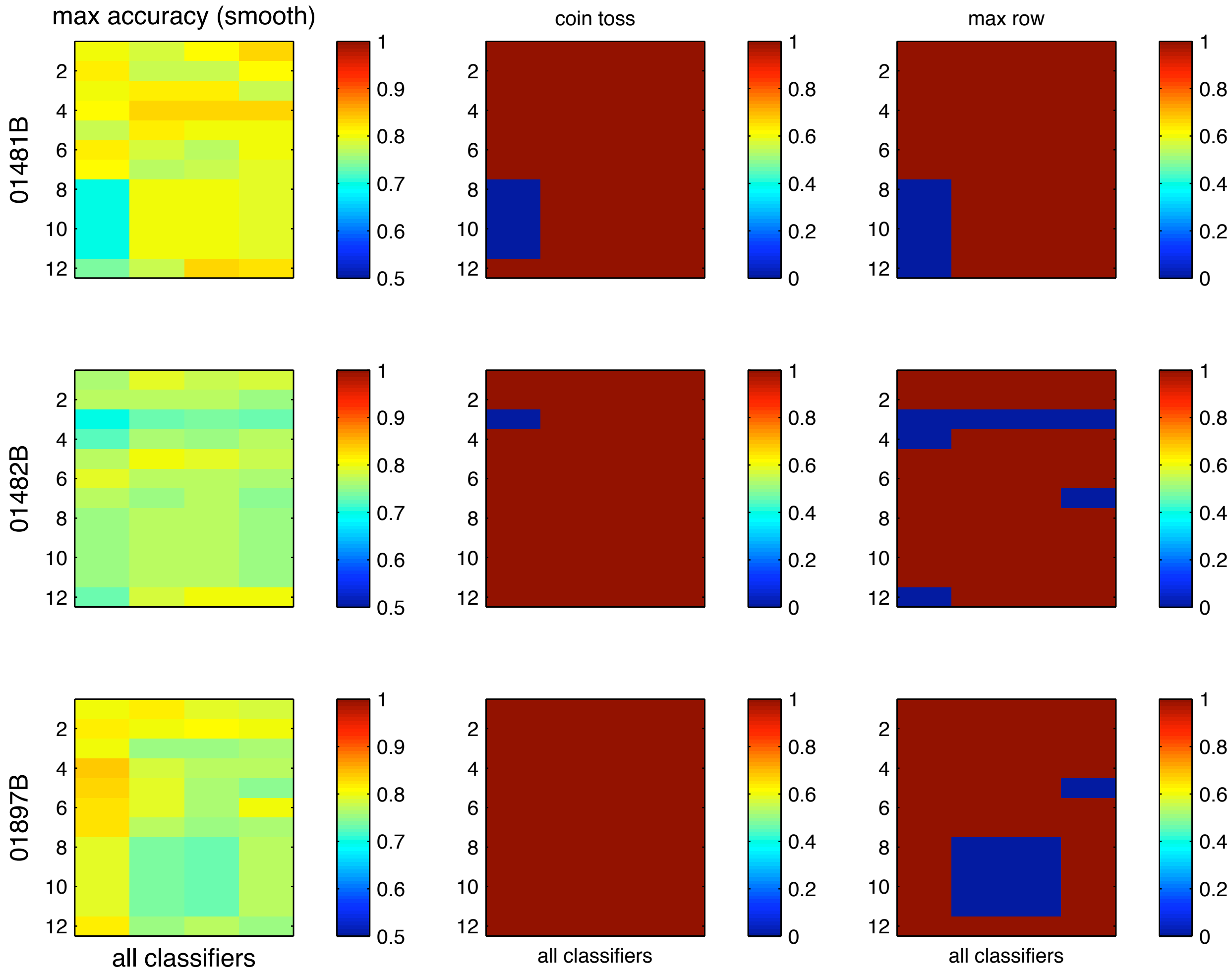


Figure 2.11: Significance of the accuracy at the best number of voxels picked using each method, for several classifiers (left column), using the coin toss (middle column) and max row null (right column) models, for dataset D1 results (one subject per row). Significance is denoted by red, no significance by blue.

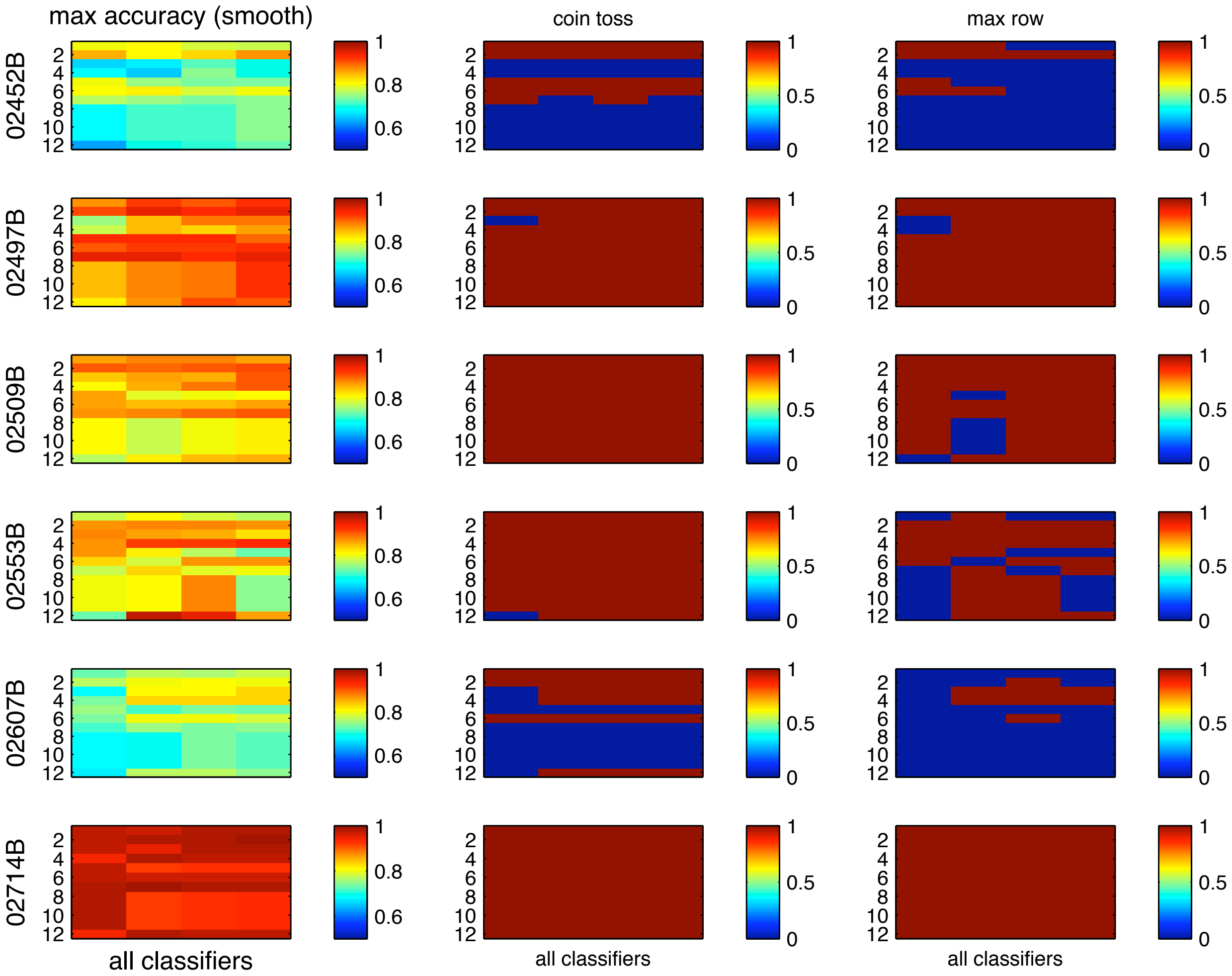


Figure 2.12: Same as Figure 2.11, for dataset D2.

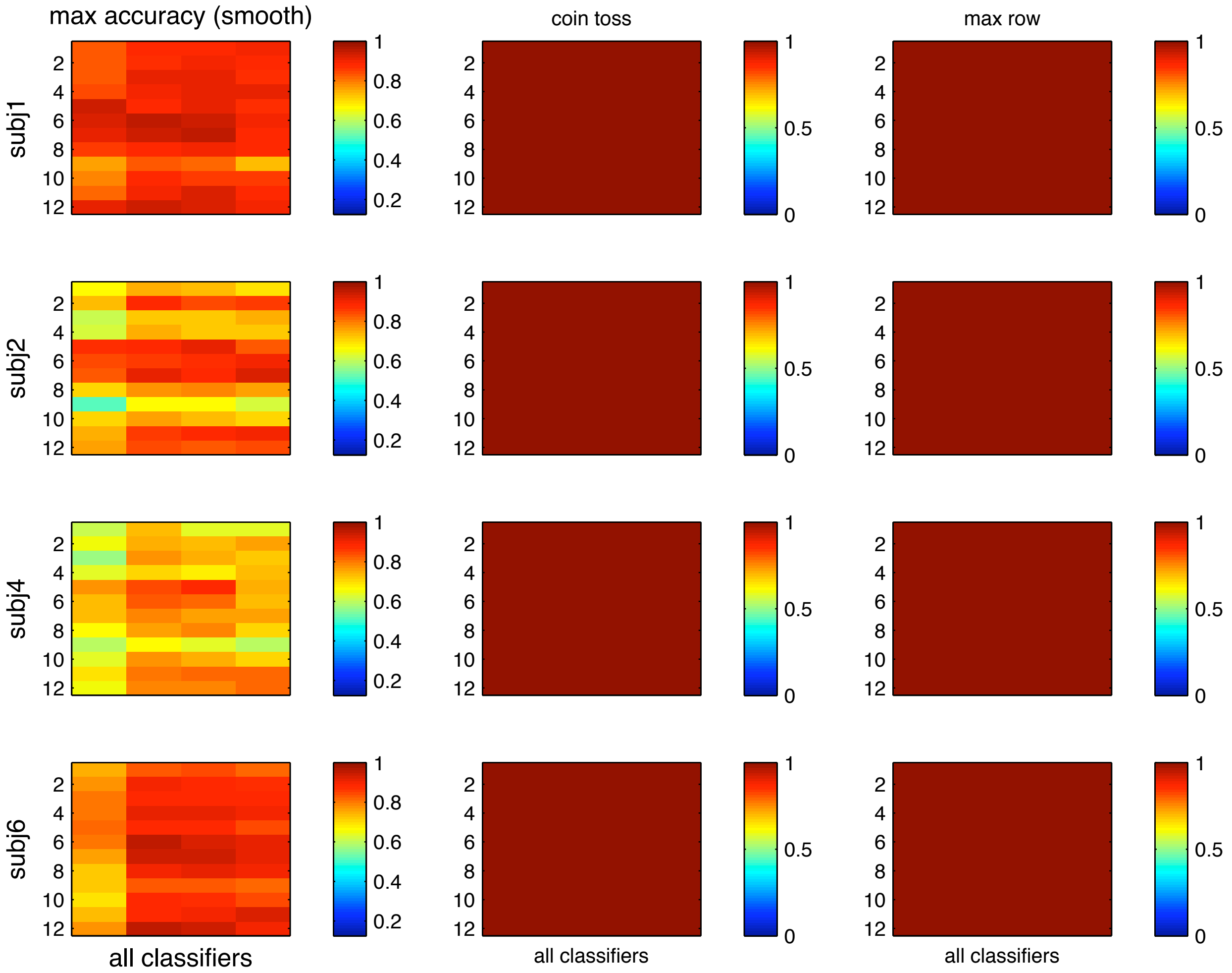


Figure 2.13: Same as Figure 2.11, for dataset D3.

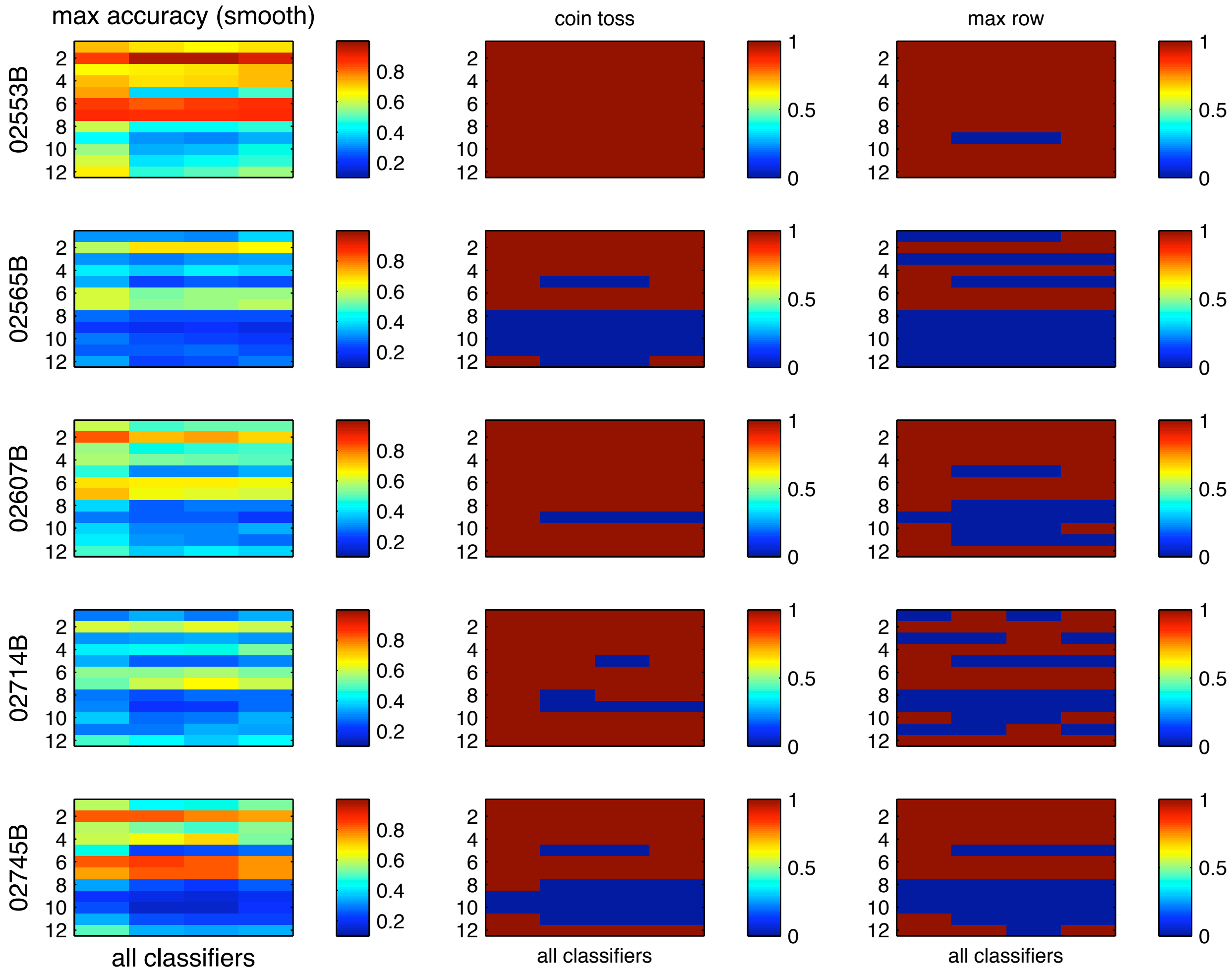


Figure 2.14: Same as Figure 2.11, for dataset D4.

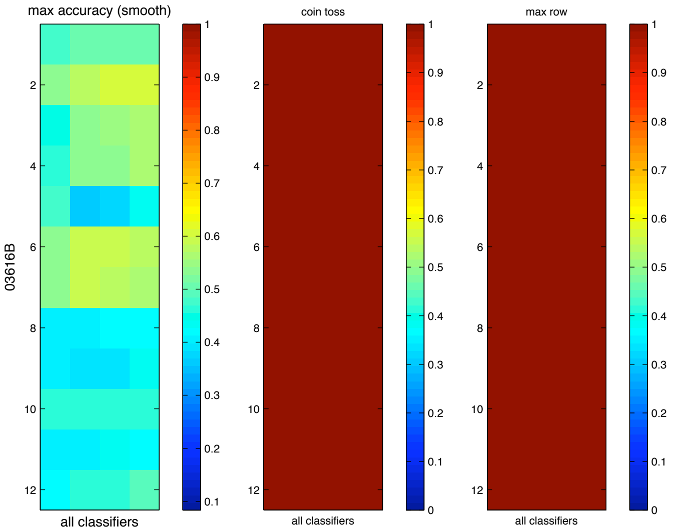


Figure 2.15: Same as Figure 2.11, for dataset D5.

2.3.5 Effects of classifier and voxel selection method choice

Given the multi-classifier results described in the previous section one can ask a different question, namely how choices of classifier or voxel ranking method could be compared. The first question one can ask is whether the choice for one of these factors has an effect across all possible choices of the other. The second is whether their *interaction* matters and thus whether any combination of choices is particularly better or worse than all others. The following two sections attempt to answer these questions, across subjects and datasets.

Effect of either choice

Consider a set of variables $E_1, \ldots, E_l$, one per example being tested, taking the value 1 if the l^{th} example was classifier correctly and 0 if not. In Section 2.2.5, we discussed how this set of variables can be viewed as a sample and how, if we want to compare the means of two such sets created from the decisions of two classifiers on the *same* test examples, a paired t-test can be used. Given that the mean of each sample is the classifier accuracy, this comparison of sample means is an accuracy comparison.

This can be generalized to comparison of the decisons made by a particular classifier using the voxels selected by a particular voxel selection method, across all possible choices of each; one way to do to it is to treat the problem as a two factor ANOVA - classifier vs selection method -

where the sample for each level of either factor - a choice for each thing - is the set of decision variables above. Because all the samples were obtained on the same test examples, this has to be a *within-subjects* two-factor ANOVA [44].

Figure 2.16, Figure 2.17, Figure 2.18, Figure 2.19 and Figure 2.20 show the maximum results for the subject(s) in each dataset (similarly to the matrices in the previous section), and the results of the ANOVA. For each subject, the title of the graph will contain *row* and/or *col*, depending on whether there is a significant difference across rows (methods) or columns (classifiers). The threshold for significance is obtained with a bonferroni correction to take into account the number of subjects, for an overall level of 0.01.

In almost all subjects and datasets there is a significant effect of voxel ranking method (row), in that the range methods (row 8-12) have worse scores than the others (with the exception of logistic regression weights (row 5), in datasets D2,D4 and D5). It appears that searchlight accuracy is almost always better than accuracy, but other contrasts are unclear.

Datasets D3, D4 and D5 (multiclass problems) have a significant effect of classifier choice (row) for at least half the subjects, with GNB being worse in dataset D3, better in D4 and worse in D5 (though the worse voxel ranking methods account for most of this).

Effect of their interaction

We also want to consider the interaction of the choice of voxel ranking method and classifier and have an idea of how well each combination does relative to all the others. Consider the indicator variables that say whether each example was labelled correctly (see Section 2.2.3) by a given classifier trained on the features selected with a given method. We can do a paired t-test on that set of indicators against the set of indicators produced by all other choices of method and classifier. This produces a set of p-values and a set of mean differences between every pair of choices. The set of p-values can then be thresholded using False Discovery Rate (given the large number of tests being performed and correlated results) to decide on a threshold to give an adjusted rate of $\frac{0.01}{\#subjects}$.

While this can be used to list all the combinations of pairs of method/classifier that have a significantly different accuracy (as done in [43] for the best scoring classifier versus others, for instance), it does not provide an obvious summary. Our approach for addressing this problem is the plot in Figure 2.21 (for dataset D4), which simultaneously plots the results of testing every combination of classifier and voxel ranking method against any other.

The figure has as many rows as voxel ranking methods, and as many columns as classifiers. Each cell i, j on that grid thus corresponds to a particular combination of voxel ranking method and classifier, and contains the results of the tests of that combination against all others.

Given the level of detail in the graph, it's better to consider a particular i, j more closely; Figure 2.22 is a magnified version of the top-left corner of Figure 2.21, and we will look at one of

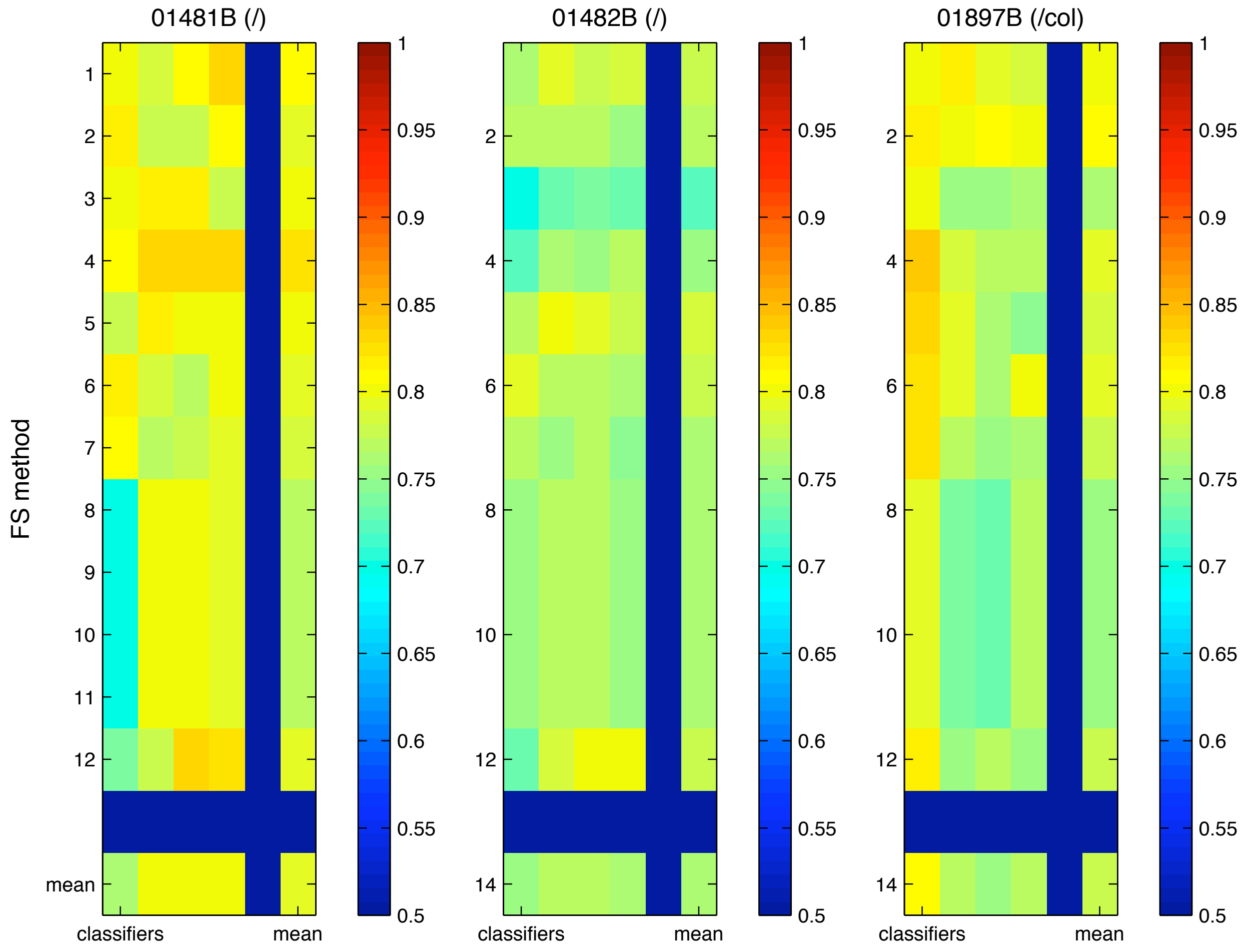


Figure 2.16: ANOVA of best classification results for each combination of voxel selection method (12 rows) and classifier (4 columns) on dataset D1, the four plots correspond to four subjects. The colour in each cell corresponds to the maximum accuracy attained by that combination (see scale). On the bottom and the right of each plot there is a dark blue row/column (padding) after which comes an extra row/column containing the mean of each column/row. The title of each subject plot contains the subject name and also encodes whether there was a significant effect of voxel selection method, if the word "row" appears, classifier, if the word "col" appears, or both.

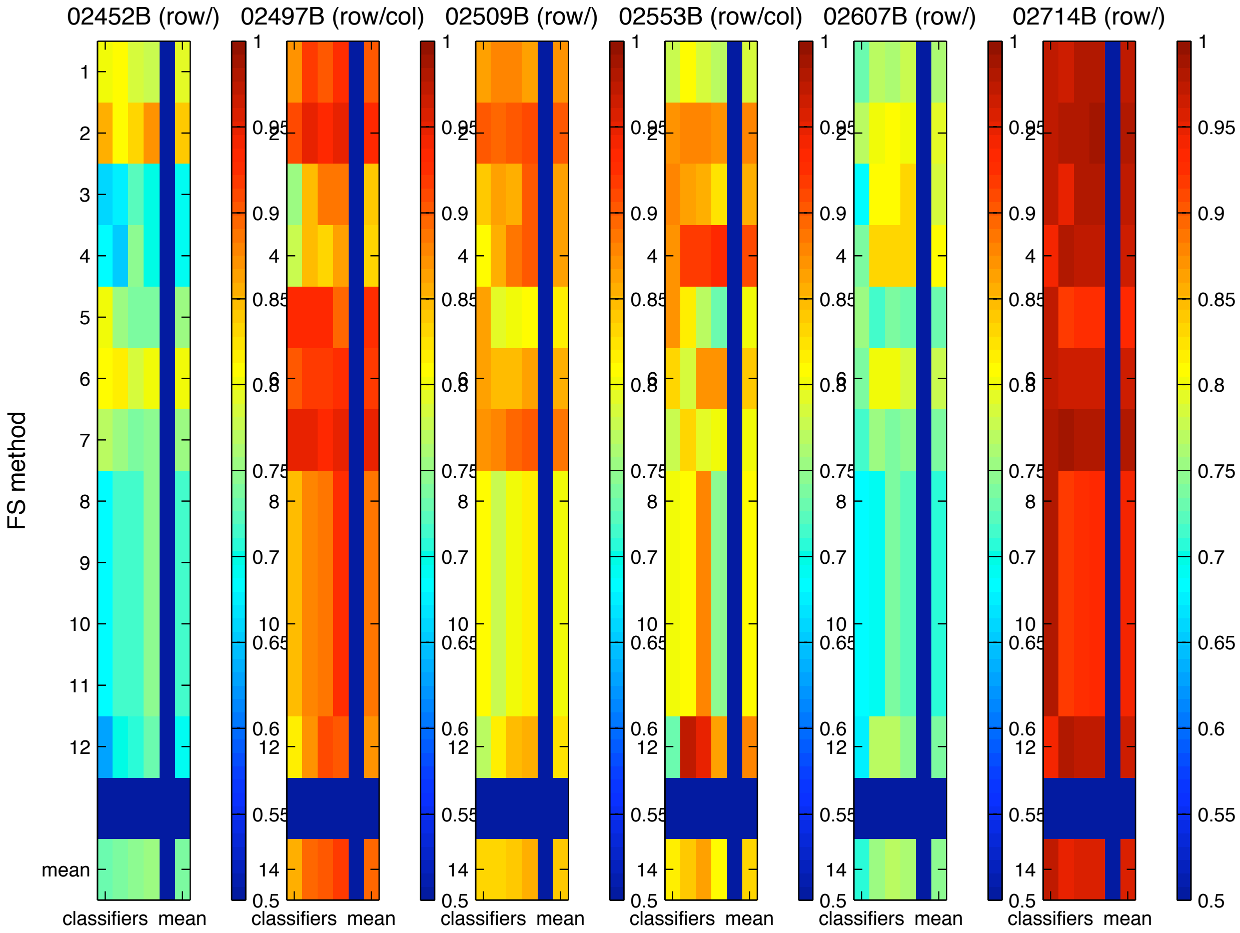


Figure 2.17: Same as Figure 2.16, for dataset D2

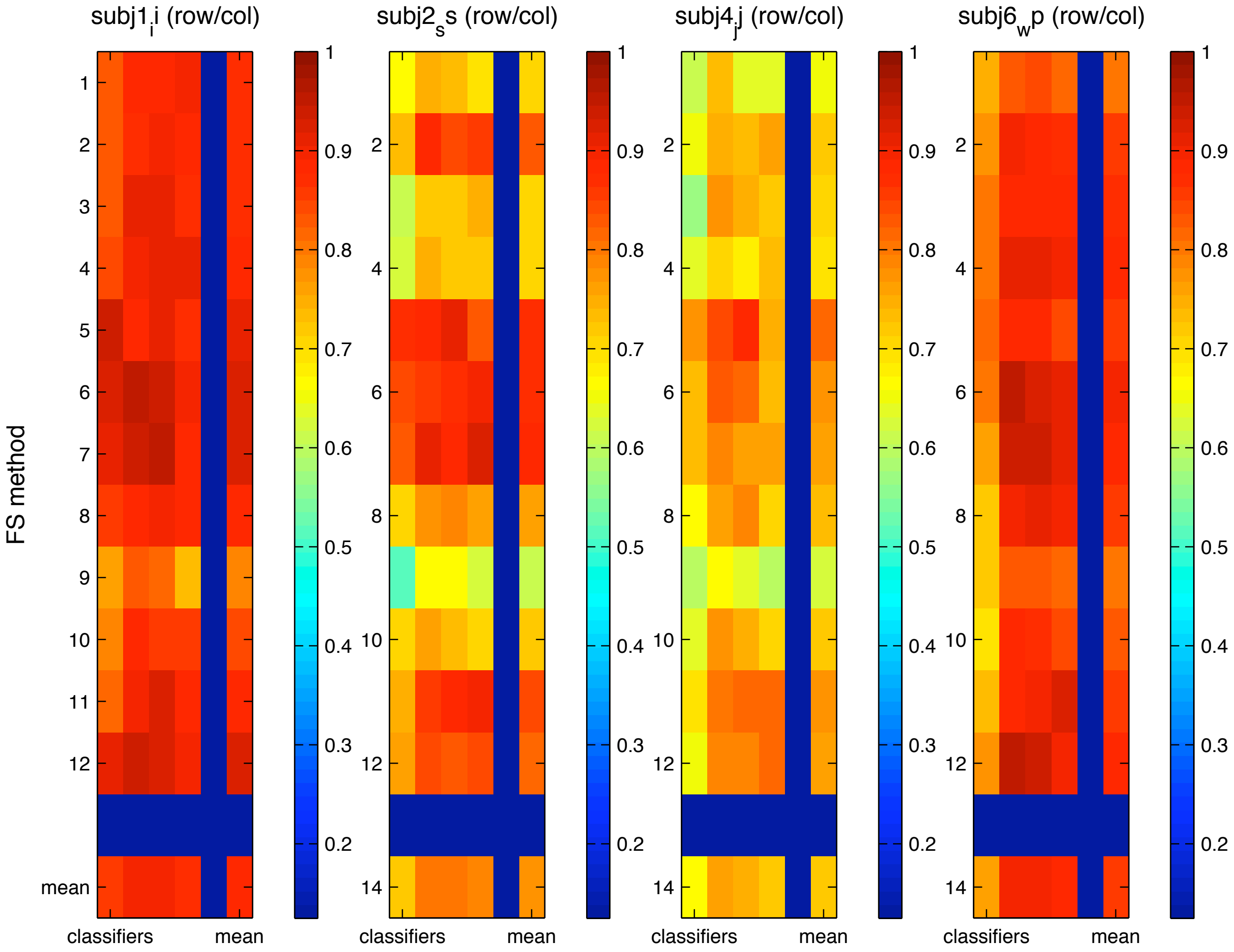


Figure 2.18: Same as Figure 2.16, for dataset D3

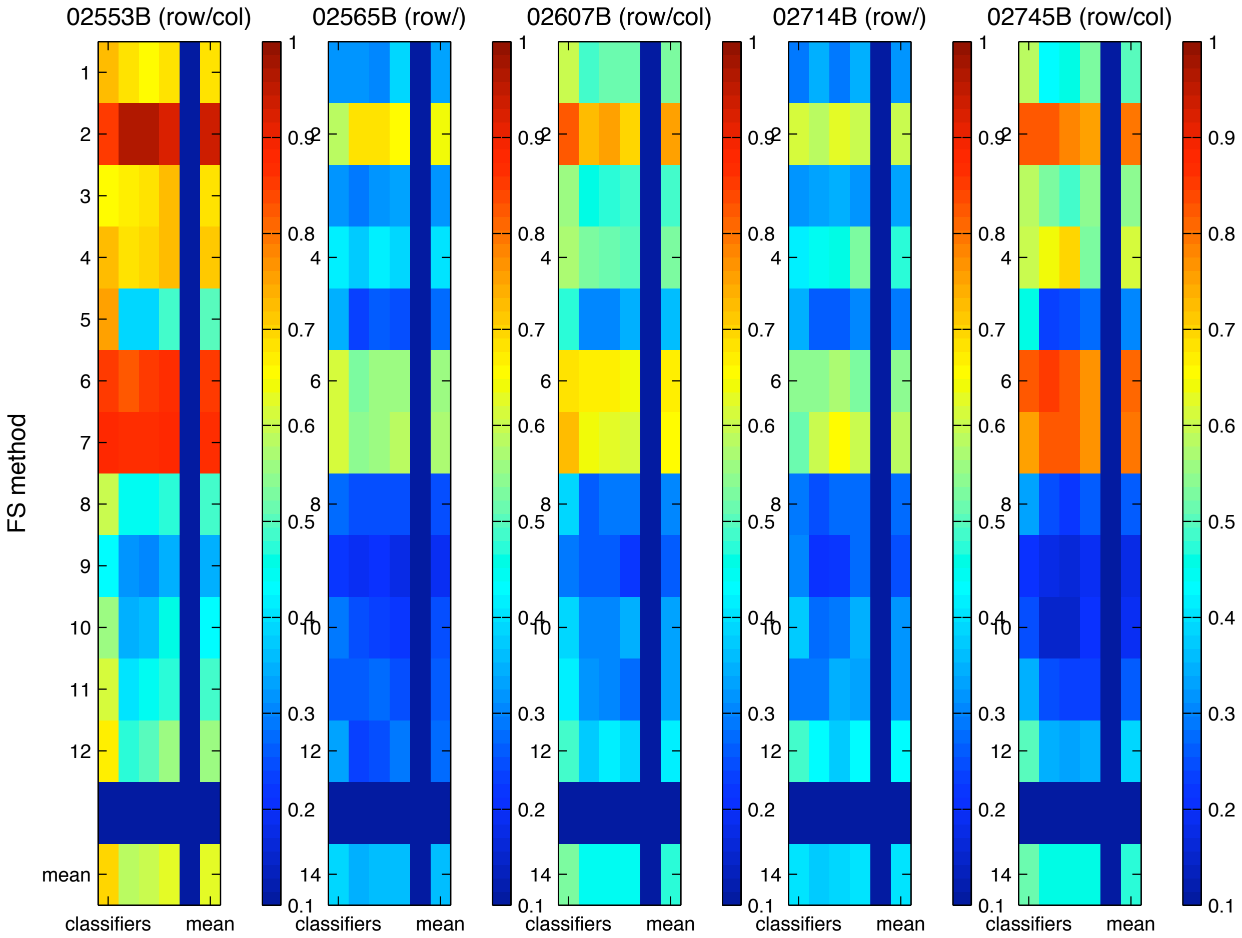


Figure 2.19: Same as Figure 2.16, for dataset D4

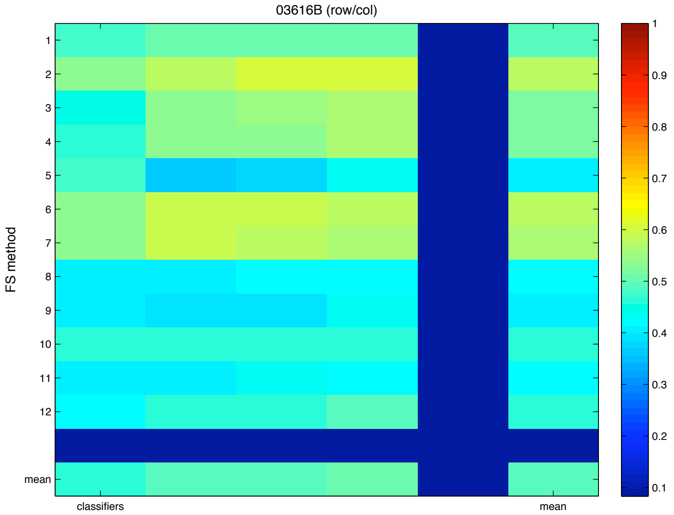


Figure 2.20: Same as Figure 2.16, for dataset D5

its plots, cell $1, 2$ (first row, second column).

Cell $1, 2$ is divided into column subplots, with the first one pertaining to the first subject, second one to the second subject, etc. Each of those subplots is a miniature plot with entries corresponding to the dimensions of Figure 2.22, with as many rows as voxel selection methods and as many columns as classifiers, with cells indexed by k, l. It captures the significance of comparisons between the $1, 2$ combination and all others, for that subject. In more detail, entry k, l in the subplot for $1, 2$ is 0 (light green) if the mean difference between the $1, 2$ and k, l samples was not significant, and has a positive (red) or negative value (blue) reflecting the sign and magnitude of a significant difference.

In this particular example, dataset D4, it's clear that for most subjects the range methods are worse than almost all others, while among the other methods 2 (searchlight accuracy), 6 (ANOVA) and 7 (replicability) work the best.

The same holds for dataset D5 and, to a lesser extent, for dataset D5. Similar plots for datasets D3 and D5 are given in Figure 2.52 and Figure 2.53, in the appendix of this chapter. Very few pairs are deemed significant in datasets D1 and D2, possibly because both are binary classification problems and most choices achieve reasonably high accuracy, so those datasets are not shown.

Overall there seems to be little difference between classifiers, except for slightly worse results with GNB.

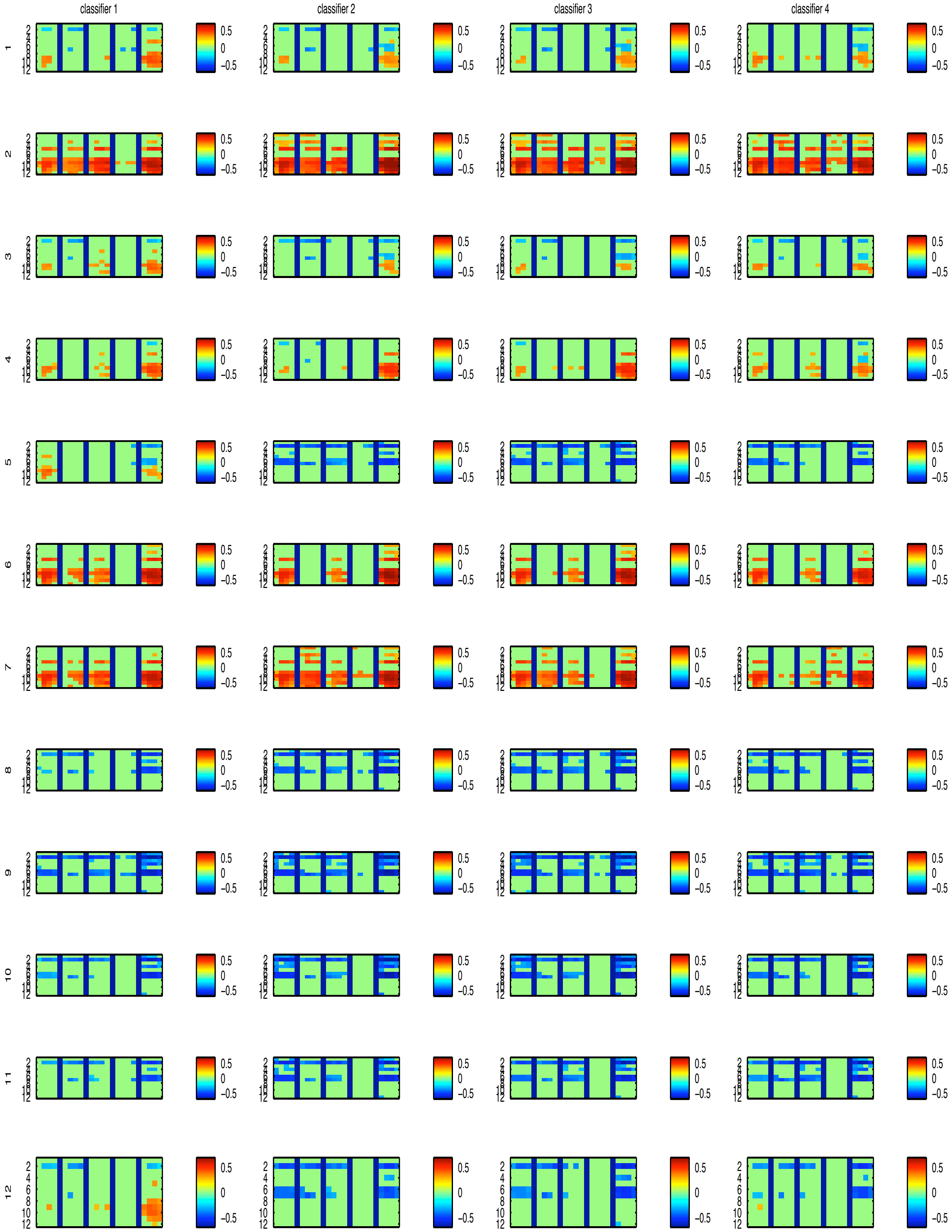


Figure 2.21: Significance of the difference between the accuracy of each combination of voxel ranking method and classifier and all others, for dataset D4. Please see Section 2.3.5 for a detailed explanation and walkthrough.

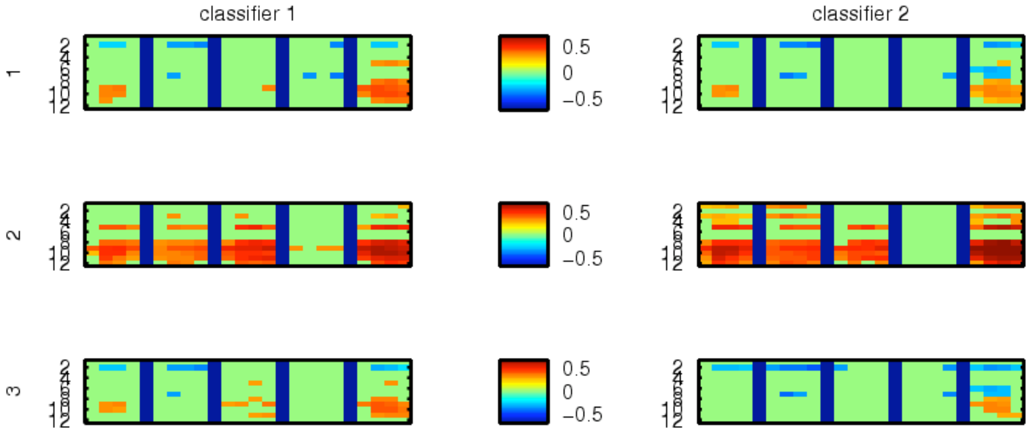


Figure 2.22: Magnified version of the upper left corner of Figure 2.21.

2.3.6 Effects of training dataset size

A natural question in face of the results in the previous sections is whether they would be very different if datasets were smaller. In particular, would certain voxel ranking methods and classifiers be more affected?

In order to examine this question we trained classifiers on a 2-fold cross-validation (even/odd runs or epochs) and compared it with the original 6-fold/12-fold cross-validations; relative to this the 2-fold situation has $40\% - 50\%$ smaller datasets. The results undergo the same smoothing and maximum selection that the original results did.

Figure 2.23 depicts a comparison of these results for dataset D1. It has 4 columns, one per subject, and three rows corresponding to the original results, the even-odd cross-validation results and the change in accuracy. Each cell in the grid contains as many rows as voxel ranking methods and columns as classifiers.

The measure of change of accuracy is $\frac{\texttt{original result}-\texttt{new result}}{\texttt{original result}}$ Figure 2.24, Figure 2.25, Figure 2.26 and Figure 2.27 depict the same information for the other datasets.

Examining these, only binary classification datasets (D1,D2) seem to remain mostly unscathed or even improve slightly. Performance can drop down 20% for many method/classifier combinations in the multiple class datasets (D3,D4,D5). A likely reason for this is the gap in the number of examples per class, which is 42 for D1 and 30 for D2 versus 6 for D3, 3 for D4 and 15 for D5; the fewer the examples, the larger the drop in performance. Even in the face of this, the methods that give the best results with the larger datasets also do so for the smaller ones.

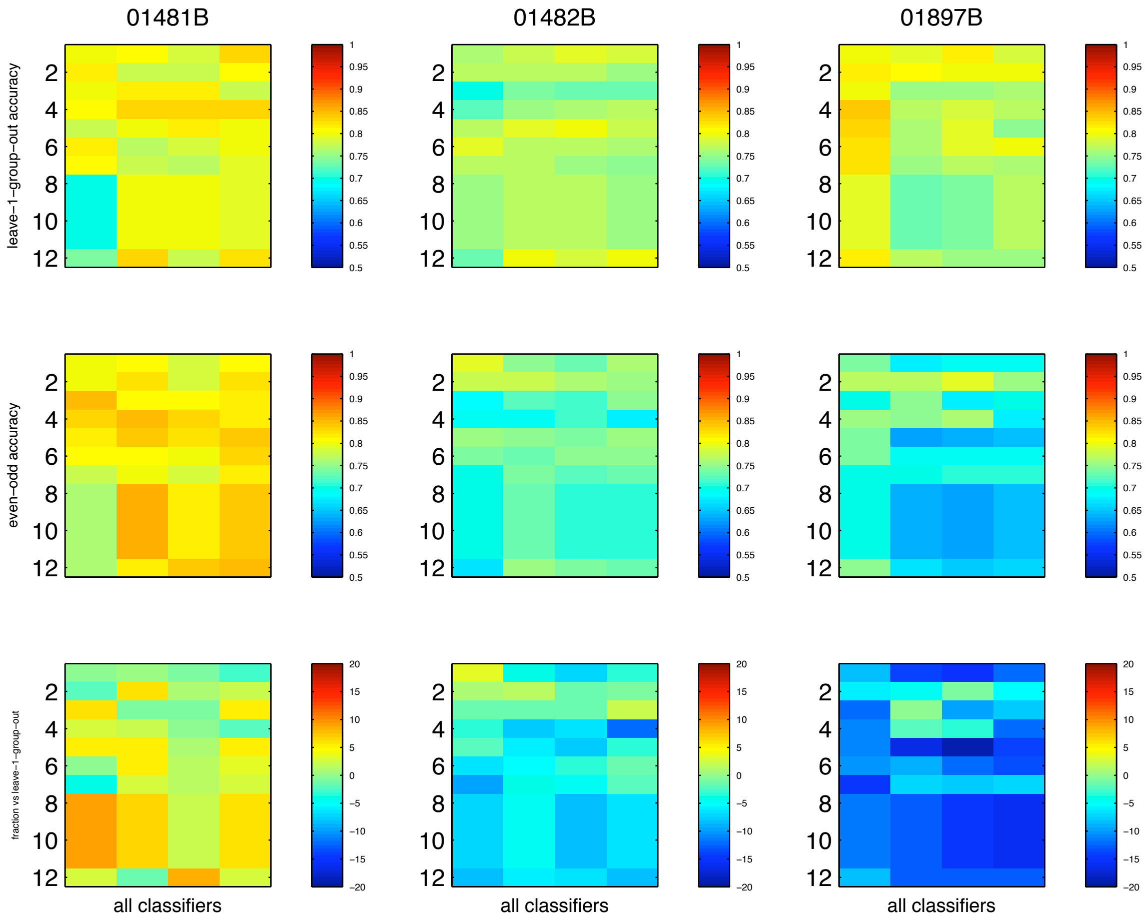


Figure 2.23: **Top:** accuracy across methods and classifiers for all subjects in dataset D1 (columns), using leave-one-out cross-validation. **Middle:** the same, using 2-fold cross-validation. **Bottom:** fraction of change between 2-fold and leave-one-out, relative to leave-one-out.

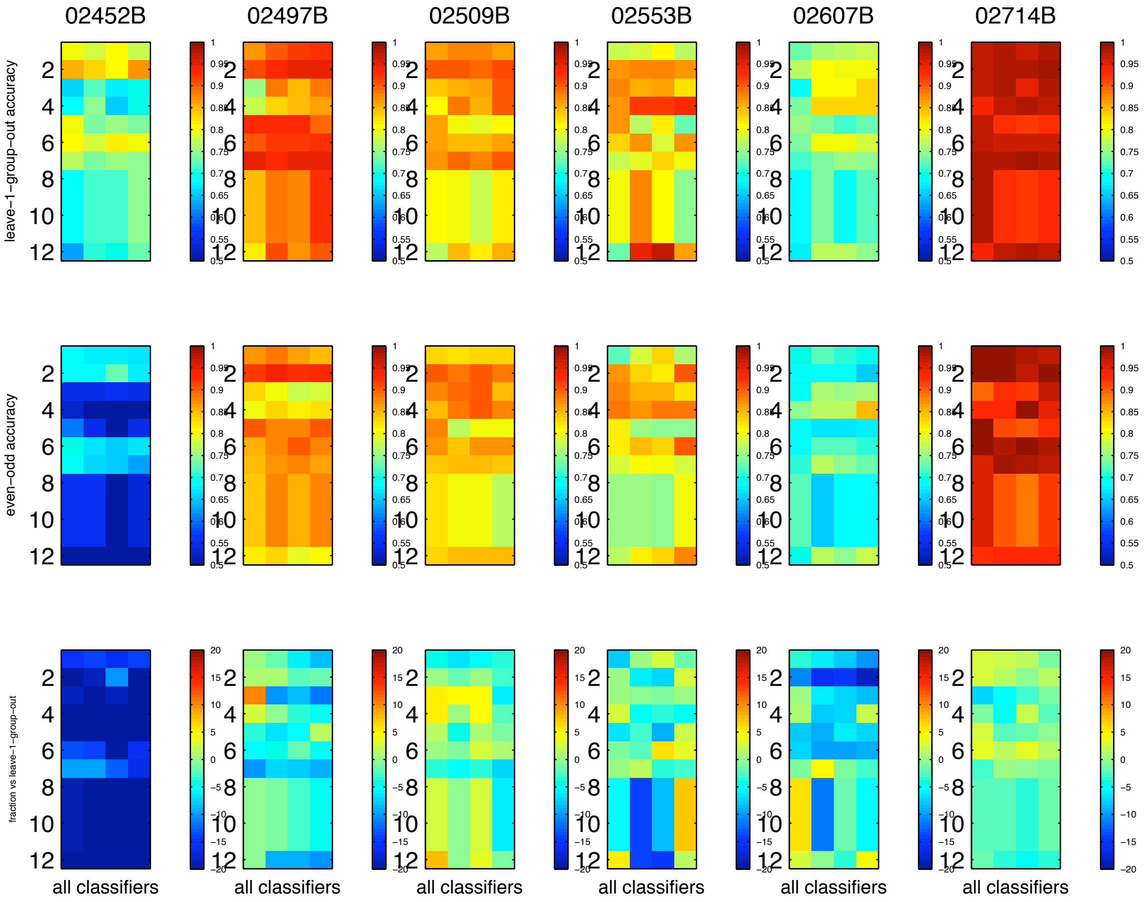


Figure 2.24: Same as Figure 2.23, for dataset D2

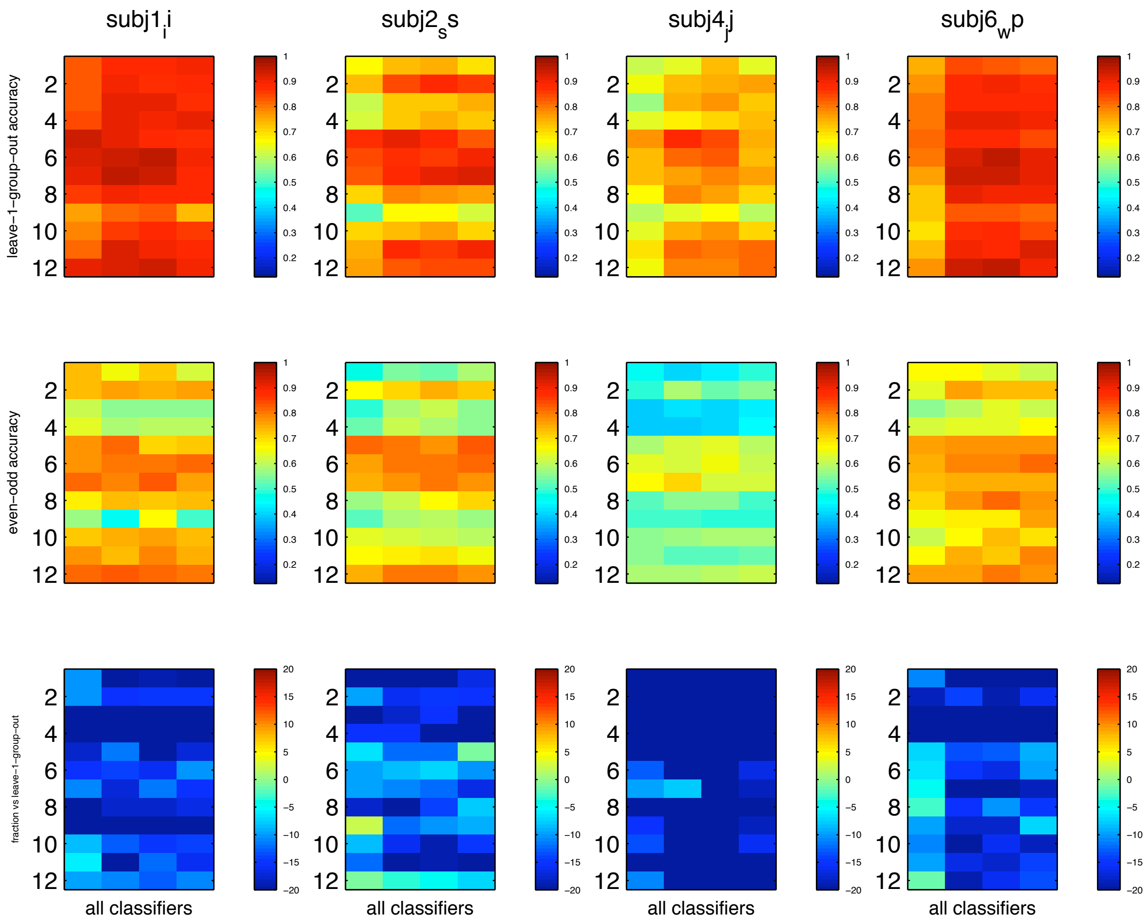


Figure 2.25: Same as Figure 2.23, for dataset D3

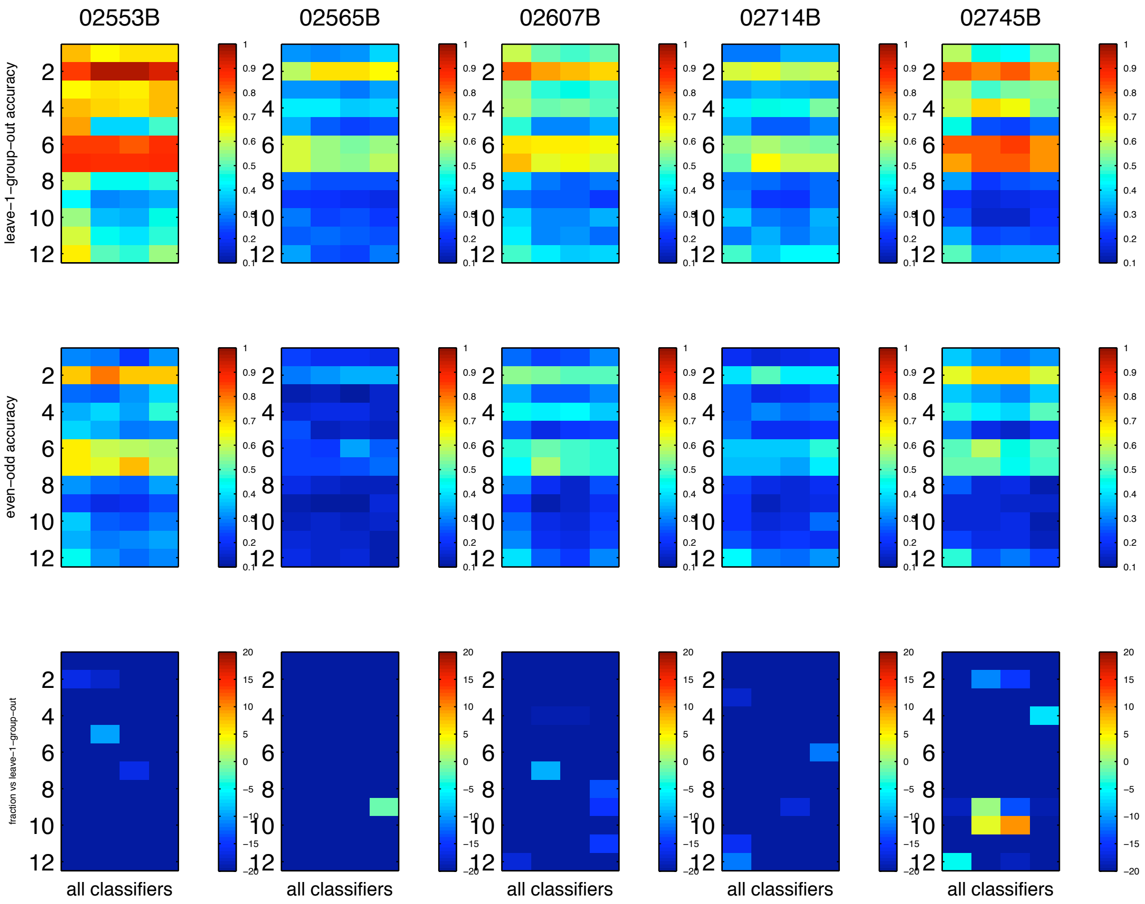


Figure 2.26: Same as Figure 2.23, for dataset D4

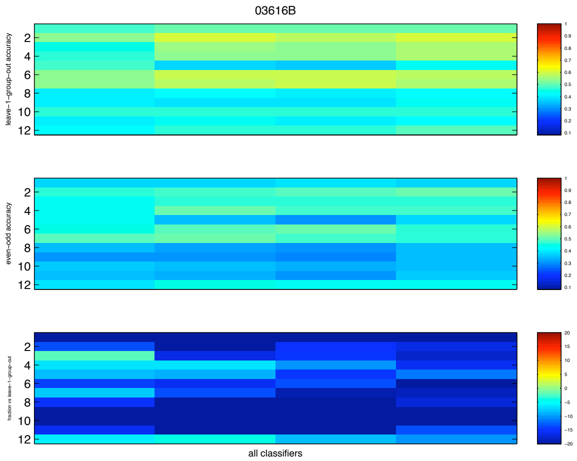


Figure 2.27: Same as Figure 2.23, for dataset D5

2.3.7 Issues with cross-validation and feature selection

Feature selection is done not only for the generic purpose of increasing accuracy but also for a domain-specific reason: to test whether features selected according to a certain criterion (e.g. location or feature behaviour) contain information that can drive a classifier (e.g. [26] and [35]). If this is being done on a single training set and using only one method for selecting features, it's easy to identify the features and examine them for location or behaviour.

More typically, however, this is being done within a cross-validation, and a different set of features can be picked in each fold. This would happen if *redundant* information is present, e.g. think of a pool of voxels where all of them have similar response characteristics across classes. A second issue is whether different feature selection methods draw from *different* pools or whether they identify many of the same features. This section examines the former question by looking at the extent to which selected features overlap across cross-validation folds. The latter question is tackled by considering the overlap of feature sets selected by different methods. Note that this is done in contrast with work such as in [41], where several univariate and multivariate methods are shown to have large differences in the activity surfaces they construct. The more analogous work would be to show different classifiers assigning weights to different voxels. Here we are concerned merely with the voxel selection methods, both within a method and across folds and between methods.

Selected voxel overlap across cross-validation folds

In order to look at the extent to which selected voxels are the same across folds we computed the fraction of overlap between the set of voxels picked on *all* folds and the set of voxels picked on *any* fold.

In an ideal situation where voxels were always ranked in the same way by the same method in all folds, this overlap fraction would be 1. In practice, we can expect a certain number of voxels fulfilling the ranking criterion well and taking turns at being near the top in different folds. Hence we expect the overlap fraction to start low as few voxels are selected (many possible voxels fit the criterion used equally well, only a few are selected in each fold), rise to a peak as the number selected increases (eventually most of the voxels that fit the criterion are in the top) and then lower as random voxels get in.

Figure 2.28, Figure 2.29, Figure 2.30, Figure 2.31 and Figure 2.32 show this overlap for all five datasets. Each figure has as many rows as subjects and two columns. The right column plots contain the smoothed row accuracy at the best number of voxels selected with each method. Each plot on the left column contains as many rows as voxel ranking methods. In each row each entry corresponds to one of the numbers of voxels selected and contains the overlap fraction between the sets selected in each fold. Following a row gives an idea of how the overlap fraction changes with the number of voxels.

The figures suggest two factors at play across datasets. The first is that if examples come from trials in an event-related design there is less overlap than if they are the average of images in a block. The second is that having more examples also increases overlap, at least for the multiclass problem datasets (D3,D4 and D5). The methods based on activity (3,4) are the only ones that have a higher amount of overlap throughout studies, together with the range method that considers maximum activity only. From this it seems reasonable to conclude that level of activation is a more consistent property than those looked at by other methods, even if in the case of the maximum it appears to select more noisy voxels (judging by the corresponding accuracy).

But is the fraction of overlap related to the accuracy in any way? Consider Figure 2.33, Figure 2.34, Figure 2.35, Figure 2.36 and Figure 2.37. In the figure for each dataset there are as many plots as subjects. Each one plots the fraction of overlap in voxels selected across folds against the corresponding accuracy at the best number of voxels, with each point corresponding to a particular method. The figures indicate that increasing overlap fraction is not accompanied by increasing accuracy and, furthermore, suggest no clear relationship.

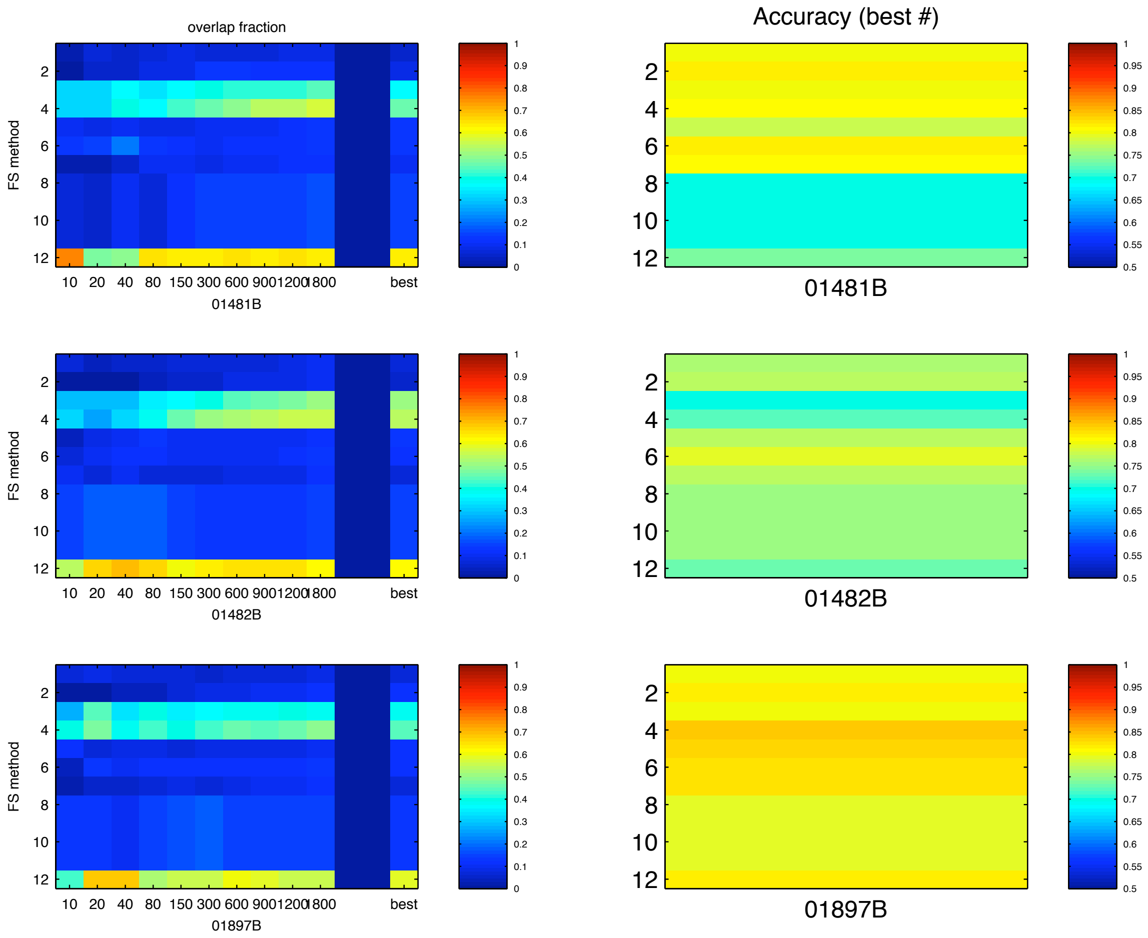


Figure 2.28: Fraction of overlap ([0,1]) between the set of voxels picked in *all* crossvalidation folds and those picked in *at least one* fold, for dataset D1 (6 fold cross-validation). The figure contains two columns of plots, one row per subject. Each plot on the left column contains as many rows as voxel ranking methods. In each row each entry corresponds to one of the numbers of voxels selected and contains the overlap fraction between the sets selected in each fold, as defined in the main text. As the numbers end there is some padding and one final column on the right with the overlap fraction at the best number of voxels selected with each method. The right column plots contain the smoothed row accuracy using the same number of voxels.

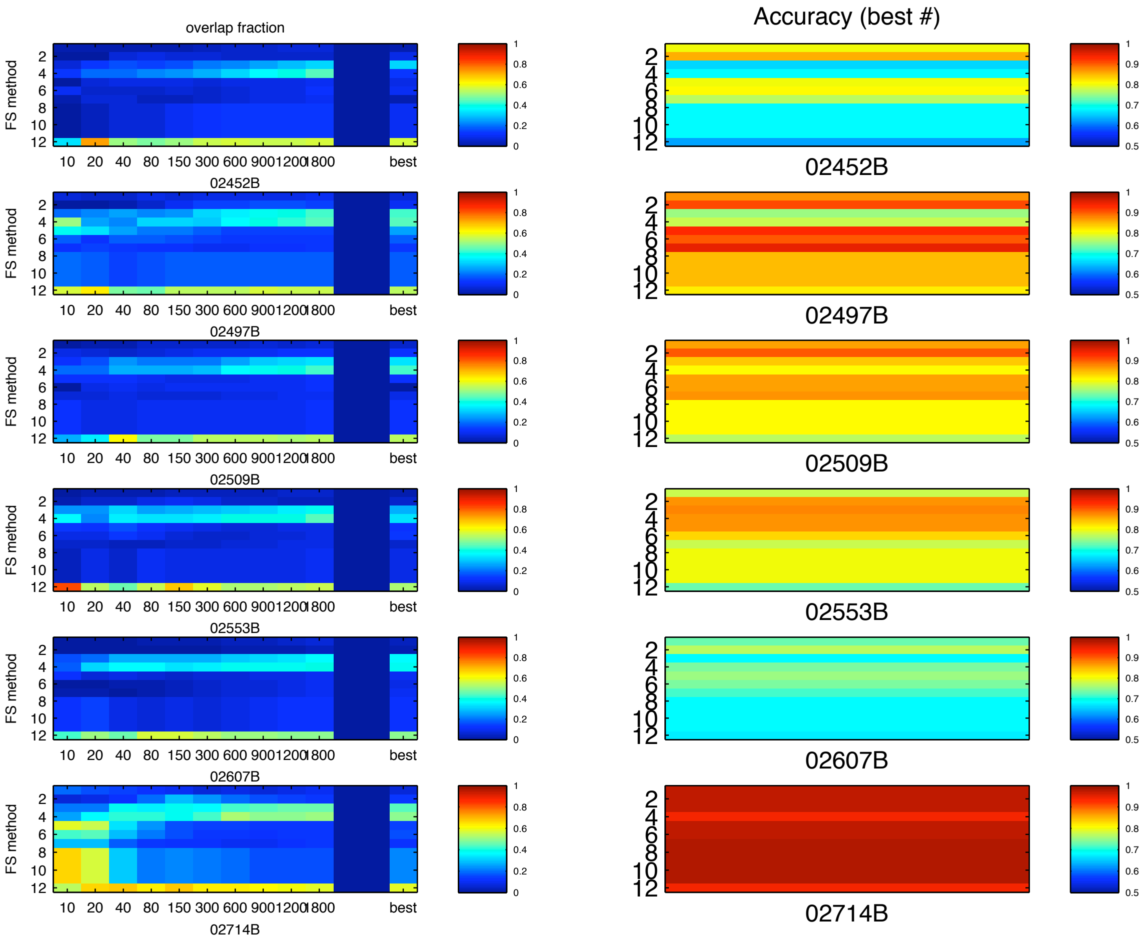


Figure 2.29: Same as Figure 2.28, for dataset D2 (6 fold cross-validation).

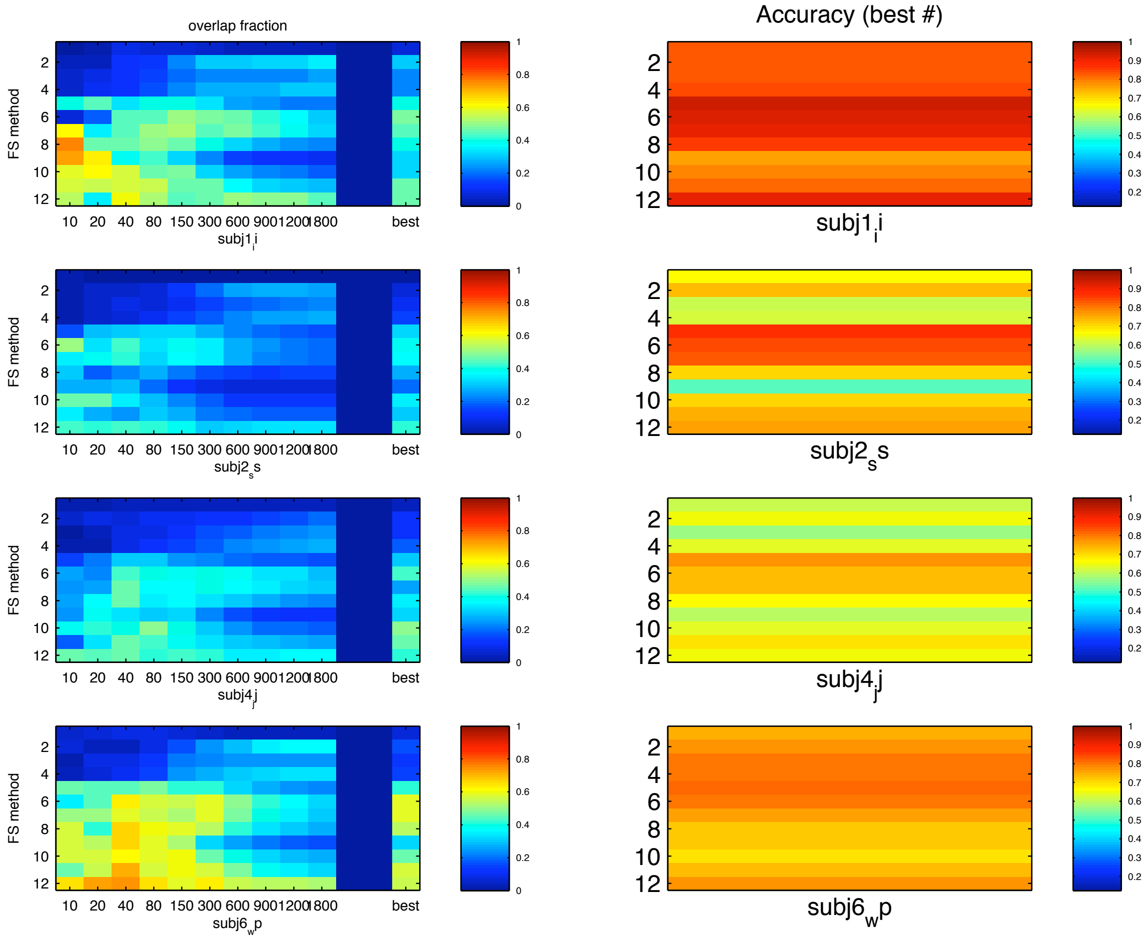


Figure 2.30: Same as Figure 2.28, for dataset D3 (12 fold cross-validation).

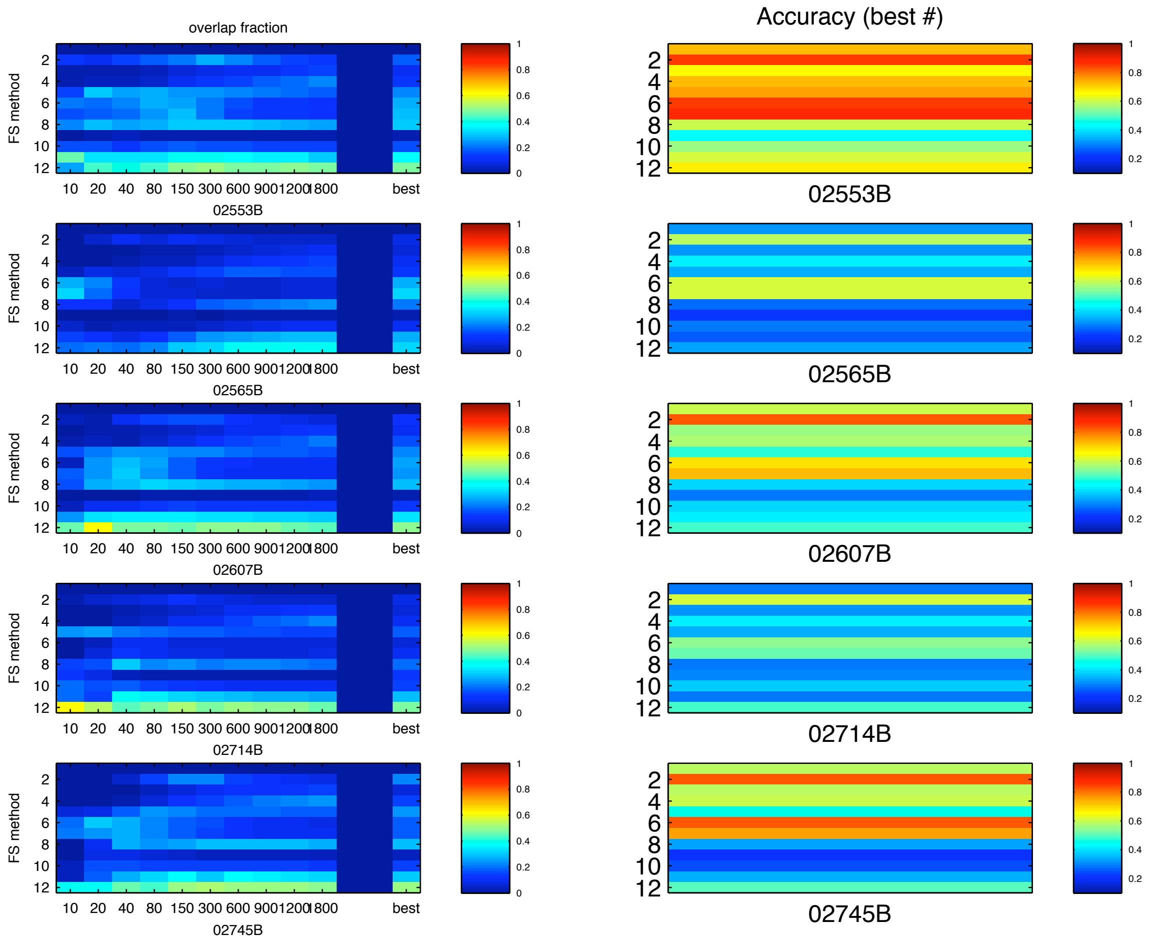


Figure 2.31: Same as Figure 2.28, for dataset D4 (6 fold cross-validation).

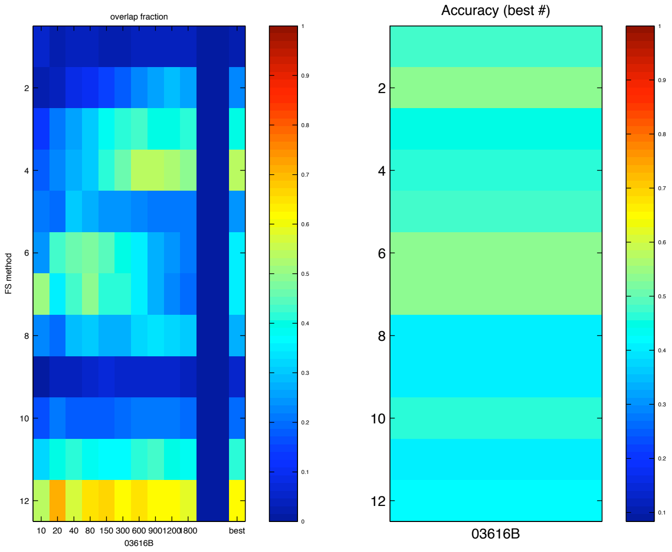


Figure 2.32: Same as Figure 2.28, for dataset D5 (6 fold cross-validation).

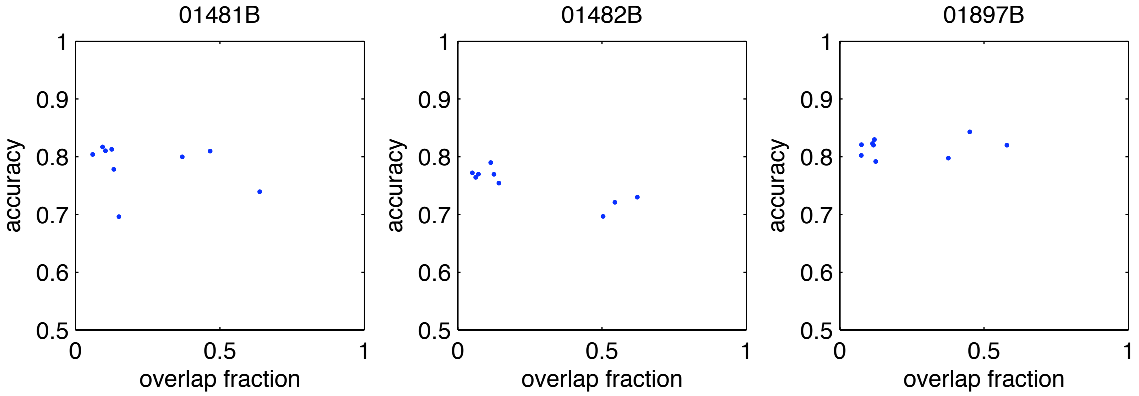


Figure 2.33: Relationship between overlap fraction (x axis, [0 1]) and classifier accuracy
.

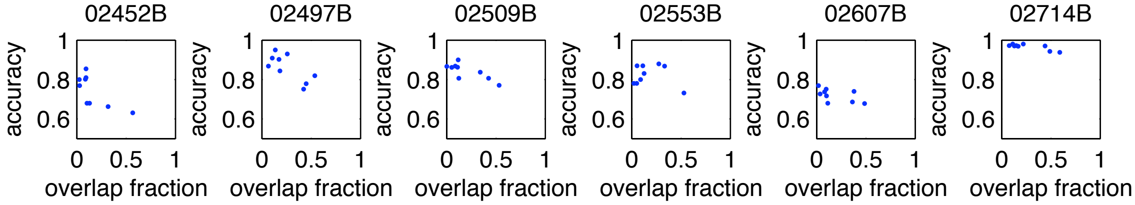


Figure 2.34: Same as Figure 2.33, for dataset D2.

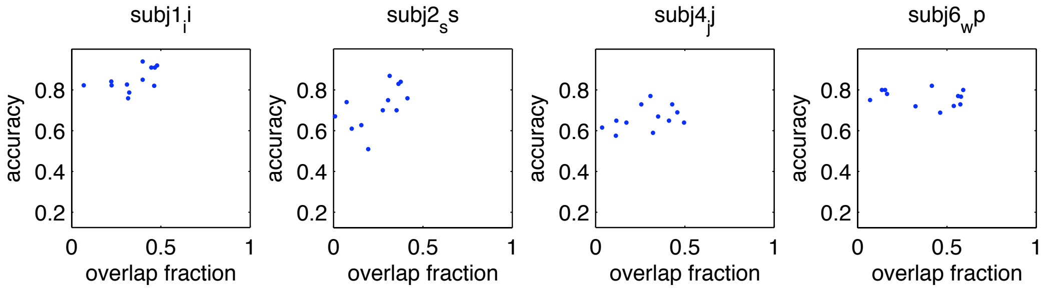


Figure 2.35: Same as Figure 2.33, for dataset D3.

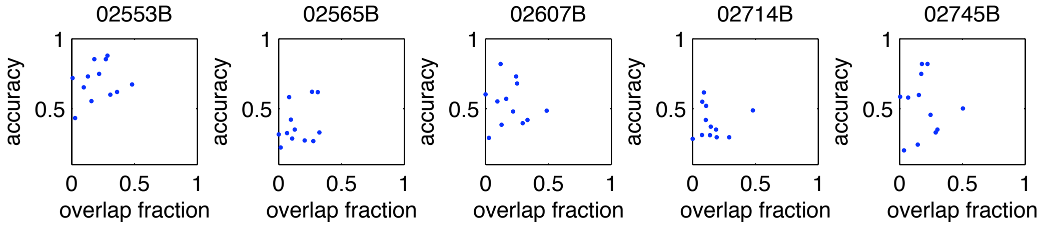


Figure 2.36: Same as Figure 2.33, for dataset D4.

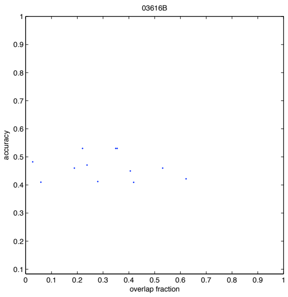


Figure 2.37: Same as Figure 2.33, for dataset D5.

Cross-method selection overlap

In order to quantify the extent to which different methods use the same voxels, we measure the fraction of overlap between the set of voxels selected on any fold by each method with the set selected by any other method. Figure 2.40 shows this for dataset D3. There are 4 columns, one per subject. The first row contains overlap matrices, where entry i, j corresponds to the fraction of overlap $\frac{\texttt{\# voxels selected in any fold by both methods}}{\texttt{\# voxels selected in any fold by either method}}$, with each method selecting the number of voxels at whatever accuracy was best for it.

Given that this could mean the overlap fraction would be low if either method was best using few voxels, the second row of plots contains montages of overlap fraction matrices using *fixed* number of voxels, for all numbers of voxels tried, as those numbers increase vertically. From this one can see that overlap does increase with numbers, but rarely to more than $50 - 60\%$. The same happens for the remaining datasets as shown in Figure 2.38, Figure 2.39, Figure 2.41 and Figure 2.42

There are a few interesting patterns that hold across datasets. The first is in the activity methods (3,4): one will be more appropriate than the other to each dataset, depending on preprocessing, but they both rank many of the same voxels highly. The second is that the two accuracy methods, voxelwise and searchlight (1,2), pick relatively different voxels (prior evidence for this was found in [58]). Given that the latter almost always leads to worse classification results, it appears that they both pick very informative voxels early on but the latter can be led astray (even with many examples, as in dataset D5). Voxelwise accuracy (3) also shows some overlap with ANOVA (6), which makes sense given that they make the same assumptions about class voxel activity distribution; as the number of voxels increases its overlap with the range methods also increases (these are voxels selected by differences in mean activity in each class, but not taking the variance into account, so possibly merely very noisy). ANOVA (6) and replicability (7) have a lot of overlap, surprisingly given that the criteria they use are unrelated, at least at first glance. One possible reason for this would be because the latter will prefer voxels that are both consistent in their activity across classes (and thus have low variance) but also have some activation driving the correlation values between runs (and thus mean differences between classes). Not surprisingly, the range methods (8-11, 12 is simply picking voxels by their maximum activity) tend to display a lot of overlap among themselves and not with all the other methods (notice the square block structure in the lower right of each graph). In datasets with binary classification problems (D1,D2) they all pick precisely the same voxels. Logistic regression weights (5) overlaps with the range methods (8-11), perhaps because voxels that help discriminate and get a large range of logistic regression weights across classes should also have a large range of mean activities across classes, which would lead to high scores in most range methods. The only range method with a somewhat different profile is “max” (12), which overlaps to a certain extent with the activity methods (3,4). Again, it’s not surprising that voxels that score high in the activity criterion would also score high here. Given the bad results obtained with the method, it most likely picks many very noisy voxels in addition.

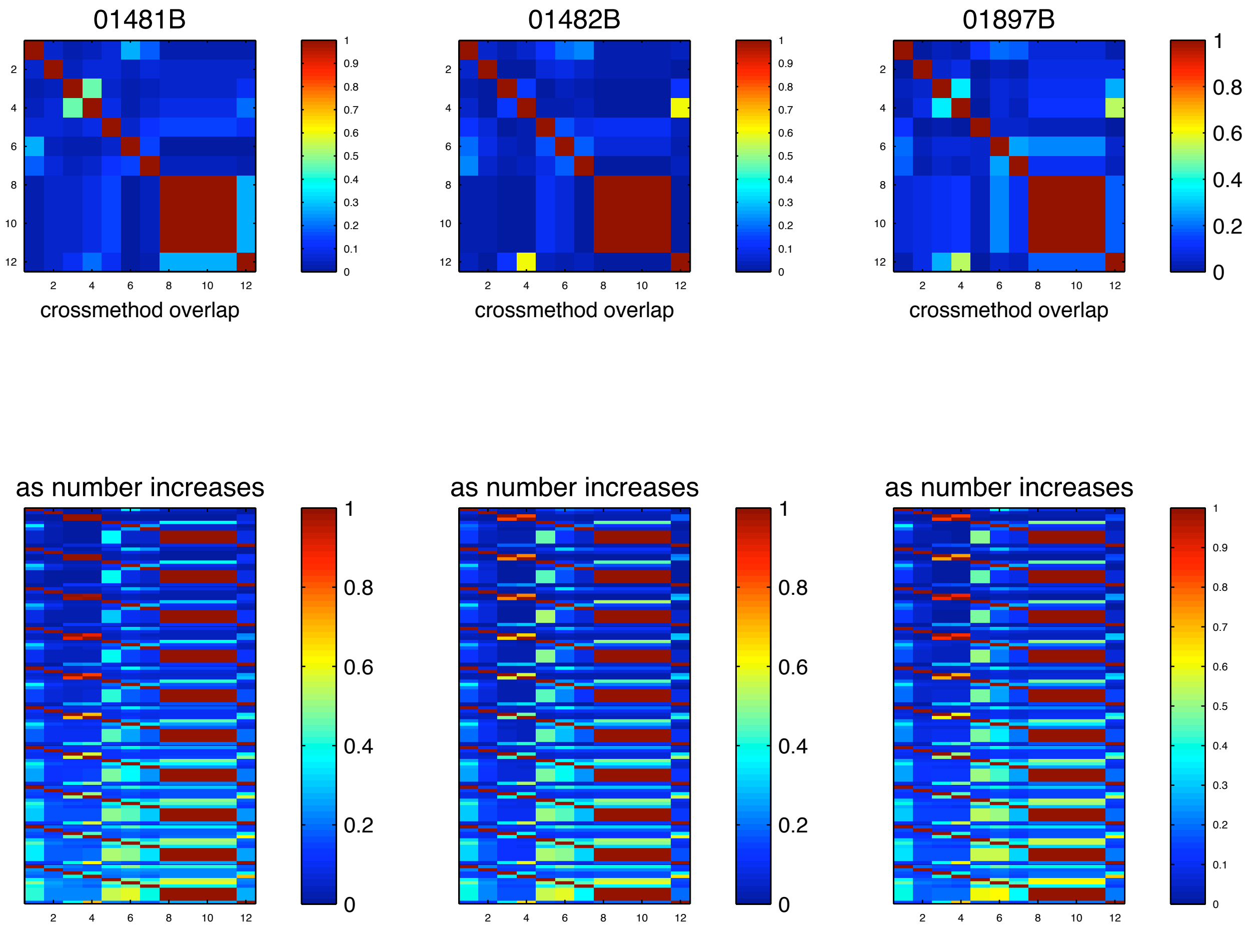


Figure 2.38: **Top:** fraction of overlap [0 1] between the set of features picked in all cross-validation folds by each pair of methods, using the best number of features for each, for dataset D1. **Bottom:** Same, forcing both methods to use the same number of features, repeated several times vertically as the number of features increases.

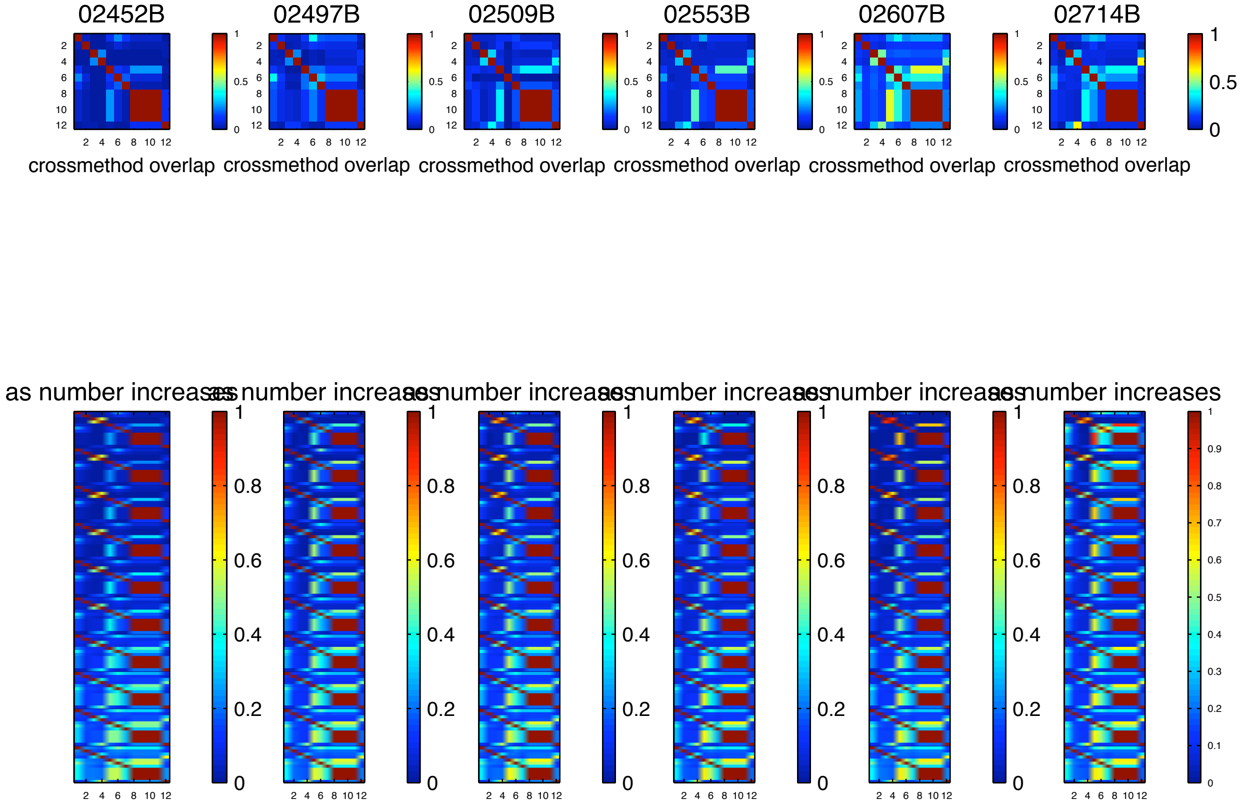


Figure 2.39: Same as Figure 2.38, for dataset D2.

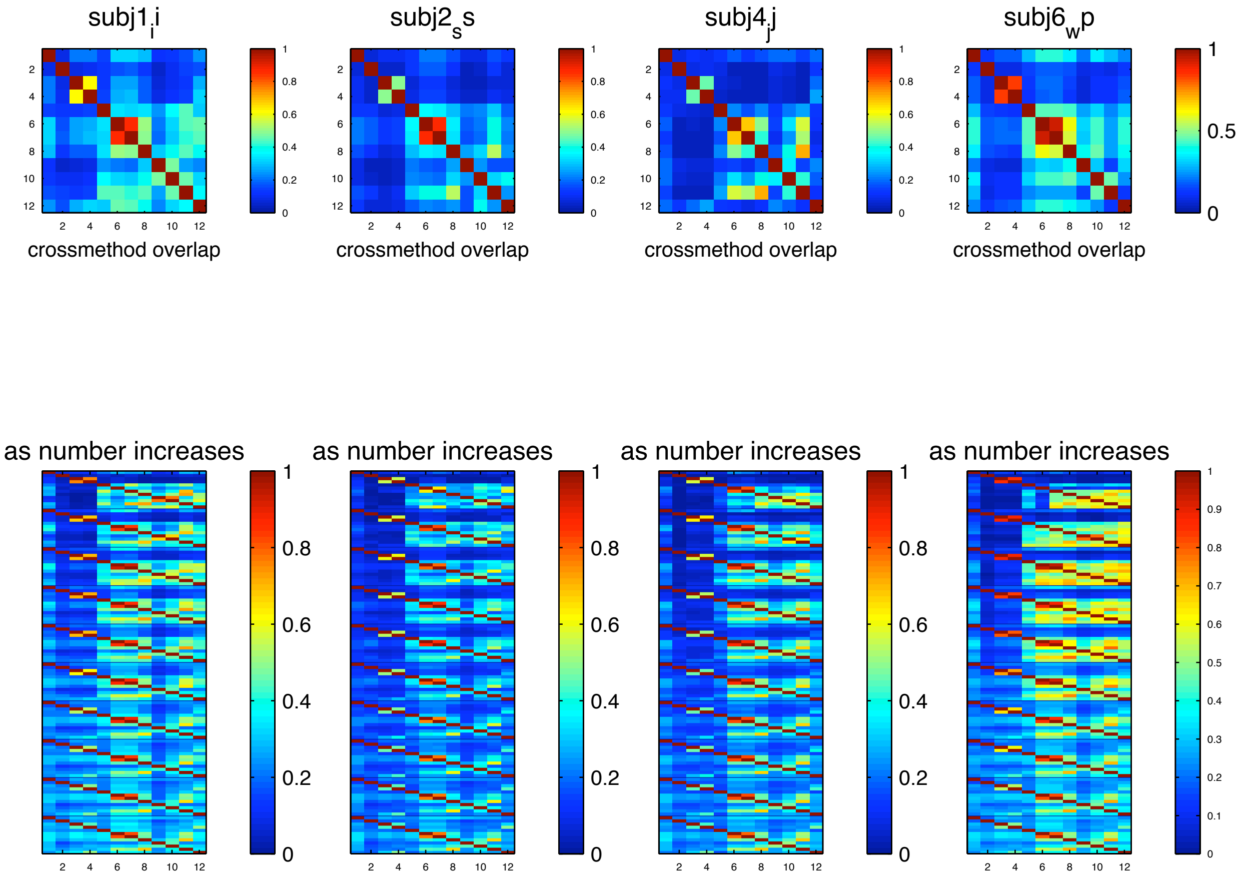


Figure 2.40: Same as Figure 2.38, for dataset D3.

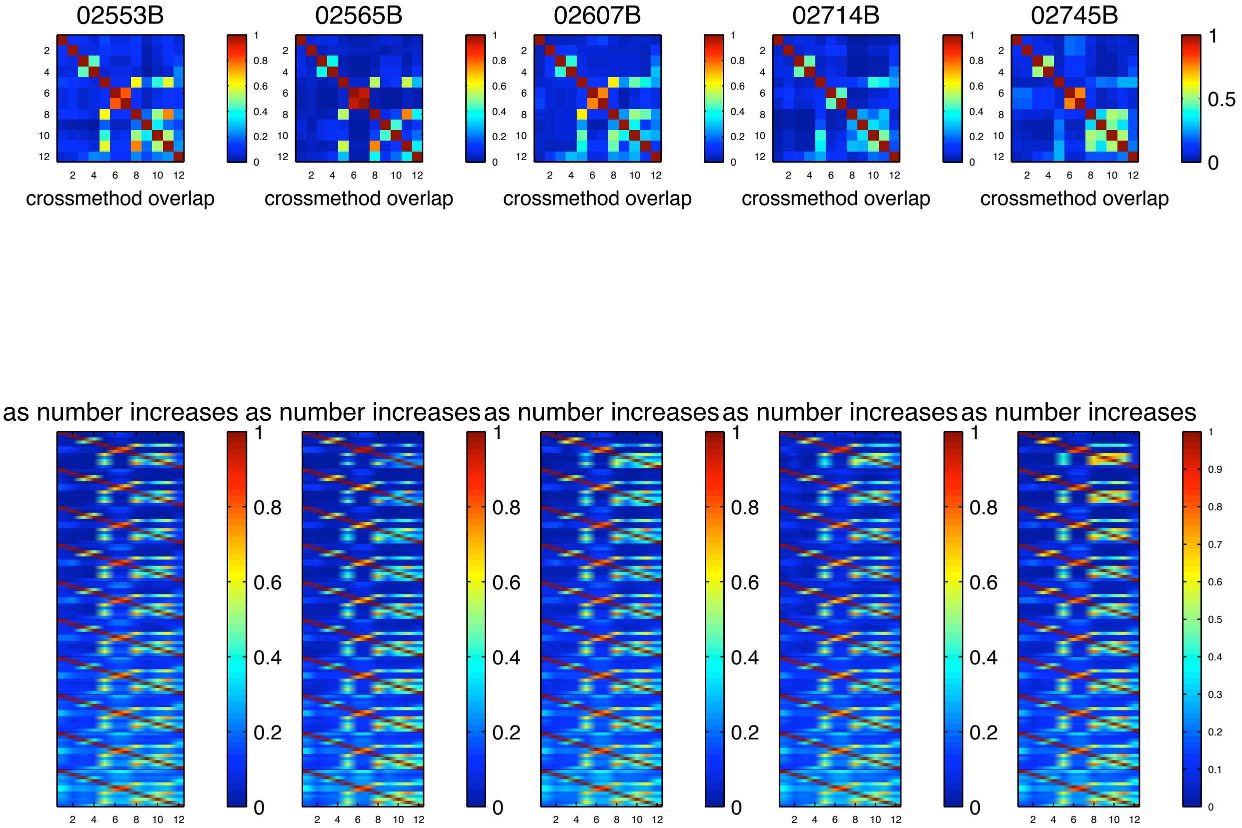


Figure 2.41: Same as Figure 2.38, for dataset D4.

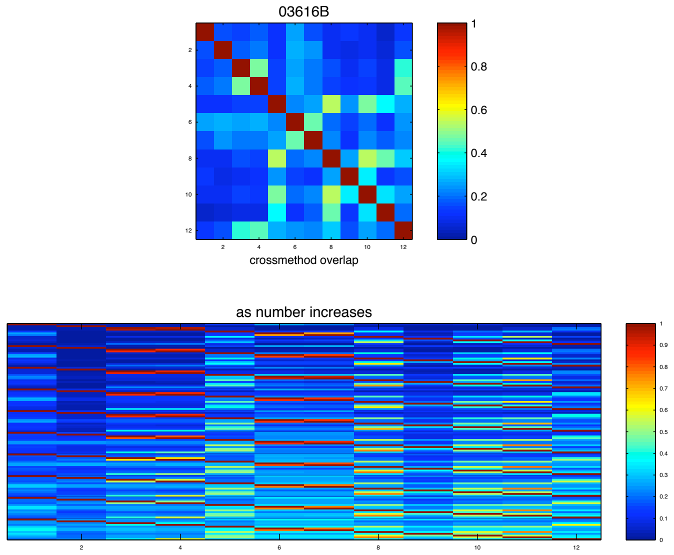


Figure 2.42: Same as Figure 2.38, for dataset D5.

Overlap in classification decisisons of the sets of voxels picked by two different voxel selection methods

In the previous section we considered looking at the overlap of the sets of voxels selected by two given methods. To some degree this tells us whether they use the same voxels, but not necessarily whether the information each set provides to the classifier is the same.

Given any two methods and a fixed classifier, we can address this question by considering which examples the classifier mislabelled when trained on the set of voxels picked with each method, with the number of voxels yielding the best result. If the classifier makes precisely the same mistakes, it's likely that it's learning a similar model and thus the information it has in each case is similar. The fraction of overlap between the sets of mistakes (number of examples on which both classifiers were wrong relative to the total number of examples mislabelelled by either classifier) is plotted for four subjects on dataset D3 on the first row of Figure 2.45.

Note that one danger of this approach is that the more accurate either of the classifier+method is the less room there is for overlap in mistakes; hence the results should be skewed towards either no overlap or almost complete overlap (if the methods happen to select voxels with similar information). In order to help gauge the effect of this, the accuracy of the classifier using the voxels selected with each of the methods is plotted on the second row.

The only overlap pattern that seems to be present in all datasets is that the two activity methods (3,4) overlap to a large extent in the mistakes they make. In the multiclass problem datasets

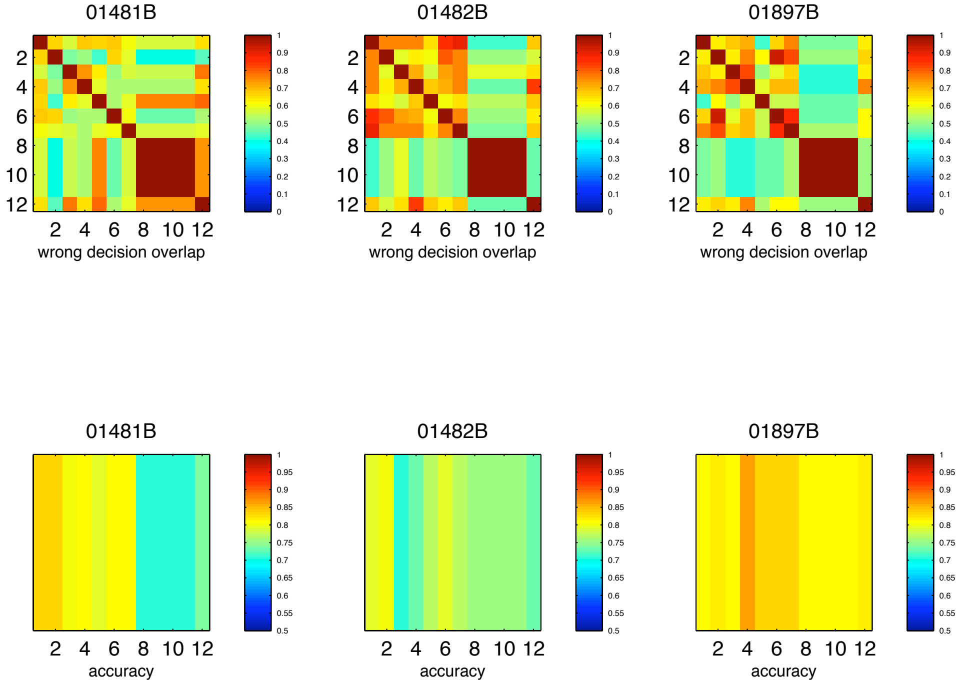


Figure 2.43: **Top:** Fraction of overlap [0 1] between sets of mislabelled examples of a classifier trained with features selected by any two methods, at the best accuracy for either, for dataset D1. **Bottom:** Accuracy of the classifier trained using the features selected by each method, at the best accuracy, for dataset D1.

(D3,D4,D5), ANOVA (6) and replicability (7) overlap very much, whereas voxelwise accuracy (1) and searchlight (2) overlap less (at the lowest levels in the graph). In dataset D4, ANOVA (6) and replicability (7) overlap very little with any of the others, as does searchlight (2). In the two class problem datasets (D1,D2) there are no clear patterns holding across all subjects.

In conclusion, there seems to be evidence that, at least for the multiclass problem datasets, there are groups of voxels with different kinds of information. This means that it should be feasible to combine groups to improve accuracy, and profitable to examine the behaviour of the voxels involved.

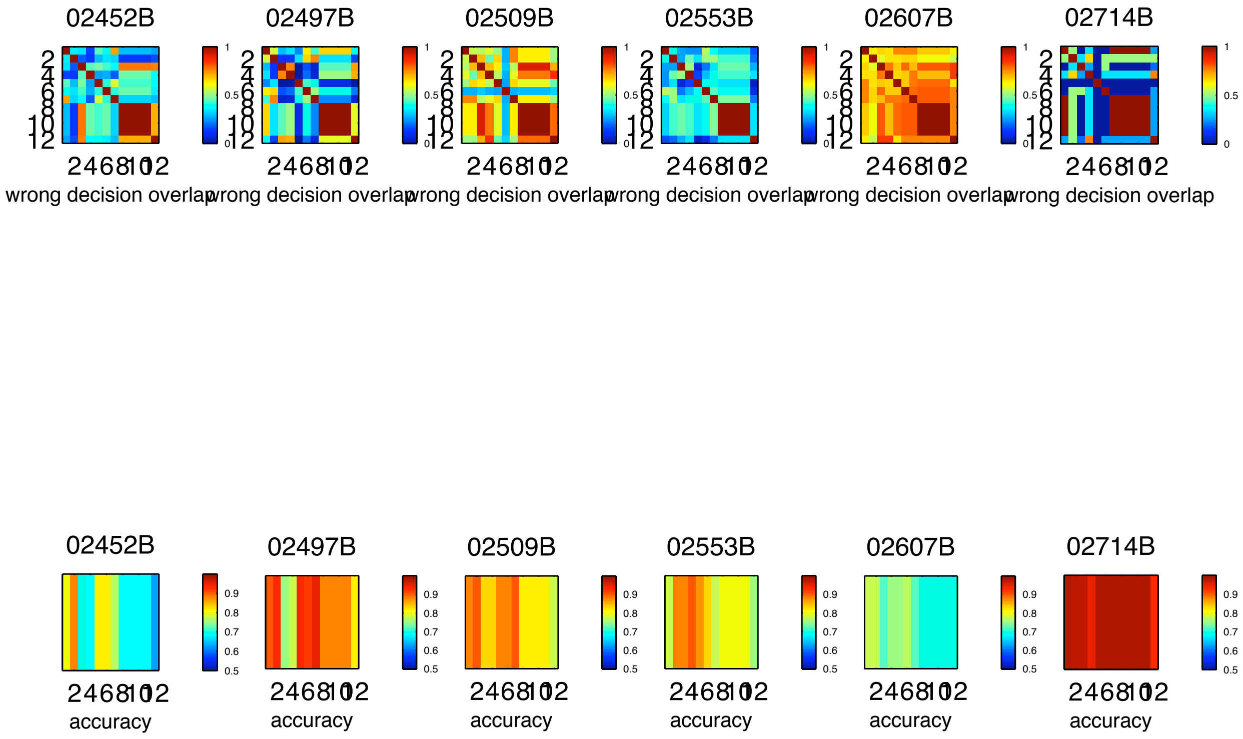


Figure 2.44: Same as Figure 2.43, for dataset D2.

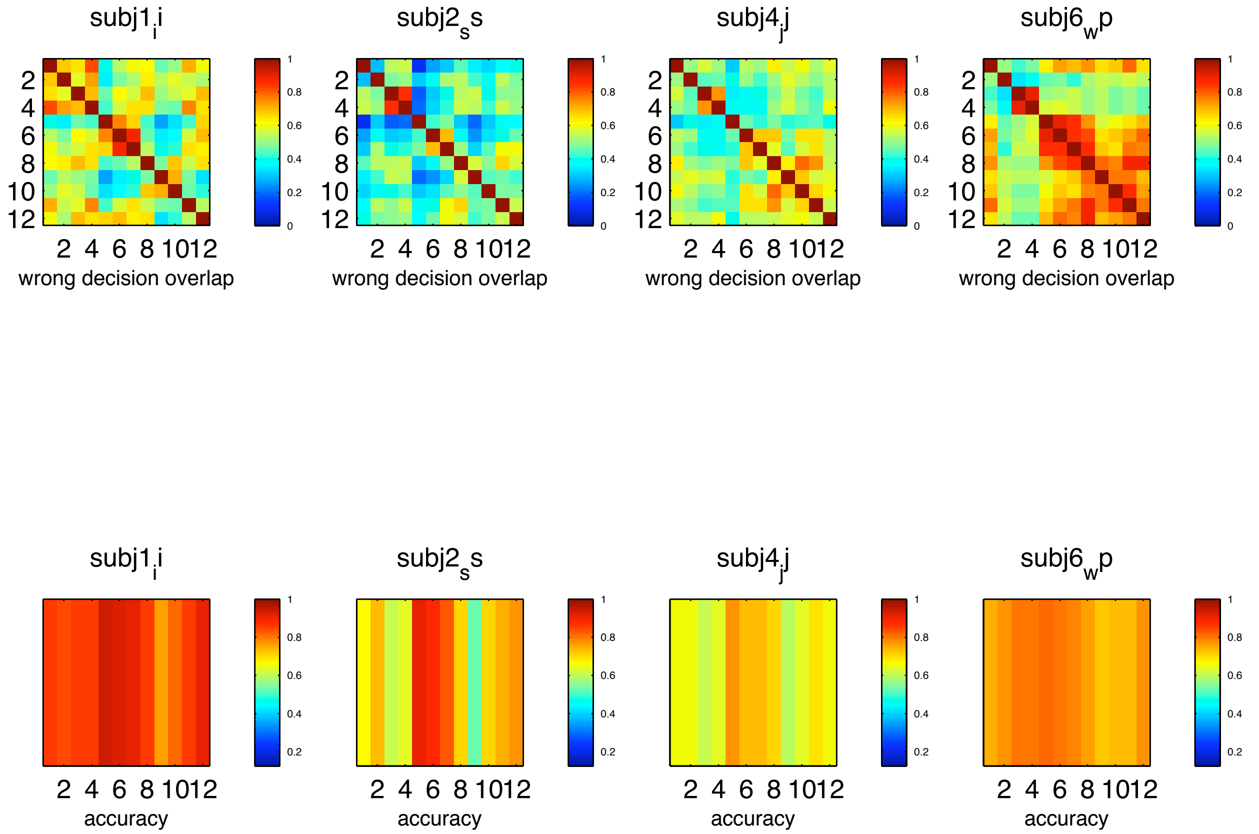


Figure 2.45: Same as Figure 2.43, for dataset D3.

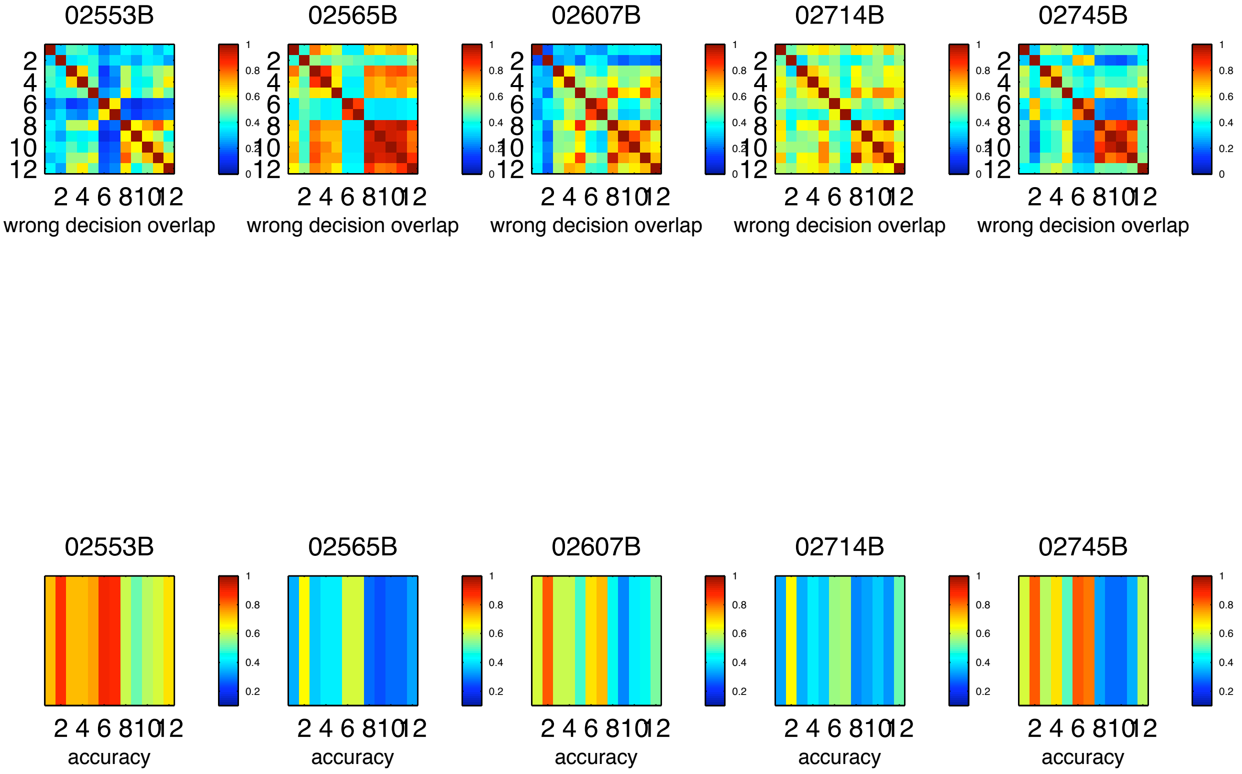


Figure 2.46: Same as Figure 2.43, for dataset D4.

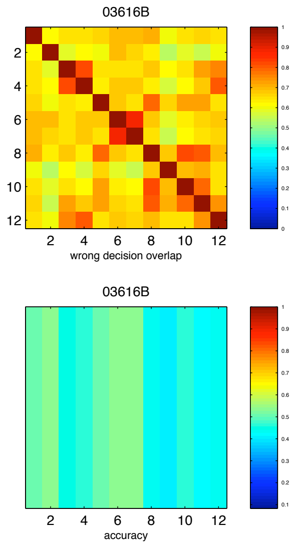


Figure 2.47: Same as Figure 2.43, for dataset D5.

2.4 An application: identifying informative locations and characterizing voxel activity in them

2.4.1 Producing information maps

The previous sections were concerned with how classification results are obtained, deemed significant and compared. In this section we try to approach the same results from a different angle, that of showing which scientifically relevant conclusions can be drawn from them (mostly derived from [58] and a manuscript in preparation).

The dataset of concern is D1 from Section 2.3.1, where the subject sees word stimuli that name items that are either tools or buildings. It has been established that machine learning classifiers can be used to decode the semantic category of *picture* stimuli from the average of tens of images acquired while the stimuli were being displayed to a subject (e.g. [24]). To our knowledge, no one had shown it is possible to do so for *word* stimuli from images acquired during a short time interval, as the task is made harder by the type of stimulus and the amount of noise due to only considering a 4 second period when averaging images. This led to our first goal: to show that enough information was present in that short a period, even when elicited by word stimuli.

Our second goal was to try to show *where* the information used by the classifier was present. Of all the voxel scoring methods outlined in Section 2.1.4, the most relevant ones for that purpose are

subject	method	number of voxels selected											max	#
		10	20	40	80	150	300	600	900	1200	1800	all	acc.	sign.
01481B	single voxel	0.77	0.83	0.81	0.81	0.80	0.79	0.73	0.73	0.74	0.71	0.65	0.83	1
	searchlight	0.81	0.80	0.80	0.82	0.83	0.82	0.80	0.79	0.75	0.75	0.65	0.83	317
01482B	single voxel	0.69	0.76	0.74	0.76	0.79	0.77	0.77	0.76	0.74	0.71	0.63	0.79	0
	searchlight	0.77	0.71	0.80	0.79	0.79	0.79	0.77	0.75	0.76	0.74	0.63	0.80	180
01897B	single voxel	0.75	0.81	0.80	0.81	0.81	0.81	0.79	0.77	0.76	0.75	0.69	0.81	3
	searchlight	0.69	0.74	0.77	0.77	0.81	0.81	0.81	0.81	0.82	0.82	0.69	0.82	449

Table 2.14: Classification results of GNB using the top n voxels ranked either by single voxel or by voxel searchlight accuracy. The final column - "number significant" - indicates how many voxels at the top of each rank have an accuracy score deemed significant.

(single voxel) *accuracy* and *searchlight accuracy*, in that they directly score voxels by whether they contain information allowing an accurate prediction. As a reminder the single voxel accuracy is simply the accuracy of a single-feature GNB classifier based on the voxel, obtained through cross-validation within the training set. Searchlight accuracy is the accuracy of a GNB classifier trained on the voxel and all the voxels adjacent to it in three dimensions, again through cross-validation within the training set. This translates into a classifier using between 1 and 27 (a $3 \times 3 \times 3$ cube) features, depending on how many neighbours the voxel has. Since the classifiers we consider are linear and add up information voxel by voxel (Section 2.1.2), the image obtained plotting the accuracy of each voxel gives us a rough approximation of how much information each voxel can contribute to the overall classifier, and that's what we consider in the next section.

All the results reported were obtained using the GNB classifier, as in this dataset the choice of classifier does not appear to affect the results very much.

2.4.2 Information map results

Table 2.14 shows the classification results obtained in 3 subjects, selecting various numbers of voxels using either single voxel or searchlight accuracy. The best results obtained with either method are similar and indicate that we can indeed decode the stimulus from the examples obtained during short time intervals.

For our second goal, let's consider images where each voxel contains either its single voxel accuracy or its searchlight accuracy. As explained in Section 2.2.1, the single voxel and searchlight accuracy scores can be converted into p-values in order to perform tests of statistical significance. More precisely, if we have n test examples we can find the p-value of the accuracy score under the null hypothesis that the true accuracy is chance level. Under the null hypothesis, the accuracy has a binomial distribution with parameters n and $p = 0.5$; the p-value we use is the probability that the accuracy under that distribution would be equal to or greater than the accuracy score obtained. This yields one p-value per voxel and thus a p-value image. This p-value image can then be thresholded

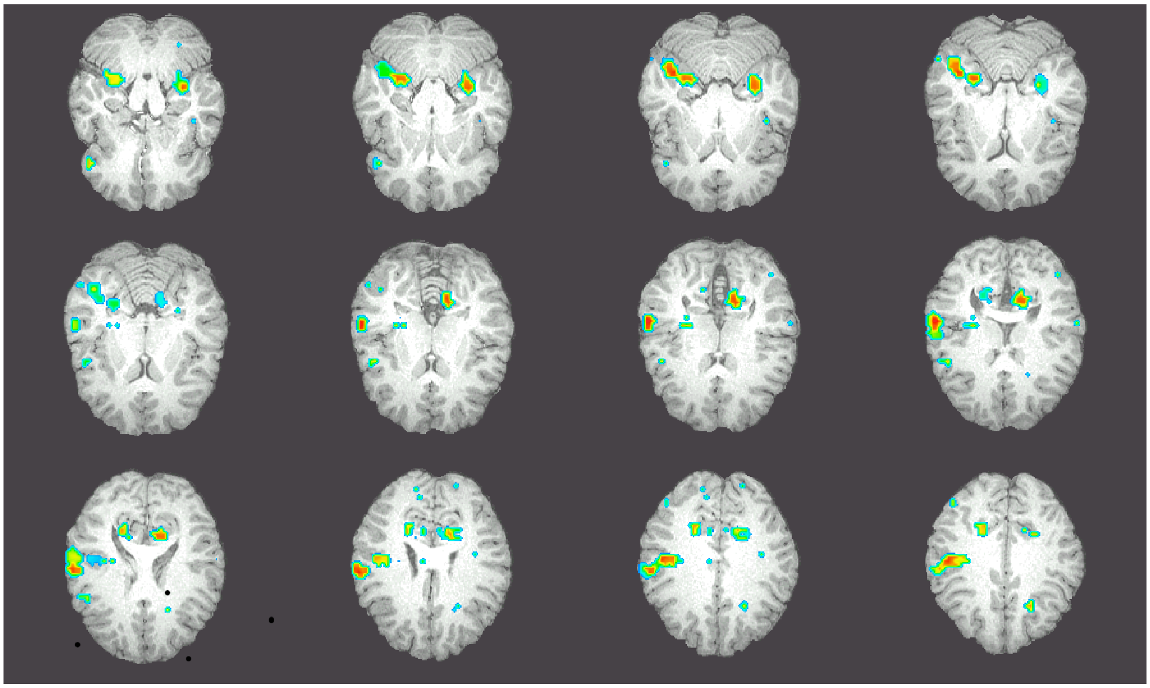


Figure 2.48: **Searchlight accuracy for voxels that have significant p-values, in subject 01481B**. The scale ranges from $\approx 70\%$(blue) to $\approx 80\%$(bright red). The slices are arrayed from inferior (top) to superior (bottom), and each slice is displayed with posterior at the top and anterior at the bottom.

in a way that accounts for multiple tests - as many tests as there are voxels - using a false discovery rate (FDR) procedure [67] with a 0.01 level. Note that the FDR procedure takes into account the fact that the scores of different voxels,and thus their p-values, can be correlated, a factor that is more acute in the searchlight case.

The last column of the table shows the number of voxels that are deemed significant under this procedure, for the single voxel and searchlight images of each subject. The *searchlight* p-value images contained hundreds of significant voxels, while the *single voxel* p-value images contained almost none. This indicates that some voxels, together with their neighbors, can form a group where the majority decides more accurately than any individual. This may happen because the noise in different voxels is uncorrelated at least to some extent, whereas informativeness in voxels will, this being a two class problem, manifest similarly in all of them as separation between the means of the distributions of values of each class.

Figure 2.48, Figure 2.49 and Figure 2.50 show the voxels whose searchlight accuracies were

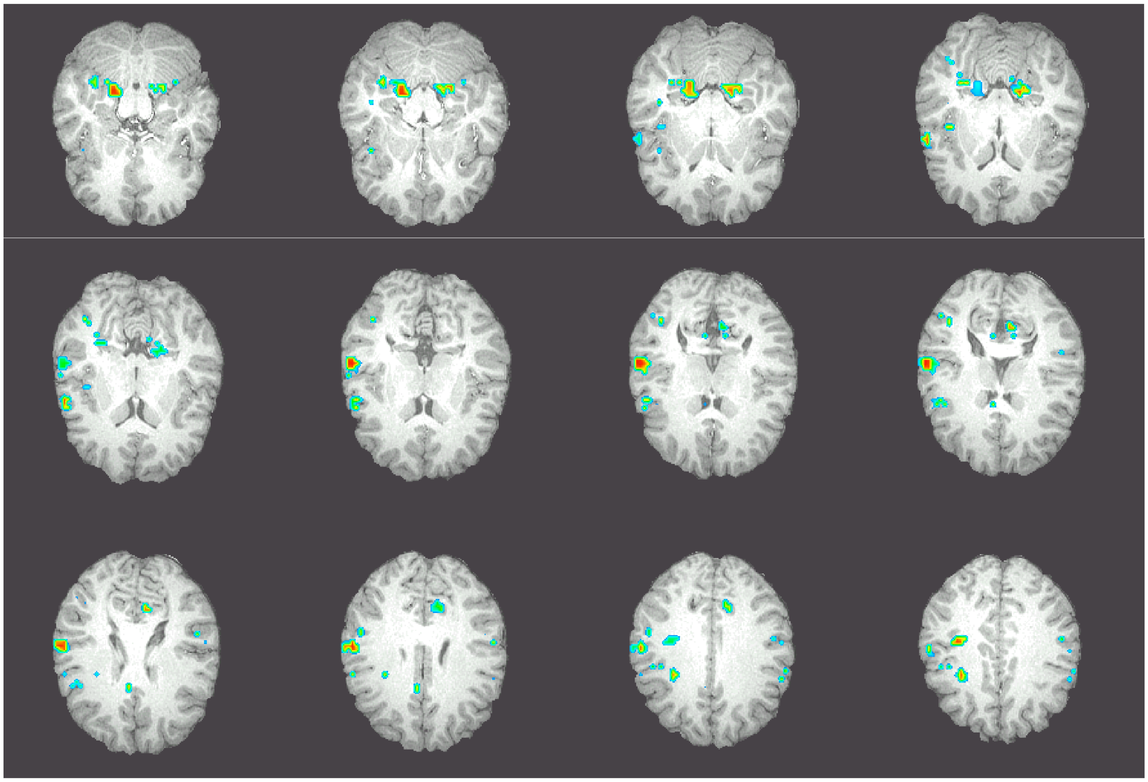


Figure 2.49: **Searchlight accuracy for voxels that have significant p-values, in subject 01482B**. The scale ranges from $\approx 70\%$(blue) to $\approx 80\%$(bright red). The slices are arrayed from inferior (top) to superior (bottom), and each slice is displayed with posterior at the top and anterior at the bottom.

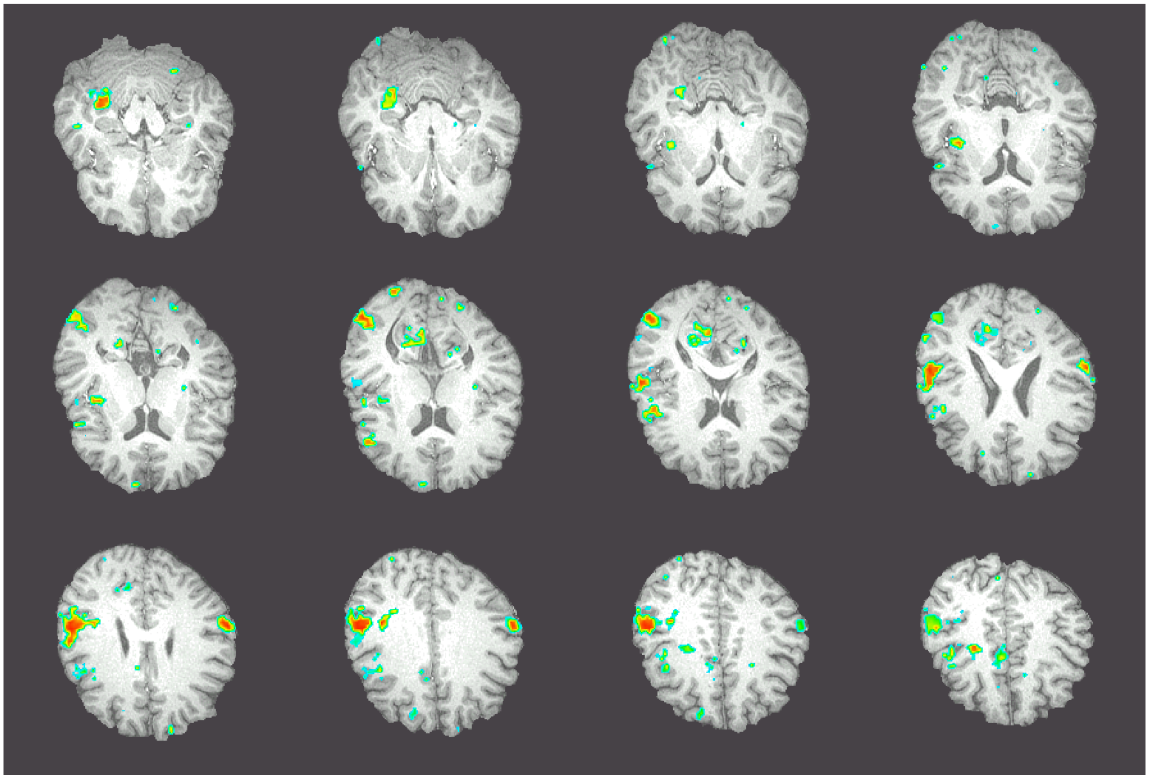


Figure 2.50: **Searchlight accuracy for voxels that have significant p-values, in subject 01897B**. The scale ranges from $\approx 70\%$(blue) to $\approx 80\%$(bright red). The slices are arrayed from inferior (top) to superior (bottom), and each slice is displayed with posterior at the top and anterior at the bottom.

deemed significant for three subjects. The accuracy of voxels in the image is in the range of $\approx 70\%(blue)$ to $\approx 80\%(red)$. There was particularly high accuracy in at least two locations of interest common to all subjects: voxels around the central sulcus and in the parahippocampal gyrus. These locations are consistent with the categories used as well as the instructions (visualizing oneself in the building or oneself using a tool) and previous reports of activation or decoding from picture stimuli of those categories (e.g. [26] or [35]). The former is reasonable given that motor cortex is on one of the sides of the central sulcus and thus likely to appear in voxels around that area if thinking of tools activates motor cortex; the latter because activation in the area has been linked to landmark/location/building stimuli. Intriguingly, other common high accuracy areas include the left inferior frontal gyrus and middle/superior temporal areas. These were not expected and might point at a different use of language when performing the task with each category and will thus be examined in more detail in future studies.

2.4.3 Characterization of informative voxel activity

By reducing dimensionality to the point where good classification is possible, voxel selection produces a dataset where one can expect voxel behaviours related to task labels to be represented. While on a binary classification problem the sole informative behaviour might be having bimodal activation where the modes correspond to the two classes, multiple class problems have room for more variety, as we shall see below.

We will use Dataset D3 introduced earlier, an 8 category problem where stimuli belong to one of 8 categories: faces, houses, cats, bottles, scissors, shoes, chairs and scrambled pictures. In Section 2.3.2 we saw that it was possible to obtain fairly high accuracy ($80\% - 90\%$) using various methods in two of the subjects. We will look at the first subject, considering the top 100 voxels as selected by ANOVA.

Figure 2.51 shows the value of each of 100 selected voxels in the average example in each of 8 classes. The voxels are ordered so that voxels with similar average levels of activation across classes are clustered together (using spectral clustering [54] with euclidean distance). They divide clearly into three groups, one that is active primarily for Face/Cat stimuli (the latter stimuli contain only cat heads), another for Houses and another mainly for the four categories of man-made objects. Note that, whereas the first two display a very selective behaviour (on for one kind of thing, off otherwise), the group active for the objects displays a a more heterogeneous pattern of activation, with subgroups active for various pairs of classes and also what might be various levels of activation. A fourth group intriguingly combines houses with some of the objects, something that we do not have a good explanation for.

It's unlikely that we would have been able to identify these classes of behaviour directly from the data containing all the voxels, be it because of the sheer number of voxels to consider in clustering or because of the many possible voxel behaviours that, despite high levels of activation, might not be structured in anyway reflecting the class structure. Using voxel selection and looking at the

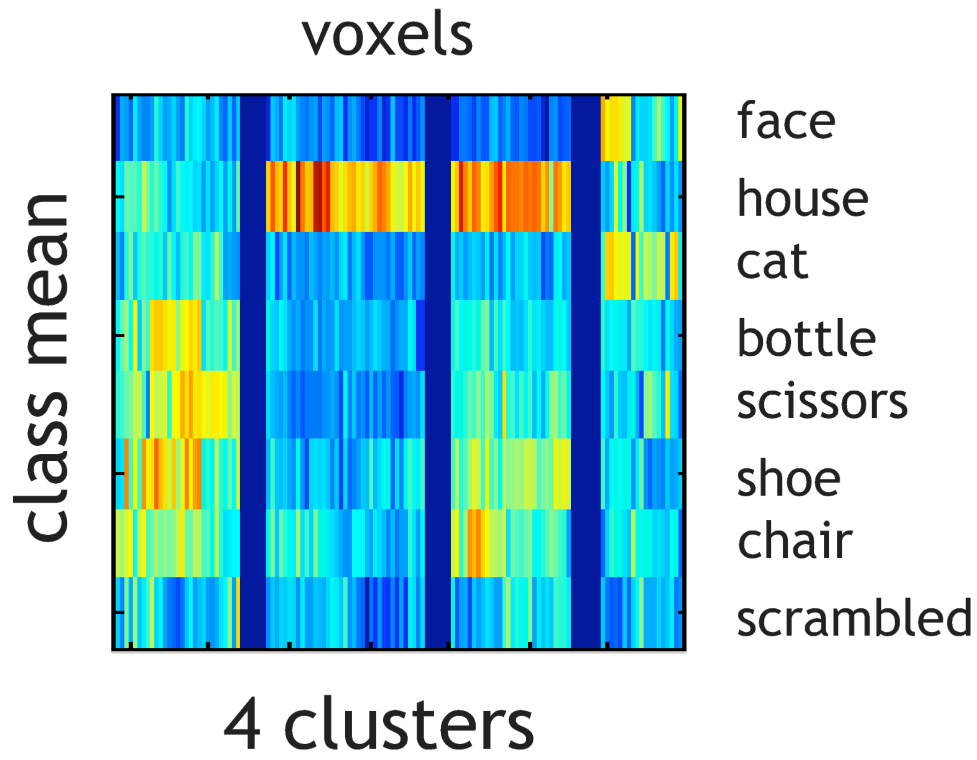


Figure 2.51: Clustering of voxel activity across 8 categories, shows the value of each 100 selected voxels in the average example for each category. The voxels are ordered so that the voxels with similar average levels of activation are grouped together.

few voxels that contain the information is thus a good way of obtaining behaviour “representatives”, which can be examined to test existing hypotheses of how the classes and their relationships are neurally encoded or suggest new ones. It’s interesting to note that a different method for voxel selection might yield different representatives, and we are currently investigating the extent to which this happens.

Appendix

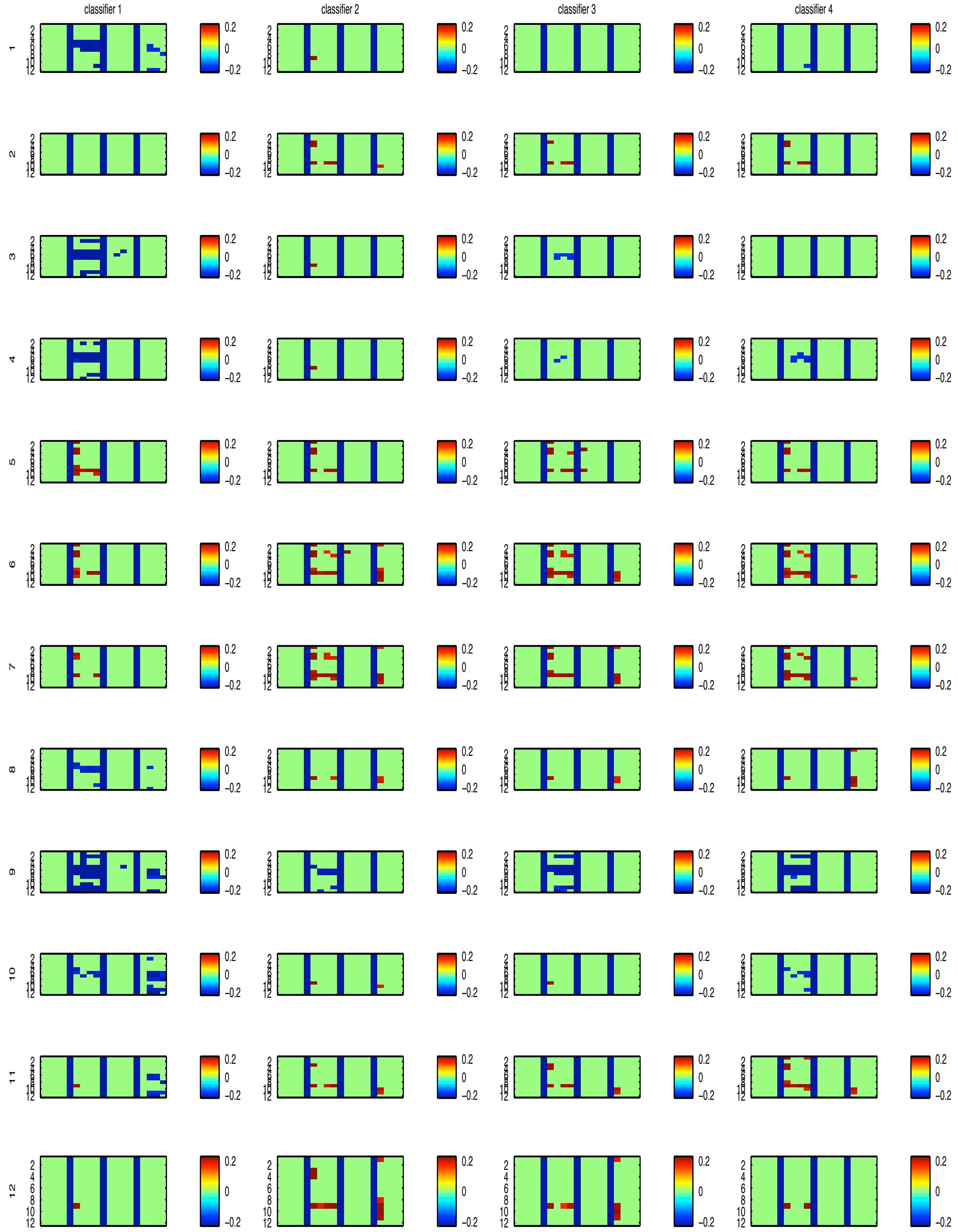


Figure 2.52: Same as Figure 2.21, for dataset D3

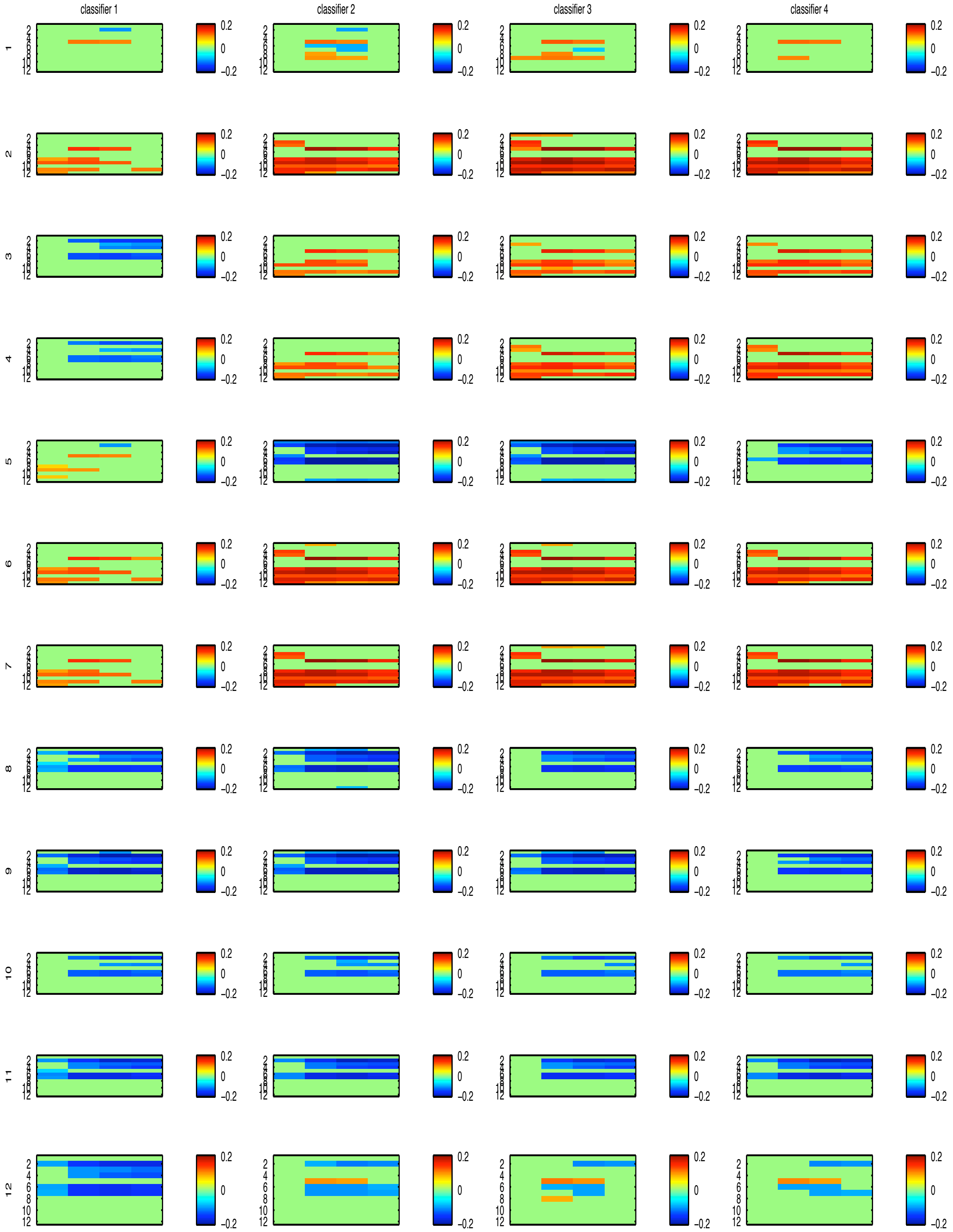


Figure 2.53: Same as Figure 2.21, for dataset D5

Chapter 3

The Support Vector Decomposition Machine

Linear dimensionality reduction in machine learning is generally performed for two purposes. The first is to make the data more interpretable by decomposing each example as weighted *linear* combination of a few basis vectors with the same dimensionality (e.g. in fMRI data each example is an image and each basis vector what is called an *eigenimage*). The vector of weights then becomes a low-dimensional representation of the high-dimensional example. The interpretability is tied to the criterion the dimensionality reduction method uses for finding a basis, e.g. basis vectors in the directions of maximum variance (in Singular Value Decomposition) or which are close to independent (in Independent Component Analysis). Knowing the criterion used and using additional domain information (e.g. anatomical locations in an eigenimage) allows one to interpret basis vectors and, through knowing which of those weigh most heavily in the decomposition of each example, the original data.

The second purpose is to help in training a classifier to predict a variable of interest from the data, by maintaining the same number of examples while reducing the number of features given to that classifier. However, SVD and ICA do not consider classification performance but rather how well the original data can be reconstructed from the basis and the low-dimensional representation; there is no guarantee that a classifier based on reduced dimensionality data produced by these algorithms will work well.

In this chapter we introduce two variations of the Support Vector Decomposition Machine (SVDM), a new algorithm that *simultaneously* solves the problem of linear dimensionality reduction and classification, outputting a solution with three components: a set of basis vectors, a low-dimensional representation of the examples and a linear classifier based on that representation. Furthermore, we demonstrate on two fMRI datasets that better classification results can be achieved than those obtained by doing SVD or ICA first and then training linear SVM or a Naive

Bayes classifiers. The low-dimensional decomposition learnt is also good for reconstruction, as well as reusable with classifiers other than the one learnt by the algorithm; most dimensions show class-related structure.

3.0.4 Singular Value Decomposition

The goal of the singular value decomposition algorithm is to find a representation of a matrix of training data as a product of lower-rank matrices. The dataset matrix contains n examples (rows) with m features (columns).

$$X_{n\times m} = \begin{bmatrix} x_1(1) & x_1(2) & \dots & x_1(m) \\ x_2(1) & x_2(2) & \dots & x_2(m) \\ \dots & & & \\ x_n(1) & x_2(2) & \dots & x_n(m) \end{bmatrix} = \begin{bmatrix} x'_1 \\ x'_2 \\ \dots \\ x'_n \end{bmatrix}$$

The SVD approximates X as a product of two[1] lower-rank matrices,

$$X_{n\times m} \approx Z_{n\times l} W_{l\times m}$$

Here l is the rank of the approximation. W is a *basis* matrix: each of its l rows is a direction of variability of the training examples. And Z is a matrix of *coordinates*: the ith row of Z gives the coefficients necessary to reconstruct the ith training example as a linear combination of the rows of W.

More precisely, the SVD minimizes the sum of squared approximation errors: we can compute Z and W by solving the optimization problem

$$\min_{Z,W} \|X - ZW\|^2_{\text{Fro}} \tag{3.1}$$

Here $\|A\|_{\text{Fro}}$ stands for the Frobenius norm of the matrix A, $\sqrt{\sum_{ij} A^2_{ij}}$.

3.0.5 Classification

We will suppose that there are $k \geq 1$ classification problems which we wish to solve. The target labels for these problems are given in the matrix $Y_{n\times k}$, with $y_{i,j} \in \{-1, 1\}$. The reason for allowing k classification problems instead of just one is that one problem may give information about features which are useful for solving another. This is a form of Multitask Learning [7] and experiments exploring it are detailed in Section 4.2 of Chapter 4.

[1] Often the SVD is written as a product of 3 matrices, $X = U\Sigma V'$, and constraints such as $U'U = I$ and $V'V = I$ are imposed on U, Σ, and V to make the solution unique. We could just as easily write it as $X = ZW$ where $Z = U\Sigma$ and $W = V'$.

To solve these classification problems using the learned low-dimensional representation from the SVD, we can seek parameters $\Theta_{l \times k}$ such that the matrix $\mathrm{sgn}(Z\Theta)$ is a good approximation to Y. Here $\mathrm{sgn}(\cdot)$ is the componentwise sign function, so for example if some element of Y is 1 we want the corresponding element of $Z\Theta$ to be positive. For convenience, we will constrain all the entries in the first column of Z to be 1, so that the corresponding entries in the first row of Θ will act as biases for the k classification problems.

As discussed above, the SVD computes Z and W without reference to Y or Θ. To improve on the SVD we will simultaneously search for values of Z, W, and Θ to minimize an objective combining reconstruction error and classification performance. To this effect we propose two different approaches that each rely on solving a particular optimization problem.

3.0.6 Chapter contents

As discussed above, the SVD computes Z and W without reference to Y or Θ. Section 3.1 and Section 3.2 introduce two algorithms that simultaneously search for values of Z, W, and Θ. Both can be viewed as solving an optimization problem that trades off reconstruction error of the example matrix X with a measure of classification performance.

The first algorithm - A - does this directly by having an objective which trades off the reconstruction error over all examples, $\|X - ZW\|^2_{\mathrm{Fro}} = \sum_{i,j}(x_{ij} - \mathbf{z_{i,:}}\mathbf{w_{:,j}})^{\mathbf{2}}$ (the Frobenius norm of the error matrix, where $\mathbf{z_{i,:}}$ is the i^{th} row of Z and $\mathbf{w_{:,j}}$ is the j^{th} column of W), with the total hinge loss of classification decisions based on the low-dimensional representation of examples, Z (this is an upper bound on training set classification error). As there are many Z/W pairs that would yield the same reconstruction error, the algorithm also requires constraints that will pick one particular solution.

The second algorithm – B – also has an objective based on feature reconstruction error and hinge loss over all examples. In contrast with algorithm A, it directly minimizes the norms of Z and Θ in the objective. This leads to optimization problems that are either the usual linear SVM problem or a very similar modification, as they will not require the constraints in algorithm A.

After Z,W and Θ are identified with either algorithm, we have a model that allows us to take a test dataset, project it to a low dimensional subspace where its coordinates are Z and the basis W; Z can then be used, together with Θ, to generate predicted labels. It's also possible to generate predicted labels from the test dataset itself, by using a discriminant in the original feature space that makes the same predictions as Θ would make given Z. Section 3.3 details how one can obtain such an equivalent feature-space discriminant.

Finally, Section 3.5.6 considers the question of constraining how many dimensions of the low-dimensional space can be used to influence the learnt Θ while using the remaining ones strictly for reconstruction.

3.1 SVDM Algorithm A

3.1.1 Optimization Problem A

As discussed in a previous section, the SVD computes Z and W without reference to Y or Θ. To improve on the SVD we will simultaneously search for values of Z, W, and Θ which minimize the following objective:

$$\|X - ZW\|_{\text{Fro}} + \sum_{i=1:n, j=1:k} h(\rho_{ij}, \mu, D)$$

where $\rho_{ij} = y_{ij} Z_{i,:} \Theta_{:,j}$. Again, the summation on i is over the n examples and the summation on j is over k binary classification problems (for all of this chapter $k = 1$, the case of $k > 1$ is discussed in the next chapter). Here h is the hinge loss function with slope $-d$ and breakpoint μ:

$$h(\rho, \mu, d) = \begin{cases} 0 & \rho \geq \mu \\ d(\mu - \rho) & \text{otherwise} \end{cases}$$

This objective trades off reconstruction error (the first term) with a bound on classification error (the second term). The parameter d controls the weight of the hinge loss term, and the parameter μ controls the target classification margin.

The relative norms of Z, W, and Θ are not constrained by the above objective; for example, if we multiply Θ by some constant λ, we can compensate by dividing Z by λ and multiplying W by λ. So, we will impose the arbitrary constraints

$$\begin{aligned} Z_{i,1} &= 1 \\ \|Z_{i,2:end}\|_2 &\leq 1 \\ \|\Theta_{:,j}\|_2 &\leq 1 \end{aligned} \tag{3.2}$$

to pick one solution out of the many possible ones. We have constrained $Z_{i,1}$ to be 1 for two different reasons: the first is to make $\Theta_{1,j}$ be a bias term for the jth linear discriminant, and the second is to make the first row of W correspond to the mean of the dataset.

The SVDM optimization problem is strongly related to both the SVD and the SVM. To gain an intuition, we can examine what happens when we set d to its extreme values. As d rises the objective will be dominated by the hinge loss term. Thus, the problems of finding Z and Θ will be similar to linear SVM problems. On the other hand, if we set d to 0, the hinge term vanishes, leaving only the Frobenius norm term. The resulting optimization is identical to the SVD problem (3.1), save for the constraints (3.2). The only one of these constraints that makes a difference is $Z_{i,1} = 1$; its effect is equivalent to centering the data by subtracting its mean before performing the SVD.

3.1.2 Solving Optimization Problem A

We can solve the SVDM optimization problem by iterating the following steps: minimize the objective with respect to the variables W while holding Z and Θ fixed, then minimize with respect to Θ with Z and W fixed, then finally minimize with respect to Z with W and Θ fixed. Holding two of the three matrices fixed at each step of the optimization procedure simplifies the optimization problem in different ways, described in the following subsections. The most important detail is that each of those subproblems is convex ([4]) and thus has a single minimum. Given this, each step reduces the overall objective value, and because the objective is positive, this alternating minimization procedure will cause the objective value to converge to a local minimum or saddle point; we stop the optimization after an iteration - a set of three minimizations, one each with respect to W, Z and Θ yielded less a 0.5% decrease in the objective. For the experiments described in Section 3.5.2, this translated into between 1 and 20 iterations.

Given Θ and Z, solve for W

As there is only one term involving W, we want to minimize

$$\|X - ZW\|_{\text{Fro}}$$

with no additional constraints. That is, we wish to predict X as a linear function of Z using coefficients W, minimizing the sum of squared prediction errors. We can find the best W by solving a linear regression problem for each column of X:

$$X_{:,j} \approx ZW_{:,j}$$

For stability we add a tiny ridge term to each regression [25].

It is relatively simple to add external constraints on the rows of W, such as sparsity or spatial smoothness, without affecting the other subproblems; this is discussed at length in Section 4.1 of Chapter 4.

Given W and Z, solve for Θ

Since W and Z are fixed, we can drop the first part of the objective as well as the constraints that don't involve Θ. The rest of the problem is then

$$\begin{array}{rll} \min_{\Theta,\rho_{ij}} & \sum_{ij} h(\rho_{ij}, \mu, C) & \\ \text{subject to} & \rho_{ij} = y_{ij} Z_{i,:} \Theta_{:,j} & i = 1 \ldots n,\ j = 1 \ldots k \\ & \|\Theta_{:,j}\|_2 \leq 1 & j = 1 \ldots k \end{array}$$

This optimization problem tells us to choose Θ so that $\mathrm{sgn}(Z_{i,:}\Theta_{:,j})$ is a good predictor of Y_{ij}; that is, it is a linear threshold classification problem. We can divide the optimization into k independent subproblems, each of which tries to find weights $\theta = \Theta_{:,j}$ which predict the jth column of Y from the features Z:

$$\begin{array}{rll} \min_{\theta,\rho_i} & \sum_i h(\rho_i, \mu, C) & \\ \text{subject to} & \rho_i = y_{ij} Z_{i,:}\theta & i = 1 \ldots n \\ & \|\theta\|_2 \leq 1 & \end{array}$$

Since the hinge loss h is piecewise linear and convex, we can replace each term $h(\rho_i, \mu, C)$ in the objective by a variable h_i and the additional constraints

$$\begin{array}{rcl} h_i & \geq & 0 \\ h_i & \geq & d(\mu - \rho_i) \end{array}$$

and hence the final problem is

$$\begin{array}{rll} \min_{\theta,\rho_i,h_i} & \sum_i h_i & \\ \text{subject to} & \rho_i = y_{ij} Z_{i,:}\theta & i = 1 \ldots n \\ & h_i \geq 0 & i = 1 \ldots n \\ & h_i \geq d(\mu - \rho_i) & i = 1 \ldots n \\ & \|\theta\|_2 \leq 1 & \end{array}$$

This problem is similar to a standard SVM optimization, but not identical: it has a constraint $\|\theta\| \leq 1$ instead of a penalty proportional to $\|\theta\|_2^2$.

Given Y and Θ, solve for Z

Z is the hardest variable to minimize over, since it appears in both terms of the objective. We can divide the optimization for Z into n subproblems, one per example (row of Z). The ith subproblem is to predict the ith row of Y using the ith row of Z as adjustable weights:

$$Y_{i,:} \approx \mathrm{sgn}(Z_{i,:}\Theta)$$

The Frobenius norm term in the objective is quadratic in Z; if it were just $\|Z\|_{\mathrm{Fro}}^2$ we would have essentially a standard SVM again, but instead we have shifted and scaled the quadratic so that it is more expensive to increase Z along a direction that hurts reconstruction accuracy.

As in Section 3.1.2 we can eliminate $h(\rho_{ij}, \mu, C)$ from the objective by adding variables h_j with appropriate constraints. With this replacement, the ith problem is to find the row vector $Z_{i,:}$ which solves

$$
\begin{array}{lll}
\min_{\zeta,\rho_j,h_j} & \|X_{i,:} - Z_{i,:}W\|^2_{\text{Fro}} + \sum_j h_j & \\
\text{subject to} & \rho_j = y_{ij} Z_{i,:}\Theta_{:,j} & j = 1 \ldots k \\
& h_j \geq 0 & j = 1 \ldots k \\
& h_j \geq d(\mu - \rho_j) & j = 1 \ldots k \\
& Z_{i,1} = 1 & \\
& \|Z_{i,2:end}\|_2 \leq 1 &
\end{array}
$$

Note that $Z_{i,:}$ affects only the i^{th} row of X, which is approximated as a linear combination of the rows of W where the weights are the entries of $Z_{i,:}$. Furthermore, $\|Z_{i,2:end}\|_2 \leq 1$ is used to force the algorithm to settle for one of many possible Z/W pairs, namely one where most of the magnitude of the signal is placed in the W matrix and puts it roughly on the same scale as X.

3.2 SVDM Algorithm B

We developed a different algorithm for learning a SVDM with the goals of simplifying the optimization subproblems and having their objectives be more similar to a traditional SVM. Given a new objective, the algorithm has the same three alternating minimization steps as algorithm A.

3.2.1 Optimization Problem B

As discussed earlier, the SVD computes Z and W without reference to Y or Θ. To improve on the SVD we will simultaneously search for values of Z, W, and Θ which minimize the following objective:

$$
\frac{1}{k_X}\|X - ZW\|^2_{\text{Fro}} + \frac{1}{k_H} \sum_{i=1:n,j=1:k} h(\rho_{ij}) + d\left(\frac{1}{k_Z}\|Z\|^2_{Fro} + \frac{1}{k_\Theta}\|\Theta\|^2_{Fro}\right)
$$

where $\rho_{ij} = y_{ij} Z_{i,:}\Theta_{:,j}$. Here h is the usual hinge loss function

$$
h(\rho) = \begin{cases} 0 & \rho \geq 1 \\ 1 - \rho & \text{otherwise} \end{cases}
$$

This objective trades off reconstruction error (the first term) and a bound on classification error via a hinge loss function (the second term) with the squared Frobenius norms of two matrices, via the parameter d. This is done with a dual purpose. On one hand, the matrix Frobenius norms become the 2-norms of the columns of Θ and rows of Z in the respective optimization subproblems; minimizing them will correspond to maximizing margin width in the classification task. A second reason for having $\|Z\|_{Fro}^2$ is to pick one of the several pairs of Z and W matrices that would reconstruct X equally well, i.e. the one whose Z has the smallest Frobenius norm.

Note that we use a standard hinge loss, as empirically we've found there was little point to having the μ parameter (changes the slope of the hinge loss) in the presence of the d parameter also allowing an increase in the weight of the hinge loss.

The constants k_X,k_H, k_Z and k_Θ compensate for the size of the matrices in each of the terms in the objective by *making the term be the average value in the matrix*. Thus we have $k_X = n \times m$,$k_H = n \times k$, $k_Z = n \times l$ and $k_\Theta = l \times k$.

We impose the constraint $Z_{i,1} = 1$ for two different reasons: the first is to make $\Theta_{1,j}$ be a bias term for the jth linear discriminant, and the second is to make the first row of W correspond to the mean of the dataset. Given that, it will be useful to introduce the following matrices. Let $\bar{X}$ be such that each row i is $\bar{X}_{i,:} = X_{i,:} - W_{1,:}$, $\bar{W} = W_{2:end,:}$, $\bar{Z} = Z_{:,2:end}$ and $\bar{\Theta}$ =Θ $_{2:end,:}$. Then our objective becomes

$$\frac{1}{k_X}\|\bar{X} - \bar{Z}\bar{W}\|_{\text{Fro}}^2 + \frac{1}{k_H}\sum_{i=1:n,j=1:k} h(\rho_{ij}) + d(\frac{1}{k_Z}\|\bar{Z}\|_{Fro}^2 + \frac{1}{k_\Theta}\|\bar{\Theta}\|_{Fro}^2)$$

where $\rho_{ij} = y_{ij}(Z_{i,:}\Theta_{:,j}) = y_{ij}(\bar{Z}_{i,:}\bar{\Theta}_{:,j} + \Theta_{1,j})$.

3.2.2 Solving Optimization Problem B

We solve problem B in the same way that we solve problem A, by iterating the following steps: minimize the objective with respect to the variable W while holding Z and Θ fixed, then minimize with respect to Θ with Z and W fixed, then finally minimize with respect to Z with W and Θ fixed.

Given Θ and Z, solve for W

As there is only one term involving W, we want to minimize

$$\|\bar{X} - \bar{Z}\bar{W}\|_{\text{Fro}}^2$$

with no additional constraints. That is, we wish to predict $\bar{X}$ as a linear function of $\bar{Z}$ using coefficients $\bar{W}$, minimizing the sum of squared prediction errors. We can find the best $\bar{W}$ by solving a linear regression problem for each column of $\bar{X}$:

$$\bar{X}_{:,j} \approx \bar{Z}\bar{W}_{:,j}$$

For stability we add a tiny ridge term to each regression [25].

Given W and Z, solve for Θ

Since W and Z are fixed, we can drop the first part of the objective as well as the constraints that don't involve Θ. The rest of the problem is then

$$\begin{array}{rll} \min_{\Theta,\rho_{ij}} & \frac{1}{k_H}\sum_{i=1:n,j=1:k} h(\rho_{ij}) + d\frac{1}{k_\Theta}\|\bar{\Theta}\|_{Fro}^2 & \\ \text{subject to} & \rho_{ij} = y_{ij}(\bar{Z}_{i,:}\bar{\Theta}_{:,j} + \Theta_{1,j}) & i = 1\dots n,\ j = 1\dots k \end{array}$$

This optimization problem tells us to choose Θ so that $\operatorname{sgn}(Z_{i,:}\Theta_{:,j})$ is a good predictor of Y_{ij}; that is, it is a linear threshold classification problem. We can divide the optimization into k subproblems, each of which tries to find weights $\theta = \bar{\Theta}_{:,j}$ and $\theta_0 = \Theta_{1,j}$ which predict the jth column of Y from the features Z:

$$\begin{array}{rll} \min_{\theta,\theta_0,\rho_i} & \sum_{i=1:n} h(\rho_i) + d\frac{k_H}{k_\Theta}\|\theta\|_2^2 & \\ \text{subject to} & \rho_i = y_{ij}(\bar{Z}_{i,:}\theta + \theta_0) & i = 1\dots n \end{array}$$

This problem is a standard linear SVM optimization.

Given Y and Θ, solve for Z

Z is the hardest variable to minimize over, since it appears in three of the four terms of the original objective, yielding

$$\begin{array}{rl} \min_{Z,\rho_{ij}} & \frac{1}{k_X}\|\bar{X} - \bar{Z}\bar{W}\|_{\mathrm{Fro}}^2 + \frac{1}{k_H}\sum_{i=1:n,j=1:k} h(\rho_{ij}) + d\frac{1}{k_Z}\|\bar{Z}\|_{Fro}^2 \\ \text{subject to} & \rho_{ij} = y_{ij}(\bar{Z}_{i,:}\bar{\Theta}_{:,j} + \Theta_{1,j}),\ i = 1\dots n, j = 1\dots k \end{array}$$

We can divide the optimization for Z into n subproblems, one per example. The ith subproblem is to predict the ith row of Y using the ith row of Z as adjustable weights:

$$Y_{i,:} \approx \operatorname{sgn}(Z_{i,:}\Theta)$$

and thus, letting $\zeta = \bar{Z}_{i,:}$:

$$\begin{array}{rll} \min_{\zeta,\rho_j} & \|\bar{X}_{i,:} - \zeta\bar{W}\|_2^2 + \frac{k_X}{k_H}\sum_{j=1:k} h(\rho_j) + d\frac{k_X}{k_Z}\|\zeta\|_2^2 & \\ \text{subject to} & \rho_j = y_{ij}(\zeta\bar{\Theta}_{:,j} + \Theta_{i,j}) & j = 1\dots k \end{array}$$

This problem can be solved directly using an optimization package or converted into one similar to the linear SVM optimization problem, in the manner we proceed to describe. The first and third terms of the objective can be transformed into a single norm minimization term through variable substitution (letting the variable be $\zeta = \bar{Z}_{i,:}$ and $c = d\frac{k_X}{k_Z}$), as follows:

$$
\begin{aligned}
\|\bar{X}_{i,:} - \zeta\bar{W}\|_2^2 + c\|\zeta\|_2^2 &= \bar{X}_{i,:}\bar{X}'_{i,:} - 2\zeta\bar{W}\bar{X}'_{i,:} + \zeta\bar{W}\bar{W}'\zeta' + c\zeta\zeta' \\
&\quad \text{(drop } \bar{X}_{i,:}\bar{X}'_{i,:}\text{, as } \zeta \text{ does not appear)} \\
&= -2\zeta\bar{W}\bar{X}'_{i,:} + \zeta\bar{W}\bar{W}'\zeta' + c\zeta I\zeta' \\
&= -2\zeta\bar{W}\bar{X}'_{i,:} + \zeta(\bar{W}\bar{W}' + cI)\zeta' \\
&\quad \text{(Cholesky factorization } R'R = \bar{W}\bar{W}' + cI\text{)} \\
&= -2\zeta\bar{W}\bar{X}'_{i,:} + \zeta R'R\zeta' \\
&\quad \text{(make the substitutions } p' = \zeta R', \zeta = p'(R^{-1})'\text{)} \\
&= -2p'(R^{-1})'\bar{W}\bar{X}'_{i,:} + p'p \\
&\quad \text{(close to a quadratic form, minus constants)} \\
&= (p - (\bar{X}'_{i,:}\bar{W}'R^{-1})')'(p - (\bar{X}'_{i,:}\bar{W}'R^{-1})') - \text{constant} \\
&\quad \text{(subst. } q = p - (\bar{X}'_{i,:}\bar{W}'R^{-1})'\text{)} \\
&= q'q \text{ (same minimum as the starting problem)}
\end{aligned}
$$

where R is a $l \times l$ matrix and p and q are $1 \times l$ vectors. The substitutions $p' = \zeta R', \zeta = p'(R^{-1})'$, $q = p - (\bar{X}'_{i,:}\bar{W}'R^{-1})'$ and $p = q + (\bar{X}'_{i,:}\bar{W}'R^{-1})'$ can then be used on the second term of the objective, the hinge loss for each example/classification problem:

$$
\begin{aligned}
h(\rho_{ij}) &= h(y_{ij}Z_{i,:}\Theta_{:,j}) \\
&= h(y_{ij}(\zeta\Theta_{2:end,j} + \Theta_{1,j})) \\
&\quad \text{(subst. } \zeta = p'(R^{-1})'\text{)} \\
&= h(y_{ij}(p'(R^{-1})'\Theta_{2:end,j} + \Theta_{1,j})) \\
&\quad \text{(subst. } p = q + (\bar{X}'_{i,:}\bar{W}'R^{-1})'\text{)} \\
&= h(y_{ij}((q + (\bar{X}'_{i,:}\bar{W}'R^{-1})')'(R^{-1})'\Theta_{2:end,j} + \Theta_{1,j})) \\
&= h(y_{ij}(q' + (\bar{X}'_{i,:}\bar{W}'R^{-1}))(R^{-1})'\Theta_{2:end,j} + \Theta_{1,j})) \\
&= h(y_{ij}(q'(R^{-1})'\Theta_{2:end,j} + (\bar{X}'_{i,:}\bar{W}'R^{-1})(R^{-1})'\Theta_{2:end,j} + \Theta_{1,j})) \\
&= h(y_{ij}(q'\text{vector } + \text{constant depending on example } i + \text{constant}))
\end{aligned}
$$

The resulting problem is then

$$
\begin{aligned}
\min_{q,\rho_j} &\quad \|q\|_2^2 + \tfrac{k_X}{k_H}\textstyle\sum_{j=1:k} h(\rho_j) \\
\text{subject to} &\quad \rho_j = y_{ij}(q'(R^{-1})'\Theta_{2:end,j} + (\bar{X}'_{i,:}\bar{W}'R^{-1})(R^{-1})'\Theta_{2:end,j} + \Theta_{1,j})
\end{aligned}
$$

which is almost the same as the usual SVM optimization, but for two differences. The first is that the hinge loss for each example adds a separate constant that depends on that example; the second is that the trade-off parameter d has moved into the computation of q. Given a solution q^*, we can use it and the substitution formulae to obtain p and then $\zeta = \bar{Z}_{i,:}$, the row of Z that we seek.

3.3 Low-dimensional and feature-space discriminants

In addition to classification performance we are also concerned with understanding which voxels provide information to the classifier. Given that we are learning a linear classifier, one such way would be to understand which voxels influence it. If we were learning a classifier with voxels as features, the weights assigned to them would be one possible gauge of such influence (but see Chapter 2 and [11] for caveats). In our case, the classifier works in component-space, hence it's not immediately clear how voxels contribute.

To find such a contribution, consider that for every component-space discriminant (column of Θ), there exists a voxel-space discriminant that is equivalent in the sense that it makes the same predictions. Let us call the matrix of equivalent discriminants T, and have each column $\mathbf{t}$ be the equivalent discriminant for the corresponding column of Θ, the vector θ.

In order to find T we need to state the requirement that equivalent discriminants make the same predictions from Z or from ZW, i.e.

$$Z\Theta = ZWT$$

and solve the resulting equation, $\Theta = WT$. $W'\Theta = W'WT$ cannot be solved by multiplying by $(W'W)^{-1}$, as W is singular. Instead, we will solve it considering that each column of T, $\mathbf{t}$, can be solved as a separate problem in terms of each column of Θ, θ:

$$\begin{array}{rl} \min_{\mathbf{t}} & \frac{1}{2}\|\mathbf{t}\|_2^2 \\ \text{subject to} & \theta = W\mathbf{t} \end{array}$$

where, of the many solutions, we choose the one with the smallest norm. This can be converted into an unconstrained optimization problem using Lagrange multipliers λ:

$$\min_{\mathbf{t}} \max_{\lambda} \quad \frac{1}{2}\|\mathbf{t}\|_2^2 + \lambda'(\theta - W\mathbf{t})$$

Differentiating with respect to λ and setting it equal to 0 gives us $\mathbf{t} = W'\lambda$. Replacing that in the problem yields:

$$\begin{array}{rl} \max_{\lambda} & \frac{1}{2}\|W'\lambda\|_2^2 + \lambda'(\theta - WW'\lambda) \\ \dots & \\ \max_{\lambda} & -\frac{1}{2}\lambda' WW'\lambda + \lambda'\theta \end{array}$$

and deriving with respect to θ and setting it equal to 0

$$-WW'\lambda + \theta = 0 \leftrightarrow \lambda = (WW')^{-1}\theta$$

The equivalent discriminant $\mathbf{t}$ is thus $\mathbf{t} = W'\lambda = W'(WW')^{-1}\theta$, a linear combination of the rows of matrix W.

It's also possible to state the problem differently by defining the equivalent discriminants as T such that:

$$Z\Theta = XT$$

Solving the problem

$$\begin{array}{rl} \min_{\mathbf{t}} & \frac{1}{2}\|\mathbf{t}\|_2^2 \\ \text{subject to} & Z\theta = X\mathbf{t} \end{array}$$

in a similar way, we get an equivalent discriminant $\mathbf{t}$ such that $\mathbf{t} = X'(XX')^{-1}Z\theta$, a linear combination of the rows of matrix X, the original examples.

3.4 Related Work

The most common approach to building low-dimensional representations that carry information about a variable of interest is neural networks ([3]). In a typical neural network the hidden layer contains a non-linear low-dimensional representation, on which one or more classifiers (output units) are trained and used to predict the variable of interest.

There are two reasons why neural networks are not suitable for our purpose of building informative low-dimensional representations. Neural networks don't perform as well as a linear classifier given as many features and as few examples as we have, if we use all the voxels as features (see Section 2.1.3 of Chapter 2). Even if successful, they would not give us a way of summarizing an example image as a sum of basis images. An auto-encoder network would give us something closer to that but, as the goal would be reconstruction rather than classification, there's no guarantee that the hidden units could be ascribed meaning by relating them to the target variable via a classifier a posteriori. Having both decoder and classifier output units has been suggested before ([32]), though we were not aware of this work at the time of the publication of the first SVDM paper ([57]). Given that we have had bad results for networks using all the voxels with tens of examples in dataset D1 – they are almost always outperformed by linear classifiers – it is unclear whether this approach would be worthy of further development.

An approach that maps the ability to predict a variable of interest to locations on a brain image is Partial Least Squares (introduced in [48] and extended in [49]), works by decomposing a covariance matrix between the values of voxels and the value(s) of the variables(s). This differs from our approach in that we use a SVM to learn the relationship between component coordinates and the target variable, rather than look for correlations.

Another issue of concern in learning low-dimensional representations is the extent to which they capture noise or other peculiarities of the training set that will not necessarily be present or relevant in a test set. This has been studied for both PCA/SVD ([23]) and ICA ([28]), using criteria that do not have anything to do with example labels. In our case, we would like to show a component is relevant by being informative.

In addition to the above, the work described in this chapter draws on several lines of research, in particular:

1. the use of additional regularization on classifiers such as logistic regression or SVM to reflect domain characteristics and provide additional leverage in learning from sparse datasets

2. the use of linear classifiers to determine feature relevance, by considering the weights placed on each feature by the learnt discriminant

3. the joint optimization of several desirable objectives, with some of them being related to classification performance and others not (e.g. characteristics of the distribution of the unlabelled data).

4. low-rank matrix factorizations of data enforcing particular constraints or regularization (e.g. all entries of one of the matrices are positive)

The work most closely related in the first line of research is [63], as it describes the use of a 1-norm linear SVM on SPECT (Single Photon Emission Computed Tomography) imaging data, using voxels as features. The authors solve a modified SVM problem that favours weight vectors where neighbouring voxels have weights of similar magnitude. The goal is to lead to a linear classifier that uses only a few contiguous brain locations, to make interpretation easier and to prevent it from assigning weight to individual noisy voxels.

In the second line of research, [21] suggested that the weights of a successful linear discriminant can provide a better indication of a feature's relevance for classification than criteria that rely on each feature by itself; our work shares this intuition. However, instead of looking at the original feature space directly, we learn a classifier on a low-dimensional subspace and propagate the weights back to the original space, thus coupling them further than a classifier by itself would. Another difference is that the authors wish to select a few features by eliminating correlated ones, while we wish to combine and summarize the collective activity of correlated features and not eliminate them. A more general treatment of combined SVM-based feature selection and classification is given in [53]. Our work deviates from it in using reconstruction error and the coupling of features via a matrix decomposition as additional regularization criteria, instead of using more traditional criteria such as various norms of the learnt weight vector.

The learning of a metric suitable for classification as well as other goals is a good example of the third line of research. Both [69] and [17] introduce algorithms for learning a Mahalanobis distance metric for use in nearest-neighbour classification, differing in the objective used and its relation to the classification error. This is in contrast with other metric learning methods insofar as it optimizes directly for classification instead of other related criteria (e.g. the clustering measure in [72]). Our work is more related to [17] in that the metric introduced there can be used to do a linear projection of the data into a low dimensional space, much as we aim to do by finding a basis and then reducing each example vector to its coordinates in that basis.

The use of the reconstruction error for regularization and the matrix decomposition to further couple weights in the original matrix - while using the learnt low dimensional representation for classification - was inspired by the maximum-margin matrix factorization described in [62], work squarely in the fourth line of research.

3.5 Experimental Results

3.5.1 Datasets

In the experiment performed to obtain these datasets the subject is shown a series of words. Each word is the name of an item which is either a kind of tool or a kind of building, and the subject performs the following task: think about the item and its properties while the word is displayed (3 seconds) and clear her mind afterwards (8 seconds of blank screen).

In all four datasets a trial is converted into an example by taking the average image during a 4 second span while the subject is thinking about the stimulus shown a few seconds earlier (these 4 seconds contain the expected peak of the signal during the trial, TR=1000msec).

- Dataset D1

  There are 7 different items belonging to each of the two categories and 6 experimental epochs. In each epoch all 14 items are named, without repetition; all 6 epochs have the same items. Using each trial to obtain an example, this yields a total of 42 examples for each of the two categories. The classification task is to predict the category of a example. This dataset has three subjects.

- Dataset D2

  This dataset is very similar to dataset D1 except for the fact that both word and picture stimuli were used, in alternation. We excised the picture stimulus portion of the dataset (the category classification task is far too easy) and retained the text stimulus portion as dataset D2. The classification task is to predict the category of a example. There are 5 items of each category and 6 experimental epochs, which gives us a total of 30 examples for each of the two categories. This dataset has three subjects (the three best ones from dataset D2 in Chapter 2)

For more details on the datasets please refer to Section 2.3.1 in Chapter 2.

3.5.2 Classification experiments (algorithm B)

Procedure

The experiments described in this chapter were almost all done using algorithm B, except for those where algorithms A and B are compared (and this is explicitly noted in the text and figure legends). This was done because algorithm B is roughly two times faster than algorithm A and also because it is easier to search for a value of its tradeoff parameter, as it is less dependent on the scale of the data (Section 3.5.4 has more details on comparing the two algorithms).

For a given dimensionality/number of components l (that is, Z has one column of 1s and l other columns) we ran a 6-fold cross-validation, corresponding to a natural division of the dataset into the 6 epochs of task performance by a subject. We also tried several settings of the d parameter to give a broader picture of its influence.

All of the experiments described in this chapter are done using 6-fold cross-validation, leaving data from each of the 6 experimental epochs as test data in turn and training on the remaining ones.

The classifiers used for comparison were either a Gaussian Naïve Bayes (GNB) [52] or a linear Support Vector Machine (SVM, using `libSVM` [10]). The linear SVM tradeoff parameter was set using cross-validation within the training set. The class conditional variance of each feature in GNB was estimated by pooling data from both classes after the class conditional means were subtracted. The cross-validation and task were exactly the same as for the SVDM experiments.

Results

Figure 3.1 displays the results obtained using SVDM models to do the category classification task in datasets D1 and D2, respectively in the top and bottom sections. The section for each dataset contains 6 graphs, one per setting of the tradeoff parameter d. d could also be set via cross-validation within the training set, by selecting the highest D that can learn an accurate prediction model, as verified in [57]. This would be analogous to setting the tradeoff parameter in an SVM to the largest possible margin that works well in the training set. We refrain from doing this here in order to show the full gamut of model behaviour, rather than just that which leads to the best classification accuracy. Each graph plots the classification accuracy of the model learnt with that setting and using between 1 and 24 components; this is the number of dimensions in the representation learnt and thus of rows in the W matrix, where each row is a basis image/component.

There are three solid curves in each graph – red, green and blue, one per subject – plotting test set accuracy in cross-validation. The numbers below each graph are the peak accuracies for each of the three subjects (first line) and the smallest number of components at which peak accuracy was attained (second line). Each accuracy number is the average accuracy over 10 runs of the algorithm using different random seeds; these affect the initial random guesses for the three matrices the algorithm learns. Finally, the dashed curves are the same, except for training set rather than test set accuracy. . The figures displaying accuracy plots through this chapter will be consistent with the style used in this one and, in the interest of brevity, most of the details will be omitted from figure captions.

Convergence

As described earlier, the algorithm iterates through solving three subproblems whose objectives are convex in alternation. The global objective never increases and converges to a *local* minimum of that global objective, with the criterion for convergence used being a change of less than 0.5% in the objective. Given that criterion, the running time is roughly linear in the number of examples and components.

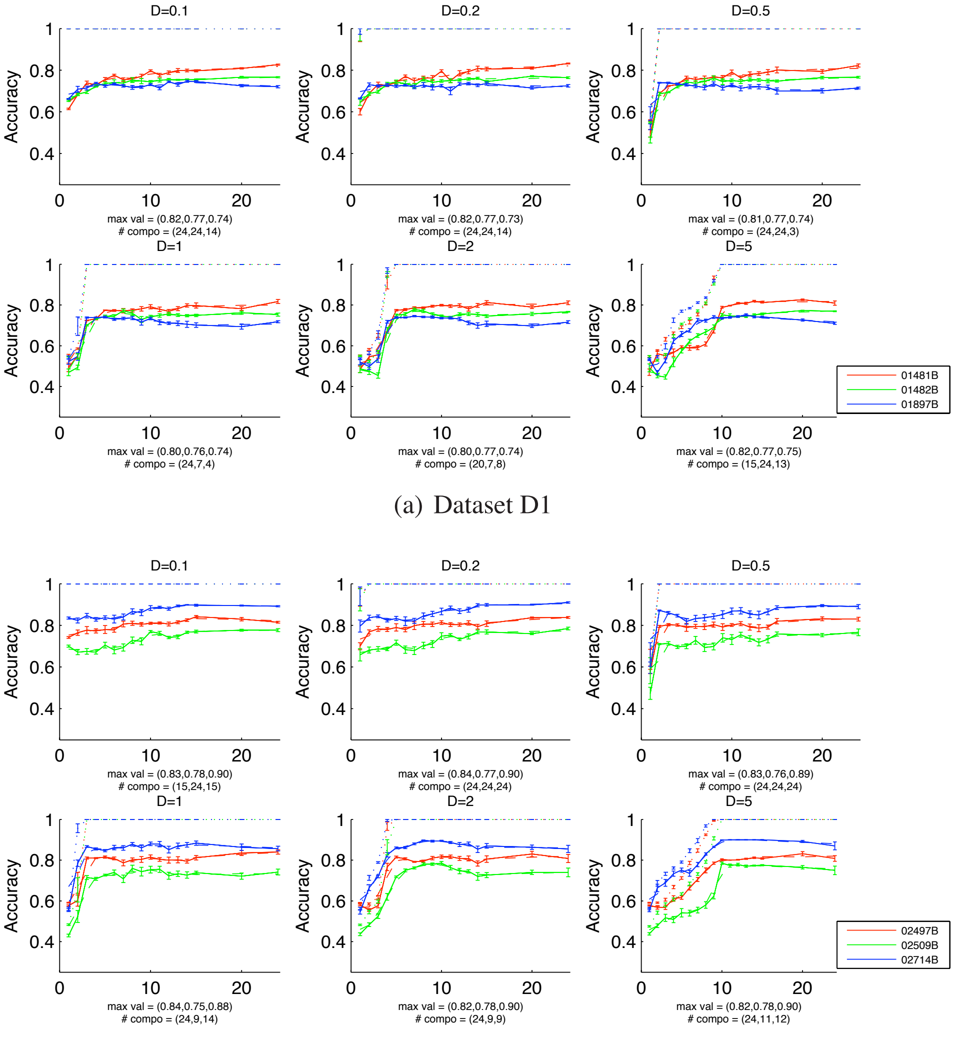
3.5.3 Comparison with classifiers

Accuracy and discriminants learnt

In order to compare the model learnt by SVDM with conventional classifiers, one can consider both the peak classification accuracy and the voxel space discriminant learnt. The latter is easy to obtain for conventional linear classifiers, and Section 3.3 describes how one can obtain a voxel space discriminant that makes the same predictions as the discriminant in the low-dimensional space of the SVDM model. In all the voxel space discriminant plots in this chapter what is being plotted is the *absolute value* of the discriminant weight, so that the weight magnitude is more evident. Without this, some weights would be dark red (high, positive, leans the classification towards class 1) and others dark blue (high, negative, leans the classification towards class 2).

Figure 3.2 contrasts, for subject 01481B in dataset D1, the best SVDM model (top, using d=2) with SVM (middle) and Naive Bayes (bottom) classifiers, in terms of both the voxel space discriminant learnt and the peak accuracy reached (the number in parentheses). Figure 3.3 and Figure 3.4 show the same for the other two subjects in dataset D1, and Figure 3.6 and Figure 3.7 for the three subjects in dataset D2.

In terms of accuracy, SVDM performs roughly as well as a linear SVM using all the voxels and much better than Naive Bayes, for both datasets. The voxel space discriminants learnt are, again, very similar between SVDM and SVM, with Naive Bayes generally producing a noisier discriminant. Note that all three put high weights on similar locations, with GNB weighing many



(a) Dataset D1

(b) Dataset D2

Figure 3.1: Classification results for SVDM models learnt with algorithm B. **Top:** Dataset D1 (subjects 01481B,01482B,01897B): each graph shows the accuracy (y axis) for one setting of the d parameter across a number of components ranging from 1 to 24 (x axis). The solid lines plot average test set accuracy across cross-validation folds for the three subjects, the dashed ones training set accuracy. Each line is the average of 10 restarts of the algorithm using different random seeds. The numbers below each graph are the peak accuracies for each of the three subjects and the smallest number of components at which peak accuracy was attained. **Bottom:** The same for the three subjects in dataset D2 (02497B,02509B,02714B).

other voxels in addition. Although we do not detail that here, we have preliminary evidence on dataset D1 and D3 that the voxels weighted by GNB but not by SVM/SVDM are voxels with fairly high correlation between each other. This is a sign of one of the effects of the regularization on the L2 norm of the weight vector done by SVM or SVDM, which lowers the weights of very correlated voxels (whereas GNB would weight them all the same way); this is a question we are investigating further.

Regions identified

A second kind of validation for the discriminant model learnt with SVDM is examining the brain regions where it places the most weight. As detailed in Section 2.4 for the case of dataset D1, we have both prior expectations of whether certain brain areas will be involved (e.g. voxels around the central sulcus or the parahippocampal gyrus) and also some information about locations of voxels deemed significant in searchlight accuracy maps.

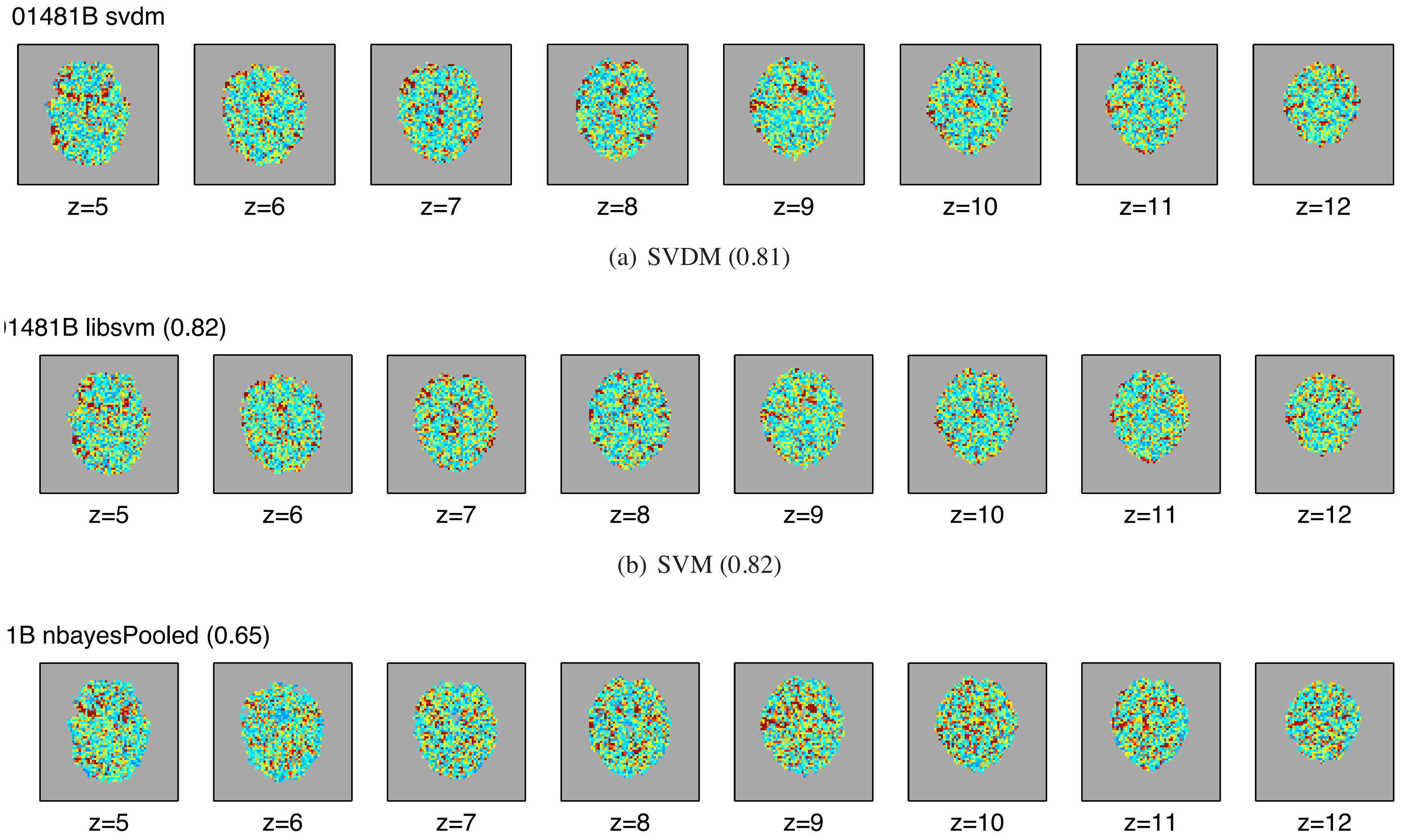
Figure 3.8 shows the voxel space discriminants learnt with SVDM for the three subjects in dataset D1 with annotations highlighting how they place weight in those two areas. The locations are similar to those found by plotting significant searchlight accuracy maps (see Section 2.4 for details and figures with the structural MRI images for the three subjects). Again, what is being plotted is the *absolute value* of the discriminant voxel weight, so that the weight magnitude is more evident. This said, the discriminant places weight in several other locations, some of which are also significant in searchlight accuracy maps and others which are not.

3.5.4 Algorithm A versus algorithm B

Motivation

Algorithm A was developed first and introduced in [57]. Algorithm B was developed subsequently, in an effort to simplify the various optimization subproblems by removing constraints and making them more similar to a conventional SVM. The main question while developing algorithm B was whether it would reach a solution to the SVDM problem similar to the one algorithm A would, both in terms of the model learnt and also of classification performance.

Why would one choose one algorithm over the other? Aside from the technical differences described in Section 3.0.6 and Section 3.2, there are other relevant practical issues. The first stems from the fact that algorithm A trades off the reconstruction error and hinge loss terms directly. This tradeoff involves both the magnitude of the error and also the size of the reconstruction error matrix (#examples by #voxels) versus the much smaller size of the hinge loss matrix (#examples by #problems). Algorithm B uses instead the average reconstruction error over all matrix entries and similar averages for the other matrices in its objective. This makes it simpler to do a search for

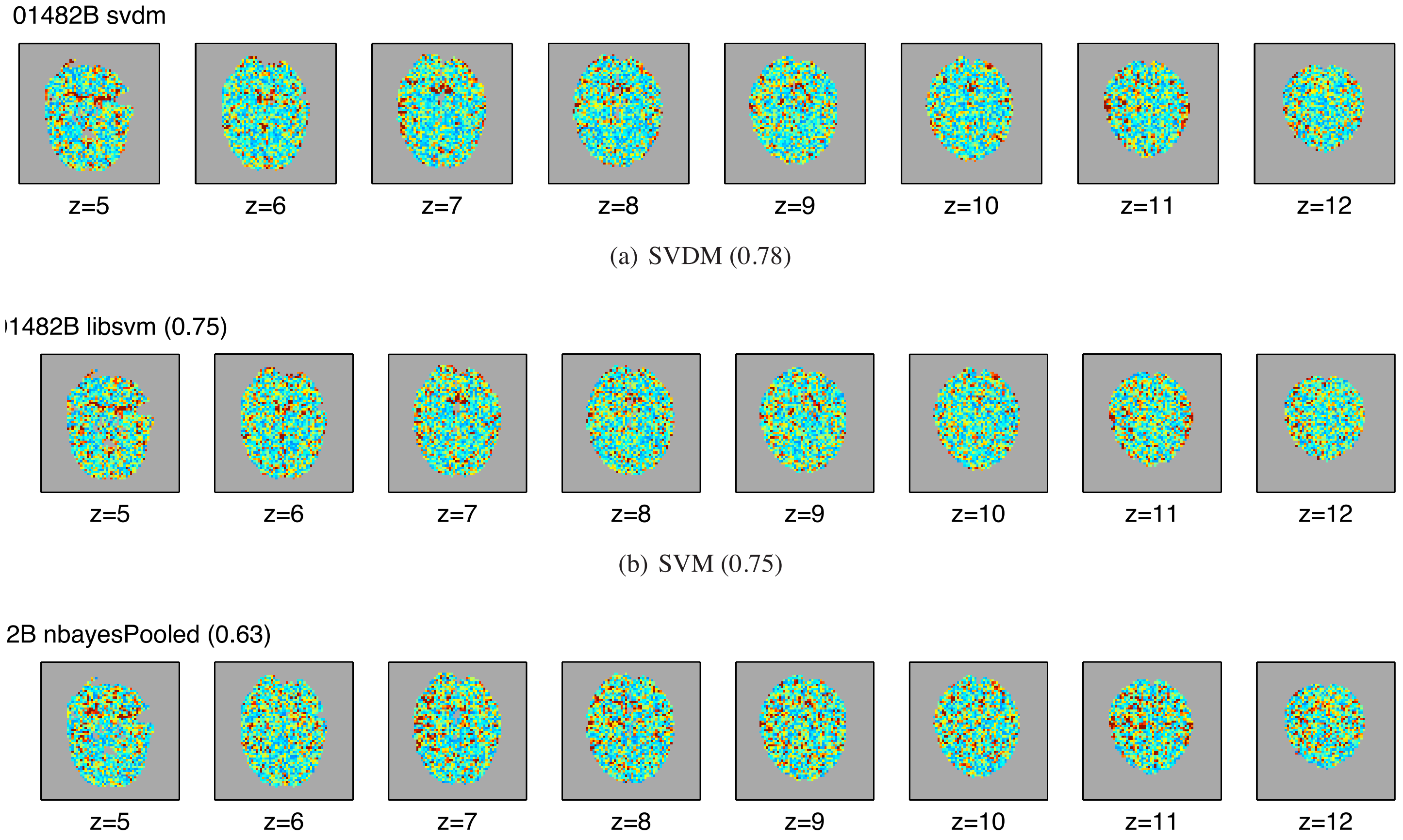


(a) SVDM (0.81)

(b) SVM (0.82)

(c) GNB (0.65)

Figure 3.2: **Top:** Voxel space discriminant for subject 01481B in dataset D1 (d=2,15 components). **Middle/Bottom:** Voxel space discriminants learnt with SVM and Naive Bayes, respectively. The colour in each voxel is the *absolute value* of the weight the discriminant gives that voxel, so that the weight magnitude is more evident. Without this, some weights would be dark red (high, positive, leans the classification towards class 1) and others dark blue (high, negative, leans the classification towards class 2). The number in parentheses is the accuracy of each classifier. The voxel space discriminants learnt are, across subjects, very similar between SVDM and SVM, with Naive Bayes generally producing a noisier discriminant. Note that all three put high weights on similar locations, with GNB weighting many other voxels in addition.

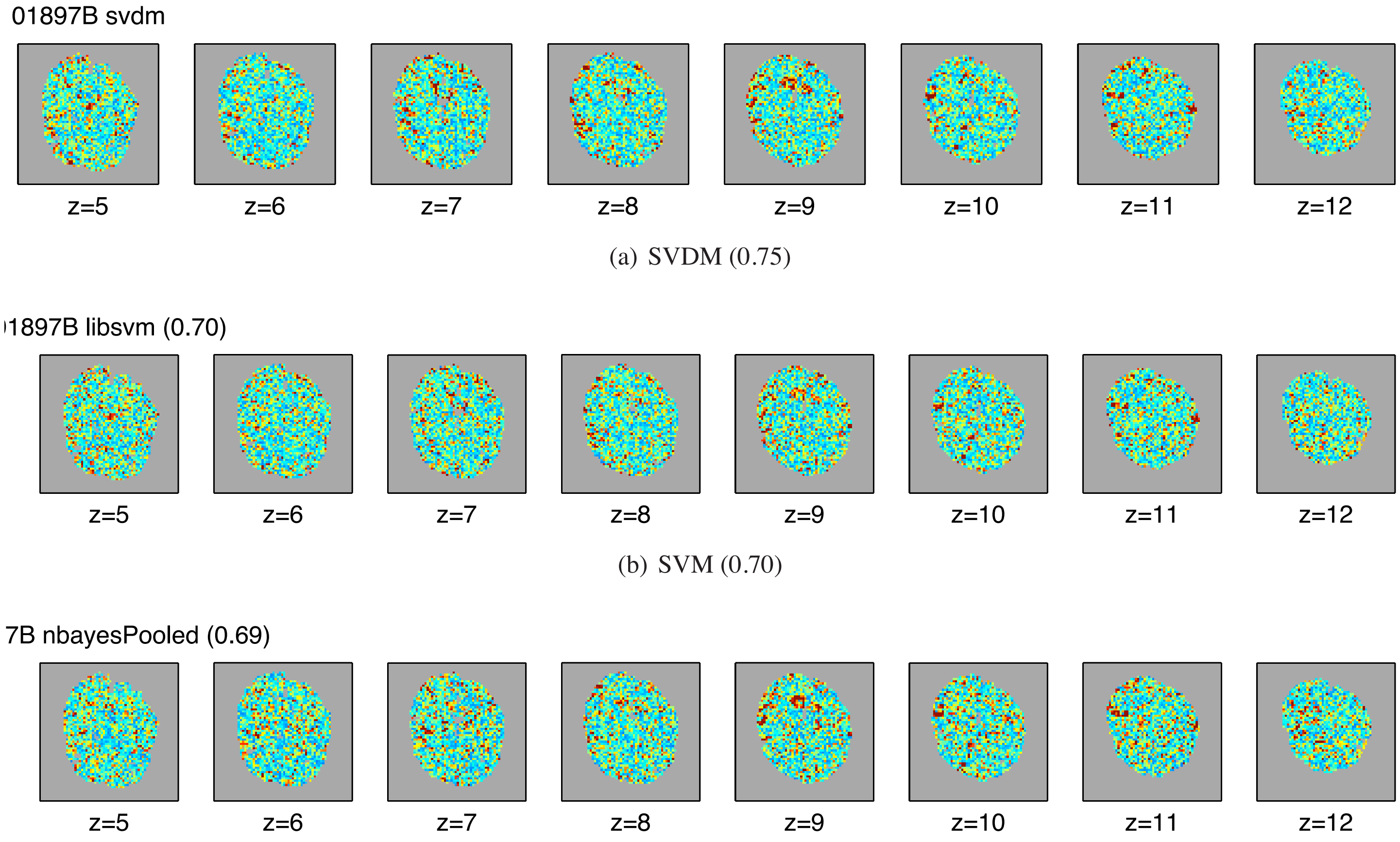


(a) SVDM (0.78)

(b) SVM (0.75)

(c) GNB (0.63)

Figure 3.3: Same as Figure 3.2, for subject 01482B in dataset D1 (d=2,7 components).

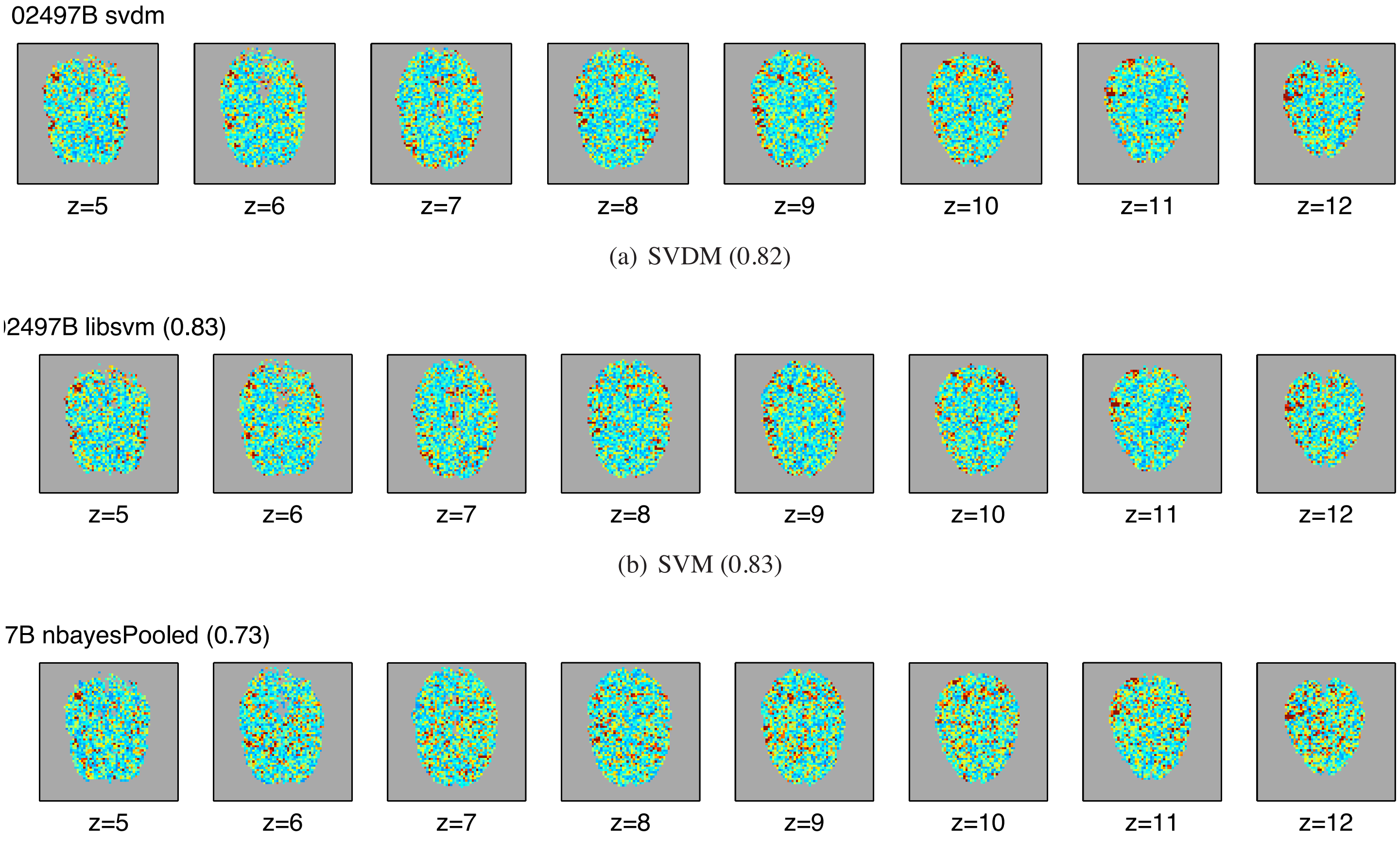


(a) SVDM (0.75)

(b) SVM (0.70)

(c) GNB (0.69)

Figure 3.4: Same as Figure 3.2, for subject 01897B in dataset D1 (d=2,7 components).

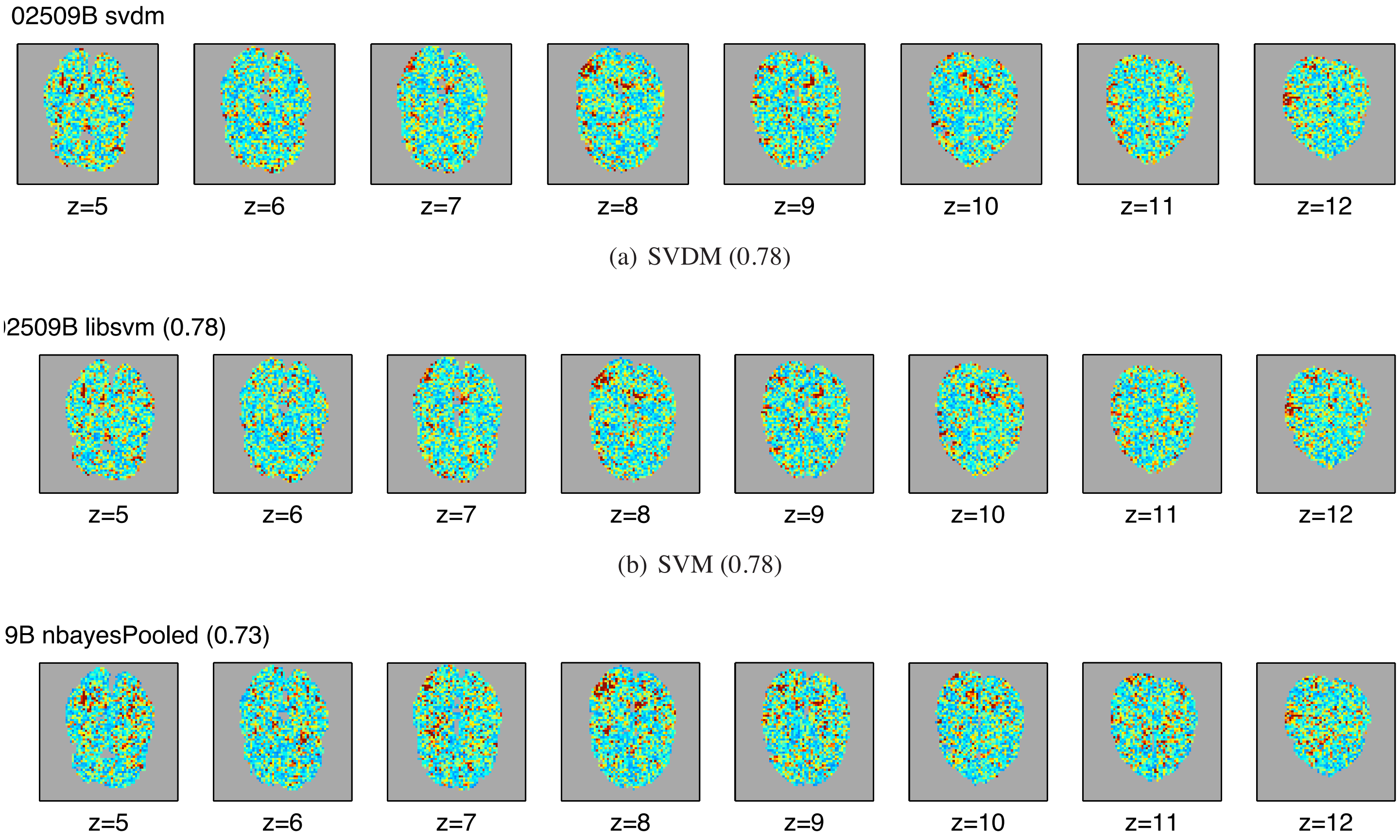


(a) SVDM (0.82)

(b) SVM (0.83)

(c) GNB (0.73)

Figure 3.5: Same as Figure 3.2, for subject 02497B in dataset D2 (d=2,10 components).

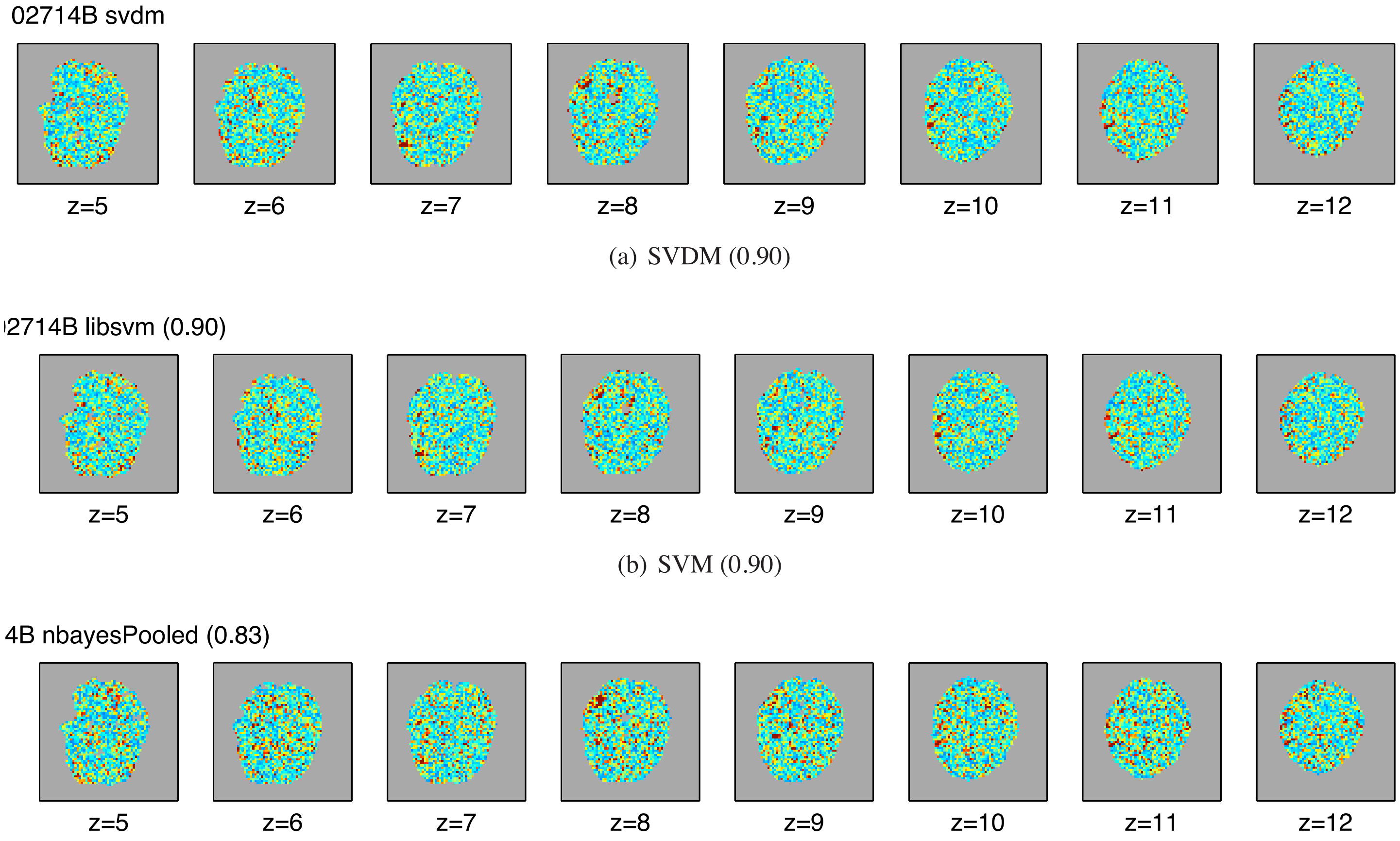


(a) SVDM (0.78)

(b) SVM (0.78)

(c) GNB (0.73)

Figure 3.6: Same as Figure 3.2, for subject 02509B in dataset D2 (d=2,10 components).



(a) SVDM (0.90)

(b) SVM (0.90)

(c) SVM (0.83)

Figure 3.7: Same as Figure 3.2, for subject 02714B in dataset D2 (d=2, 8 components).

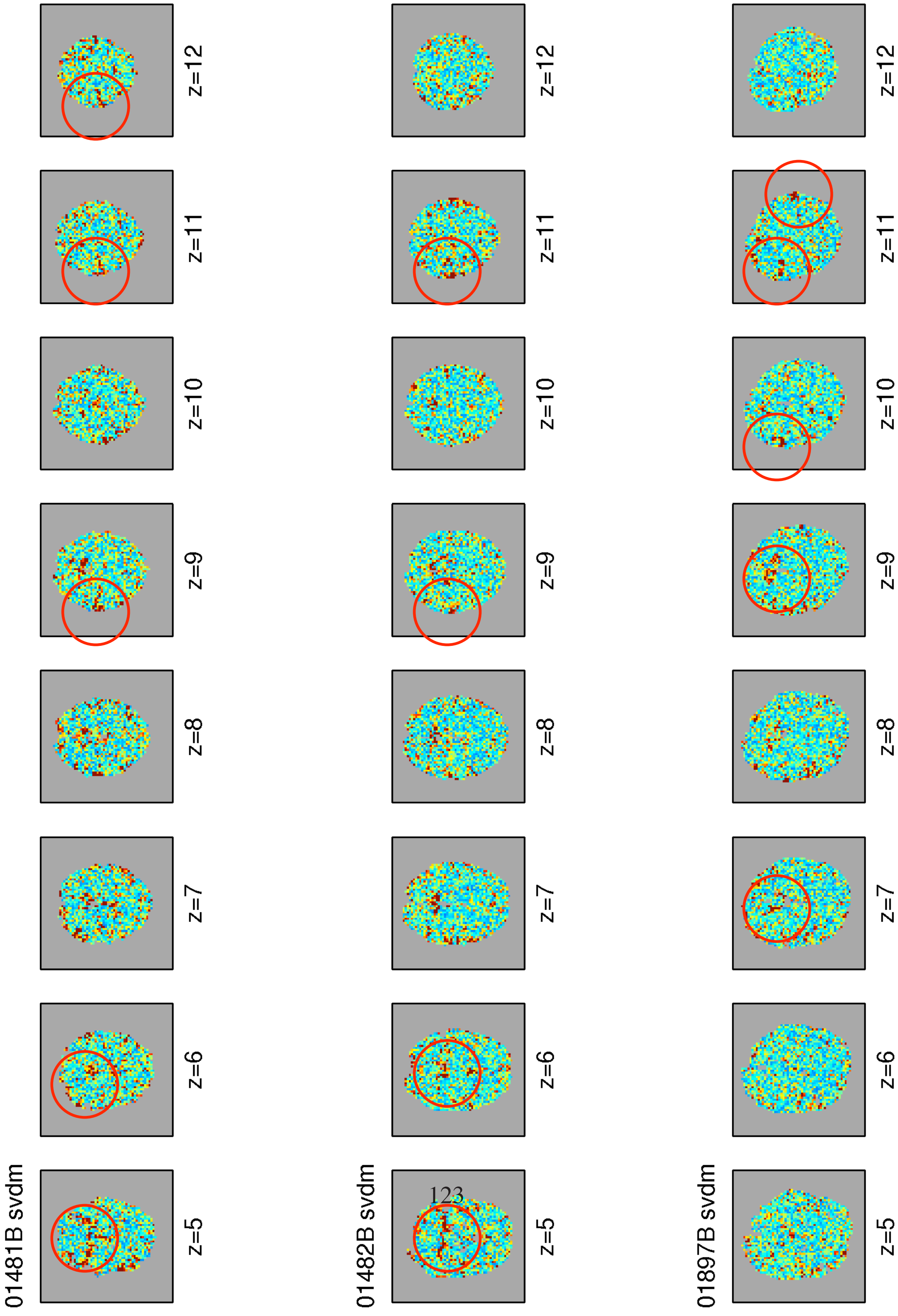

01481B svdm
z=5
z=6
z=7
z=8
z=9
z=10
z=11
z=12
01482B svdm
z=5
z=6
z=7
z=8
z=9
z=10
z=11
z=12
01897B svdm
z=5
z=6
z=7
z=8
z=9
z=10
z=11
z=12

Figure 3.8:

the value of the d parameter, as a small range (say $[0, 10]$) seems to suffice to cover the gamut of model behaviours.

A second issue is the speed at which the SVDM can be trained. Empirically, we've found that algorithm B is roughly 2 times faster than algorithm A at training a model with the same number of components, over a range of numbers of components and settings of the tradeoff parameter. In addition, the optimization sub-problem for Θ is a standard SVM problem and the one for Z can be modified to be similar to one; this would allow better performance if we replaced those steps with off-the-shelf packages (or simple modifications of those).

Results

Figure 3.9 shows the classification results in dataset D1 using the two algorithms, for a range of settings of the d parameter in each. The peak accuracies attained with either algorithm are comparable, as are the curves as the number of components is increased; the results appear to be slightly more variable across seeds in algorithm B results, for a given d and number of components. The most likely source for this is the optimization objective for the Z subproblem leading to some numerical instability near the solution.

A few discriminants learnt for the three subjects are shown in Figure 3.10, Figure 3.11 and Figure 3.12. The discriminants were obtained using the same number of components for all subjects and a seed that gave the same accuracy results for both algorithm A and B in all cross-validation folds, yielding very similar discriminants. Given that and the faster speed of algorithm B we will use algorithm B for the remainder of this chapter and also Chapter 4, except where otherwise noted.

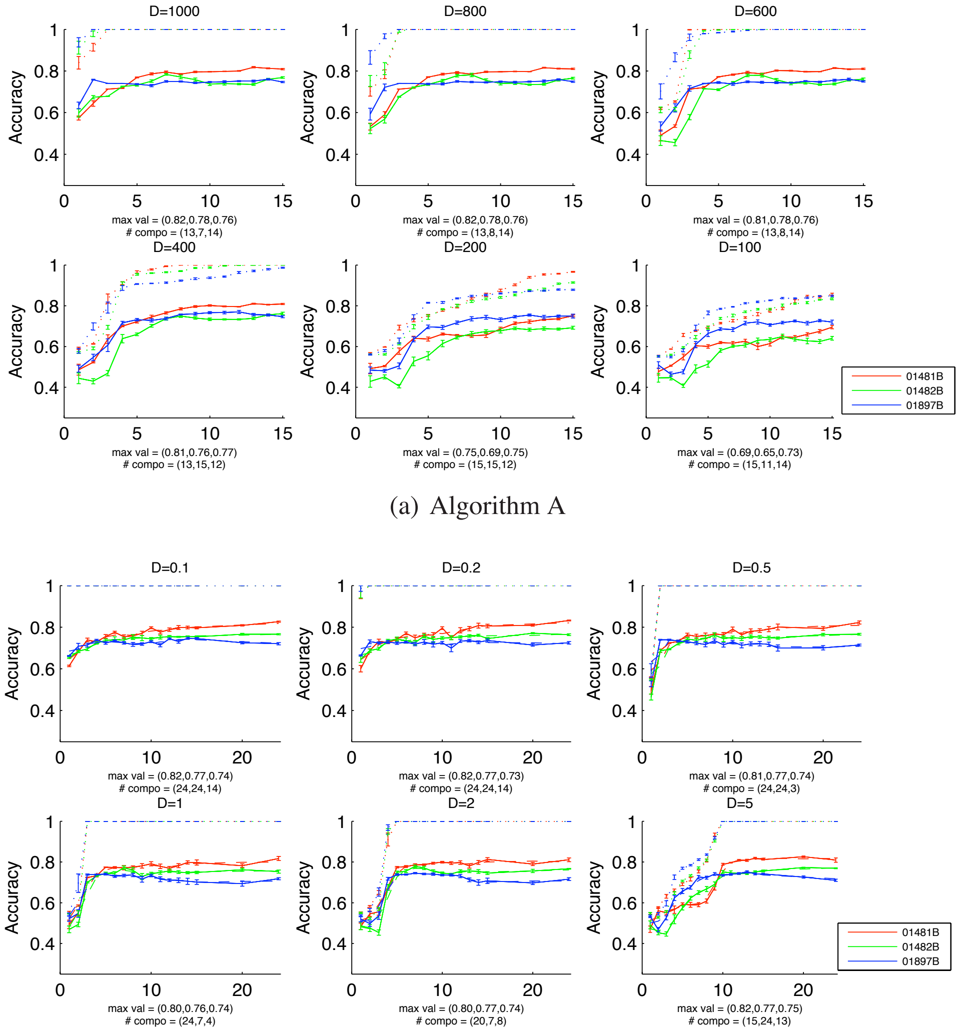
3.5.5 Algorithm B versus SVD and ICA

Procedure

Singular Value Decomposition (SVD) and Independent Component Analysis (ICA) are two popular methods for generating low-dimensional representations of a dataset without referring to class labels. The question we are interested in is whether these low-dimensional representations are *informative*, i.e. can they be used to predict class labels; if so, we also want to know how they fare compared with SVDM.

Both methods can be used to generate matrix factorizations $X = ZW$ akin to that produced with SVDM:

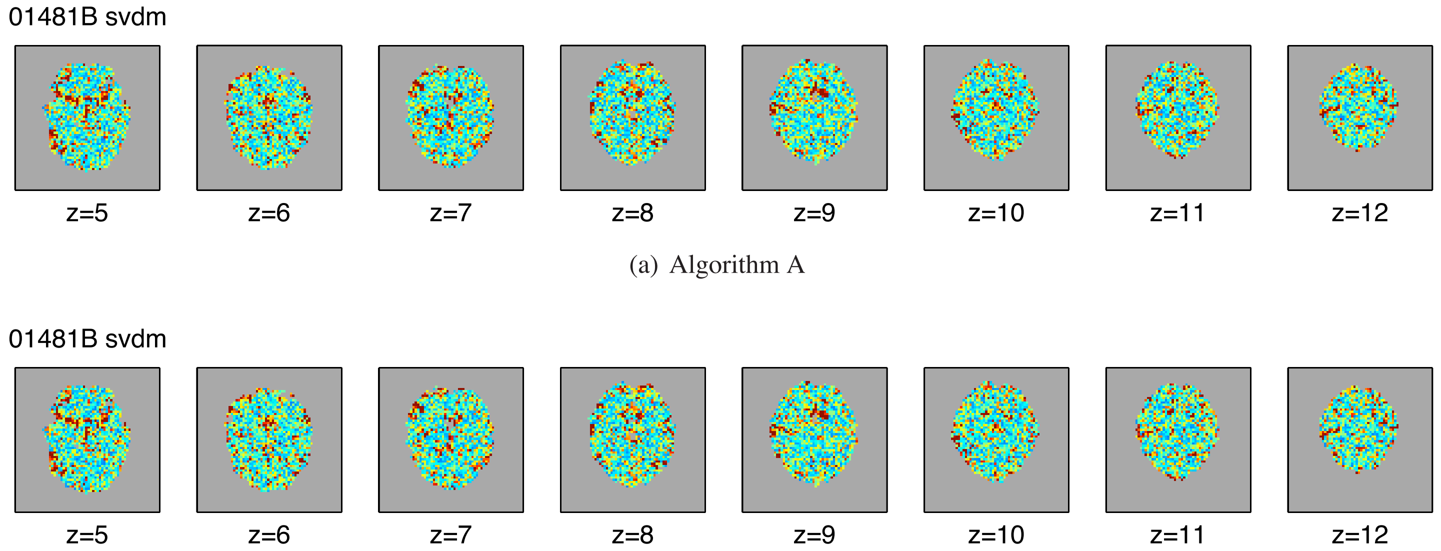
- SVD: $X = USV'$ $Z = U * S$ $W = V'$ (using MATLAB and our own code)
- ICA: $X = ZW$, where W are the independent components and Z is the mixing matrix (using FastICA, [22])



(a) Algorithm A

(b) Algorithm B

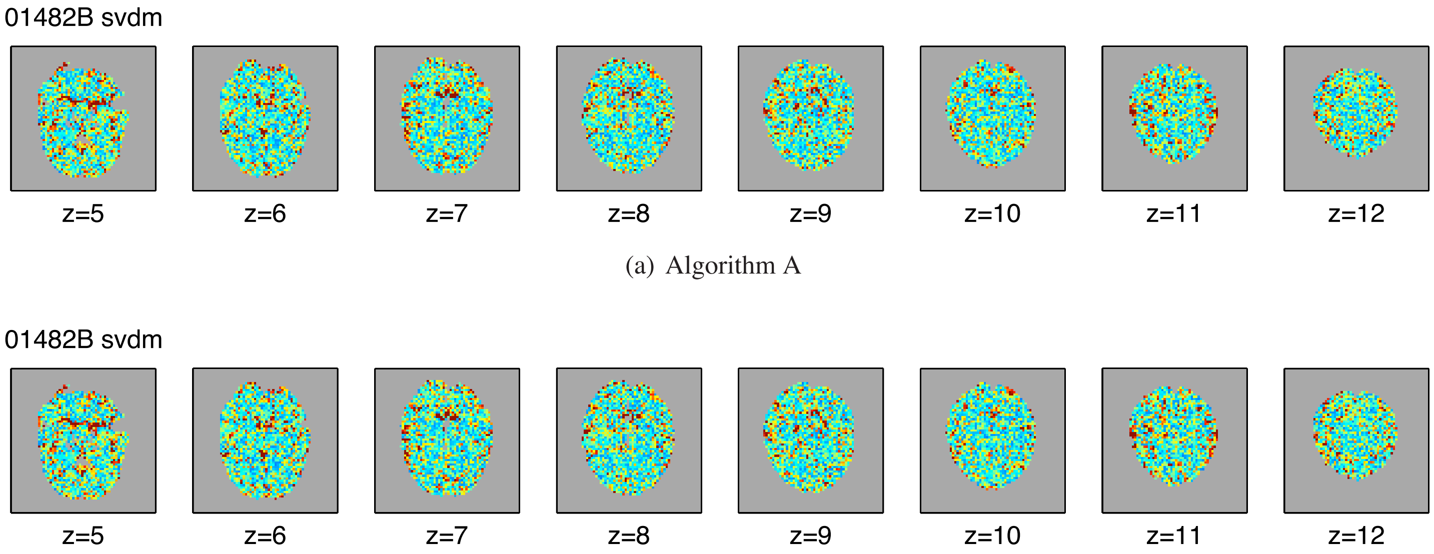
Figure 3.9: **Top:** Classification results for Algorithm A SVDM models trained on the three subjects in dataset D1. Each pane shows the results for one setting of the d parameter across a number of components ranging from 1 to 24 [TODO: redo A plot for 24 components]. The solid lines plot average test set accuracy across cross-validation folds for the three subjects, the dashed ones training set accuracy. Each line is the average of 10 restarts of the average using different random seeds. **Bottom:** The same for Algorithm B SVDM models (same results as in Figure 3.1).



(a) Algorithm A

(b) Algorithm B

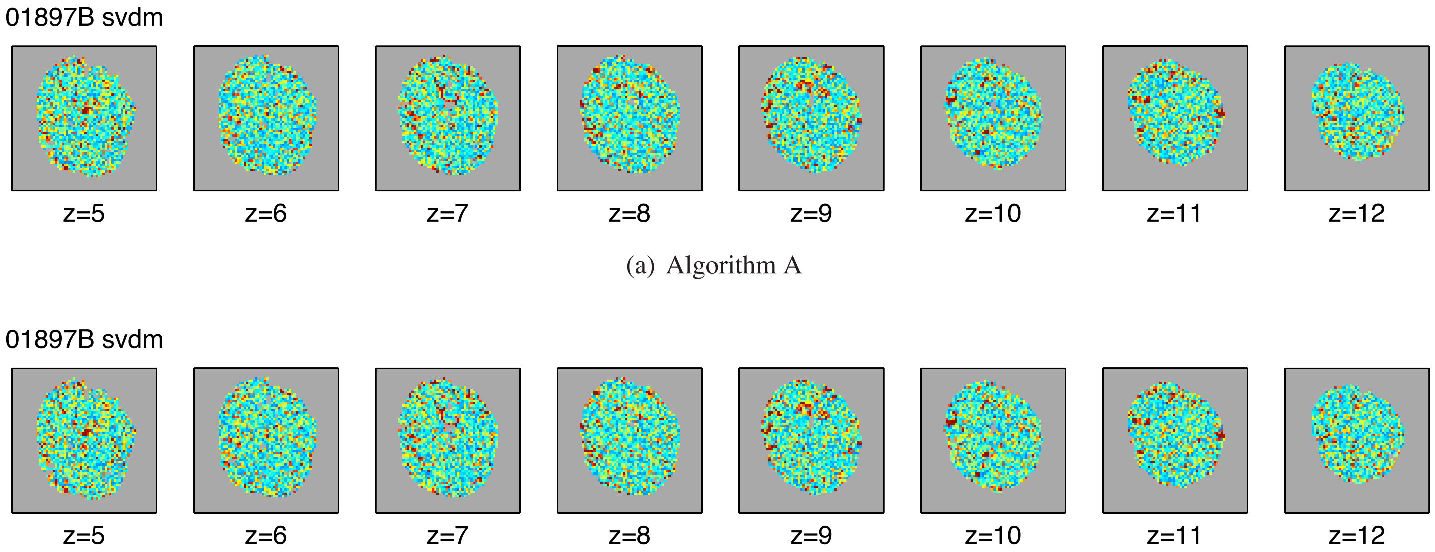
Figure 3.10: **Top:** Voxel space discriminant for subject 01481B in dataset D1, obtained from an Algorithm A SVDM model (10 components, d=800). **Bottom:** Same using an Algorithm B SVDM model (10 components, d=200).



(a) Algorithm A

(b) Algorithm B

Figure 3.11: Same as Figure 3.10 for subject 01482 in dataset D1



(a) Algorithm A

(b) Algorithm B

Figure 3.12: Same as Figure 3.10 for subject 01897 in dataset D1

and those representations can be tailed to use a given number of components (columns of Z or rows of W).

In order to compare the low-dimensional representation of the data learnt with SVDM with that learnt with SVD or ICA, we trained classifiers – Naive Bayes and linear SVM – from those representations to perform the same classification task described in previous sections.

In this comparison we are concerned mainly with accuracy and how quickly it is attained. Another direction of comparison would be to see whether the learnt basis images produced by either SVD or ICA were, in a sense, interpretable, similarly to what is done in Section 3.5.3. While we will not purse this issue further here, there are indications that spatial ICA basis images tend to be more interpretable than SVD, as judged by the time course corresponding to each basis image and the fact that they will generally be sparse and have non overlapping locations [5]. In spatial ICA, however, it's also often the case that task related activity in different classes that overlaps spatially (e.g. the one in [26]) will tend not be separated into different basis images, which would be the desired behaviour in order to classify from the corresponding time courses.

Results

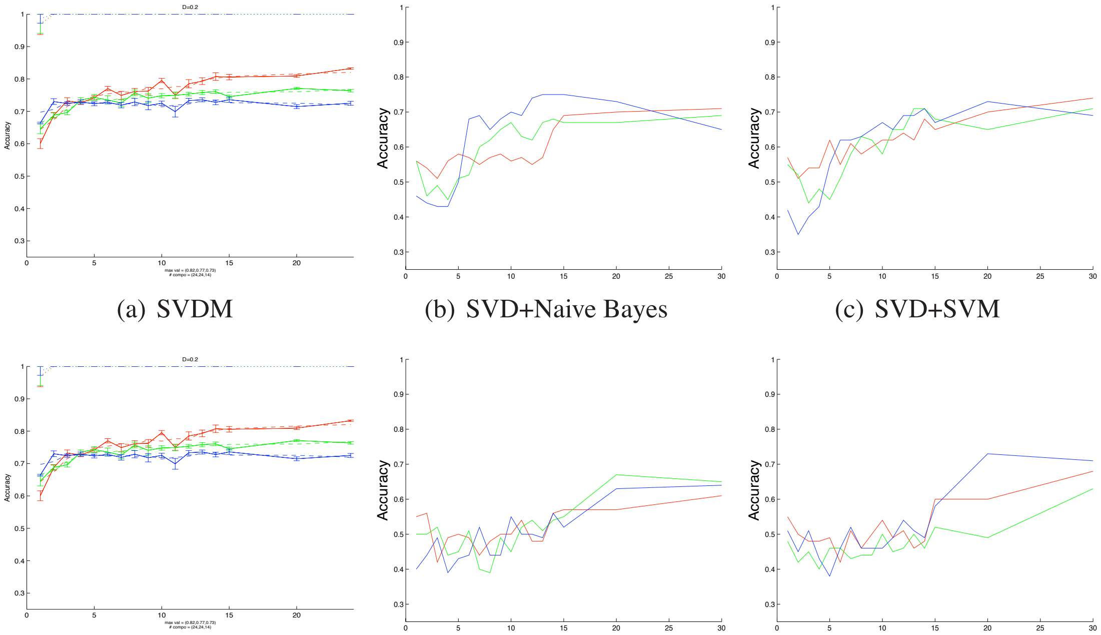
We generated low-dimensional representations using from 1 to 24 components from the training data in each cross-validation fold using both methods. Using those low-dimensional representations we then trained both Naive Bayes and linear SVM classifiers and used them to classify the test data. The results of this are shown in Figure 3.13 for dataset D1 and in Figure 3.14, for dataset D2, together with similar result curves for a SVDM model (d=0.2) using the corresponding number of components.

We can also compare the various methods by looking at the peak accuracy they can reach regardless of the number of components, by considering Table 3.1. The main conclusions are that SVDM attains peak accuracy using fewer components than SVD or ICA and, moreover, that peak accuracy is the same or higher.

Conversely, one could ask whether it is the classifier part of SVDM (the Θ matrix) or the low-dimensional representation (Z) that are responsible for the performance. We verified in [57] that the low-dimensional representation learnt with SVDM can be used with linear SVM or Gaussian Naive Bayes classifiers with virtually the same results as using the classifier part of SVDM, showing that informative components were learnt. We do not show the result curves here mainly because they are practically indistinguishable.

3.5.6 Constraining the number of informative components

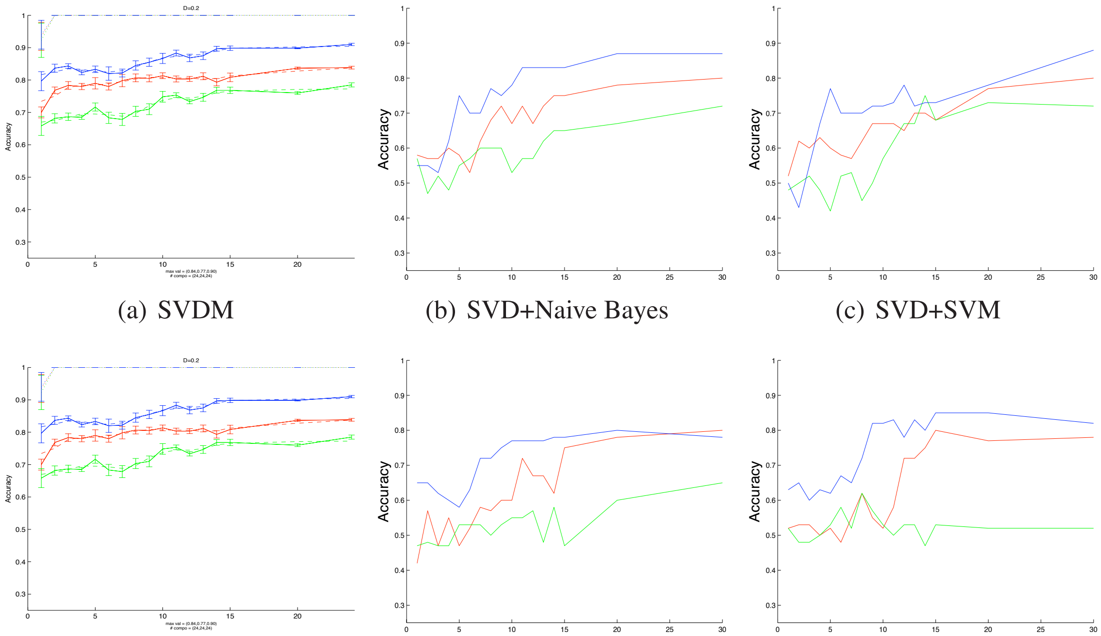
One important question to ask of a learnt SVDM model is which components have a relationship with the classification label(s) and thus are *informative* for the classifier portion of the model. Ex-



(a) SVDM (b) SVD+Naive Bayes (c) SVD+SVM

(a) SVDM (b) ICA+Naive Bayes (c) ICA+SVM

Figure 3.13: **Row 1:** Classification results for Algorithm B SVDM models (left) for the three subjects in dataset D1 versus the results of Naive Bayes (middle) or SVM (right) trained on SVD-generated low-dimensional representations. Each pane shows the results across a number of components ranging from 1 to 24, using d=2 for the SVDM. **Row 2:** Same as Row 1, but on ICA-generated low-dimensional representations.



(a) SVDM (b) SVD+Naive Bayes (c) SVD+SVM

(a) SVDM (b) ICA+Naive Bayes (c) ICA+SVM

Figure 3.14: **Rows 1/2:** Same as Rows 1 and 2 in Figure 3.13, respectively, but using dataset D2.

	method				
subject/dataset	SVD		ICA		all voxels
	peak	#comp.	peak	#comp.	
Dataset D1	linear SVM				
01481B	0.75	40	0.71	40	0.82
01482B	0.71	13	0.69	40	0.75
01897B	0.74	40	0.75	40	0.70
	GNB				
01481B	0.75	40	0.64	30	0.65
01482B	0.73	40	0.65	30	0.63
01897B	0.75	13	0.69	30	0.69
Dataset D2	linear SVM				
02497B	0.83	40	0.80	15	0.83
02509B	0.78	40	0.75	40	0.78
02714B	0.90	30	0.88	30	0.90
	GNB				
02497B	0.82	40	0.80	30	0.73
02509B	0.75	40	0.73	40	0.73
02714B	0.88	30	0.80	20	0.83

Table 3.1: Peak accuracy of classifier (linear SVM or Gaussian Naive Bayes) trained on a low-dimensional representation produced with SVD or ICA (using between 1 and 40 components) or on all voxels. The top half of the table pertains to dataset D1, the bottom half to dataset D2.

amining the voxel-space equivalent discriminant considers the aggregate effect of all components and is thus not suitable for this purpose. A more appropriate approach is to examine the Z matrix directly and consider the columns for which the corresponding Θ weights are not zero. These columns correspond in turn to the components that affect the learnt discriminant.

To make this more concrete, Figure 3.15 shows the Z matrices learnt on each of six cross-validation folds for one subject, using 10 components (top left), and the corresponding Θ vectors (top right). The higher the absolute value of an entry in a Θ vector, the more it influences classification; note also that *all* non-zero entries will influence the classification. In each training dataset the example labels alternate in groups between the two classes, "tools" and "buildings", and this is visible as the red/blue alternation in column 7 of the Z matrix, which is one with the highest absolute value. The value of Θ indicates how helpful to prediction each component is but, given that most of them influence prediction and that there are positive and negative weights (which can make components offset each other) it's harder to say which components are truly informative.

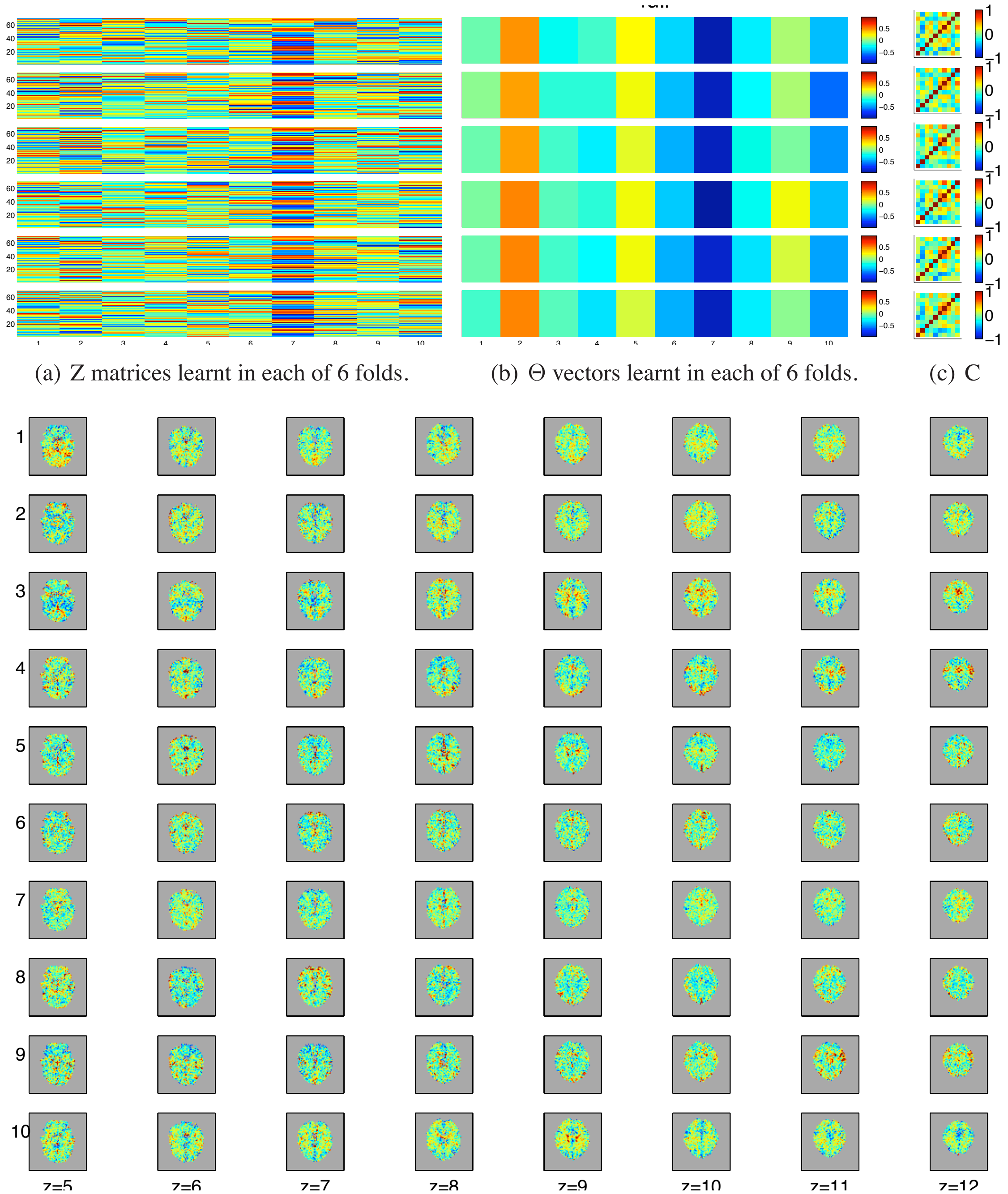
This gives rise to an interesting possibility: constraining most Θ weights to be zero, freeing the respective components to be used in lowering reconstruction error and forcing the remaining components to focus on classification. As an illustration of the effect, consider Figure 3.16, which shows the same matrices depicted in figure Figure 3.15, using the same number of components but constraining all but 2 of them to have Θ $=0$. The effect is to make the 2^{nd} column of Z have a much more pronounced alternation between examples of the two classes, with the corresponding component being sparser than the others.

Given that this approach has the desired effect on the learnt model, the next question to ask is whether if affects classification performance. Figure 3.17 shows the result curves obtained letting all, half and a quarter (rounded down) of the components influence the learnt discriminant, for the two datasets and with two different settings of D. From these it seems reasonable to conclude that constraining does not affect the accuracy of the learnt models and, moreover, leads to similar performance curves as the number of components is increased.

Given that classification performance is very similar regardless of restriction, the final question to ask is how restriction affects the learn model in general. Figure 3.18 shows the learnt components for the three subjects, letting all, half and quarter of the components influence the learnt discriminant, (when using 10 components, for the sake of example). The breakdown of informative and non informative components is clearly visible in the columns that alternate between the two labels. Figure 3.19 shows the same for subjects in dataset D2.

Finally, one can ask whether constraining component influence leads to different voxel space discriminants. Figure 3.20 contrasts the voxel space discriminants obtained letting all, half and quarter of the components influence the learnt discriminant for subject 01481B, with Figure 3.21 and Figure 3.22 doing the same for the other two subjects in D1 and Figure 3.23, Figure 3.24 and Figure 3.25 for the three subjects in dataset D2.

Even when severely constraining the number of components the discriminants learnt are still



(a) Z matrices learnt in each of 6 folds.

(b) Θ vectors learnt in each of 6 folds.

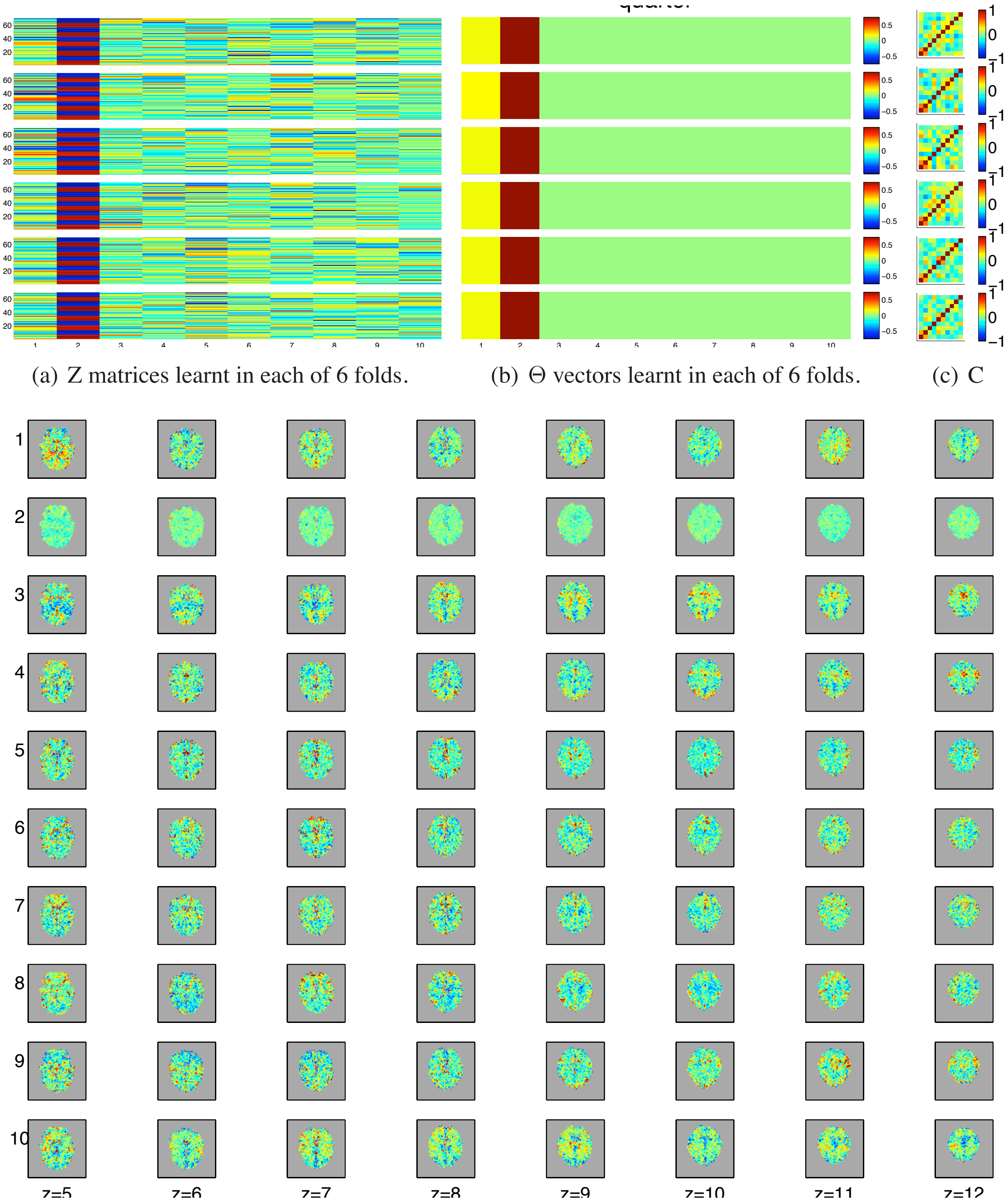
(c) C

(d) W matrix learnt on the 5^{th} fold (one row: all slices of the corresponding row of W).

Figure 3.15: Matrices learnt for one subject in 6 crossvalidation folds (Z and Θ, top left and middle), correlation of columns of the Z matrices (top right).

very similar, and this happens for all subjects in the two datasets. One possible reason for this is the fact that this is a two class problem and hence not too many components are necessary to produce a reasonable discriminant.

In conclusion, restricting the number of components that can affect the learnt discriminant by setting most of the discriminant weights to 0 is feasible and produces the desired effect, without a performance penalty. This approach still requires setting the number of components that can affect prediction, but Chapter 4 will describe a method for doing so automatically.



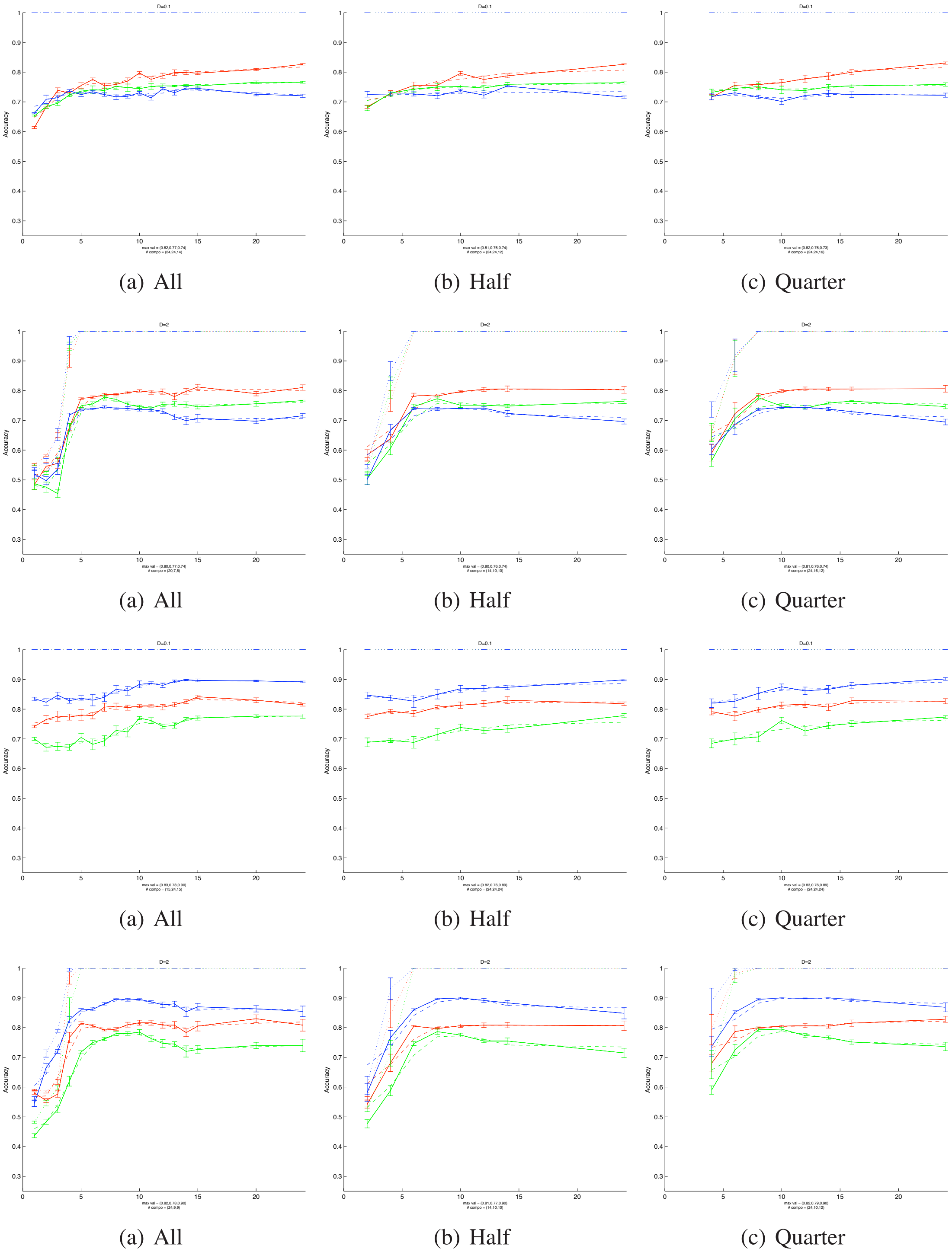
(a) Z matrices learnt in each of 6 folds.

(b) Θ vectors learnt in each of 6 folds.

(c) C

(d) W matrix learnt on the 5^{th} fold (one row: all slices of the corresponding row of W).

Figure 3.16: Same as Figure 3.15, but allowing only the first two entries in each Θ vector to be different from 0.



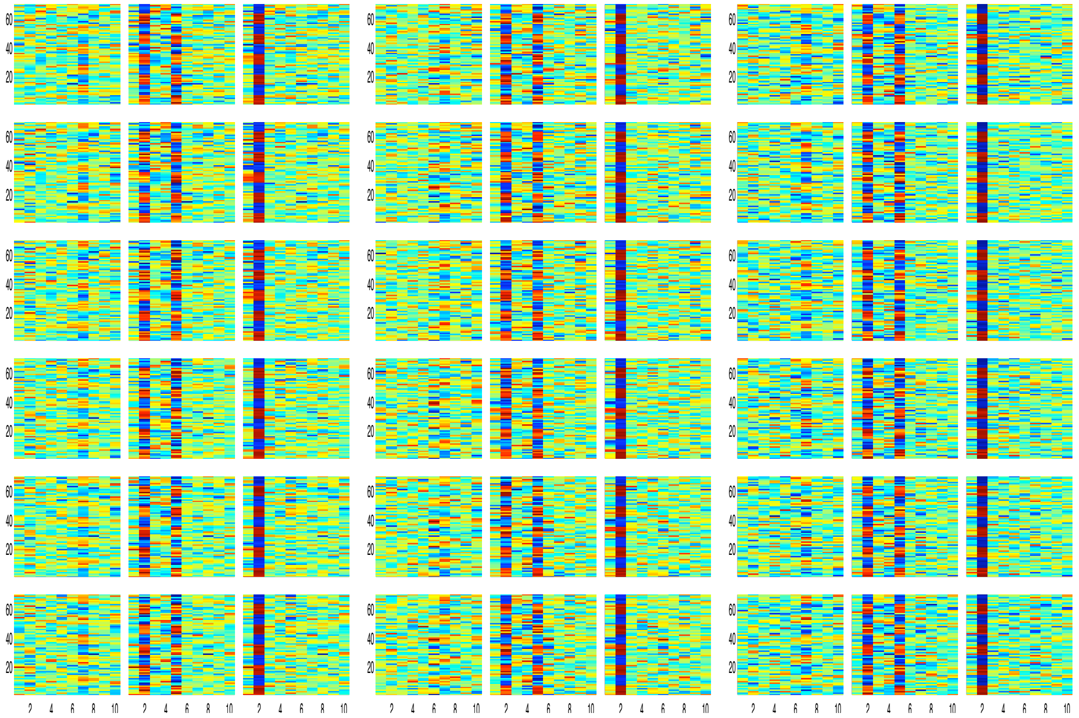
(a) All (b) Half (c) Quarter

(a) All (b) Half (c) Quarter

(a) All (b) Half (c) Quarter

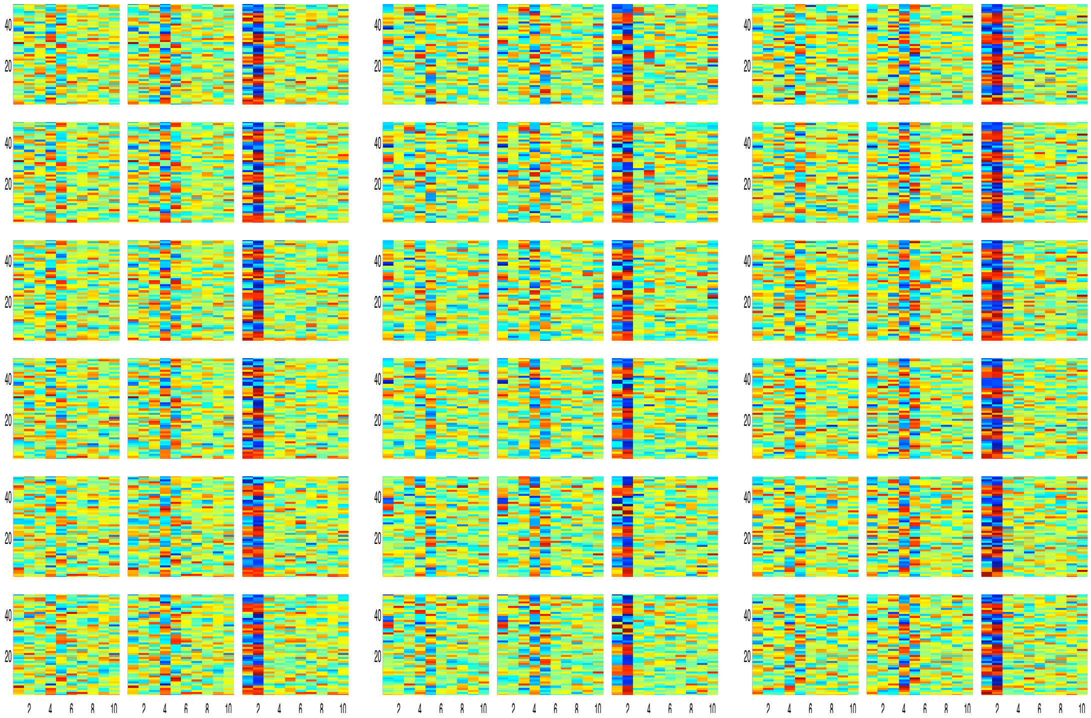
(a) All (b) Half (c) Quarter

Figure 3.17: **rows 1/2:** Classification results for Algorithm B SVDM models trained on the three subjects in dataset D1, letting different numbers of components influence the learnt classifier (row 1: d=0.5, row 2: d=2). Left is all the components, middle is half the components and right is a quarter of the components, rounded down. **rows 3/4** The same for dataset D2.



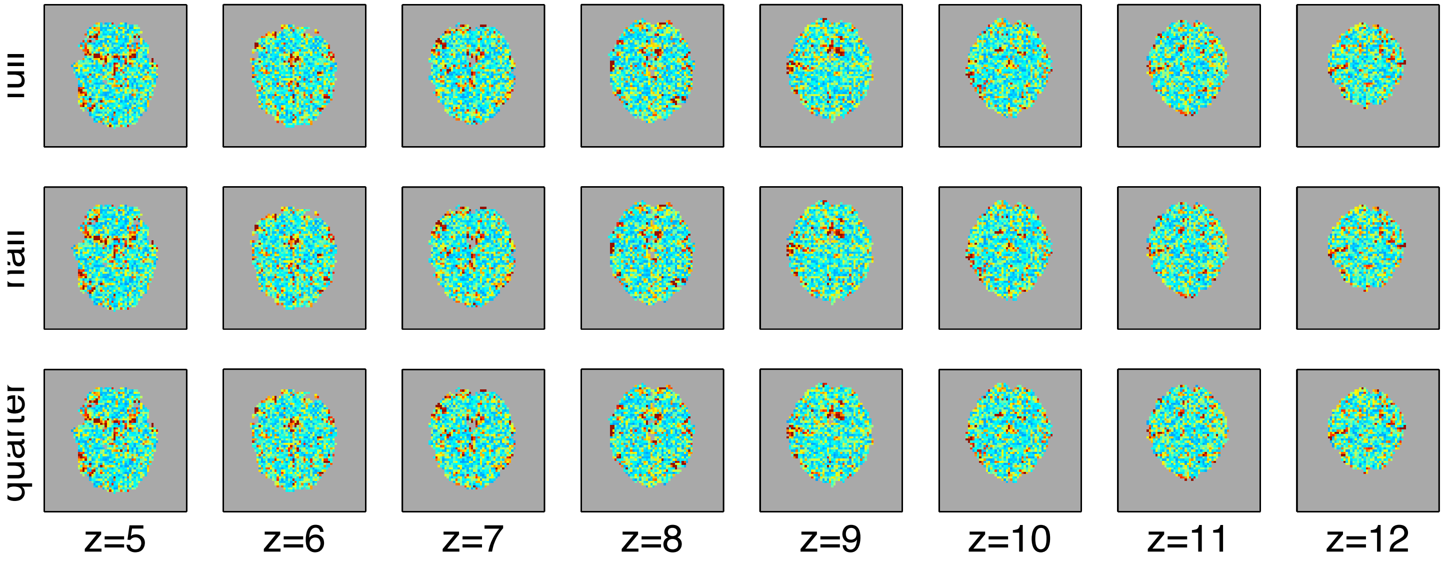
(a) 01481B (all/half/quarter) (b) 01482B (all/half/quarter) (c) 01897B (all/half/quarter)

Figure 3.18: Low-dimensional representations Z learnt for the three subjects in dataset D1, letting different numbers of components influence the learnt classifier while using 10 components (for each subject there are three columns, all, half or a quarter of the components, rounded down). The six rows correspond to the six cross-validation folds, as in Figure 3.16.



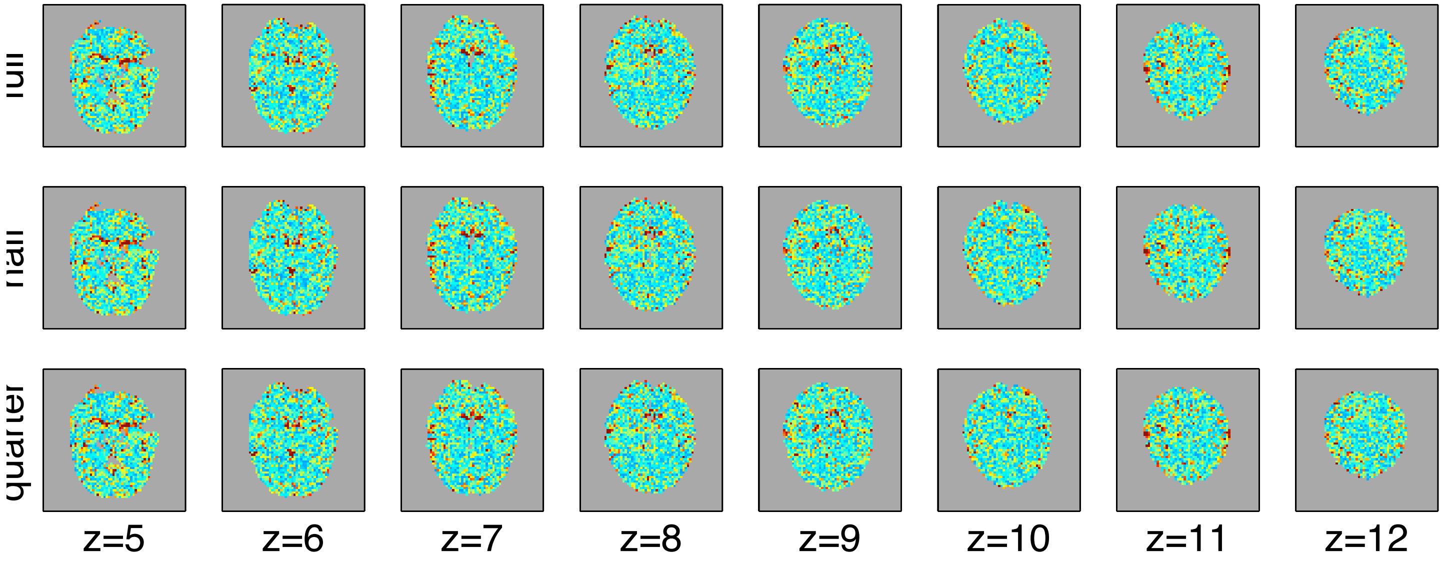
(a) 02497B (all/half/quarter) (b) 02509B (all/half/quarter) (c) 02714B (all/half/quarter)

Figure 3.19: Same as Figure 3.18, for dataset D2.



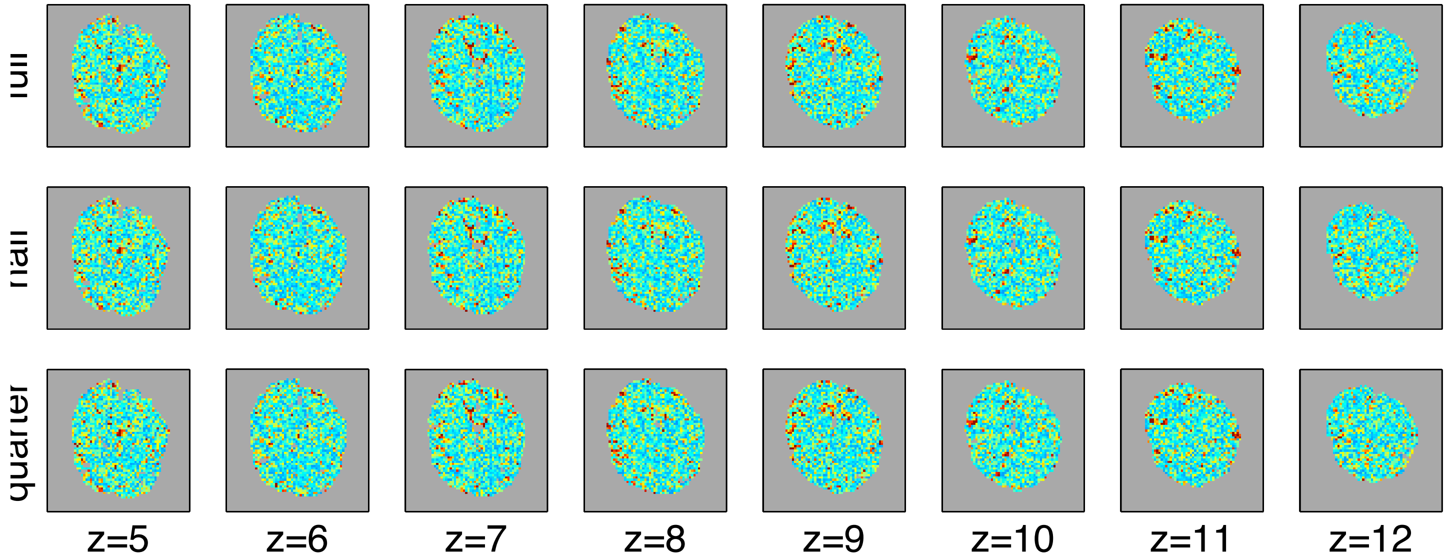
(a) 01481B (all/half/quarter)

Figure 3.20: Voxel-space discriminant for Algorithm B SVDM models for subject 01481B in dataset D1, letting different numbers of components influence the learnt classifier (using 10 components, full=10, half=5, quarter=2).



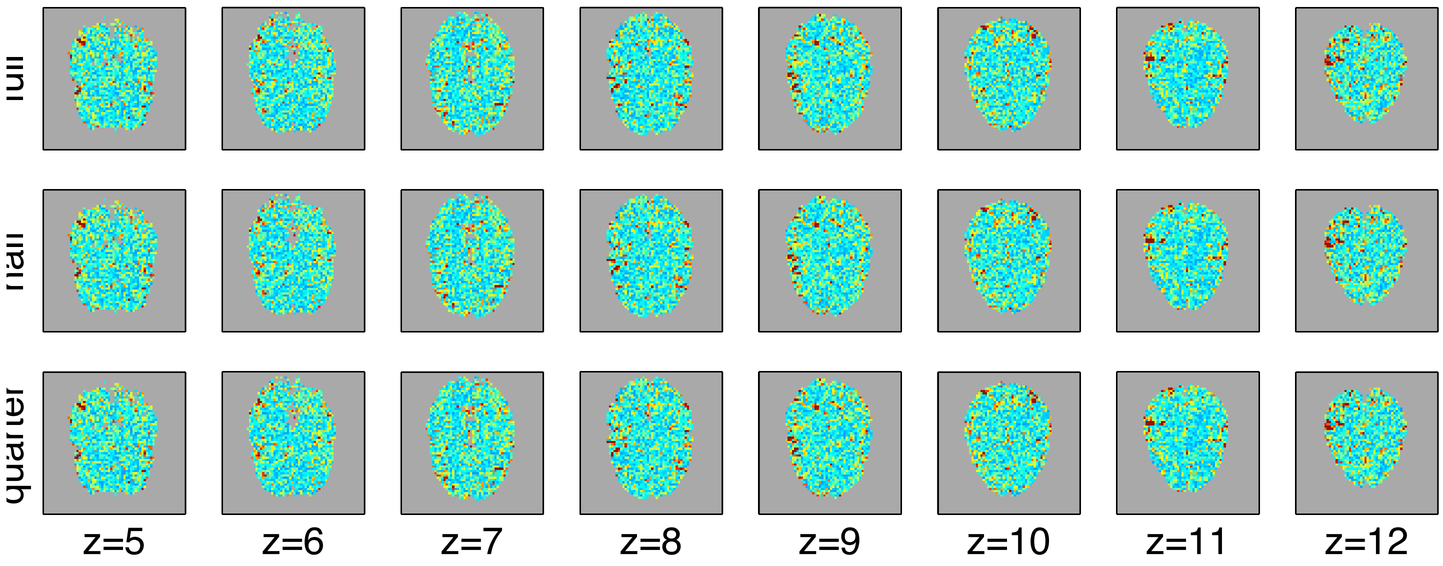
(a) 01482B (all/half/quarter)

Figure 3.21: Voxel-space discriminant for Algorithm B SVDM models for subject 01482B in dataset D1, letting different numbers of components influence the learnt classifier (using 10 components, full=10, half=5, quarter=2)



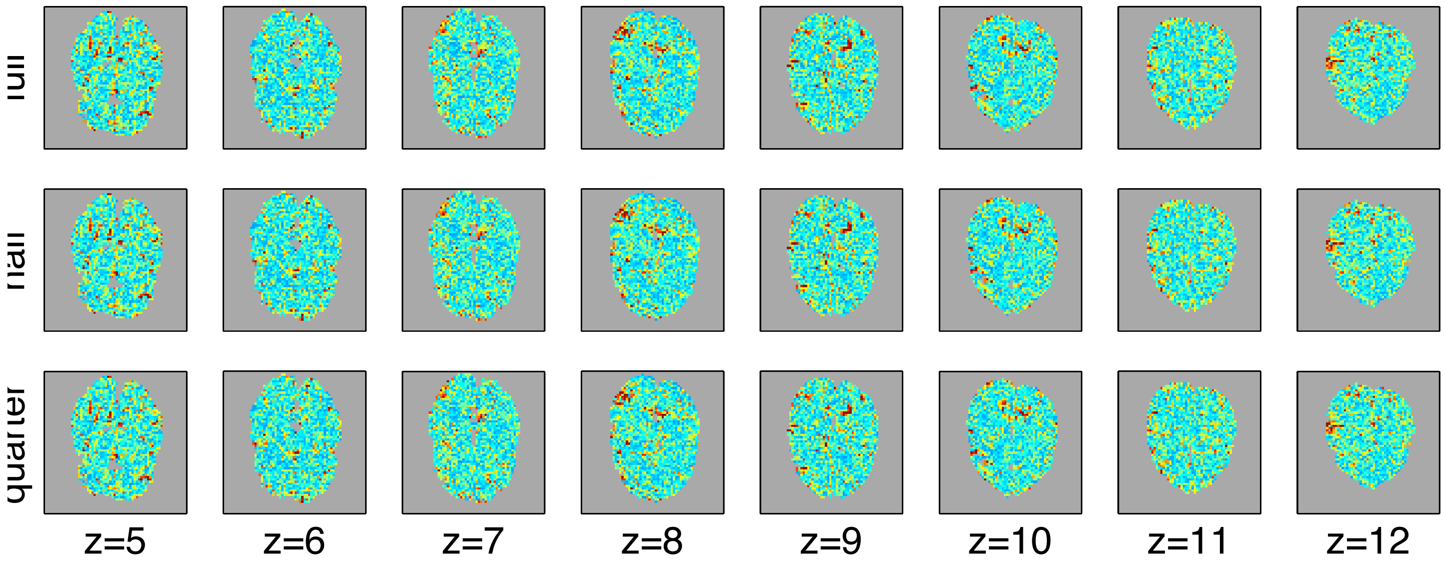
(a) 01897B (all/half/quarter)

Figure 3.22: Voxel-space discriminant for Algorithm B SVDM models for subject 01897B in dataset D1, letting different numbers of components influence the learnt classifier (using 10 components, full=10, half=5, quarter=2)



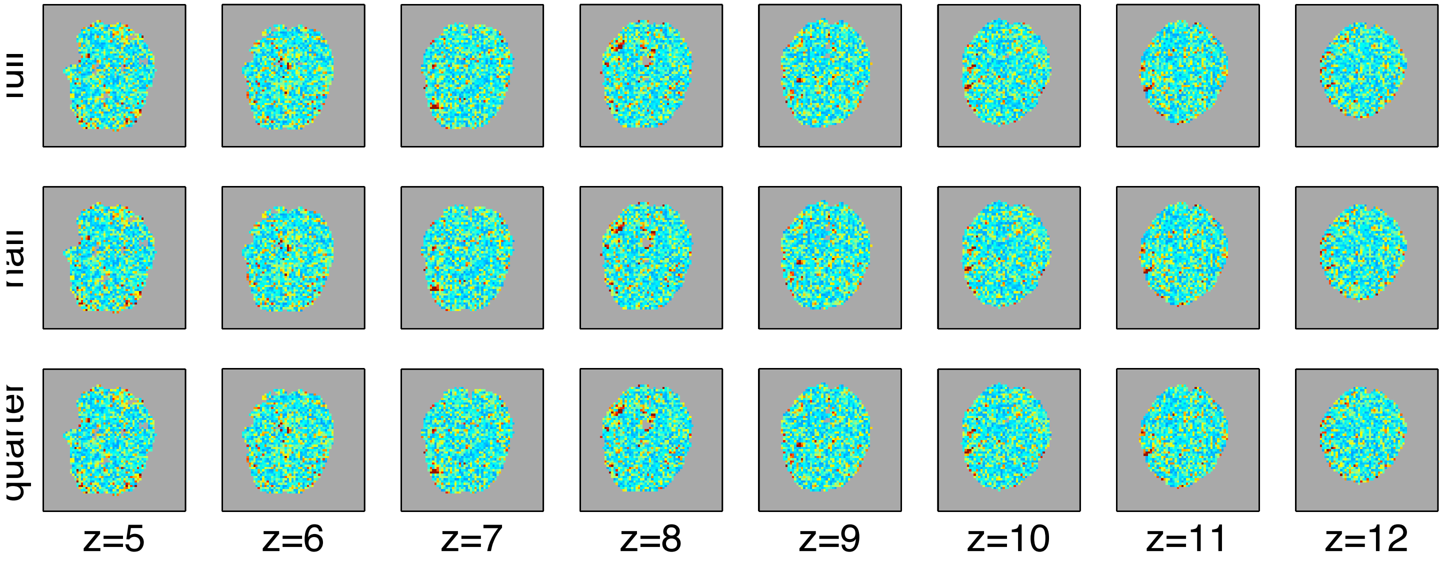
(a) 02497B (all/half/quarter)

Figure 3.23: Voxel-space discriminant for Algorithm B SVDM models for subject 02497B in dataset D2, letting different numbers of components influence the learnt classifier (using 10 components, full=10, half=5, quarter=2)



(a) 02509B (all/half/quarter)

Figure 3.24: Voxel-space discriminant for Algorithm B SVDM models for subject 02509B in dataset D2, letting different numbers of components influence the learnt classifier (using 10 components, full=10, half=5, quarter=2)



(a) 02714B (all/half/quarter)

Figure 3.25: Voxel-space discriminant for Algorithm B SVDM models for subject 02714B in dataset D2, letting different numbers of components influence the learnt classifier (using 10 components, full=10, half=5, quarter=2)

Chapter 4

SVDM extensions

In this chapter we show how the SVDM algorithm can be modified to produce solutions where very few dimensions are used in prediction, while maintaining classification and reconstruction performance at the same level of the original algorithm. Furthermore we show how the algorithm can be adapted to work with more than two classes, apply SVDM to relevant fMRI datasets and show that it compares favourably with linear SVMs in terms of classification accuracy. In addition we show that the low-dimensional representations learnt can capture relationships between classes. Finally, we introduce two different ways of extending the SVDM algorithm so that fMRI data from multiple subjects can be used. We show that it is possible to learn multisubject models that perform as well as single subject models, using either approach.

4.1 Adding regularization through constraints and penalties

One advantage of the matrix formulation in algorithm B, described in Section 3.2 of Chapter 3, is that it is conceptually easy to add constraints or penalties on any of the three matrices in the solution to the optimization problem. The most obvious of these are constraints on the components, the rows of the W matrix, which can be viewed as basis images for reconstruction. The basis images can be made sparse (few voxels active), smooth (neighbouring voxels tend to have similar values) or have activation fixed or encouraged in brain locations known to be relevant. It is also possible to constrain voxel behaviour across basis images, each voxel being a column of the W matrix. Voxels can be encouraged to be active sparsely (in as few basis images as possible) or to be active in examples belonging to a subset of experimental conditions, for instance. Constraints and penalties can also be added to the Z or Θ matrices, respectively the low-dimensional representation and the linear classifier weights. Doing it on the former would make sense if one wanted to make sure a given basis image is shared by particular experimental conditions, for instance. Doing it on the latter is similar to penalizing the weight vector in a SVM.

In general, any penalty can be added to the global optimization problem as long as it is a convex function of one of the three matrices W,Θ or Z and its coefficient is positive. If this is the case, each of the the subproblems, where any two of the three matrices are fixed, remains convex. This section illustrates the effect of several different penalities on the solution, several of which will be used in other sections.

4.1.1 Implementation of a smoothness penalty

The algorithms described in the previous chapter find W by solving one regression problem for each of its m columns (voxel across all basis images). In order to place a constraint or a penalty over an entire row of W, it is simpler to solve for entire rows of W simultaneously instead. This can be done by noting that $X \approx ZW$ can be decomposed into a sum of matrices where each matrix involves a single row of W, i.e.

$$X \approx ZW = Z_{:,1}W_{1,:} + \dots Z_{:,l}W_{l,:}$$

Given this, the problem of finding W can be decomposed into a series of least squares problems: find row $W_{i,:}$ such that, multiplying by $Z_{:,i}$, gives the best fit to the residual

$$X - (Z_{:,1}W_{1,:} \dots Z_{:,i-1}W_{i-1,:} + Z_{:,i+1}W_{i+1,:} + \dots Z_{:,l}W_{l,:})$$

in the least squares sense. This is iterated until W converges.

The objective of each of these sub-problems contains only a squared error term. It is thus easy to add terms penalizing the difference between each voxel j and its neighbours in the i^{th} row of W, $W_{i,:}$:

$$\sum_{j=1}^{m} \sum_{k=1}^{\#\texttt{neighbours}_j} (W_{ij} - W_{ik})^2$$

Adding this to the optimization problem for W in Section 3.2.2 in Chapter 3 gives us the objective ($\bar{X}$ is the dataset matrix with its mean subtracted):

$$\|\bar{X} - \bar{Z}\bar{W}\|^2_{\mathrm{Fro}} + \lambda \sum_{i=1}^{l} \sum_{j=1}^{m} \sum_{k=1}^{\#\texttt{neighbours}_j} (\bar{W}_{ij} - \bar{W}_{ik})^2$$

The results presented in the next section show the effect of introducing this penalty using $\lambda = 1$ as the setting for the tradeoff parameter. We've found the degree to which smoothness was present could vary noticeably with changes of a few orders of magnitude in λ, and in general it will have to be set to take into account the value scale of X.

4.1.2 Results

Figure 4.1 compares the accuracy of the model learnt by algorithm B incorporating the smoothness penalty – with a fixed weight of 1 on the penalty relative to the reconstruction error term in the objective – with the one produced using just algorithm B, setting $D = 0.1$, on datasets D1 and D2.

Given that the accuracies reached are similar, to see the effect of adding this penalty one needs to either look directly at the basis images learnt or at the voxel space discriminants, a linear combination of those basis images (as shown in Section 3.3 in Chapter 3). The latter are shown in Figure 4.2, Figure 4.3 and Figure 4.4 for the subjects in dataset D1 and Figure 4.5, Figure 4.6 and Figure 4.7 for the subjects in dataset D2. It is clear that the approach used leads to smoother basis images and thus to a smoother voxel space discriminant.

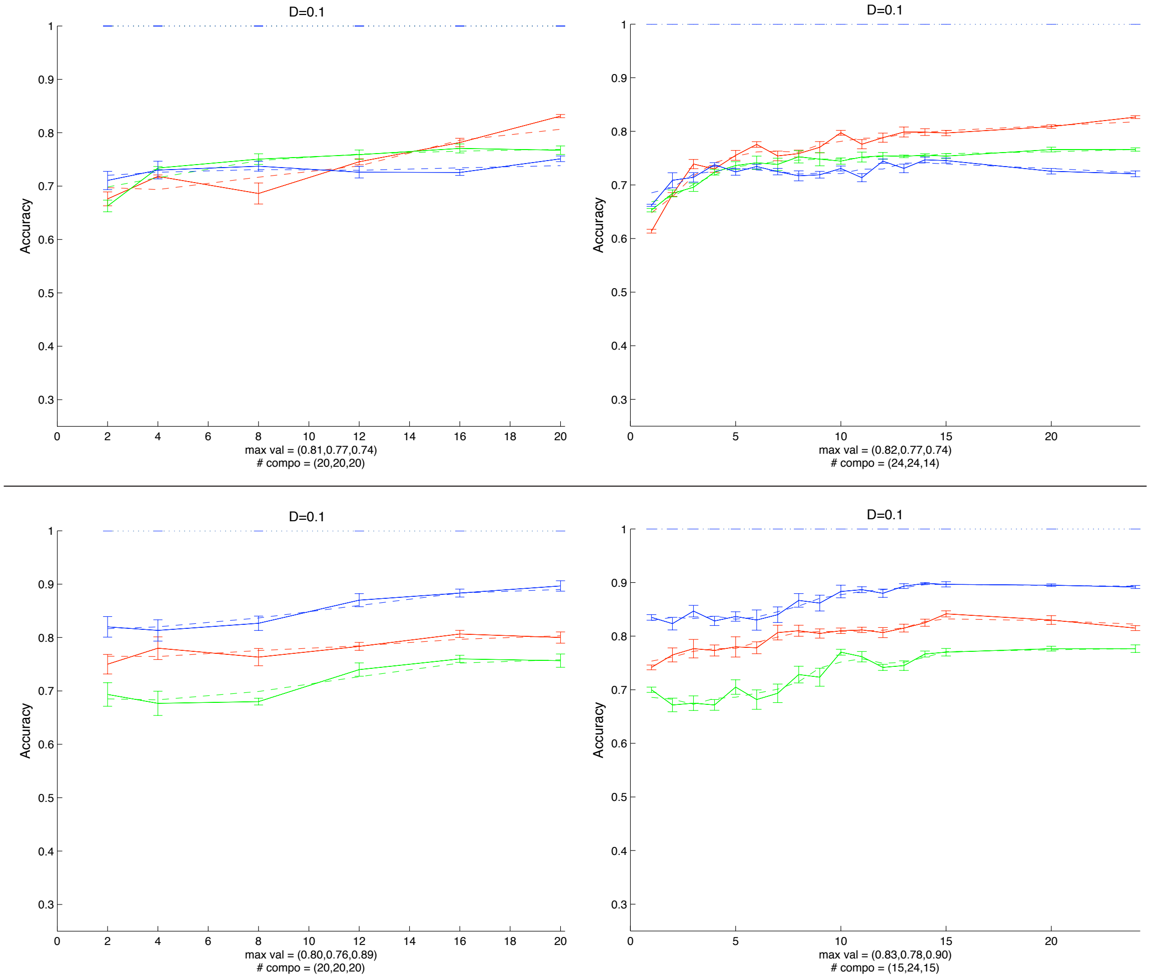


Figure 4.1: A comparison of the accuracies obtained using the smoothness penalty (left) and without using it (right), for datasets D1 (top) and D2 (bottom), using D=0.1.

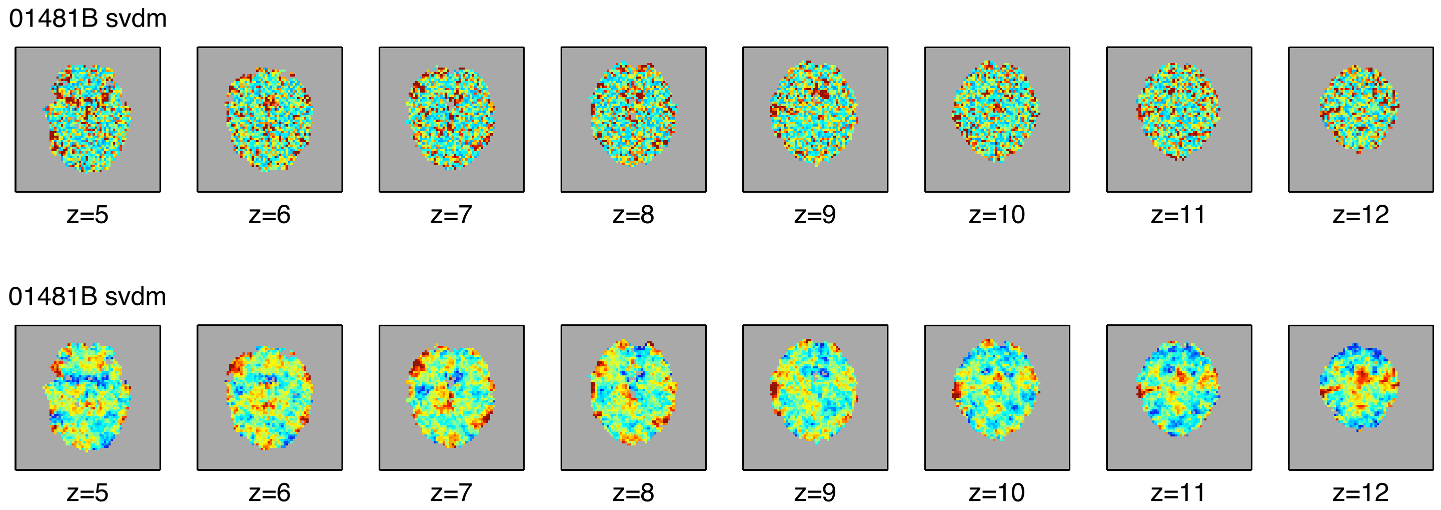


Figure 4.2: A comparison of voxel-space discriminants learnt with the original algorithm (12 components) and the original algorithm with a smoothness penalty (12 components), for subject 01481 in dataset D1

.

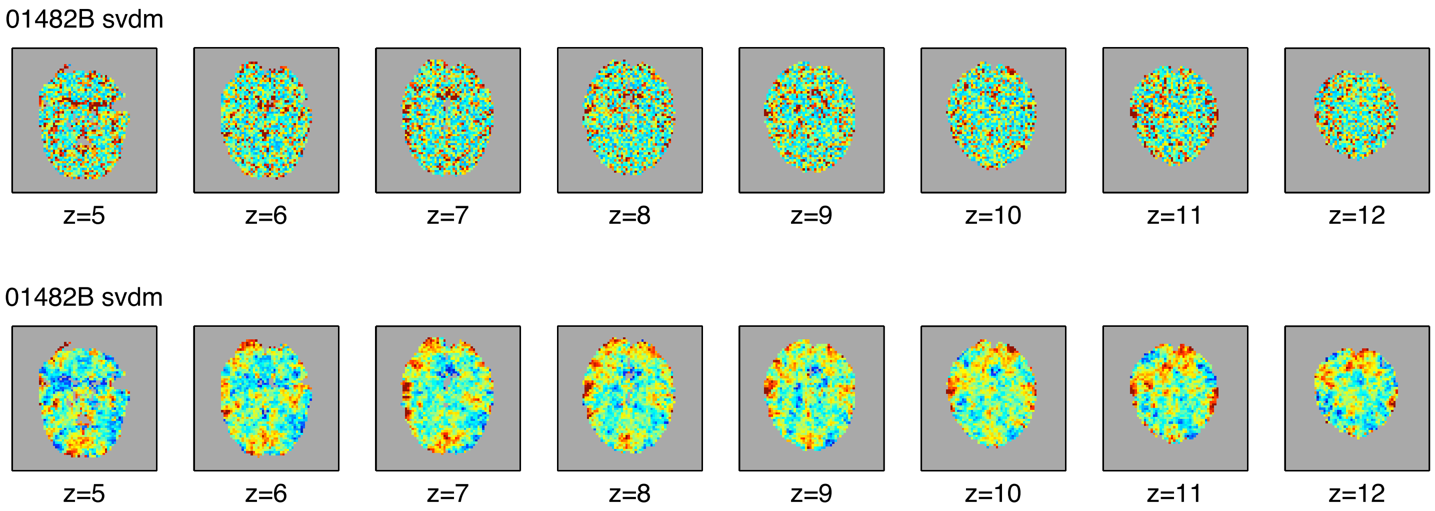


Figure 4.3: Same as Figure 4.2 for subject 01482B in dataset D1.

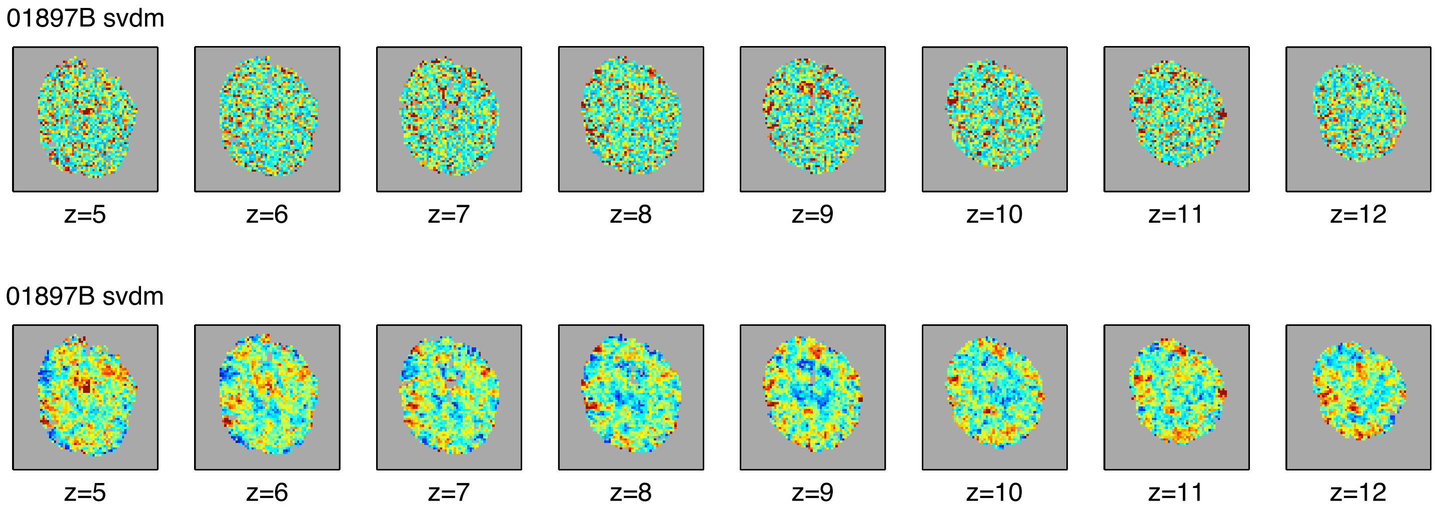


Figure 4.4: Same as Figure 4.2 for subject 01897B in dataset D1.

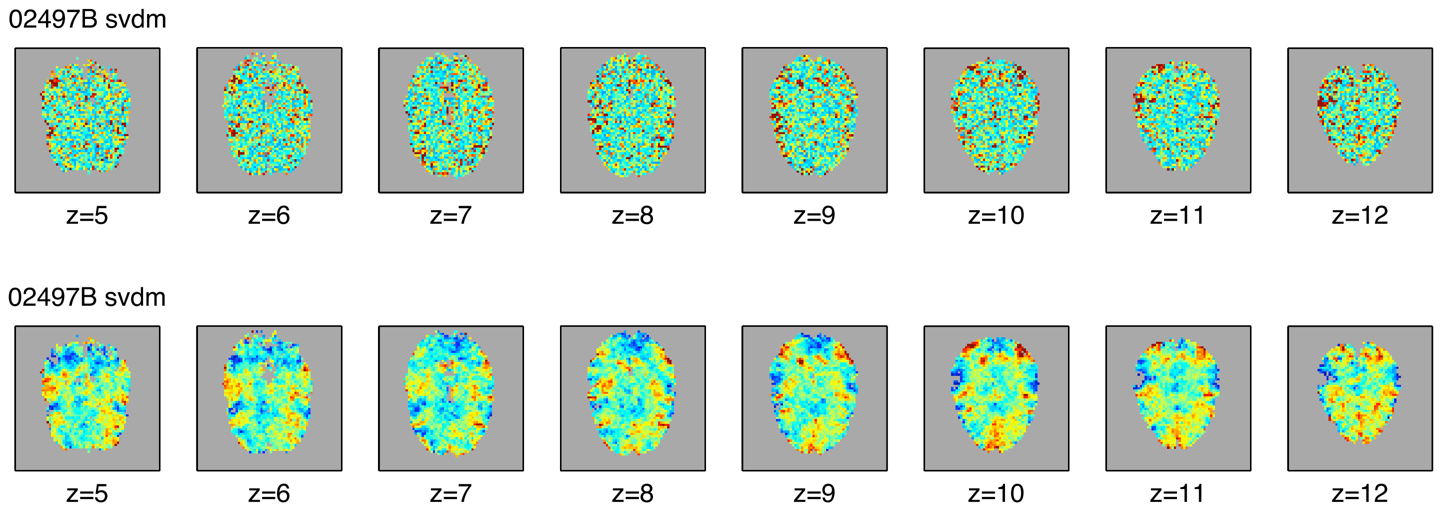


Figure 4.5: Same as Figure 4.2 for subject 02497B in dataset D2.

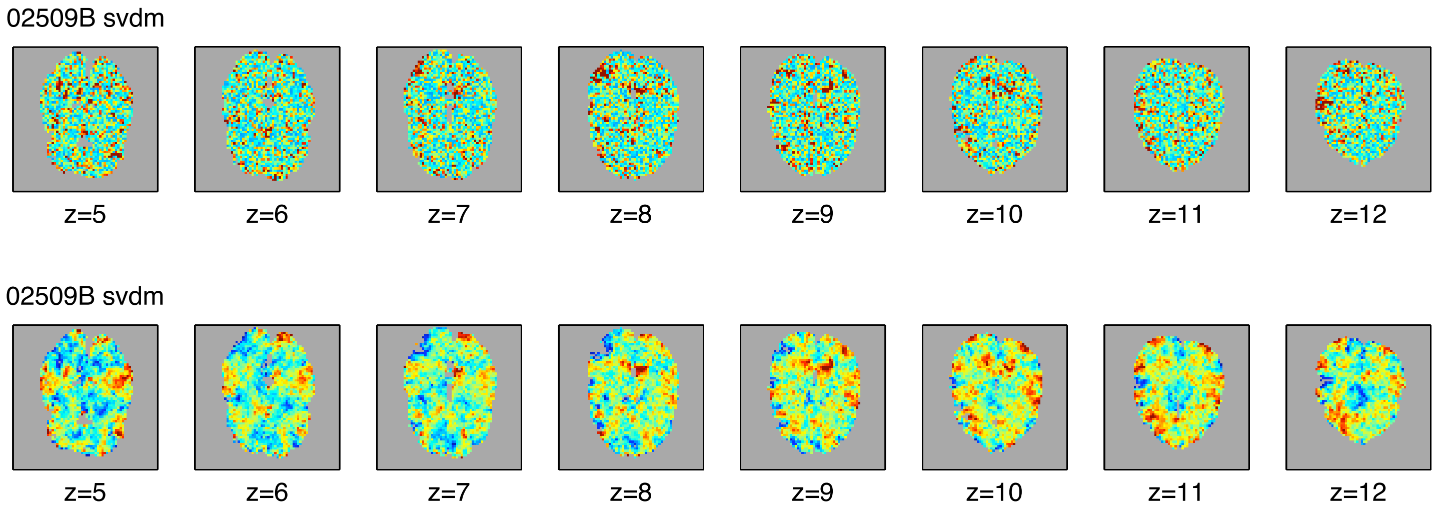


Figure 4.6: Same as Figure 4.2 for subject 02509B in dataset D2.

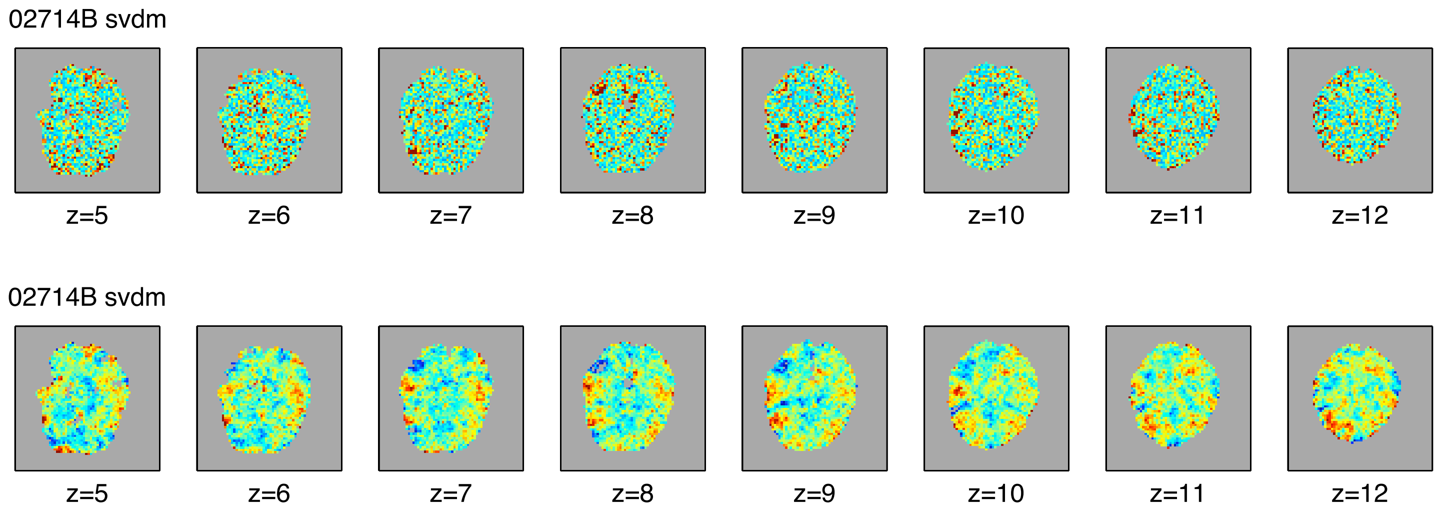


Figure 4.7: Same as Figure 4.2 for subject 02714B in dataset D2.

4.1.3 Automatically constraining the number of components used

Implementing a sparsity penalty

An original goal for the SVDM algorithm was to find informative components, i.e. which could be related to the classification label(s). The way such a relationship manifests is through the Z matrix having structure related to the labels (e.g. a column being negative for examples with one label or positive for examples with another label). The informative components are those for which the corresponding column of Z has Θ weights which are not zero.

Intuitively, the fewer columns are related to the classification problem the easier it will be to determine what the relationship is. As evidenced in Section 3.5.6 in Chapter 3, the algorithm will find many such components that are related to the target variable, even though one would suffice in the two class problems considered there. In the same Section we've seen that explicitly constraining a number of the entries of Θ to be 0 is a viable approach to ensuring that the corresponding components do not affect the learnt classifier; this makes it easier to relate the remaining basis images to the classification task. This does raise the question of how to decide how many components are allowed to affect the learnt classifier.

A more principled way of encouraging the algorithm to use a sparse Θ is to penalize having too many of entries of Θ be non-zero, rather than explicitly zeroing them. The easiest way to enforce this penalty is to alter the $\|\Theta\|_{Fro}^2$ term in the global objective, as this term appears only in the Θ subproblem. Replacing the term by $\|\Theta\|_1 = \sum_{ij} |\theta_{ij}|$ should encourage sparsity in Θ and, by making entries 0, reduce the number of columns of the low-dimensional representation Z influencing the learnt classifier.

Results

Figure 4.8 shows that the same peak accuracy can be obtained with or without the penalty encouraging sparsity. The voxel-space discriminants learnt are similar, as shown in Figure 4.9, Figure 4.10 and Figure 4.11, for dataset D1, and Figure 4.12, Figure 4.13 and Figure 4.14, for dataset D2. The difference in the learnt models is visible in Figure 4.15 and Figure 4.16, for datasets D1 and D2 respectively. Each figure is divided into three parts (top,middle,bottom) corresponding to three subjects. Inside each part, the left side shows the low-dimensional Z matrices learnt with the original algorithm in each of six cross-validation folds, and the right side the ones learnt using the penalty. Right next to each group of six matrices there is a scale where examples from one class are blue and those from the other class are red. Under each of the six matrices is a square correlation matrix showing how the columns in the matrix correlate (dark blue and red are -1 and 1 correlations, respectively). The effect of the penalty is to make most columns of the learnt Z matrix be unrelated to the example labels, while leaving one or two that are strongly related (very high or very low for all the examples). In contrast, the original algorithm will make most columns be somewhat related to the target label (visible as a colour breakdown in the learnt matrices) and, incidentally, correlated to a large extent (expected, given that this is a binary classification problem).

Given that the SVM and SVDM discriminants are so similar and that, in the limit, we could end up with a single informative component in a two-class situation, it's legitimate to ask whether the SVDM can learn a classification model that is different from a SVM. One possible way of doing this is to add regularization reflecting domain specific feature characteristics that would be harder to do directly on the SVM model. We'll also see in Section 4.2 that a different model *can* be learnt in the multiclass case where there is scope for the low-dimensional representation to capture commonalities between classes.

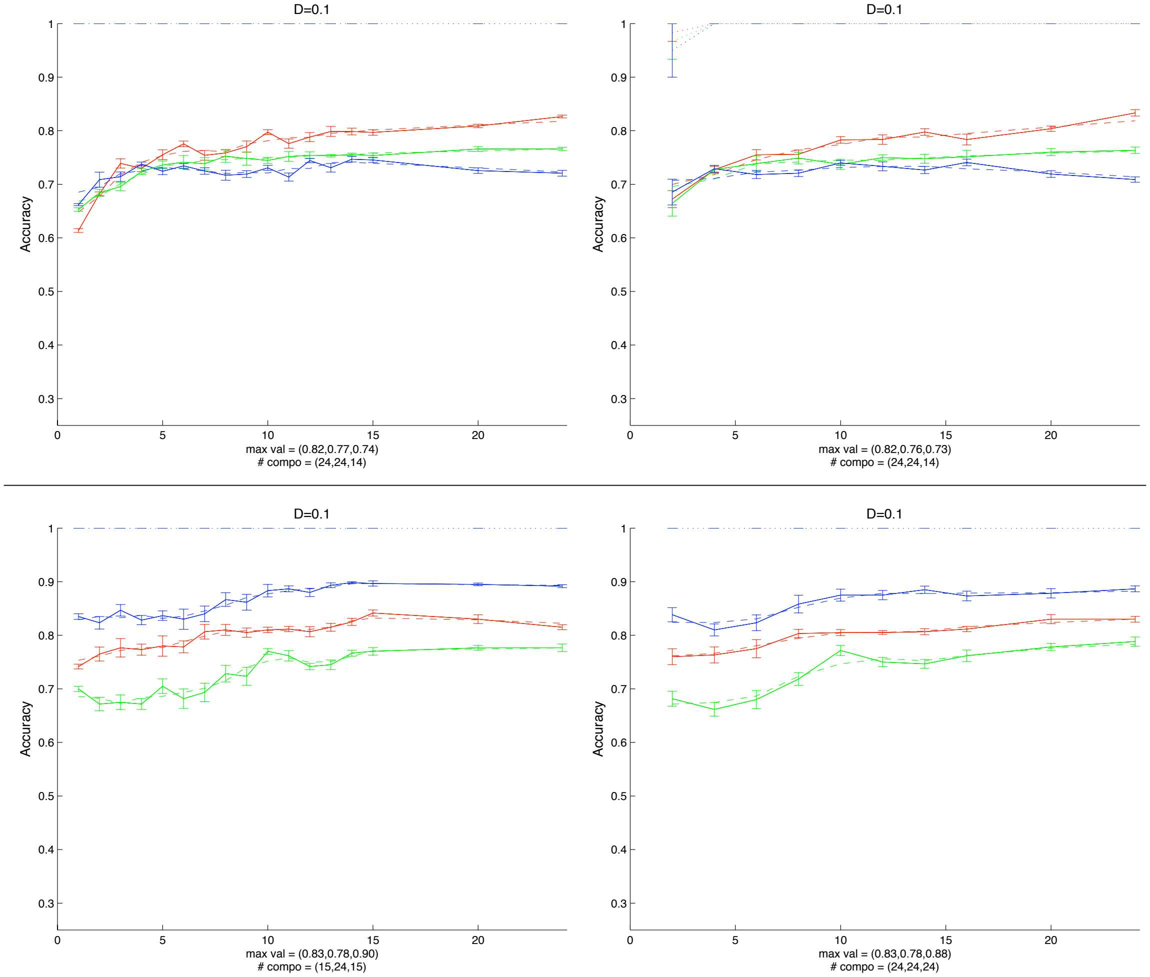


Figure 4.8: A comparison of the accuracies obtained using the penalty encouraging Θ sparsity (right) and without using it (left), for datasets D1 (top) and D2 (bottom), using D=0.1.

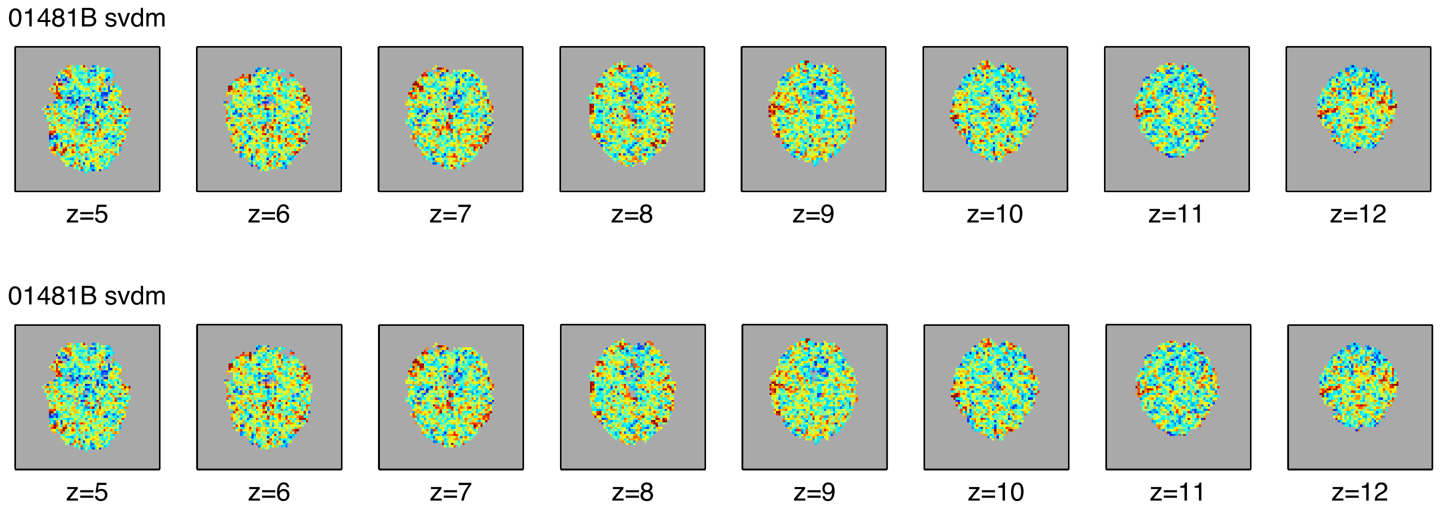


Figure 4.9: A comparison of voxel-space discriminants learnt using the original algorithm (24 components,D=0.1) and the the algorithm with a penalty encouraging Θ sparsity (24 components,D=0.1), for subject 01481B in dataset D1

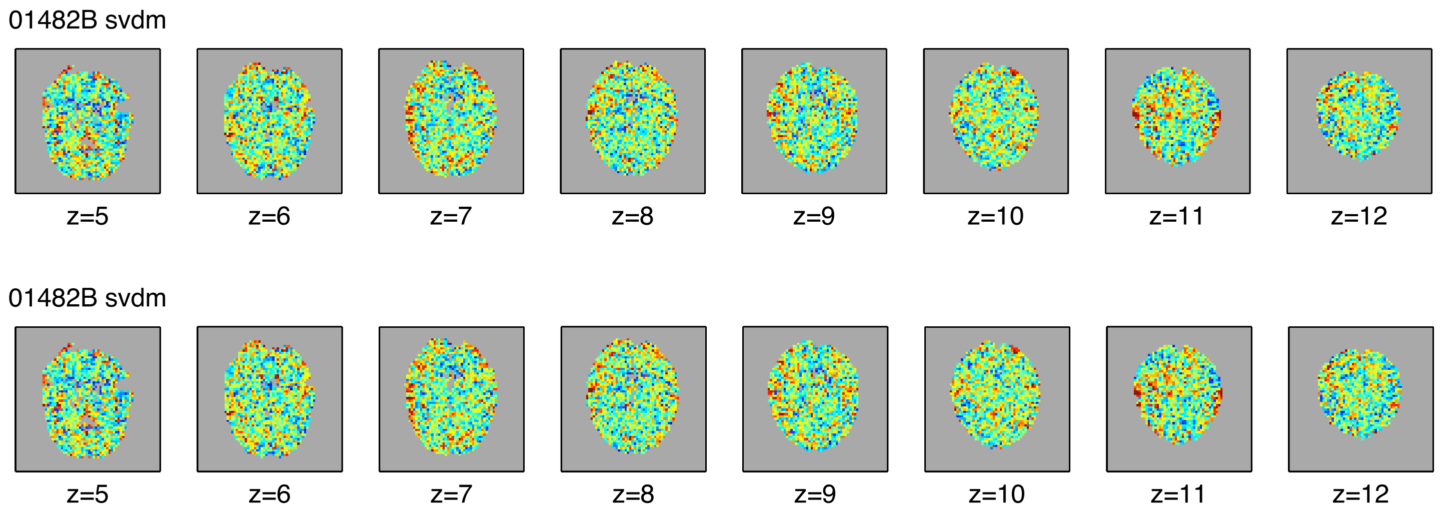


Figure 4.10: Same as Figure 4.9 for subject 01482B in dataset D1.

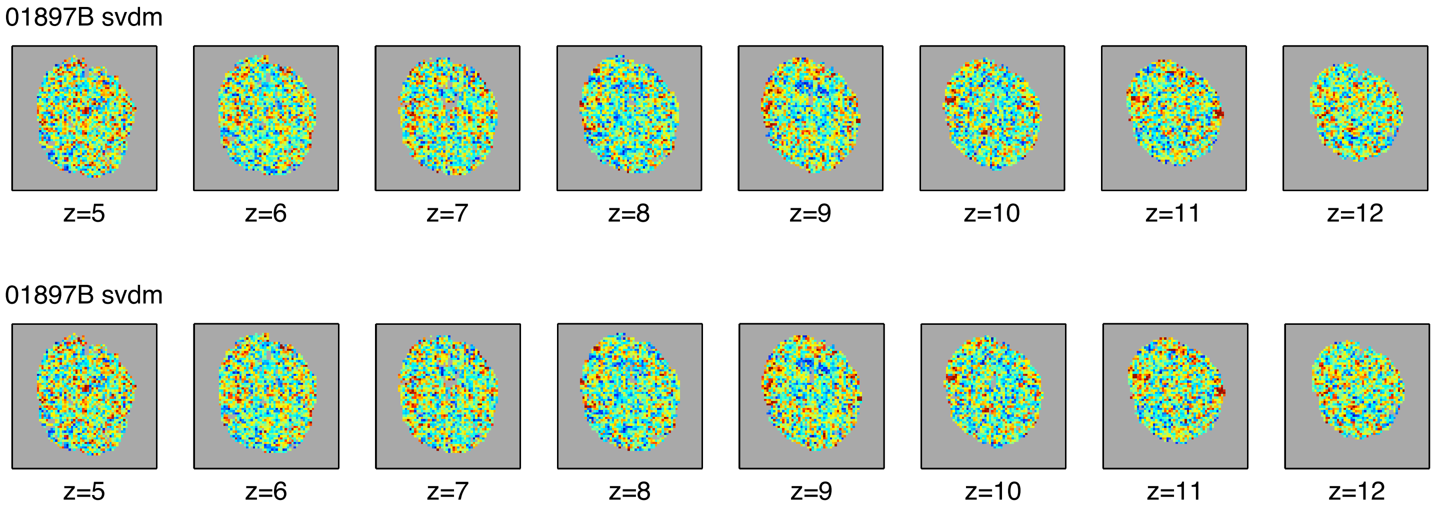


Figure 4.11: Same as Figure 4.9 for subject 01897B in dataset D1.

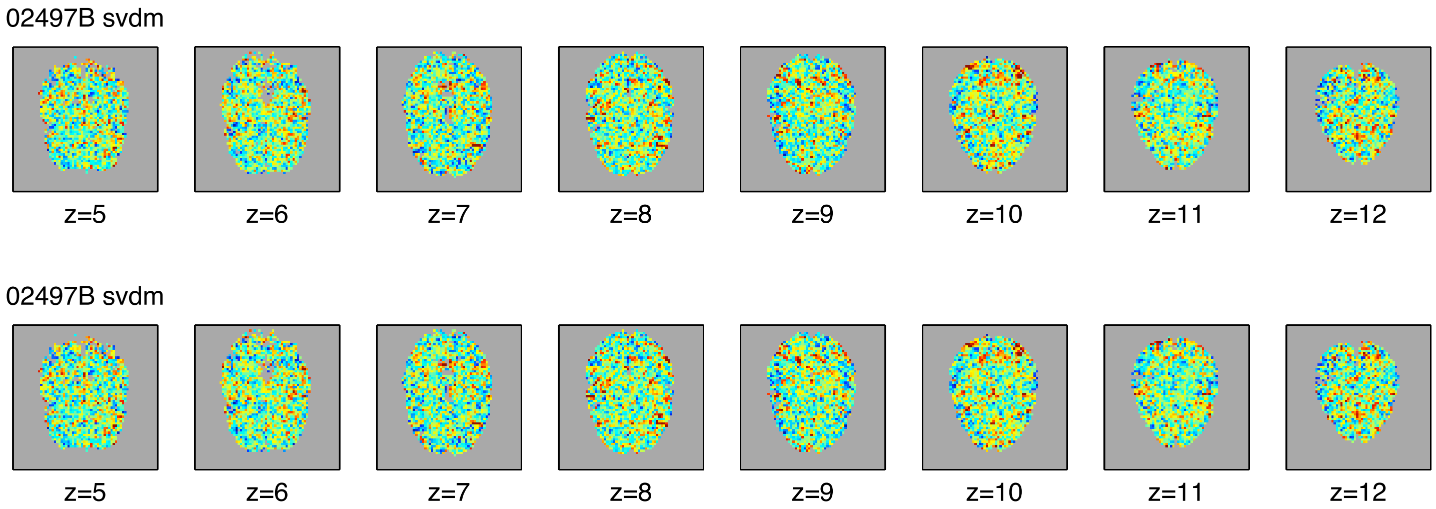


Figure 4.12: Same as Figure 4.9 for subject 02497B in dataset D2.

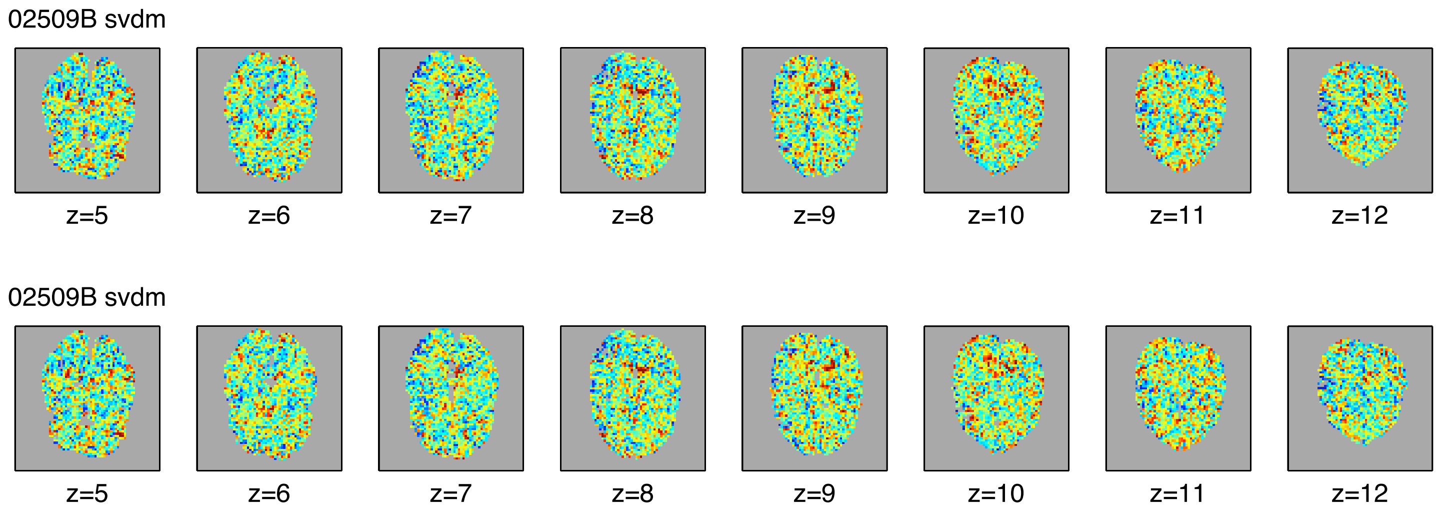


Figure 4.13: Same as Figure 4.9 for subject 02509B in dataset D2.

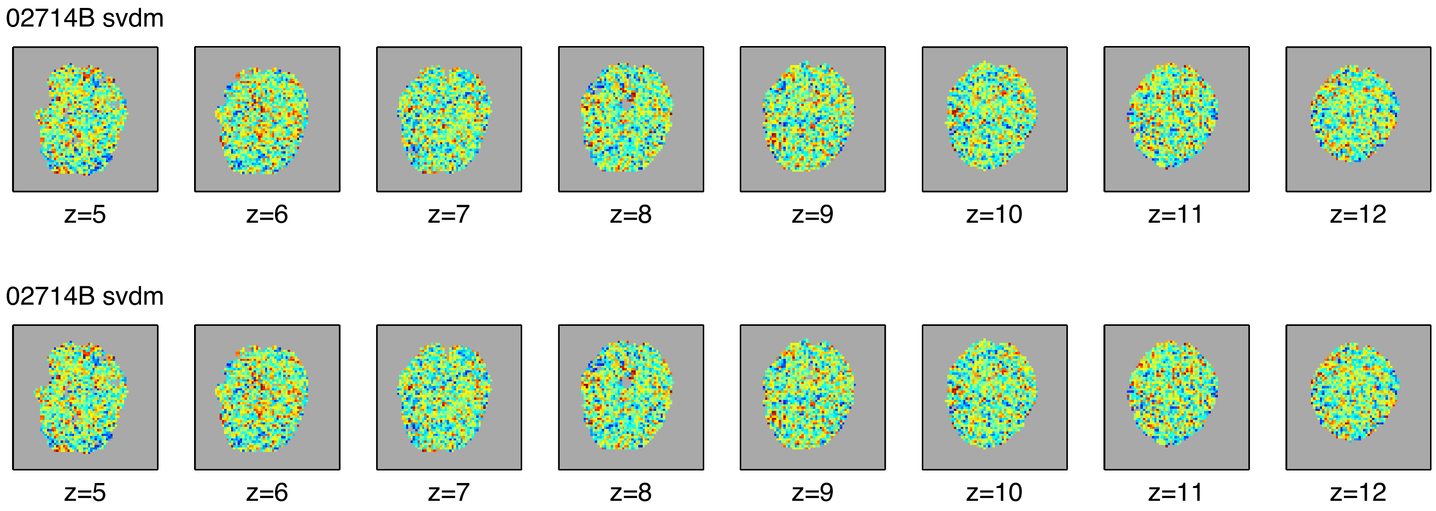


Figure 4.14: Same as Figure 4.9 for subject 02714B in dataset D2.

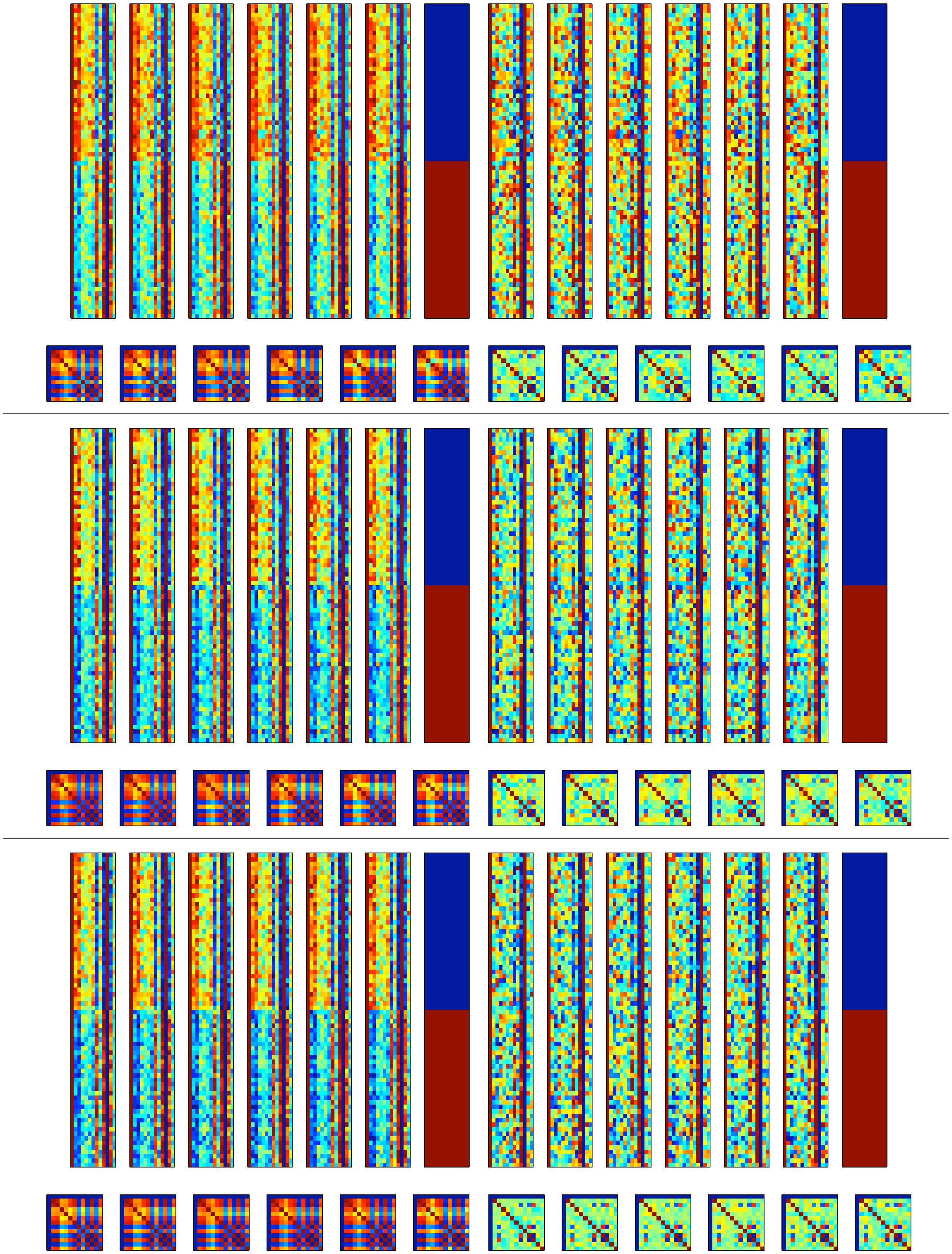


Figure 4.15: A comparison of the Z matrices learnt using the original algorithm (left, 12 components,D=1) and the algorithm with a penalty encouraging Θ sparsity (right, 12 components,D=1), for the three subjects in dataset D1 (top, middle, bottom). For each algorithm/subject combination we can see six Z matrices (one per cross-validation fold) and, underneath, six square matrices with the correlation between the columns of the corresponding Z matrix. Next to each set of six a scale shows example class (blue or red). The penalty makes most columns of the learnt Z matrices be unrelated to the example labels.

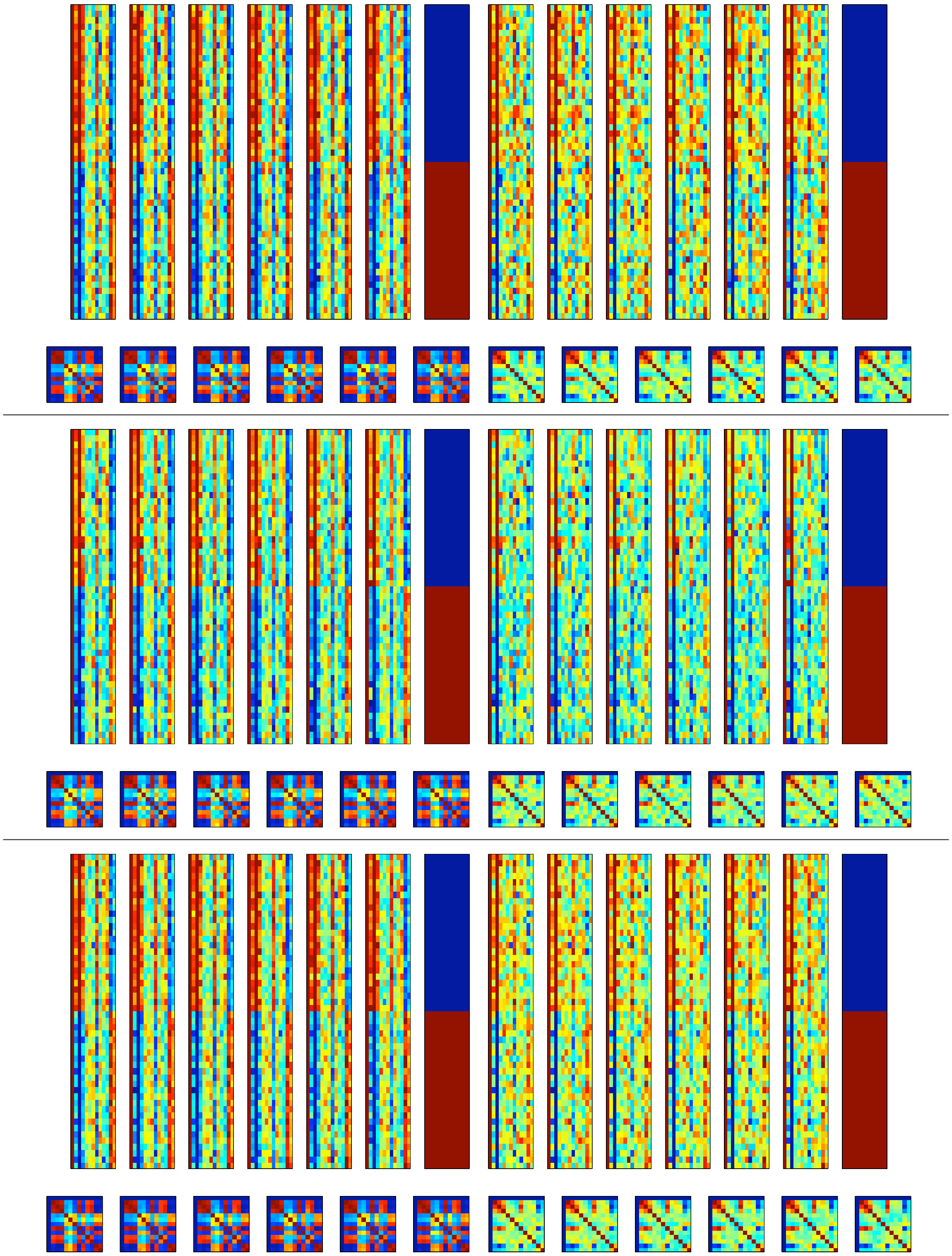


Figure 4.16: Same as Figure 4.15 for dataset D2.

4.1.4 Other extensions

Composite penalties

The two previous sections show how to encourage desirable properties in the Θ and W matrices separately. It is also possible to do both things simultaneously if, again, each of the three matrix optimization subproblems still remain convex. For certain problems, however, the only feasible approach is to divide the optimization procedure into stages.

To illustrate this, let's consider a situation where we would like to have a sparse (low L_1-norm) voxel space discriminant. One way of doing this would be to try to have the rows of W be sparse, so that the weighted combination of those rows making up the discriminant would be sparse as well; this would be similar to making the rows smooth, as described in Section 4.1.1. This seems inappropriate, however, in that most rows of W need not be sparse, as they will not affect the discriminant. Making such rows sparse is also at odds with the goal of reconstructing the data matrix well, in that the rows would be a worse basis (because most voxels would have less activation and more components would be needed than before). The question is then how to enforce the sparsity *only* for the rows of W that correspond to columns of Z involved in classification.

Our approach to doing this has two stages. The first is to solve the usual SVDM problem with the L_1 penalty encouraging sparsity on Θ, which will tell us which columns of Z are informative for each classification problem. In the second stage, Θ is held fixed and we solve for W and Z such that, for columns of Z that affect classification, the corresponding row of W is sparse. In practice this does work, in that it produces sparser rows of W when the corresponding entries in Θ are used in classification. However, it does not produce any substantial improvements in classification and thus we will not provide any additional material regarding it; we present the procedure to illustrate a way of specifying a relatively complex property for the solution of the optimization problem while maintaining the convexity of the subproblems.

Different bases

A different direction for extension would be to change the criteria for what a good reconstruction is or even away from reconstruction altogether. An example of the former would be to penalize reconstruction error with something other than squared error, which would be appropriate if we thought the data were corrupted by non-gaussian noise. For the latter, we might want to consider making the rows of W be independent (as in spatial ICA, [5]), for instance.

For the first option, we would need the penalization to still be a convex function when W and Z are variables in their respective optimization sub-problems. For the second option, one would have to treat the W problem as that of finding a basis of independent components given the mixing matrix Z. The Z subproblem would be similar to what we currently have.

4.2 Multiclass and multitask learning

4.2.1 Setting

The classification tasks thus far have all been binary. As pointed out in Section 3.1 and Section 3.2 of Chapter 3, the SVDM algorithm is designed to learn an arbitrary number of classification problems simultaneously. Those problems are represented by the k columns in the label matrix Y, and predicting label j of example i correctly entails predicting entry (i, j) in the label matrix correctly.

The following are some possible circumstances where it would be necessary to have multiple columns in the label matrix Y:

1. multiple classes (e.g. predict which of 8 categories a subject is thinking about)
2. multiple problems (e.g. predict which of 2 categories a subject is thinking about, predict whether the stimulus is a word or a picture)
3. hierarchical (e.g. predict which of 2 categories a subject is thinking about, predict which exemplar in that category)

The first situation is the usual kind of multiple class problem; the others are multitask learning situations, where one expects that learning how to make one of several predictions will help one do better in others than one would learning them by themselves. Regardless of the situation, the first goal is to be able to classify correctly in a multiclass situation and the second is to use the low-dimensional representation learnt by SVDM to identify the relationship between the different classes.

4.2.2 Algorithm

The algorithm used to learn the SVDM is the one described in Section 3.2 of Chapter 3, Given that the algorithm can only deal with binary classification problems, one per column of the Y matrix, it is necessary to encode a multiclass problem as a series of binary problems. We considered two ways of doing this: converting to a set of one-class-versus-all-others problems or using an error-correcting output code (ECOC, [14]). It's possible that, in using either method to convert to binary problems, there is an imbalance between -1 and +1 labels in some of the columns of Y. If that happens, the hinge loss term $h(\rho_{ij})$ for examples with the label in the majority can prevail and the model may sometimes opt to predict that label for all examples.

To address this issue, we weigh every hinge loss term with a constant weight that reflects the ratio of labels. For instance, imagine that there are 3 -1 labels for each $+1$, which could happen if encoding a 4 class problem as 4 one-class-versus-rest problems. Then, the hinge loss terms in each column for $+1$ examples will receive weight 1 and the those for -1 examples will have weight $\frac{1}{3}$. Empirically, across various datasets and tasks, we've found that this had the desired effect in preventing default classification with the majority label.

The optimization objective then becomes

$$\frac{1}{k_X}\|X - ZW\|^2_{\text{Fro}} + \frac{1}{k_H}\sum_{i=1:n,j=1:k} a_{ij}h(\rho_{ij}) + d(\frac{1}{k_Z}\|Z\|^2_{Fro} + \frac{1}{k_\Theta}\|\Theta\|^2_{Fro})$$

where $\rho_{ij} = y_{ij}Z_{i,:}\Theta_{:,j}$ and a_{ij} is the weight described above.

4.2.3 Experiments

Datasets

The following sections present results obtained using two multiclass datasets, M1 and M2. The datasets were chosen for the fact that a classifer using all the voxels can still perform much better than random and reasonably well when compared to a classifier using voxel selection achieving peak performance. Furthermore, and given that this is currently work in progress, we considered only the best subject from each of the studies for the purposes of comparison with SVM and examining the model learnt.

The source studies use picture stimuli, with each stimulus belonging to one of several semantic categories. Dataset M2 is the same as dataset D3 in Chapter 2. It contains all the voxels in the subject's cortex, and each example is the average image in a block where the subject sees stimuli belonging to the same category.

Dataset M1 does not contain the whole cortex but only a subset of voxels selected with a localizer scan (separate dataset, used to select voxels shown to behave differently in data from the four classes pooled together versus a baseline of scrambled object stimuli). Stimuli are shown in a fast event related paradigm and examples created from an entire run by using the coefficients of a regression of each voxel on regressors containing predicted time courses for responses to each condition. Table 4.1 summarizes this information. The datasets were kindly provided by Nancy Kanwisher and Leila Reddy (M1,unpublished) and by James Haxby (M2, same as D3,[26]).

Learnt Models

The SVDM models trained all used the penalty encouraging sparsity on Θ (described on Section 4.1.3), in order to narrow the number of informative components. The SVMs trained on all voxels for accuracy comparisons had their parameter set using cross-validation over the training data. In all cases the classification task is to predict the category of the stimulus corresponding to an example, using a one-class-versus-rest binary problem encoding.

dataset	# classes	# examples	#voxels	stimulus categories
M1	4	40	$\sim 4K$	faces, objects, scenes, body parts
M2	8	96	$\sim 24K$	faces, houses, cats, scissors shoes, chairs, bottles, scrambled

dataset	design	example source
M1	fast event related	1 example per category, computed from multiple items, ×10 runs
M2	block	1 example per category block, one block of each category, ×10 runs

Table 4.1: Characteristics of the two multiclass datasets used.

Table 4.2 shows the results obtained, averaging the performance of the algorithm starting with five different seeds in the case of SVDM. SVDM can perform better than SVM and the pattern holds for other subjects in the case of dataset $M1$ (not shown here). It's also possible to obtain slightly better results - a few $\%$ higher in accuracy, in the limited experiments we ran - by using an error correcting output code; this happens at the cost of having more trouble in relating informative components to labels and vastly increased computation time, given that the procedure requires generating tens of learning problems rather than just as many as there are classes. For that reason we will not pursue that avenue further in this chapter and will instead examine models learnt with a one-versus-rest encoding.

study	SVM (all voxels)	SVDM (5-seed avg)	chance	with voxel selection
M1	0.60	0.74 (#components=20,$D = 1$)	0.25	high 70s to low 80s
M2	0.57	0.65 (#components=40,$D = 1$)	0.125	70s

Table 4.2: Results of training a one-class-versus-rest SVDM and a linear SVM on three different datasets.

Figure 4.17 shows the Θ and Z matrices learnt on dataset M1, broken down in a particular way. First, note that there are as many columns as there are cross-validation folds, with each column containing the matrices learnt from the training data in the corresponding fold.

Figure 4.18 focuses on a single column (the 10^{th}), to make it easier to describe the figure. From the top, there are 4 classification problems (each class versus all the others, with the 4 classes being faces, scenes, objects and people). Following that, we have the informative portion of the Θ matrix (the rest is 0 and not shown, since we are using the L_1 SVDM described earlier). Θ is transposed so that row i of each matrix contains the coefficients used in the classifier for problem i (class i versus the others). Each colum corresponds to a column in the informative part of Z. Below Θ we have the informative part of the Z matrix (columns that have corresponding non 0 Θ coefficients) and below that the uninformative part (used solely for reconstruction), respectively.

Note how some of the rows of Θ have multiple non zero entries (indicating that more than one component contains information about that classification problem) and some of the columns also have multiple non zero entries (indicating that a component is shared across classification problems). The latter may indicate an area with opposite patterns of activations for scenes and people. Similarly, there is another area with opposite activation between scenes and objects. To examine this, a domain expert would have to consult the third basis image in Figure 4.19, which shows the basis images (rows of W) corresponding to the informative dimensions in the Z matrix. We were not able to obtain structural images and do this in time for the completion of this dissertation, but intend to pursue this direction in the future.

Figure 4.20 shows a comparison of the average cross-validation fold discriminants learnt by SVDM and a linear SVM for each of the four one-vs-rest problems. Each pair of rows contains the SVDM and SVM voxel-space discriminants for one problem, with columns corresponding to brain slices. The discriminants are very similar at a large scale (as seen in a slice by slice correlation between the two discriminants in Figure 4.21), so whatever difference there is that is responsible for the difference in performance between the two methods lies in small pattern details. This is visible in Figure 4.19, where the rows of the matrix W corresponding to the informative components are plotted. The components place small, localized portions of activation that get added into the discriminants in Figure 4.20.

Figure 4.22 shows the informative portions of Θ and Z for dataset M2. In this dataset it's almost always the case that each classification problem resorts to a single component (only one positive entry per row) and each component serves a single problem. Figure 4.23 shows the average cross-validation fold discriminants learnt by SVDM and a linear SVM for each of the eight one-vs-rest problems. Looking at Figure 4.21 we see that, unlike in the previous dataset, the discriminants produced with the two methods are not too similar. SVM seems to be learning sparser discriminants, with SVDM placing weights in the same locations as SVM and others as well. Moreover, there is no component relating two classes (in almost all folds) and, if a classifier puts weight in more than one component, it is for one of the object categories. This is especially puzzling given that the datasets share some of the same classes.

Possible reasons for this start with details of how the study for each dataset was designed. M1 has smaller voxels ($2 \times 2 \times 2$mm) and three of the four classes have known locations that respond very strongly and selectively to their stimuli. It's thus likely that SVDM will find voxels that can be assigned to a single component. M2 has larger voxels ($3 \times 3.5 \times 3$mm) and a few categories (objects) where the responses are less selective, which might justify the fact that those correspond to the classification problems for which we see components being combined into a decision. Given that faces and houses are present in both datasets, though, it's still necessary to explain why the model is so qualitatively different for those categories.

One possibility is that there are enough components in M1 to reconstruct the data and still leave a few out to capture very localized responses; this is both because there are fewer categories but also because the total number of voxels is ≈ 4000. In M2, there are both many more voxels (≈ 24000) and more categories, but only twice as many components. The conjecture that can be tested is that learning models with many more components will lead to those models being more similar to those in M1; again, doing this was not feasible in time to complete this dissertation, but will be an area of further work.

Multitask learning

We also carried out preliminary experiments in multitask learning, in two variants of the tool/building category experiments described in Chapter 2. In the first variant the stimuli are words and come in one of two different languages, portuguese or english. In the second variant the stimuli come in two modalities: either words or pictures, with matched items. The multitask problems considered were predicting category+language or category+modality, with the hypothesized result being that learning to predict both tasks would make predictions of category in particular more accurate than learning to predict category alone. The language or modality problems were very easy by themselves (accuracy close to 100%) and thus not very suitable to gauge improvements brought about by multitask learning. In practice, there was no visible advantage to training a multitask model in this case, both in a leave-1-fold-out cross-validation and in a more data-restricted even-odd split-half cross-validation, in any of the few subjects tried from either study.

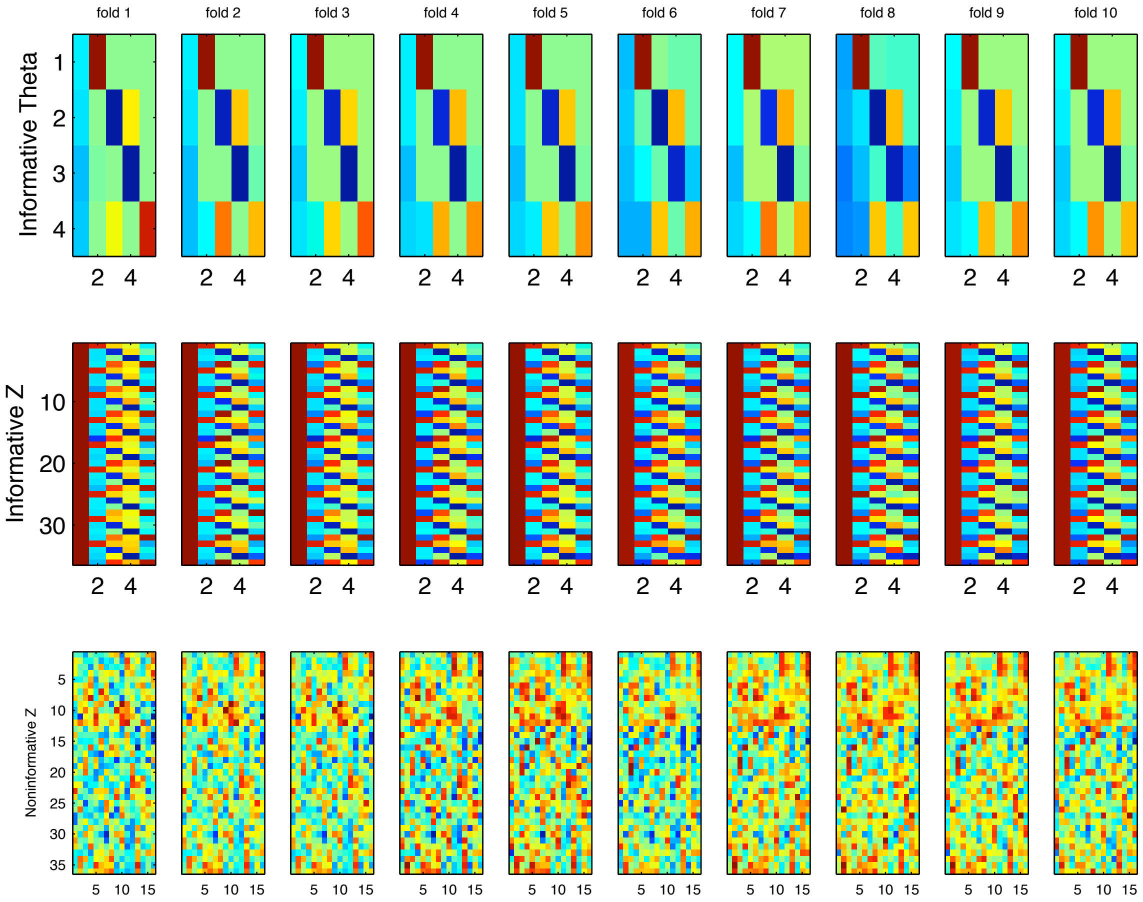


Figure 4.17: Z and Θ matrices learnt across folds (columns) in dataset M1 (please consider a single column for the rest of the caption). **Top:** The informative (non-zero) portion of Θ, with row i corresponding to classification problem i and containing the weights over informative components in the discriminant for that class (row 1 is faces, 2 is scenes, 3 is objects, 4 is people, e.g. for faces-vs-nonfaces, weight is only placed on column 2 of Z, which has a different value for examples labelled faces and all others). **Middle:** The corresponding informative portion of the low-dimensional representation Z, with as many columns as the information portion of Θ, and as many rows as examples. **Bottom:** The remaining non-informative portion of Z, with many additional columns and as many rows as examples.

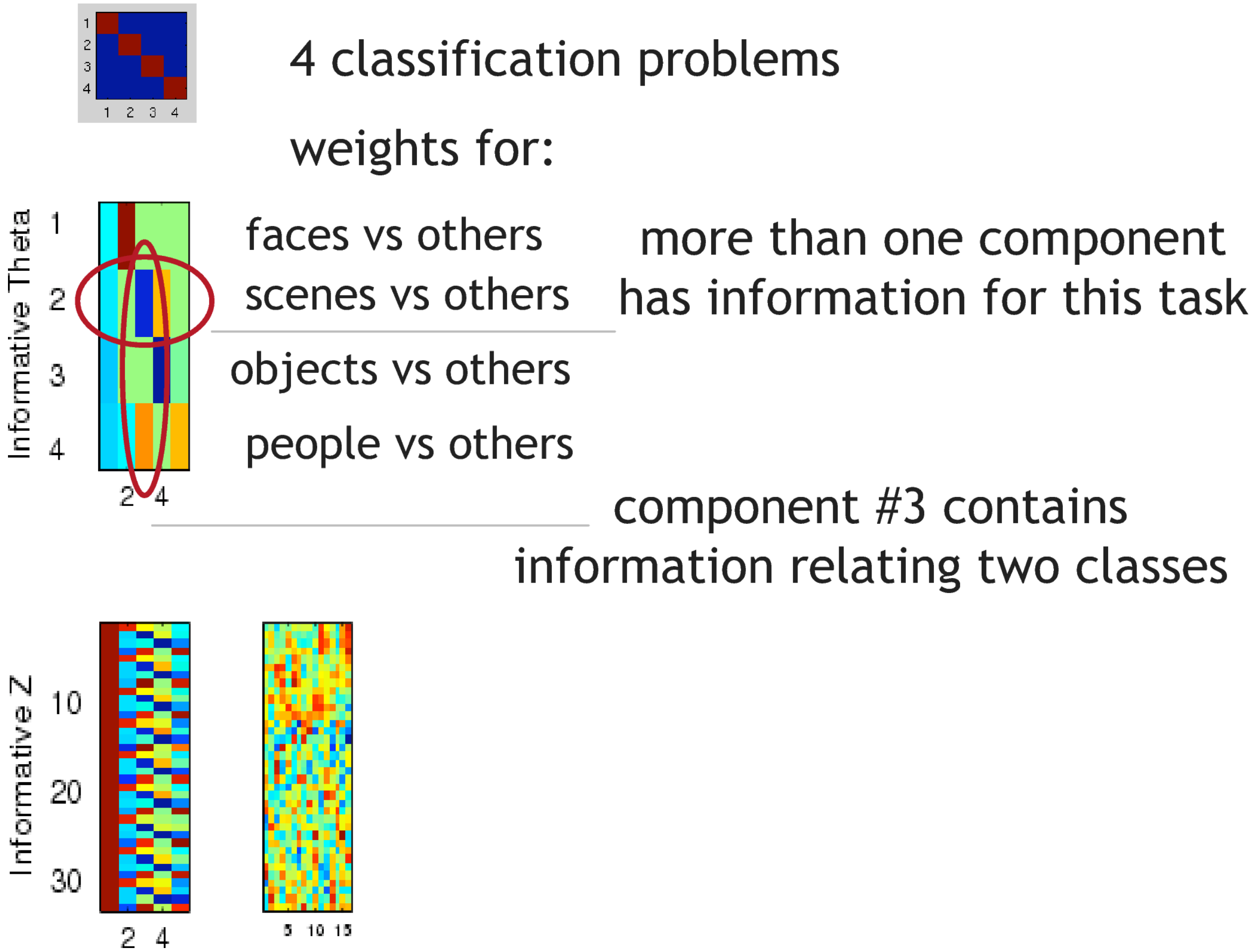


Figure 4.18: Detailed view of the Θ matrix and informative and non-informative parts of the Z matrix learnt in the 10^{th} fold in Figure 4.17. The informative Θ matrix columns are aligned with the informative ones in the Z matrix.

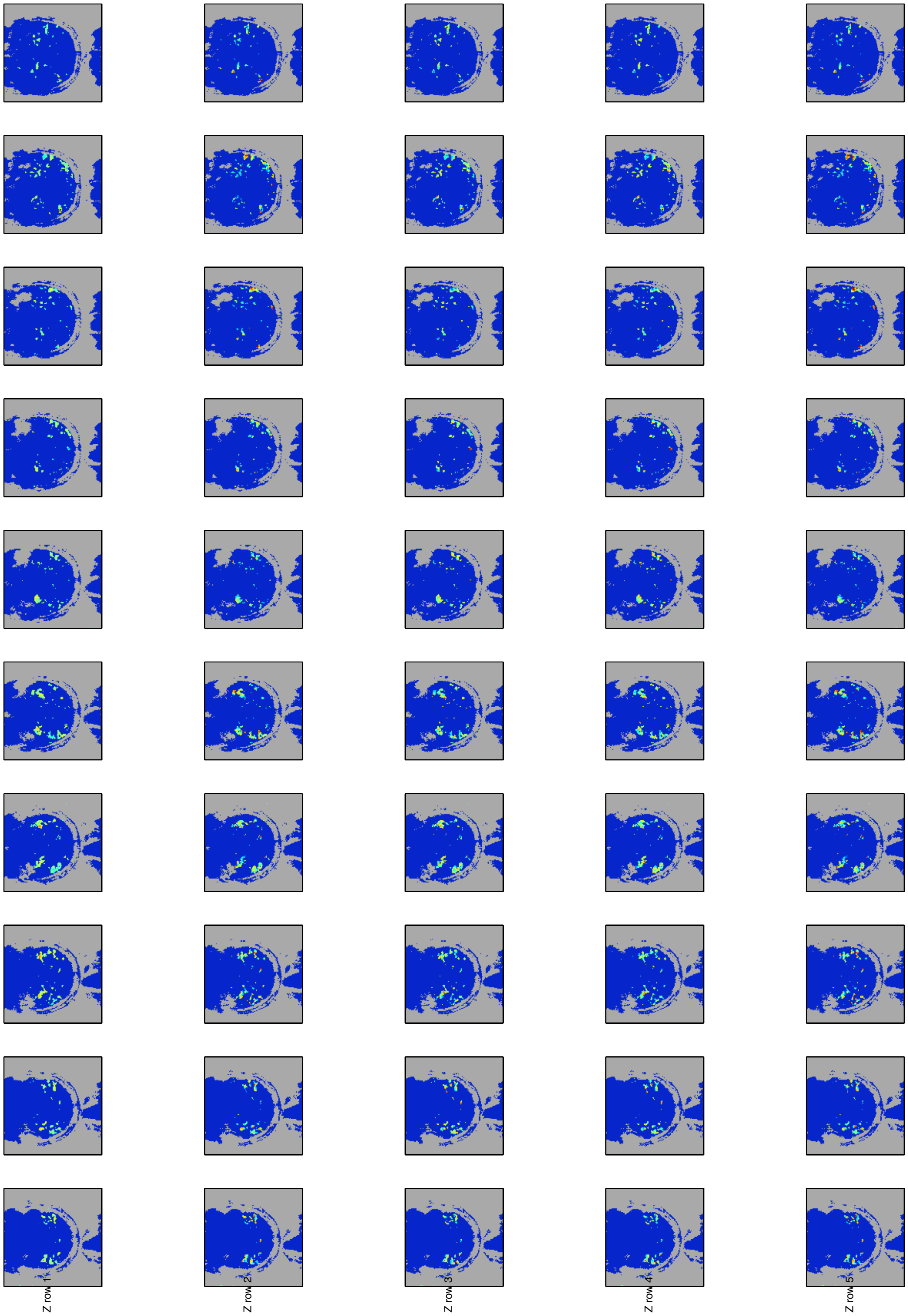


Figure 4.19: **View sideways:** The rows of the W matrix corresponding to informative columns of the Z matrix in the fold of dataset M1 (as shown in Figure 4.18). The full brain is shown in dark blue for reference, lighter voxels are the $\sim 4K$ given to the algorithm.

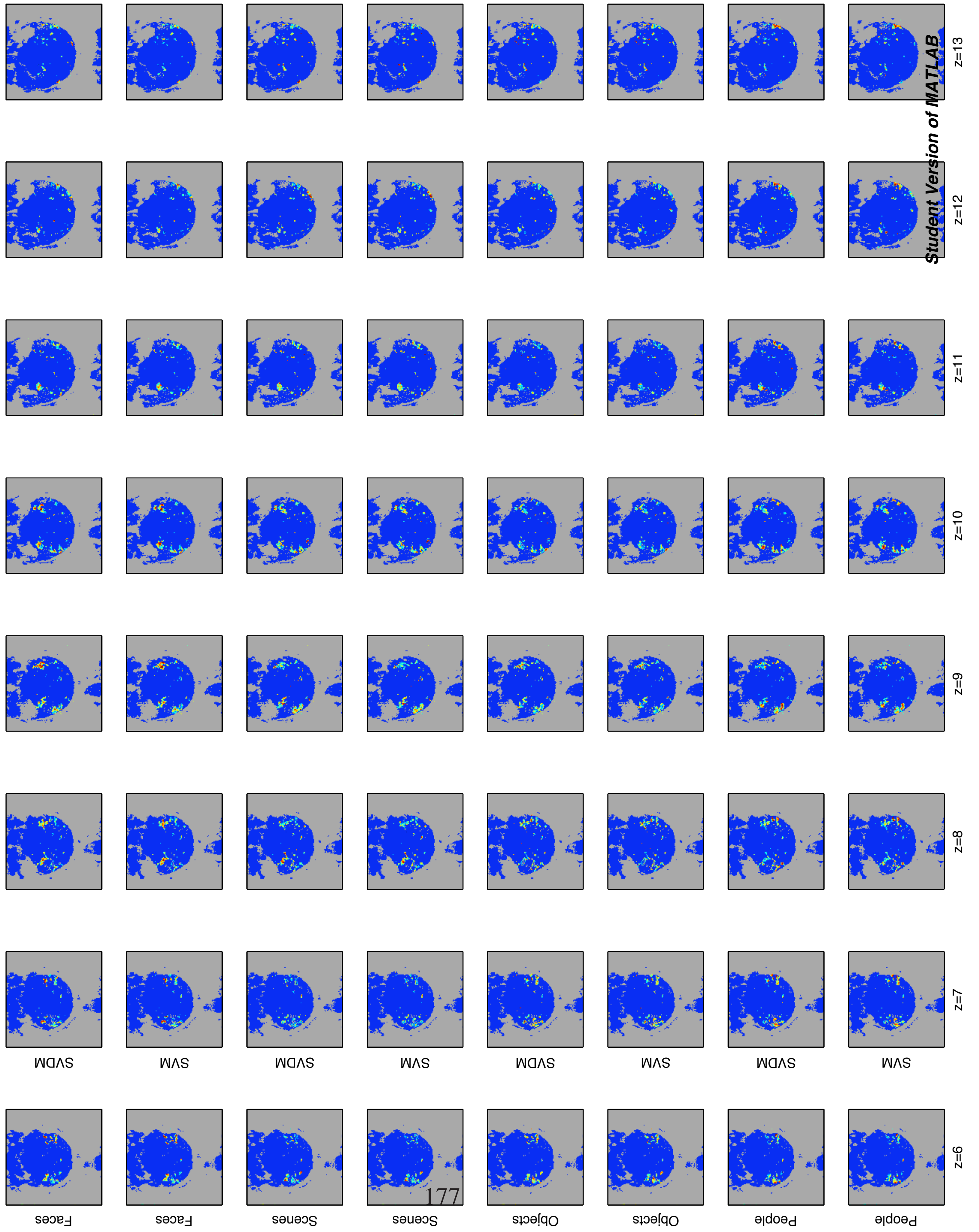
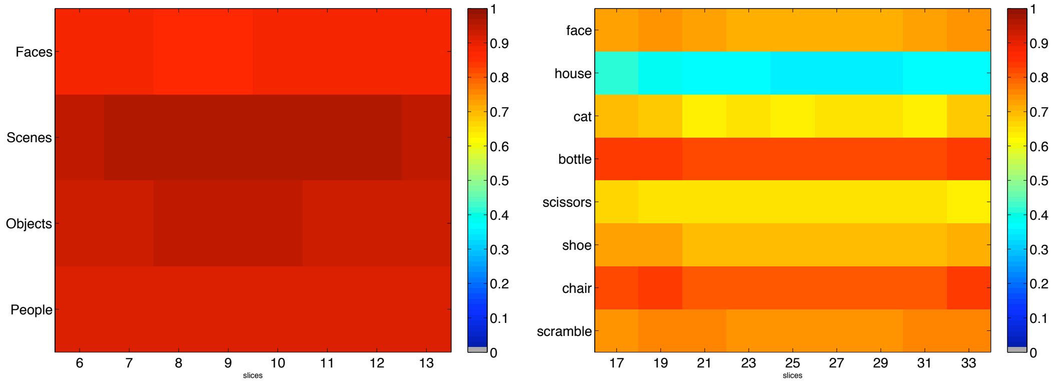


Figure 4.20: A comparison of the discriminants learnt by SVDM and a linear SVM for each of the four one-vs-rest problems (faces,scenes,objects and people). Each pair of rows contain the SVDM and SVM discriminants for one problem, with columns corresponding to brain slices The full brain is shown in dark blue for reference, lighter voxels are the $\sim 4K$ given to the algorithm.



(a) Dataset M1

(b) Dataset M2

Figure 4.21: **Left:** correlation between SVM and SVDM discriminant for each class-vs-others problem (row) and brain slice (column), for dataset M1 (slices shown in Figure 4.20). **Right:** same for dataset M2 (slices shown in Figure 4.23).

4.3 Multisubject models

Thus far the SVDM models considered have always been learnt and deployed on each subject by itself, with accuracy being the main criterion for judging their worth. In the fMRI domain, however, models are judged not just by their prediction accuracy but also by the degree to which the model learnt in one subject shares characteristics with those learnt in others, as well as the extent to which it is consistent with existing knowledge. One example of this would be having a model that places weight on similar locations across all subjects and, moreover, chooses regions that are known to be involved in the task being performed.

This suggests trying to ensure that models do share characteristics across subjects as they are trained, rather than look for them a posteriori. Why would one expect this to work, intuitively? The first reason would be that forcing subject models to agree on something would provide bias in addition to that already present in the SVDM. The second is that more data would be brought to bear on the problem than would be available for any single subject.

The way the optimization problem has been formulated leads to three straightforward ways of building a multi-subject model, similar to the two types of matrix stacking for ICA suggested in [1]. These are illustrated in Figure 4.24:

1. Shared Θ (“sharedT”): learn a single discriminant for all subjects, ensuring that each coordinate in the low dimensional space plays the same role in all of them (e.g. coordinate 2 is positive for examples that belong to the “Tools” category, negative otherwise). Separate W

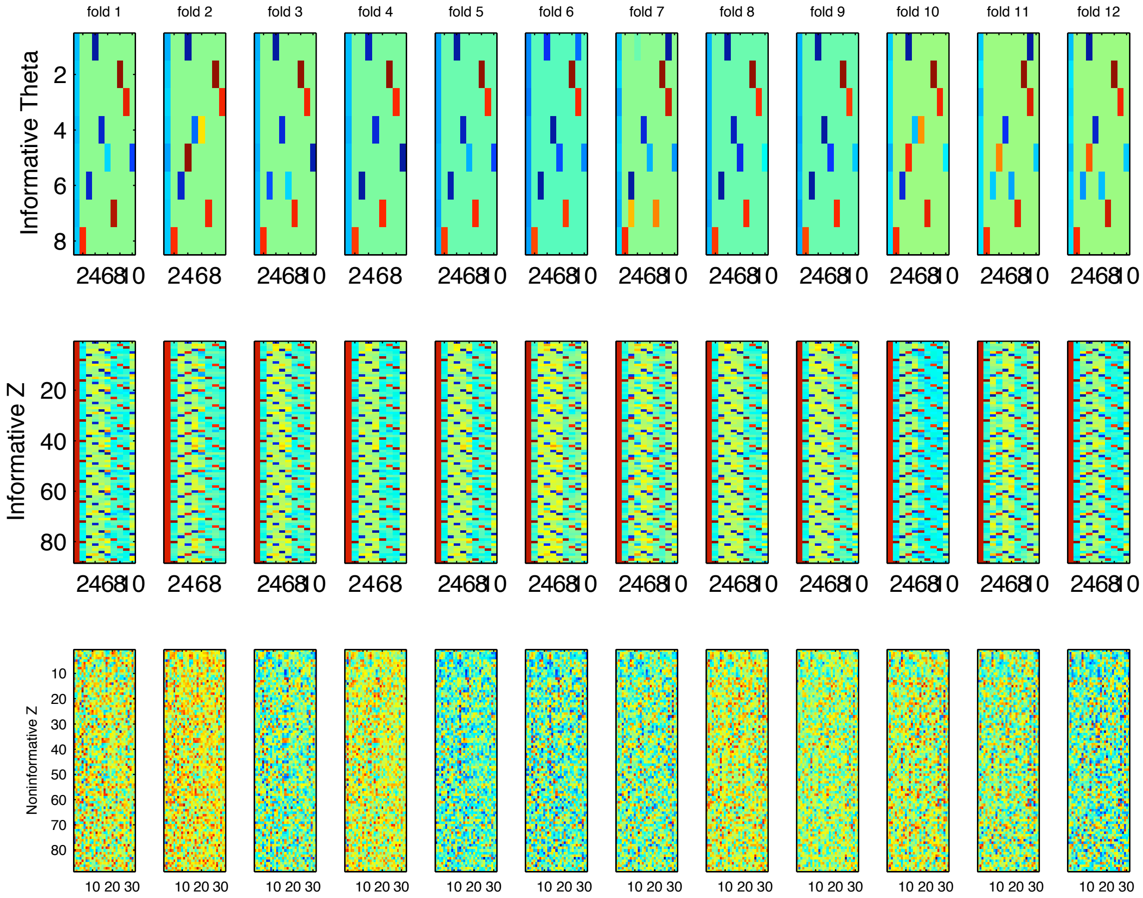


Figure 4.22: Matrices learnt across folds (columns) in dataset M2, analogous to Figure 4.17 for dataset M1.

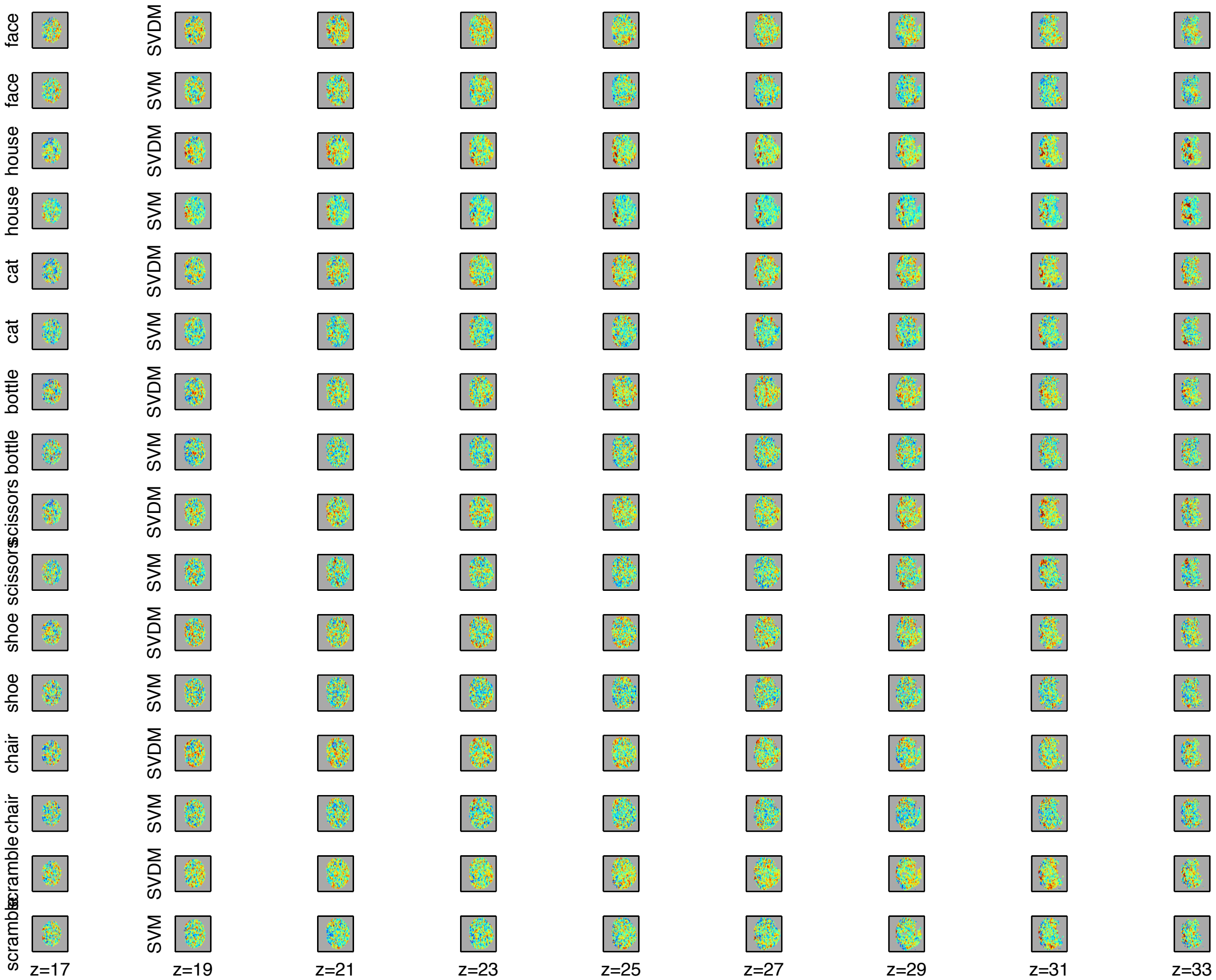


Figure 4.23: **View sideways:** A comparison of the discriminants learnt by SVDM and a linear SVM for each of the eight one-vs-rest problems (faces,houses,cats,bottles,scissors,shoes,chairs,scrambled), similar to Figure 4.20 for dataset M2

.

Student Version of MATLAB

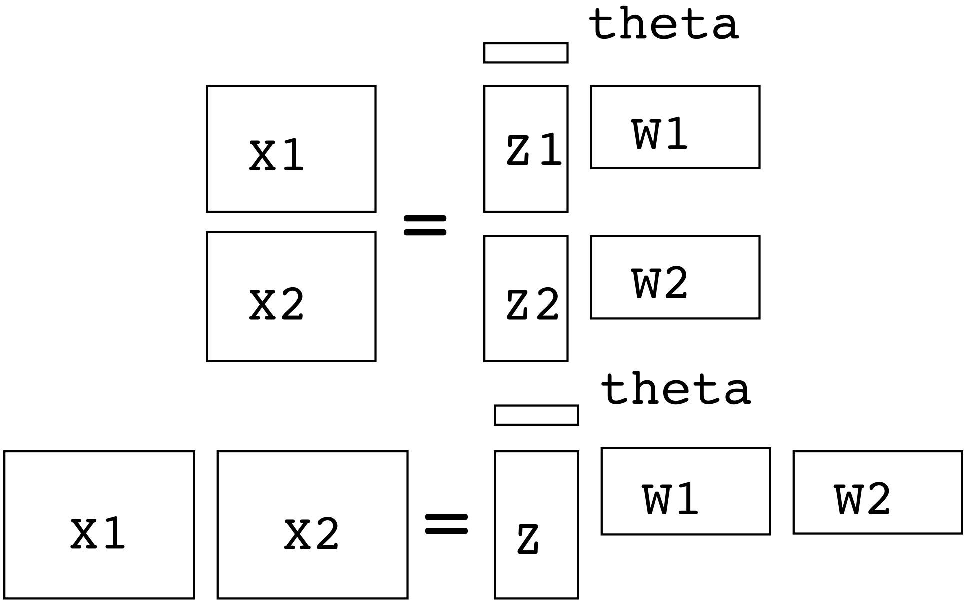


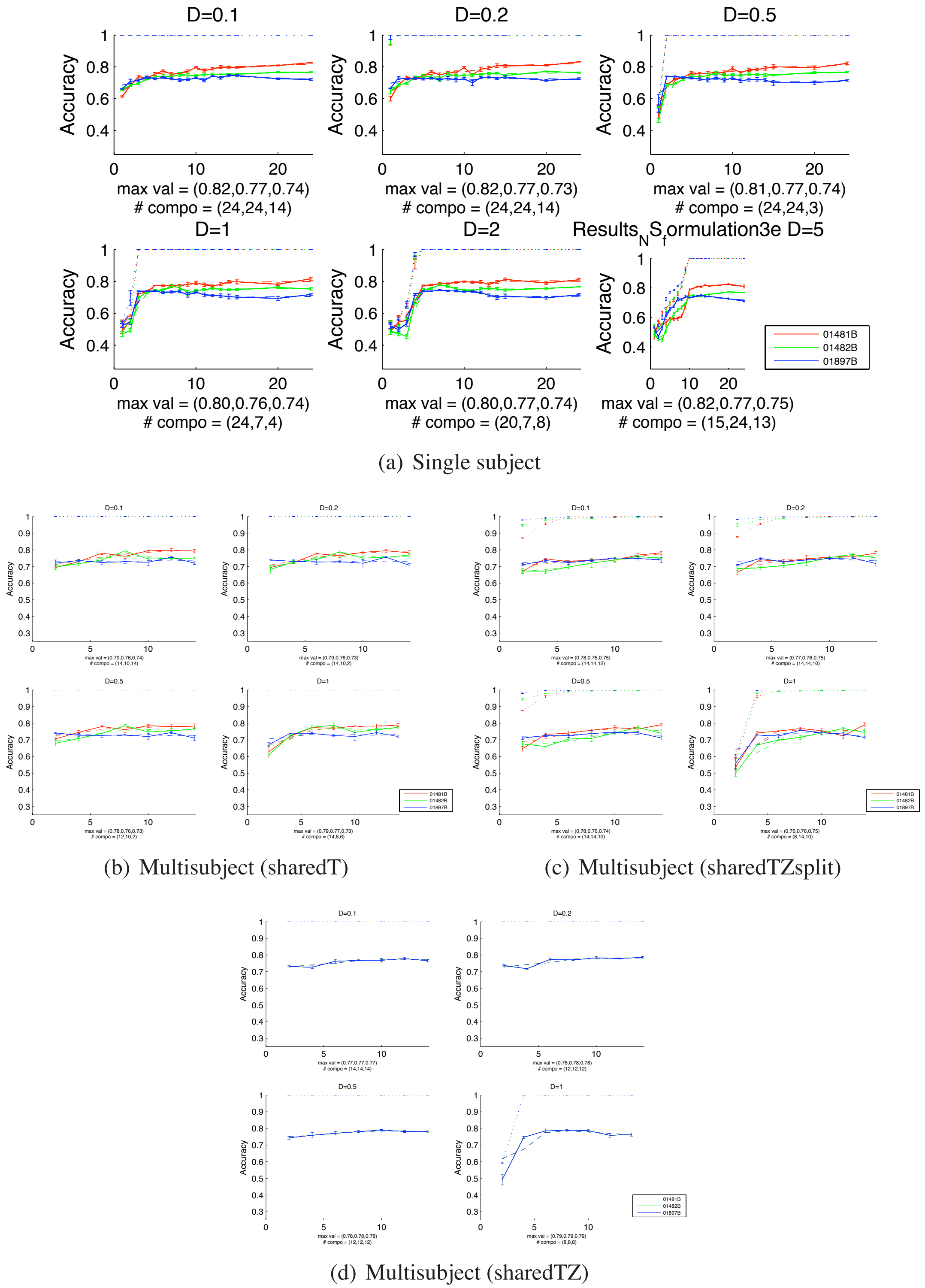
Figure 4.24: Two possible ways of learning a SVDM model combining data from subjects 1 and 2, shared Θ (“sharedT”,top) and shared Θ and Z (“sharedTZ”,bottom).

and Z are produced for each subject.

2. Shared Θ and Z (“sharedTZ”): learn a single discriminant *and* a single low dimensional representation Z shared by all the subjects. This would have the advantage of constraining Z further by ensuring that the role a component plays relative to the labels is the same for all subjects.
3. Shared Θ and Z, followed by shared Θ (“sharedTZdecouple”): A third possibility is to train a model using “sharedTZ” and a penalty encouraging sparsity on the discriminant, yielding a subject-shared Z where only a few columns are informative and a subject specific W. After this, a second stage starts where the Θ learnt is fixed and the model is retrained using “sharedT”, yielding subject-specific Z. The idea is to prevent overfitting by forcing the model make a few informative components agree across subjects, by making the columns of Z similar (the advantage of “sharedTZ”); at the same time, it should not force it to agree on the remainder of the components, since they are used for reconstruction and need not play similar roles across classes for the different subjects (the weakness of “sharedTZ”). An approach related to this one is introduced in [45], for an ICA model with subject-shared and subject independent bases.

How should one measure success of a multisubject model? The most obvious measure is comparing the accuracy of the predictions for each specific subject made by the multisubject model with those made by single subject models. The contrast is made in Figure 4.25 between the performance of the single subject models, the two multisubject models that can generate subject specific predictions and also of the multisubject model that uses all the subjects, for subjects in dataset D1. Figure 4.26 makes the same contrasts for the subjects in dataset D2. The first conclusion one can draw from the graphs is that there does not seem to be any significant advantage (or disadvantage) in terms of accuracy to training a multisubject rather than a single subject model. The second conclusion is that "sharedTZ", the multisubject model that makes a single prediction using all the subjects, has a performance curve that is roughly the average of that of the three single subject models.

Finally, we have the question of whether training with multiple subjects will help in a situation where datasets are smaller. In order to test this we performed a modified 6-fold cross-validation. In each fold we used all of the data for 2 of the three subjects and only $\frac{1}{6}^{th}$ of the data for the third to build a shared Θ model, testing on the remaining $\frac{5}{6}^{th}$. This was done for each of the three subjects and then compared with the same procedure done training a single subject model on the $\frac{1}{6}^{th}$ of the data. The accuracy results were roughly the same for both single and multisubject models, in both datasets.



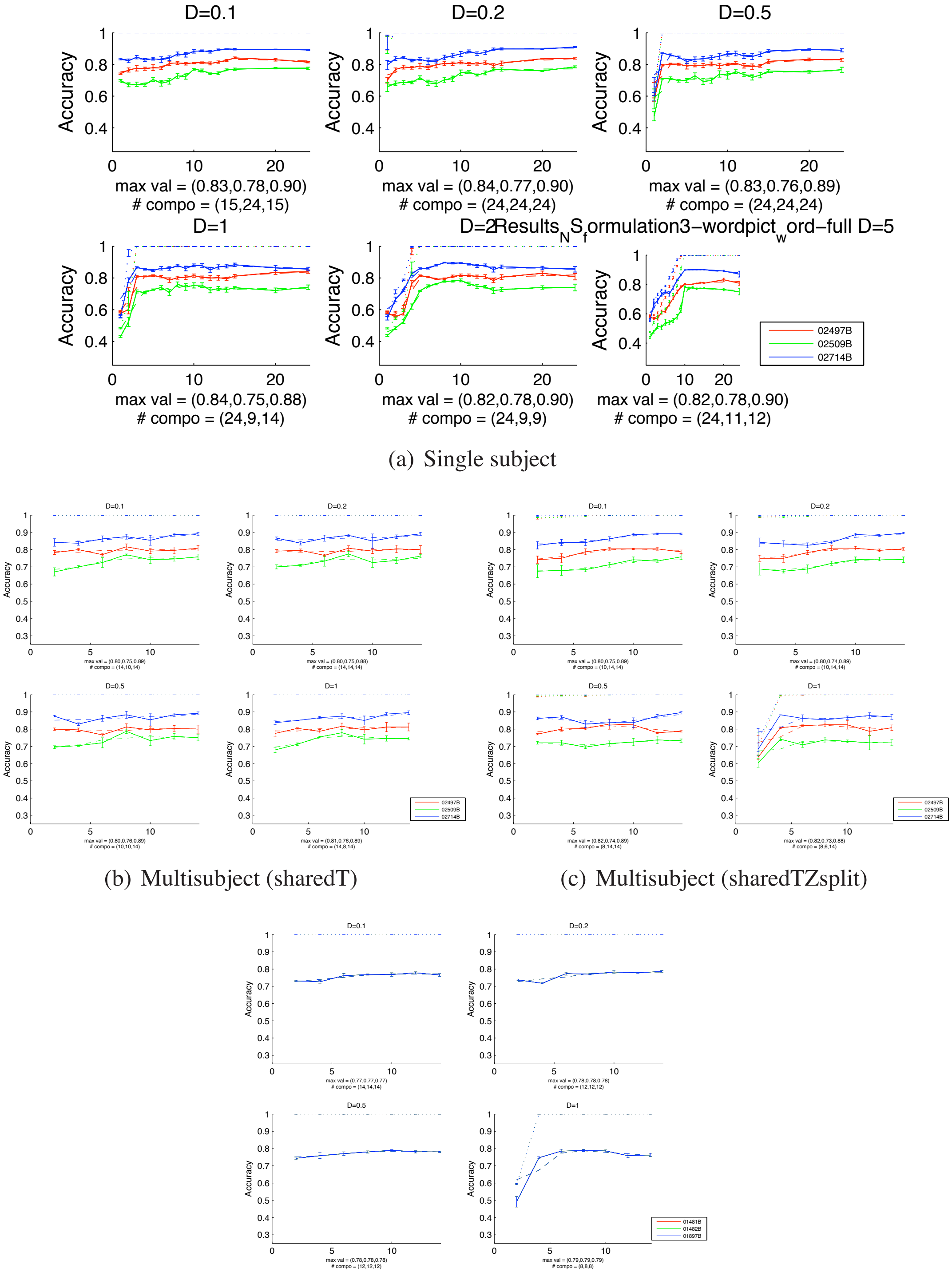
(a) Single subject

(b) Multisubject (sharedT)

(c) Multisubject (sharedTZsplit)

(d) Multisubject (sharedTZ)

Figure 4.25: **Top:** The classification results learning separate models for each subject in dataset D1, for different values of D (same as Figure 3.1 in Chapter 3). **Middle:** Multisubject model subject accuracy curves, for shared Θ or shared Θ and Z. **Bottom:** Multisubject shared Θ and Z model prediction using all the subjects.



(a) Single subject

(b) Multisubject (sharedT)

(c) Multisubject (sharedTZsplit)

(d) Multisubject (sharedTZ)

Figure 4.26: Same as Figure 4.25 for dataset D2

Chapter 5

Conclusions

The thesis put forth in this dissertation is that machine learning classifiers can be used as instruments for decoding variables of interest from functional magnetic resonance imaging (fMRI) data. As stated earlier, there are two main goals in decoding:

- Showing that the variable of interest can be predicted from the data in a statistically reliable manner (i.e. there's enough information present).
- Shedding light on how the data encode the information needed to predict, taking into account what the classifier used can learn and any criteria by which the data are filtered (e.g. how voxels and time points used are chosen).

Chapter 2 considered the issues that arise when using traditional linear classifiers and several different voxel selection techniques to strive towards these goals. Addressing the former requires being able to decide on what a significant result is if starting from many different classification error estimates. Addressing the latter brought to issues such as whether there is redundant, as well as different kinds of, information in fMRI data and how those facts complicate the task of determining whether voxel subsets encode the desired information.

Chapter 3 and Chapter 4 introduced the Support Vector Decomposition Machine, a new algorithm that attempts to sidestep the use of voxel selection by learning a low dimensional representation of the data that is also suitable for decoding variables of interest and has the potential to allow incorporation of domain information directly, rather than through the proxy of voxel selection criteria.

Decoding Information From fMRI Data With Linear Classifiers (Chapter 2)

This chapter starts from the knowledge that is possible to train classifiers to predict variables of interest from fMRI data (from our own work and that of others) and seeks to answer the following questions:

1. For each classifier and voxel ranking method, what is the "best" number of voxels, i.e. at what number of voxels is the true accuracy of the classifier using them higher?
2. Is any classification result significant, taking into account that one has to choose a voxel selection method and a number of voxels, in addition to trying different classifiers and doing all of this for multiple subjects?
3. Are there better classifiers or voxel ranking methods, or does it depend on how they interact?

In order to answer the first question, we introduced a procedure for determining the best number of voxels to use for a given combination of voxel selection method and classifier (rather than just assume the best number is that with the highest classification accuracy, which is very susceptible to noise in the presence of small datasets).

The second question is generally answered by testing the significance of only a few results taken out of a larger set produced with the combination of voxel selection and the use of various numbers of voxels and classifiers. Our view is that selecting the best number of voxels out of several tried is an optimistic procedure. In order to test this we considered the use a null model derived through a permutation test as "ground truth". In face of its computational cost, we introduced a new analytical null model and also a permutation test shortcut. Experiments in fMRI data revealed that the commonly used approaches to testing the significance of classification results in fMRI data are indeed optimistic, in comparison to the significance determined with the permutation test. Further experiments revealed that both the analytical null model and the permutation test shortcut provide significance results closer to that of the permutation test than the usual approach, albeit optimistic (analytical) and pessimistic (shortcut) relative to it.

Given best results for each voxel selection method combined with a classifier, we used ANOVA to answer the third question by testing whether method or linear classifier choice make a difference, across several fMRI studies. The main conclusion is that method choice has much more impact than classifier choice, if voxel selection is used (not the case if all the voxels are used). From the machine learning standpoint, the *simultaneous* comparison of voxel selection and classifier is novel. With a view on using this methodology with more factors or situations where the effect is not so clear cut, we also introduced a visualization of mean difference significance for all interactions of ANOVA factor level. To the best of our knowledge this is a new way of simultaneously gauging whether a significant different exists and, if so, what its magnitude and sign are for all interactions simultaneously.

Beyond the three questions above we also considered the question of whether the use of voxel selection allows for replicable results in terms of which voxels are used in folds of a cross-validation procedure and concluded that it mostly does not. The lack of replicability does not, however, seem to interfere with accuracy, thus indicating a certain degree of redundancy in the set of voxels with the characteristics we are selecting for.

Finally, we addressed the question of whether different kinds of information are present in fMRI data, judging by how a given classifier learns different models that make different prediction mistakes if given data selected by different criteria. There seems to be evidence that, at least for the multiclass problem datasets, there are groups of voxels with different kinds of information. This means that it should be feasible to combine groups to improve accuracy, and profitable to examine the behaviour of the voxel populations involved, as well as their locations.

The conclusions suggest a number of promising directions for future work. The first direction is tackling the correlation between entries in a result table row (classification results using successively larger nested sets of voxels picked from a ranking), with two aspects to consider. Handling correlated errors between successive points is the first aspect, using nonparametric regression methods suited for that setting, and studying it on simulated small dataset result tables with injected correlation. The other aspect is to find a way to extend the analytical null model for result significance to take this correlation into account directly. The second direction is to study the effect of regularization by comparing linear classifier discriminant weights when using all the voxels, where there are noticeable differences at least between GNB and the other classifiers. Preliminary work suggests the effect of regularization is to prevent a lot of discriminant weight from being placed in highly correlated voxels; this may have a beneficial effect for two reasons, the elimination of redundant information and also the fact that, given that noise is spatially correlated, a group of similar noisy voxels that appear informative are prevented from having too much influence in the classification decision. The third direction is to examine the effect of preprocessing choices on the effectiveness of voxel selection methods, in order to ascertain whether those choices remove or enhance the voxel behaviours the methods look for. These choices can take place at a very early stage (such as the choice of frequencies to filter out of data) or right before voxels are selected (such as normalizing feature ranges or making all examples have the same norm across features). The framework we've used to compare classifiers and voxel selection methods in a two factor ANOVA can easily be extended to add preprocessing choices as a third factor. The goals would be not just to identify best practices in terms of preprocessing before classifier use but also to help decide on whether particular choices would interfere with precisely the hypothesized voxel behaviours one may be seeking.

The Support Vector Decomposition Machine (Chapter 3)

In this chapter we considered the problem of learning a low-dimensional representation of sparse, high-dimensional fMRI data capable of decomposing such data in such a way that the low-dimensional dataset can be related to a variable of interest. Such a low-dimensional representation should be *informative* as well as *predictive*, in that the relationship must be understandable as well as useful in classification.

We've introduced two algorithms that learn a low-rank matrix factorization of the data together with a classifier to predict the variable of interest from one of the matrices in the factorization pair; this matrix contains the coordinates of each datapoint in a basis of components given by the other matrix, and thus acts as a low-dimensional representation. Both algorithms work by trading off the reconstruction error of the original matrix from the product of the two low-rank matrices and the error of a classifier based on the low-dimensional representation. Given that the decomposition into coordinates and basis, each example can then be seen as a weighted sum of components.

We've shown that using either algorithm to learn the low-dimensional representation is not only feasible but, also, that a classifier trained on the learnt representation predicts better than the same classifier trained on a representation learnt separately using SVD or ICA, the most obvious competing methods. On the binary classification problems in the fMRI data domain the algorithm performs as well as the best regularized classifiers learning using all the voxels.

From the standpoint of interpretability, the relating of the components to the variable of interest, it's not clear that either algorithm is as successful. There are many possible low-dimensional representations that do equally well in terms of reconstruction or classification error, both while they are being trained and also in test data. Hence, it's possible for a representation to be good for prediction and yet have different components interacting in a complicated way to make that prediction, making it hard to say which of them are informative. This can be addressed by constraining the learnt solution, in this case by limiting the number of components that can affect prediction, leaving us with only one or two components to consider informative in a binary classification task.

Chapter 4 considers several other approaches to constraining the solution space or expanding the capabilities of this class of algorithm, namely:

- Adding regularization to shape the basis images learnt or automatically determined how many informative components to have.

- Learning low-dimensional representations for several subjects in the same study, sharing parts of those representations over all of them and using this as an additional source of constraints.

- Behaviour of the algorithm with more than two class labels.

Support Vector Decomposition Machine Extensions (Chapter 4)

The main conclusion is that SVDM can be extended to support additional regularization beyond the simple L_2 norm penalty in the original formulation, while maintaining convexity in all its subproblems. It can also be modified to automatically determine the number of informative components necessary. Furthermore, it can handle multiclass problems and, to the limited extent tested, outperform linear SVMs in that setting. Finally, it can be modified in an elegant way to handle multiple subjects in a single model with subject independent and subject specific parts.

On the negative side, if using binary classification problems, a model with an automatically determined number of informative components essentially learns a linear SVM. If building a multiclass model there is no guarantee that there will be any component sharing (one of the original goals of building an SVDM), though the results suggest it can happen and will likely depend on the dataset. Finally, there seems to be no no advantage to building multiple subject models, even in the presence of very small datasets where one would expect the pooled information to be particularly useful.

Given the above, the most obvious directions for future work are:

- Develop more regularization approaches, drawing from domain specific knowledge and the feature selection work stemming from Chapter 2.
- Develop the extensions suggested, in particular other types of penalization and making the learnt basis closer to spatial ICA than to SVD.
- Explicitly enforce more component sharing or certain types of component relationships (e.g. products of the respective columns in the Z matrix), developing this in tandem with synthetic data models with known or postulated relationships between stimulus classes (e.g. “cats” and “faces” must share some activation).
- Study the characteristics of models with many more components, now that we have a means of ensuring that only as few as necessary will be used for classification.
- Develop methods for interpreting the models learnt using error correcting output codes, given that they can both be more accurate and also explicitly consider many breakdowns of the classes into binary problems; eventually, this might be useful for finding class groupings not predicted by domain experts but that can be accurately distinguished from a particular voxel support (that would be captured by a component).
- Understanding why no extra information is derived from multi-subject models and whether combining multisubject with multiclass models will give SVDM more scope to learn subject-independent structure.

- Experimenting with datasets with several tasks performed by the same subject which are non-trivial from a classification standpoint, in the hope that this will be more favourable for multitask learning through component sharing.

Machine Learning Classifiers and Neuroscience

The goal of this thesis was to show that machine learning classifiers allow us to examine fMRI data in ways that go beyond looking for individual voxels matching a specified pattern of behaviour. This can mean anything ranging from showing that there is enough information in a multivariate pattern of activation to predicting a variable of interest to revealing how that information is organized; in the limit, we would like to show how the classes being predicted are related to each other, how those relationships are underpinned by parts of the pattern of activation and how they, in turn, reflect known information about the neural substrate and observable phenomena of cognition.

The process of doing this also yielded more general benefits in terms of machine learning applicability. The lessons learned in terms of training, applying and evaluating classification results generalize to other domains with sparse, high-dimensional datasets (such as DNA microarrays) where feature selection is necessary; furthermore, it's unclear that there is *a* better feature selection method. Rather, there may be many different feature behaviours all of which can have some - but not all - relevant information. Several of the domain characteristics considered also go against the "traditional" setting for machine learning: examples drawn i.i.d. from an example distribution, with more examples than features and uncorrelated features. Our belief is that the work in this domain shows that something can be learnt even when reality is far from this setting.

SVDM is an attempt to deal with the fact that the sparse high-dimensional data may, in addition, be more complex than usual due to the presence of many sources of activation. Most of these are unrelated, or only partially related, to what we would like to predict and thus act as a complex, structured noise. Furthermore, it has both a spatial and a temporal dimension (especially in fast event-related designs) that may both provide additional leverage (e.g. spatial smoothness prior for classifier parameters) and hinder us through violated assumptions (e.g. examples no longer being drawn independently, or appearing only as mixtures). Finally, that same spatial structure also manifests in relationships between features: we can expect correlation, mutual inhibition or even more complex combinatorial codes as spatial resolution improves.

In conclusion, we hope that this dissertation convinces the reader that cognitive neuroscience has a lot to offer to machine learning as an application area, both in terms of challenging data properties but also of the richness of the domain knowledge and potential payoff in terms of influencing current scientific practice.

Bibliography

[1] M. N. Wernick A. S. Lukic, Y. Yang, L. K. Hansen, K. Arfanakis, and S. C. Strother. Effect of spatial alignment transformations in pca and ica of functional neuroimages. *IEEE Transactions on Medical Imaging*, 26 (8):1058–1068, 2007. 4.3

[2] AFNI. Analysis of functional neuroimaging. Technical report, NIMH (http://afni.nimh.nih.gov/afni). 2.3.1

[3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995. 3.4

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. 3.1.2

[5] V. Calhoun, T. Adali, L. Hansen, J. Larsen, and J. Pekar. Ica of functional mri data: an overview. In *4th international symposium on Independent Component Analysis and Blind Signal Separation (ICA 2003)*, 2003. 3.5.5, 4.1.4

[6] T. A. Carlson, P. Schrater, and S. He. Patterns of activity in the categorical representations of objects. *Journal of Cognitive Neuroscience*, 5(15):704–717, 2003. 1, 2.1.2, 2.1.4

[7] R. Caruana. Multitask learning. *Machine Learning Journal*, 28:41–75, 1997. 3.0.5

[8] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *International Conference on Machine Learning*, 2006. 2.2.5

[9] G. Casella and R. Berger. *Statistical Inference*. Duxbury, Pacific Grove, CA, 2002. 2.2.3, 2.2.5

[10] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines (http://www.csie.ntu.edu.tw/ cjlin/libsvm)*, 2001. 3.5.2

[11] Xu Chen, Francisco Pereira, Wayne Lee, Stephen Strother, and Tom Mitchell. Exploring predictive and reproducible modeling with the single-subject fiac dataset. *Human Brain Mapping*, 27(5):452–461, 2006. 3.3

[12] 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*, 19:261–271, 2003. 2.1.3, 2

[13] T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923, 1997. 2.2.5

[14] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995. 2.1.2, 4.2.2

[15] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003. 2.1.4, 2.2.2, 2.2.5

[16] Karl Friston. *Statistical Parametric Mapping: The Analysis of Functional Brain Images*. Academic Press, 2006. 1, 1

[17] A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems*, 2005. 3.4

[18] P. Golland, F. Liang, S. Mukherjee, and D. Panchenko. Permutation tests for classification. In *Conference on Learning Theory (LNCS 3559)*, pages 501–515. Springer, 2005. 2.2.4

[19] Phillip Good. *Permutation, Parametric and Bootstrap Tests of Hypotheses*. Springer, New York, 2005. 2.2.2, 2.2.4

[20] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003. 2.1.4

[21] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning Journal*, 46:389–422, 2002. 3, 3.4

[22] H. Gvert, J. Hurri, J. Srel, and A. Hyvrinen. *FastICA (http://www.cis.hut.fi/projects/ica/fastica/index.shtml)*, 2005. 3.5.5

[23] LK Hansen et al. Generalizable patterns in neuroimaging: How many principal components? *Neuroimage*, ?:?, 1999. 3.4

[24] 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, 2004. 2.1.3, 2.4.1

[25] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer-Verlag, 2001. 1, 2.1.2, 2.1.2, 3.1.2, 3.2.2

[26] J. V. Haxby, M. I. Gobbini, M. L. Furey, A. Ishai, J. L. Schouten, and P. Pietrini. Distributed and overlapping representations of faces and objects in ventral tempora l cortex. *Science*, 293(28 September):2425–2430, 2001. 2.3.7, 2.4.2, 3.5.5, 4.2.3

[27] J. Haynes and G. Rees. Decoding mental states from brain activity in humans. *Nature Reviews Neuroscience*, 7:523–34, 2006. 1, 1, 2.1.4

[28] J. Himberg, A. Hyvarinen, and F. Esposito. Validating the independent components of neuroimaging time-series via clustering and visualization. *Neuroimage*, ?:?, ? 3.4

[29] G. Hinton, R. Neal, R. Tibshirani, and DELVE team. Assessing learning procedures using delve. Technical Report 461, Department of Computer Science, University of Toronto, 1995. 2.2.5

[30] S. A. Huettel, A. W. Song, and G. McCarthy. *Functional Magnetic Resonance Imaging*. Sinauer, 2004. 1

[31] A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley and Sons, 2001. 1

[32] N. Intrator. On the combination of supervised and unsupervised learning. *Physica A*, pages 655–661, 1993. 3.4

[33] T. Joachims. Making large-scale svm learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999. 1, 2.1.2

[34] Y. Kamitani and F. Tong. Decoding the visual and subjective contents of the human brain. *Nature Neuroscience*, 8:679–685, 2005. 1, 2.2.2

[35] N. Kanwisher. The ventral visual object pathway in humans: Evidence from fmri. In L. Chalupa and J. Werner, editors, *The Visual Neurosciences*. MIT Press, 2003. 2.3.7, 2.4.2

[36] A. J. Klockars and G. Sax. *Multiple Comparisons*. Sage University Paper (no. 61), Beverly Hills, 1985. 2.2.2, 2.2.5

[37] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, 1995. 2.2.3

[38] N. Kriegeskorte, R. Goebel, and P. Bandettini. Information-based functional brain mapping. *Proceedings of the National Academy of Sciences of the USA*, 103:3863–3868, 2006. 2.1.3, 2.1.4

[39] R. Kustra and S. C. Strother. Penalized discriminant analysis of [o-15]-water pet brain images with prediction error selection of smoothness and regularization hyperparameters. *IEEE Transactions on Medical Imaging*, 20:367–387, 2001. 1, 2.1.2, 2.1.3

[40] S. LaConte, S.C. Strother, V. Cherkassky, J. Anderson, and X. Hu. Support vector machines for temporal classification of block design fmri data. *Neuroimage*, 26:317–329, 2005. 2.1.1

[41] N. Lange, S. C. Strother, J. R. Anderson, F. A. Nielsen, A. P. Holmes, T. Kolenda, R. Savoy, and L. K. hansen. Plurality and resemblance in fmri data analysis. *Neuroimage*, 10:282–303, 1999. 2.3.7

[42] J. Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6:273–306, 2005. 2.2.3

[43] T. Lim, W. Loh, and Y. Shih. A comparison of prediction accuracy, complexity and training time of thirty-three old and new classification algorithms. *Machine Learning Journal*, 40:203–229, 2003. 2.2.5, 2.3.5

[44] Richard Lowry. *Concepts and Applications of Inferential Statistics*. http://faculty.vassar.edu/lowry/webtext.html, 1999-2007. 2.1.4, 2.2.2, 2.2.5, 2.3.5

[45] A. S. Lukic, M. N. Wernick, L. K. Hansen, J. Anderson, and S. C. Strother. A spatially robust ica algorithm for multiple fmri data sets. In *IEEE International Symposium on Biomedical Imaging*, pages 839–842, 2002. 3

[46] J. K. Martin and D.S. Hirschberg. Small sample statistics for classification error rates i: Error rate measurements. Technical Report 21, Computer Science Dept., University of California Irvine, 1996. 2.2.3

[47] J. K. Martin and D.S. Hirschberg. Small sample statistics for classification error rates ii: Confidence intervals and significance tests. Technical Report 22, Computer Science Dept., University of California Irvine, 1996. 2.2.3, 2.2.5

[48] A. R. McIntosh, F. L. Bookstein, J. V. Haxby, and C. L. Grady. Spatial pattern analysis of functional brain images using partial least squares. *NeuroImage*, (3):143–157, 1996. 3.4

[49] A. R. McIntosh and N. J. Lobaugh. Partial least squares analysis of neuroimaging data - applications and advances. *NeuroImage*, 23:250–263, 2004. 3.4

[50] R. S. Menon and S. Kim. Spatial and temporal limits in cognitive neuroimaging with fmri. *Trends in Cognitive Sciences*, 3(6):207–216, 1999. 1

[51] T. Mitchell. *Machine Learning*. McGraw Hill, New York, 1997. 2.1.2, 2.1.3

[52] T. M. Mitchell, R. Hutchinson, R. S. Niculescu, F. Pereira, X. Wang, M. Just, , and S. Newman. Learning to decode cognitive states from brain images. *Machine Learning*, 57:145–175, 2004. 1, 1, 1, 2.1.1, 2.1.4, 1, 3.5.2

[53] J. Neumann, C. Schnrr, and G. Steidl. Combined svm-based feature selection and classification. *Machine Learning*, 61(1-3):129–150, 2005. 3.4

[54] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems*, 2001. 2.4.3

[55] K. Norman, S. M. Polyn, G. Detre, and J. V. Haxby. Beyond mind-reading: multi-voxel pattern analysis of fmri data. *Trends in Cognitive Sciences*, 10, 2006. 1, 1, 2.1.4

[56] J. Opsomer, Yuedong Wang, and Yuhong Yang. Nonparametric regression with correlated errors. *Statistical Science*, 16 (2):134–153, 2001. 2.2.3

[57] F. Pereira and G. Gordon. The support vector decomposition machine. In *International Conference on Machine Learning*, 2006. 2.1.3, 3.4, 3.5.2, 3.5.4, 3.5.5

[58] F. Pereira, T. Mitchell, R. Mason, M. Just, and N. Kriegeskorte. Spatial searchlights for feature selection and classification of functional mri data. In *Abstracts of the* 12^{th} *Conference on Human Brain Mapping*, 2006. 2.1.3, 2.3.7, 2.4.1

[59] S. M. Polyn, V. S. Natu, J. D. Cohen, and K. A. Norman. Category-specific cortical activity precedes recall during memory search. *Science*, 310:1963–1966, 2005. 2.1.1

[60] M. Rogati and Y. Yang. High-performing and scalable feature selection for text classification. In *Conference on Information and Knowledge Management*, 2002. 2.2.2, 2.2.5

[61] SPM. Statistical parametric mapping. Technical report, Functional Imaging Laboratory, UCL (http://www.fil.ion.ucl.ac.uk/spm). 2.3.1

[62] N. Srebro. *Learning with Matrix Factorizations*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 2004. 3.4

[63] J. Stoeckel and G. Fung. A mathematical programming approach for automatic classification of spect images of alzheimer's disease. In *International Conference on Data Mining*, 2005. 2.1.3, 3.4

[64] S. C. Strother. Evaluating fmri preprocessing pipelines. *IEEE Engineering in Medicine and Biology Magazine*, 25 (2):27–41, 2006. 1, 2.2.5

[65] S. C. Strother, J. Anderson, L. K. Hansen, U. Kjems, R. Kustra, J. Siditis, S. Frutiger, S. Muley, S. LaConte, and D. Rottenberg. The quantitative evaluation of functional neuroimaging experiments: The npairs data analysis framework. *Neuroimage*, 15:615–624, 2002. 1, 2.1.2, 2.1.4, 2.2.3

[66] S. C. Strother, J. Anderson, L. K. Hansen, U. Kjems, R. Kustra, J. Siditis, S. Frutiger, S. Muley, S. LaConte, and D. Rottenberg. The quantitative evaluation of functional neuroimaging experiments: The npairs data analysis framework. *NeuroImage*, 15:747–771, 2002. 1

[67] Larry Wasserman. *All of Statistics*. Springer, New York, 2004. 2.2.2, 2.2.3, 2.2.4, 2.2.5, 2.4.2

[68] Larry Wasserman. *All of Nonparametric Statistics*. Springer, New York, 2006. 2.2.3

[69] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbour classification. In *Advances in Neural Information Processing Systems*, 2005. 3.4

[70] E. P. Xing. Feature selection in microarray analysis. In D. P. Berrar, W. Dubitzky, and M. Granzow, editors, *A Practical Approach to Microarray Data Analysis*. Kluwer Academic Publishers, San Francisco, 2003. 2.1.4

[71] E. P. Xing, M. I. Jordan, and R. M. Karp. Feature selection for high-dimensional genomic microarray data. In *International Conference on Machine Learning*, 2001. 2.2.5

[72] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, 2002. 3.4

[73] Y. Yang and J. Pedersen. A comparative study on feature selection in text categorization. In *International Conference on Machine Learning*, pages 412–420, 1997. 2.1.4, 2.2.2, 2.2.5