THE REPUBLIC OF TURKEY
BAHÇEŞEHİR UNIVERSITY

EVALUATION AND IMPROVEMENT OF FEATURE SELECTION TECHNIQUES FOR COGNITIVE STATE CLASSIFICATION USING fMRI DATA

Master's Thesis

CEYHUN CAN ÜLKER

İSTANBUL, 2012

THE REPUBLIC OF TURKEY
BAHÇEŞEHİR UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
COMPUTER ENGINEERING

EVALUATION AND IMPROVEMENT OF FEATURE SELECTION TECHNIQUES FOR COGNITIVE STATE CLASSIFICATION USING fMRI DATA

Master's Thesis

CEYHUN CAN ÜLKER

Supervisor: ASST. PROF. DR. TEVFİK AYTEKİN

İSTANBUL, 2012

THE REPUBLIC OF TURKEY
BAHÇEŞEHİR UNIVERSITY
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
COMPUTER ENGINEERING

Title of the Master's Thesis	:	Evaluation and Improvement of Feature Selection Techniques for Cognitive State Classification using fMRI Data
Name/Last Name of the Student	:	Ceyhun Can ÜLKER
Date of Thesis Defense	:	14 May, 2012

The thesis has been approved by The Graduate School Of Natural And Applied Sciences.

Assoc. Prof. Dr. Tunç BOZBURA
Acting Director

This is to certify that we have read this thesis and that we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Science.

Examining Commitee Members:	Signature
Asst. Prof. Dr. Tevfik AYTEKİN (Supervisor)	: ....
Asst. Prof. Dr. Kemal Egemen ÖZDEN	: ....
Asst. Prof. Dr. Mehmet Alper TUNGA	: ....

ACKNOWLEDGEMENTS

I would like to dedicate this thesis to beloved my father and my family. Without their support, encourgement and belief in me, I don't think I would be able to accomplish this.

I would like to specially thank my thesis advisor Dr. Tevfik Aytekin for being far more than an advisor. His interest and belief in me saved me many times when in doubt and despair, he had such an incredible tolarance for me. His being opened to new horizons, his eagerness in learning and teaching, gave me an invaluable experience and knowledge. I wish I could make much greater progress in return.

I am thankful to my jury members Dr. Egemen Özden and Dr. Alper Tunga for giving their valuable time, helpfulness and costructive critisism. I also need to thank our department chair Dr. Taşkin Kocak and every other faculty members for their tolerance, and making life at Bahçeşehir University easy and enjoyable.

Lastly, I would must mention and thank Emin Akşehirli, Murat Yağcı, Ertunç Erdil, Jbid Arsenyan, Erdem Erzurum, Bengi Aygün, Selçuk Keskin, Davut Özcan, Dilan Şahin, Efsun Karaca, Necati Kılıç, Ali Karaali, Ömer Tura, İlkay Öksüz, Onur Önder, Sezer Ulukaya, Güneş Akşehirli many other heroes that I forgot to name here for being really helpful both technically and everyday life issues.

It was a great honor and pleasure for me to be met with each person mentioned in here.

14 May 2012 Ceyhun Can ÜLKER

ABSTRACT

EVALUATION AND IMPROVEMENT OF FEATURE SELECTION TECHNIQUES FOR COGNITIVE STATE CLASSIFICATION USING fMRI DATA

Ülker, Ceyhun Can

Computer Engineering
Supervisor: Asst. Prof. Dr. Tevfik AYTEKİN

May 2012, 45 Pages

Recent research has shown that it is possible to classify cognitive states of human subjects based on fMRI (functional magnetic resonance imaging) data. One of the obstacles in classifying fMRI data is the problem of high dimensionality. A single fMRI snapshot consists of thousands of voxels and since a single experiment contains many fMRI snapshots, the dimensionality of an fMRI data instance easily surpasses the order of tens of thousands. So, feature selection methods become a must from both classification and running time performance points of view. To this end several feature selection methods are studied, either general or specific to fMRI data. So far, one of the best such methods, which is specific to fMRI data, is called the "active" method. In this work we combine genetic algorithms with the active method in order to improve the performance of feature selection. Specifically, we first reduce the feature dimension using the active method and search for informative features in that reduced space using genetic algorithms. We achieve similar levels of classification performance using much less number of voxels than active method offers.

Keywords: fMRI, Feature selection, Voxel selection, Genetic algorithm, Cognitive state prediction

ÖZET

fMRI VERİSİ KULLANARAK BİLİŞSEL HAL TASNİFİNDE ÖZNİTELİK SEÇİM TEKNİLERİNİN DEĞERLENDİRİLMESİ VE İYİLEŞTİRİLMESİ

Ceyhun Can Ülker

Bilgisayar Mühendisliği
Tez Danışmanı: Yrd. Doç. Dr. Tevfik AYTEKİN

Mayıs 2012, 45 Sayfa

Son zamanlardaki araştırmalar insan deneklerin fMRI (fonksiyonel manyetik rezonans görüntüleme) verisini kullanarak bilişsel hal ayırt etmenin mümkün olduğunu göstermiştir. fMRI verisinin sınıflandırılmasını güçleştiren en büyük engellerden biri verinin yüksek boyutlu ve seyrek olmasıdır. Tek bir fMRI enstantanesi binlerce voxel bulundurabilir ve bir deney bir çok fMRI enstantanesi barındırdığından verinin boyutu kolaylıkla on binleri geçebilir. Şu halde öznitelik seçimi yöntemlerinin kullanılması hem sınıflandırma hem de çalışma zamanı başarımları bakımlarından zorunluluk halini almıştır. Bu yüzden gerek genel gerekse fMRI verisine özgü bir çok öznitelik seçim yöntemi çalışılmıştır. Şimdiye kadarki en iyi yöntemlerden biri de "aktif" olarak adlandırılan fMRI verisine özgü öznitelik seçim yöntemidir. Bu çalışmada genetik algortima öznitelik seçimi başarı mının arttırılmasını sağlamak için aktif yöntemi ile birleştirilmiştir. Özel olarak, öncelikle aktif yöntem kullanılarak öznitelik boyutunu azaltıp, sonra bu indirgenmiş uzayda genetik algoritma kullanılarak diğerlerinden daha çok bilgi taşıyan öznitelikler aranmıştır. Bu yöntem yardımıyla aktif yöntemi ile benzer başarı seviyesi, aktif yöntemin sunduğundan çok daha az sayıda voxel kullanılarak, sağlanabilmiştir.

Anahtar Kelimeler: fMRI, Öznitelik Seçme, Voksel Seçme, Genetik Algoritma, Bilişsel Hal Tahmini

CONTENTS

TABLES

FIGURES

ABBREVIATIONS

fMRI	:	Functional Magnetic Resonance Imaging
BOLD	:	Blood Oxygen Level Dependent Response
TR	:	Time Resolution
SVM	:	Support Vector Machines
KNN	:	K-Nearest Neighbour
GNB	:	Gaussian Naïve Bayes
Voxel	:	Volume Element

SYMBOLS

Population	:	$\boldsymbol{\Omega}$
k^{th} Generation	:	$\boldsymbol{\Omega}^{(k)}$
i^{th} Individual of Population $\boldsymbol{\Omega}$	:	$\boldsymbol{\Omega}_i$
j^{th} Attribute of i^{th} Individual of Population $\boldsymbol{\Omega}$	:	$\boldsymbol{\Omega}_{i,j}$
Set of Indices	:	$\mathcal{X}$
Individuals from Population $\boldsymbol{\Omega}$ whose Indices Given by $\mathcal{X}$	:	$\boldsymbol{\Omega}_{\mathcal{X}}$
ω Attributes of Individuals with Indices $\mathcal{X}$ in Population $\boldsymbol{\Omega}$	:	$\boldsymbol{\Omega}_{\mathcal{X},\omega}$
Binary Vector, Typically Denoting a Specific Individual	:	ω
j^{th} Attribute of Individual ω	:	ω_j
Optimum Individual	:	ω^*
Number of Examples in a Dataset	:	N
Number of Features or Number of Voxels	:	n
Number of Individuals in a Population	:	p
Proportion of Individuals to Survive *Selection*	:	ρ
Proportion of Selected Attributes in an Initial Population	:	τ
Proportion of Mutated Attributes in an Individual	:	μ
Proportion of Exchanged Attributes in *Crossover*	:	κ
Dataset	:	$\mathcal{D}$
Error	:	ϵ, ε
Estimated Error on Dataset $\mathcal{D}$ Using Learning Algorithm Ψ	:	$\hat{\varepsilon}_{\mathcal{D}}(\Psi)$
Learning Algorithm Parameters	:	$\boldsymbol{\Theta}$
Threshold Parameter	:	Θ
Learning Method	:	Ψ
Indicator Function of x	:	$1\{x\}$
Set of classes	:	$\mathcal{C}$
k-Nearest Neigbour of x	:	$\mathcal{N}_k(x)$

1. INTRODUCTION

Machine learning techniques have been successfully applied to classify fMRI data in order to decode cognitive states of humans (Mitchell et al., 2004), (Davatzikos et al., 2005), (Mourão-Miranda et al., 2005), (Zhang & Lee, 2009), (Formisano et al., 2008), (Shinkareva et al., 2008), (Yoshida & Ishii, 2005). One of the biggest challenges in applying machine learning methods to fMRI data lies in the nature of the fMRI dataset. Typically, in fMRI datasets the dimensionality of the feature space is very high (in the order of tens of thousands) whereas the number of examples is very limited (in the order of hundreds or even less). High dimensionality effects both running time and classification performance. From the classification point of view, more dimensionality generally means more irrelevant information, which in turn may mean more noise. From the running time point of view, high dimensionality increases both training and test times of machine learning algorithms. These issues brings the need to reduce the number of features to a reasonable level.

In cases where the number of features is large feature selection methods are typically employed. There are different feature selection methods (Guyon & Elisseeff, 2003). One class of such methods is named as the *wrapper methods* (Zongker & Jain, 1996) which wraps a quality measure and based on that measure incrementally adds (starting with an empty set) or removes (starting with a full set) features that are best (in case of removal, worst) at each iteration. In this work, we utilize genetic algorithms, which can also be used as a wrapper method and which is especially known to be performing well in searching huge spaces (Siedlecki & Sklansky, 1989), (Kudo & Sklansky, 2000), (Liu et al., 2009). As we will explain below in detail, using genetic algorithms we achieve similar levels of classification performance with much less number of voxels.

1.1 OUTLINE

This thesis is organized as follows:

- In Chapter 1 a brief introduction is given to the problem at hand, feature selection problem of cognitive state prediction using fMRI data.
- In Chapter 2 machine learning is explained, some of the machine learning techniques are introduced. Also a brief introduction is given on fMRI type medical imaging and BOLD signal. A thorough review done to literature, explaining what has been done with machine learning so far to solve the high dimensionality problem, what works have been done with fMRI classification, available works on Genetic Algorithm used as feature selection methods, etc.
- Chapter 3 puts the main work done throughout this thesis. The datasets are introduces by giving information about the sources, structure of the dataset and the experiments that resulted in those data. Under the methods section, reader is given the methodology used for applying genetic algorithms as a feature selection method to classify fMRI data. It follows with other methods tried such as functional dependency graphs, random subspaces, etc.
- In Chapter 4 gives the experimentation framework employed and the experimental results obtained by mentioned methods, by means of graphs and tables along with their explanations.
- In Chapter 5, finally the thesis is wrapped up with the conclusion, thoughts and experiences gained throughout the thesis work process.

2. LITERATURE REVIEW

2.1 MACHINE LEARNING

Some problems in Computer Sciences has well defined algorithmic solutions. Using those algorithm, one guarantees to obtain a reasonable solution by conforming necessary conditions. Let's take searching for a certain element in an array as an example. By using the Binary Search algorithm we can identify the location of that certain element if it exists, given that array is in sorted order. It is not the case that for some instances of the problem (for example, with some different array) we can't guarantee to find location or determine existence. This is what makes a solution algorithmic, due to the properties of an algorithm.

On the other hand there are still many problems either currently does not have a algorithmic solution or there are solutions but not practical enough to be widely applied. One such problem is recognizing human faces. We as humans do not even think how we recognize faces, since we just can. But when it comes to program a machine so that it can do that task, scientists face a serious problems. In such a problem, one is generally given an image that contains a face of human. Finding a human face in an image is also a task at a similar difficulty level with just recognizing a face in images with all the same angle, pose, expression, illumination, location, size, etc. Leaving this problem aside, recognizing faces with similar conditions have still have problems. Until now there are some methods involving finding spatial distance between local facial features, such as distance between eyes, mouth width, distance between nose and mouth and so on (Jain & Li, 2005). Even if we can successfully find these metrics without human intervention, we need to come up with some rules to so that we can assign importance to each of those metrics.

Contrary to algorithmic and rule based approaches, when human recognition is considered, rules become implicit and not easily visible all the time. A human, or in general living beings, instead of being given a set of rules, they are rather exposed to many examples of that learning subject. Such agents implicitly find the underlying rules by simple trial & error. If this kind of recognition was available to machines than it could be possible for them to capture very complex relationship between given example and it's associated target mapping. This is what *Machine Learning* is about. According to Mitchell (1997), a computer program is said to learn from experience E with respect to some class of tasks T

and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*. It can also be said that using machine learning one may expect to get a function $\hat{f}$ for which the gap between target variables y (or real function images $f(\mathbf{x})$) and answer of learned agent $\hat{f}(\mathbf{x})$ for each example data $(\mathbf{x}, y)$ is sufficiently small.

2.1.1 Learning Algorithms

There are plenty of useful tools under the umbrella of Machine Learning. Here in this section some of the algorithms used throughout in this thesis will be given.

K-Nearest Neighbour

Suppose we are trying to approximate a continuous function. According to Weierstrass definition for continuous functions $f : X \longrightarrow Y$ in Royden (1988) equation (2.1) basically tells us that images of two close points under a continuous function cannot be too far apart, according to distance metrics d_X and d_Y in spaces X and Y, respectively.

$$\forall \varepsilon \geq 0, \exists \delta \geq 0; \quad \forall \mathbf{x}_1, \mathbf{x}_2; \qquad d_X(\mathbf{x}_1, \mathbf{x}_2) \leq \delta \Longrightarrow d_Y(f(\mathbf{x}_1), f(\mathbf{x}_2)) \leq \varepsilon \tag{2.1}$$

From this one can conclude that target values of nearby points can be used for approximating value at an unknown point which is the intuition behind the *K-Nearest Neighbour* algorithm. The algorithm first finds the nearest k points $\mathcal{N}_k(\mathbf{x})$ in the input space using some metric d, then combines those target values y of points to obtain $f(\mathbf{x})$ for an unseen example x. For example if one wants to approximate a function $\hat{f}$ using some known $f(\mathbf{x})$ for each $(\mathbf{x}, y) \in \mathcal{D}$, which is known to be a *Regression* type problem, he/she could use following equation (2.2) after collecting nearest k points in set $\mathcal{N}_k(\mathbf{x})$.

$$\hat{f}(\mathbf{x}) = \frac{1}{k} \sum_{(\mathbf{x}_i, C_i) \in \mathcal{N}_k(\mathbf{x})} f(\mathbf{x}_i) \tag{2.2}$$

If the problem is finding a function that will assign each input value to a member of a finite set of discrete values, which is called a *Classification* problem, then one can just check the majority class within the nearest neighbours. Following equation (2.3) implements a function with such a behaviour.

$$f(\mathbf{x}) = \underset{C}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, C_i) \in \mathcal{N}_k(\mathbf{x})} 1\{C = C_i\} \tag{2.3}$$

Gaussian Naïve Bayes

This algorithm is based on Bayes formula (2.4). Bayes formula is used to invert conditional probability $X|Y$ to obtain $Y|X$. From machine learning viewpoint, since X is input and Y is associated target class, $X|Y$ is probability distribution of X of some certain class Y and $Y|X$ class probability based on input X.

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)} \tag{2.4}$$

After calculating $Y|X$ for each Y for a given X, one obtains the amount of evidence X gives to support membership to class Y. From now on, what needs to be done to find the associated class Y of X, is to just find the class that maximizes the evidence. Equation (2.5) gives the function that approximates class of a given X.

$$\hat{f}(x) = \underset{c}{\operatorname{argmax}}\, p(c|x) \tag{2.5}$$

But as seen in (2.4), the denominator $p(X)$ in calculation of $p(Y|X)$ is all the same for a fixed X, which makes it irrelevant for maximization. Therefore the classification function can be further reduced to obtain its final simpler form in (2.6).

$$\hat{f}(x) = \operatorname*{argmax}_{c} p(x|c)p(c) \tag{2.6}$$

Yet $p(Y|X)$ still needs to be defined. This is where *learning* occurs. Firstly a model needs to be defined. Although there are many possible probability distribution models, for $p(X|Y)$ Gaussian Distribution (2.7) and for $p(Y)$ Multinomial Distribution (2.8) are used throughout this thesis.

$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}, \qquad \text{where } x \in \mathbb{R} \tag{2.7}$$

$$p(c) = \phi_c, \qquad \text{where } \phi_c = \frac{\sum\limits_{(x_i,\, c_i)\in\mathcal{D}} 1\{c_i = c\}}{|\mathcal{D}|} \tag{2.8}$$

Finding parameters μ and σ is also pretty straightforward. Since a dataset $\mathcal{D}$ is given, one can calculate conditional sample mean (2.9) and conditional sample variance (2.10).

$$\mu_j|c = \frac{\sum\limits_{(x_i,c_i)\in\mathcal{D}} 1\{c_i = c\}\, x_{i_j}}{\sum\limits_{(x_i,c_i)\in\mathcal{D}} 1\{c_i = c\}} \tag{2.9}$$

$$\sigma_j^2|c = \frac{\sum\limits_{(x_i,\, c_i) \in \mathcal{D}} 1\left\{c_i = c\right\}(x_{i_j} - \mu_j|c)^2}{\sum\limits_{(x_i, c_i) \in \mathcal{D}} 1\left\{c_i = c\right\}} \tag{2.10}$$

The method explained so far is enough to implement a Bayes classifier for single feature dataset (where $X \in \mathbb{R}$). On the other hand if *Conditional Independence* assumption (2.11) is done, it can be generalized to many features (where $X \in \mathbb{R}^n$).

$$X \perp Y|Z \quad \Longleftrightarrow \quad p(X,Y|Z) = p(X|Z)p(Y|Z) \tag{2.11}$$

After this observation, one can easily generalize Bayes classifier to have n conditionally independent variables x_i each of which are modeled by Gaussian Distribution given in (2.7).

$$\hat{f}(\mathbf{x}) = \operatorname*{argmax}_c p(c) \prod_{j=1}^{n} p(x_j|c) \tag{2.12}$$

Support Vector Machines

All of the two-class classification algorithms put a separating hypersurface between positive and negative examples. In case of linear classifiers, the hypersurface becomes a $(n+1)$-dimensional hyperplane as in (2.13) where n is the dimensionality of input space, $\mathbf{w}$ is the coefficients of linear separator and b is the intercept term.

$$\hat{f}(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b \tag{2.13}$$

Since points $\mathbf{x}$ satisfying $\hat{f}(\mathbf{x}) = 0$ are located on the plane, we can assign class labels $\{-1, 1\}$ by just checking whether this function outputs a positive number (label -1) or a negative number (label 1).

There may be infinitely many separating hyperplanes found for a given dataset. But when generalization capability concerned, it was shown by Vapnik (1995) that the hyperplane that maximizes margin between closest points to the decision boundary gives the optimal results. The optimization objective given in (2.14) tries to maximize the margin between separating hyperplane and the closest points.

$$\begin{aligned} &\underset{\mathbf{w},b}{\text{maximize}} \quad \left\{ \frac{1}{||\mathbf{w}||} \min_{n} \left[y_n \hat{f}(\mathbf{x}_n) \right] \right\} \\ &\text{subject to} \quad y_n \hat{f}(\mathbf{x}_n) \geq 1, \quad n = 1, \ldots, N \end{aligned} \tag{2.14}$$

Since we know that minimum margin is considered to be 1, then the above maximization problem turns into maximizing the reciprocal of magnitude of $\mathbf{w}$, which is same as minimizing of $||\mathbf{w}||^2$. To solve this optimization problem is *Lagrangian Multipliers* method can be utilized which involves multiplying each of the constraints with a distinct coefficient called lagrangian multiplier and adding them up with the objective function, then finding either minimum or maximum of resulting function which is called *Lagrangian Function* (2.15). One point to be careful about if constraints are not with respect to zero, one side of equality/inequality should be brought to other side to obtain zero on one side. For example $y_n \hat{f}(\mathbf{x}_n) \geq 1$ should be rewritten as $y_n \hat{f}(\mathbf{x}_n) - 1 \geq 0$.

$$L(\mathbf{w}, \mathbf{a}, b) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{n=1}^{N} a_n \left(y_n \hat{f}(\mathbf{x}_n) - 1 \right) \qquad \textbf{(2.15)}$$

To minimize this lagrangian function one just needs to derivate $L(\mathbf{w}, \mathbf{a}, b)$ with respect to $\mathbf{a}$ and b, set both to 0, and solve them. Solving the first equation after deriving L with respect to $\mathbf{a}$ and setting to 0 yields equation (2.16), while the latter procedure yields equation (2.17).

$$\mathbf{w} = \sum_{n=1}^{N} a_n y_n \mathbf{x}_n \qquad \textbf{(2.16)}$$

$$0 = \sum_{n=1}^{N} a_n y_n \qquad \textbf{(2.17)}$$

When $\mathbf{w}$ is given in (2.16) plugged into (2.15), *Dual Lagrangian Function* (2.18) will be obtained and this function only depend on lagrange multipliers.

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m y_n y_m \mathbf{x}_n{}^{\mathrm{T}} \mathbf{x}_m \qquad \textbf{(2.18)}$$

The newly obtained function is again our optimization objective, but this time with respect to lagrange multipliers $\mathbf{a}$, and the constraint is the equation given in (2.17) along with $a_n \geq 0$. The new optimization problem will have the same aim as the initial optimization problem and its counterpart. That's why the first problem is called *Primal Optimization Problem* and the second one is called *Dual Optimization Problem.* Since second one is a tractable one rather than the first, solving it will give the maximal margin separating hyperplane. Again plugging the equation (2.16) into (2.13), one obtains (2.19) which is the equation for separating hyperplane.

$$\hat{f}(\mathbf{x}) = \sum_{n=1}^{N} a_n y_n {\mathbf{x}_n}^{\mathrm{T}} \mathbf{x} + b \tag{2.19}$$

Solving the dual optimization parameters will yield $\mathbf{a}$, but one still needs to find b. For the points where corresponding a_n is non-zero means that those points are closest points to the margin, and $y_n \hat{f}(\mathbf{x}) = 1$ will hold (Bishop, 2006). By plugging $\hat{f}(\mathbf{x})$ given in (2.19) into those constraints with non-zero a_n, and solving for b will give (2.20).

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in N_{\mathcal{S}}} \left(y_n - \sum_{m \in N_{\mathcal{S}}} a_m y_m {\mathbf{x}_n}^{\mathrm{T}} \mathbf{x}_m \right) \tag{2.20}$$

where $N_{\mathcal{S}} = \{n | a_n \neq 0\}$ are called *Support Vectors.*

One can further simplify the decision function for efficiency so that those points that are not support vectors are not included in the decision function, yielding final form of SVM's classification function (2.21).

$$\hat{f}(\mathbf{x}) = \sum_{n \in N_{\mathcal{S}}} a_n y_n {\mathbf{x}_n}^{\mathrm{T}} \mathbf{x} + b \tag{2.21}$$

2.1.2 Ensemble Methods

Instead of using just one classifier, using a group of classifiers may improve classification accuracy significantly. Using a collection, or an ensemble of classifiers in such manner is called *Ensemble Learning* or *Ensemble Methods*. To understand how such a method will perform better consider the case where the input space is 2-D plane, and positive examples lie in a triangular region while negative ones are outside of that triangle. It is clear that a single linear decision boundary cannot give a good classification performance. But if one builds three linear decision boundaries each of which are co-linear with an edge of the triangle, such an ensemble of classifiers would yield the optimum classifier. In such cases it can also be thought as each classifier somehow captures a different aspect of classification objective and contributes in their own way.

A more generalized approach to the above solution could be having K distinct classifiers $\hat{f}_k$ each trained isolated from others then taking the majority of responses of those classifiers as the final classification. This approach is called *committees* (Bishop, 2006) since each classifier giving votes for their deserved classes. Final classification function is given in (2.22).

$$\hat{f}(\mathbf{x}) = \underset{c}{\operatorname{argmax}} \sum_{k=1}^{K} 1 \left\{ \hat{f}_k(\mathbf{x}) = c \right\} \qquad \textbf{(2.22)}$$

One can also use classifiers with different input spaces by mapping inputs to a different space through a function $\phi_k(\cdot)$ and then training different models in each respective space as given in (2.23). Such a method is also employed in this work and will be demonstrated in section 3.2.2.

$$\hat{f}(\mathbf{x}) = \underset{c}{\operatorname{argmax}} \sum_{k=1}^{N} 1 \left\{ \hat{f}_k \left(\phi_k(\mathbf{x}) \right) \right\} \qquad \textbf{(2.23)}$$

There are also various other techniques within ensemble methods such as cascaded classifiers or hierarchy of classifiers (Russell & Norvig, 2009), but they are beyond the scope of this thesis.

2.1.3 Feature Selection

Even though machine learning is applied to variety of types of data, some data may pose problem due to its dimensionality. As the number of dimensionality increases in input space, size of the input space (i.e. number of possible inputs) increases exponentially. This is the so called *Curse of Dimensionality* (Chávez et al., 2001). Since the sample density is reduced number of possible classification functions increases significantly, but only few of them are capable of generalizing well over the unseen data.

One of the biggest challenges in applying machine learning methods to fMRI data lies in the nature of the fMRI dataset, which will further be explained in 2.2. Typically, in fMRI datasets the dimensionality of the feature space is very high (in the order of tens of thousands) whereas the number of examples is very limited (in the order of hundreds or even less). High dimensionality effects both running time and classification performance. From the classification point of view, more dimensionality generally means more irrelevant information, which in turn may mean more noise. From the running time point of view, high dimensionality increases both training and test times of a machine learning algorithms. These issues brings the need to reduce the number of features to a reasonable level. Such techniques are called *Feature Reduction* techniques. One of the well known subbranch of feature reduction techniques is *Feature Selection* where number of features are reduced by just eliminating some certain number of features from data based on some criteria. Following section gives brief information about a class of feature selection techniques that is combined with Genetic Algorithm which will be further detailed in section 2.1.4 in this thesis.

Wrapper Methods

One class of such methods is named as the *wrapper methods* (Zongker & Jain, 1996) which wraps a quality measure and based on that measure incrementally adds (starting with an empty set) or removes (starting with a full set) features that are best (in case of removal, worst) at each iteration. In this work, we utilize genetic algorithms, which can also be used as a wrapper method and which is especially known to be performing well in searching huge spaces (Siedlecki & Sklansky, 1989; Kudo & Sklansky, 2000). As we will explain below in detail, using genetic algorithms we achieve similar levels of classification performance with much less number of features.

2.1.4 Genetic Algorithm

A genetic algorithm is a biologically inspired search algorithm which tries to mimic evolutionary process by the aid of computers to find an optimal solution to a problem (Goldberg, 1989). In a genetic algorithm, possible solutions of a problem are represented by *individuals*. A set of individuals form a *population*. All individuals have the same set of attributes but the values of these attributes may change from individual to individual. Values of the attributes are discrete, most often binary (as in our case).

We will represent a population as a matrix and use the symbol $\mathbf{\Omega}$ to denote it. Rows of the matrix $\mathbf{\Omega}$ represent individuals, i^{th} row is denoted by $\mathbf{\Omega}_i$. If a set $\mathcal{X}$ is used as a subscript, which is represented with a capital letter, then $\mathbf{\Omega}_{\mathcal{X}}$ means the matrix composed of by taking the rows denoted in the index set $\mathcal{X}$ from the matrix $\mathbf{\Omega}$. $\mathbf{\Omega}_{i,j}$ is i^{th} individual's j^{th} attribute. ω, row vector, is used to denote some specific individual. When we use a binary vector ω as a second subscript of a matrix as in $\mathbf{\Omega}_{\mathcal{X},\omega}$, it means we are selecting rows denoted in $\mathcal{X}$, and columns whose indices correspond to a 1 in vector ω. $1\{x\}$ is the indicator function as shown in (2.24).

$$1\{x\} = \begin{cases} 1, & \text{if } x \text{ is } \textbf{true} \\ 0, & \text{otherwise} \end{cases} \tag{2.24}$$

The algorithm starts with a predetermined number of individuals p whose attributes are generated randomly. (From here on we will denote the number of individuals in the populations with a p.) In our case all the attributes are in binary domain, so each attribute is a single bit binary number drawn from Bernoulli distribution with parameter ρ, which is the parameter that determines the proportion of ones in the initial population.

$$\mathbf{\Omega}_{i,j} \sim_{i.i.d} \text{Bernoulli}(\rho) \quad (\text{where } \mathbf{\Omega} \in \{0,1\}^{p\times n}) \tag{2.25}$$

where n is the number of voxels that the genetic algorithm will be working on. This initial randomly generated population is called the first *generation* $\mathbf{\Omega}^{(1)}$.

All the individuals' *fitnesses* are measured through some metric f from the set of binary vectors onto real numbers.

$$f : \{0,1\}^n \longmapsto \mathbb{R}$$

The population is sorted by their rows, so that the matrix conforms the condition given in (2.26).

$$\forall i, j;\; f(\mathbf{\Omega}_i) \geq f(\mathbf{\Omega}_j) \Longleftrightarrow i \leq j \tag{2.26}$$

The individuals with the best fitnesses are retained (some criterion could be selected, e.g. best 30% of the population) and the rest are eliminated from the population. This phase is called *selection*. Let τ be the ratio of survival, then the number of remaining individuals will be $p' = \lfloor p\tau \rfloor$.

Then the algorithm does the *crossover* operation among some of the individuals to create new ones. Crossover is generally done in such a way that either predefined or randomly selected attributes are interchanged (swapped) between two individuals. In case of uniform crossover (which we use in the experiments), attributes are selected randomly by generating a binary string from a Bernoulli distribution with success parameter κ. Some predetermined proportion τ of the population reproduces with crossover to form new individuals which are called *offsprings*. If $\mathcal{P} = \{1, \ldots, p'\}$ is the set of indices of surviving individuals and if $\mathcal{A} \subset \mathcal{P} \times \mathcal{P}$ is the set of randomly selected pairs of individuals then $|\mathcal{A}| = (p - p')/2$ individuals will be subject to crossover.

Finally the algorithm picks randomly a small proportion of individuals for *mutation*, which causes some of their attributes to alter values. Resulting population is called the next generation. From now on the algorithm continues to iterate in the same manner until some stopping criterion is met such as the average or minimum fitness being above some threshold. One then can select the individual with the highest fitness as the result of the search which will be the output of the genetic algorithm as a solution to the current search problem.

2.1.5 Performance Evaluation

In building a classifier there are two important points that needs to be considered. First, a classifier needs to have a good overall quality in *prediction performance*. This means we need to make use of maximum possible number of examples, otherwise we would have poorly estimated the true performance of the trained system. Second, the performance estimate of the classifier should be *reliable*. For a performance estimation to be robust, we need to have enough examples to be used in testing.

The facts given above forbids us to use the *hold-out* methods which involves removing some portion of the examples (typically 30-40%). This would not be an efficient way of using of the available examples since we are using only 70% of the examples for training purposes. It would yield a poorer system in terms of performance compared to the system which could be obtained through training over the whole dataset. Another option might be *k-fold cross-validation*, where data is divided into k pieces and take one of the pieces as test and others as train and repeat this processes for every piece. If k is small ($k = 5$ and $k = 10$ are the most preferred values) then similar to hold-out method 20% or 10% (corresponding to $k = 5$ and $k = 10$, respectively) of the data will not be used in the training phase. On the other hand, using cross-validation in a *leave-one-out* ($k = N$) fashion does the best in both perspectives, since it almost uses all the examples (except one) as training, which results in a better trained system. And since it does many tests (as many as the number of examples) our performance estimates are reliable. One issue to argue against leave-one-out cross-validation is that it is computationally too expensive, which causes the number of trainings required to evaluate the system to scale linearly with the number of examples. But in our case, the number of examples is limited (only 80 examples) which makes the leave-one-out cross-validation the most favourable choice for evaluation.

2.2 fMRI

fMRI stands for *Functional Magnetic Resonance Imaging*. fMRI images contain neural activity associated with a specific region in the brain. These images are often 3-dimensional which can be thought as a stack of 2-dimensional images. fMRI images may consist of a single snapshot or a sequence of snapshots with a certain amount of time interval in between each of the snapshots, called *time resolution*.

2.2.1 Medical Imaging & BOLD

It is thought that blood oxygenation (hemodynamics) is strongly correlated to neural activity. Since activation of a neuron requires energy, glucose consumption (burning of glucose) increases. This brings a demand in oxygen so that glucose could be burnt using it. After 1-2 seconds, this demand is supplied by the increased oxygenated blood cell (hemoglobin) rate. This effect fades approximately in 4-6 seconds, making the oxygenation rate return to the baseline. Since oxygenated hemoglobin cells are paramagnetic as opposed to deoxygenated hemoglobin cells which are diamagnetic, these changes can be observed through magnetic resonance detectors. This type of neural activity measurement is called *Blood Oxygen Level Dependent (BOLD)* response.

Since fMRI devices can capture 3-dimensional images, we can have information about which spatial location has how much neural activity. Spatial locations are separated from each other, and each such unit of volume is called a *volume element*, in short, *voxel*. Typically a voxel is a rectangular prism having sides about 1.5 mm long. This may vary by the fMRI device, setting and experiment. Besides, having a higher resolution (i.e. having smaller voxels) causes time resolution to be greater, and vice versa.

2.2.2 Cognitive State Decoding Task

By the help of development in neural imaging techniques, in recent studies, neuroscientists concluded that neural activation in the brain has strong correlation with the currently thought of the subject which is called *cognitive state*. Since fMRI images are sequences of mental snapshots, such data is being examined to extract information related to the current cognitive state, so that just by using such images predictions against which cognitive state the subject is in can be held. In sections 3.1.2 and 3.1.1 two such examples will be given. In the first one involves a task where it is tried to distinguish between whether the image belongs to a subject who looks at a picture or belongs to a subject who looks at a sentence. The second one is an example of such task where aim is to distinguish between several semantic categories (such as buildings, tools, insects) by taking fMRI snapshot of a subject who looks at a word describing an instance of those semantic categories.

2.2.3 fMRI Specific Feature Selection Methods

Active

As we mentioned above well-known general feature selection methods can be used for dimensionality reduction. However, Mitchell et al. (2004) shows that instead of using some general feature selection method, if we use some problem specific feature selection method we can achieve better results. This problem specific feature selection method proposed in Mitchell et al. (2004) is named as *'active'* method. This method looks at each voxel's activity for each target class and compares it to the activity of that voxel in the fixation period. The intuition is that the more the activation of a voxel during a task (looking at a sentence or a picture) differs from its activation during the fixation period the more informative that voxel is for that task. The method then chooses most active n voxels. Value of n may be determined through cross-validation.

Accuracy

Another feature selection method mentioned in Mitchell et al. (2004) and Pereira et al. (2009) is called the *'accuracy'* method. This method ranks each voxel by its ability to correctly classify an unseen example. This method is less efficient in terms of both computational complexity and classification performance compared to active method, since it requires evaluation of each voxel by some machine learning algorithm. Another downside of this method is that it needs label information unlike active method. This brings in the need of having a separate validation set. On the other hand, while using active method one can just apply the feature selection technique and then continue without having to worry about whether there is an information flow from test set to training set since it doesn't peek at labels of any examples.

2.2.4 Similar Previous Works

Genetic algorithms are widely utilized for feature selection and proven to be working well with many machine learning problems (Kudo & Sklansky, 2000). Hong & Cho (2006) employs a genetic algorithm approach to make problems more tractable and to increase classification performance. Genetic algorithms are also applied to instantaneous cognitive state discrimination problem. Ramirez & Puiggros (2007) applies genetic algorithm in a similar manner as in our work and reports the results evaluated on fMRI images taken while subjects are listening to either high or low pitch tones. Our work diverges from Ramirez & Puiggros (2007) as we use genetic algorithms to search in a reduced feature space and explore different approaches to make use of the final population as will be explained in section 3.2.1. Boehm et al. (2011) also applies genetic algorithms as a feature selection method, although their main concern is to demonstrate the potential of one-class machine learning techniques.

3. MATERIALS & METHODS

3.1 DATASETS

There are a huge variety of data to awaiting to be applied on by machine learning. Fields like astronomy, medical, economy, finance; all have data either in public or private repositories. While some of these fields have scarce amount of data, others have plenty of them. Unfortunately, fMRI field is one of these fields mentioned that has too little amount of data.

3.1.1 StarPlus Dataset

StarPlus is a publicly available dataset that is collected in Carnegie Mellon University (Keller et al., 2001), (Mitchell et al., 2004) where the subjects are volunteering students.

Task: Picture vs. Sentence

It is a result of the experiment called Picture vs. Sentence. In this experiment setting subjects are shown in sequence a picture and a sentence. After viewing the picture and the sentence subjects are asked to answer whether the sentence correctly describes the picture. Pictures are different arrangements of the symbols +, *, and/or $, such as shown below

$$\frac{+}{*}$$

Sentences are descriptions (either correct or incorrect) of the pictures such as “It is true that the plus is below the star.”

An experiment consists of *trials*. In a trial, there are 4 phases: *Picture*, *Fixation*, *Sentence*, *Fixation*. *Picture* and *Sentence* phases may interchange from trial to trial. Each phase takes 4 seconds, and since the time resolution of fMRI device in this experiment is 500 milliseconds, there are 8 fMRI snapshots in each of these phases. For each subject there are a total of 40 trials. In half of those trials pictures come before sentences, in the other half sentences come before pictures. In fixation periods subjects are asked to stare at a blank screen with a dot at its center, so that activation related to that task is decayed and the activation at the next task will be independent from the previous activation.

The dataset is formed by taking the two subsequent 8 seconds interval from each trial, and labelling each of them depending on which stimulus is shown, that is, if the stimulus is a sentence then the interval is labelled as ‘S’ (for sentence) and if the stimulus is a picture then the interval is labelled as ‘P’ (for picture). Therefore two examples are extracted out of each trial. Since the number of trials is 40, we have a total of 80 examples.

The size of a 3-dimensional snapshot is $64 \times 64 \times 8$, but many of the coordinates are empty because they don’t correspond to a neural activity area in the brain. This means that, even though there are $64 \times 64 \times 8 = 32,768$ available coordinates, only a smaller portion of it is actually occupied. Number of occupied coordinates, i.e. voxels, are about 5,000 on average. The coordinates of the occupied voxels differ from subject to subject which makes it difficult to build classifiers across subjects. Although there are methods (Collins et al., 1994), (Mitchell et al., 2004) which bring all subjects to a common space, we don’t employ such methods in this work. Subjects with different number of voxels do not create any problems since the classifiers are built and evaluated on a per-subject basis.

As we mentioned above an example consists of not a single but a sequence of snapshots. Since an example consists of an 8 seconds timespan and since after each 500 milliseconds a snapshot is taken, a single example contains 16 snapshots. This means, there are approximately $16 \times 5,000 = 80,000$ features in the original problem. Speaking in machine learning terms, we have a design matrix of size $80 \times 80,000$ which is considered to be a very high dimensional and sparse dataset.

3.1.2 Science Dataset

This dataset again is a publicly available dataset. (Mitchell et al., 2008)

Task: Semantic Categories

In this experiment subjects are shown example names of certain 12 categories, such as buildings, animals, insects, tools, etc. And they are asked to think about the those semantic category they belong to. The classification task of a learning agent is also similar, they need to classify the cognitive state of the subject, namely the semantic category, by looking at the fMRI image taken during the presentation of the stimulus, which is the example of any category mentioned before.

Data Structure & Format

This data has a similar structure to the former one since it is also a 3-dimensional fMRI image. The only significant difference is the temporal dimension is not used. There are 12 categories. For each category there are 5 examples. These presentations also repeated 6 times. Excluding the temporal dimension, one can conclude that number of examples are $12 \times 5 \times 6 = 360$, and the dimensionality is just the number of voxels, approximately about $10,000$. So in this dataset, each presentation of stimuli has one associated (which is repeated 6 times), this eliminates the temporal dimension in a single data point. Another point to note in this data is the number of examples in this data is not too scarce. So doing k-fold cross validation is also feasible in this dataset compared to the previous one. Such evaluations also done on this data set and results will be presented in corresponding following sections.

3.2 FEATURE SELECTION METHODS

As already described before, this thesis mainly concentrates on improving classification efficiency by focusing on feature selection methods. This section will describe the methods used as feature selection phase in training a learner.

3.2.1 Genetic Algorithm

As we describe before in section 2.1.4 active method sorts voxels according to their activity levels and selects the top most active voxels. This methodology has one assumption which might be an important drawback: it assumes that less active voxels are always less informative. Although activity of a voxel during a task is important, a strict ordering according to activity might not be the optimal solution. For example, there might a set of, say, 50 voxels inside the most active 250 voxels which are more informative than the most active 50 voxels. However, the active method cannot select a set of active voxels which are not contiguous with respect to activity. In this thesis, we want to test this idea, i.e., search for a more informative set of voxels among the ones selected by the active method.

Transforming fMRI Data

Medical images can be of huge sizes and they don't necessarily contain just spatial dimensions, they can also exhibit a temporal dimension. In case of fMRI images, it is more frequent to have an image as a sequence of instantaneous images related to an experiment (like 'trials' in the StarPlus experiment) rather than a single snapshot. To treat such data we need to come up with a way of transforming it so that we have a standard input space for any given machine learning algorithm. One basic approach described in Pereira et al. (2009) is to serialize the 3-dimensional images into a flat vector in some order (for example, the voxel order available in data); then each such vector that belongs to a single time instance that is taken during the presentation of a single stimulus is concatenated one after another which makes a single example. For example, assuming that a single snapshot contains 5000 voxels and a picture phase of a trial takes 16 snapshots then we will have 5000×16 features in a single example. We may refer this form of data as classification-ready form. This approach also used as prerprocessing in each of the method that comes after this.

Adapting Genetic Algorithm

Now we describe how we apply genetic algorithm as a voxel selection method. The solution space consists of binary vectors each of which represents a subset of the available voxels. In a single binary vector the i^{th} component indicate whether the i^{th} voxel will be selected or not (1 if selected, 0 otherwise). In this way we can represent any possible solution to our problem with such binary vectors. If we can define a metric to measure how fit an individual is and define a stopping criterion for the evolution process, these definitions can be plugged into a standard genetic algorithm.

The fitness function has the crucial role here since it is the essential part that makes the genetic algorithm applicable to the voxel selection problem of fMRI data. Fitness function maps an individual (a binary vector as described above) ω to its fitness value. We can consider this function as parameterized by the dataset $\mathcal{D}$ comprising of the fMRI data X (in its classification-ready form as described in section 3.2.1) and the corresponding labels Y, and a learning algorithm Ψ. $\mathcal{D}$ and Ψ are used to evaluate the fitness of input ω by approximating classification accuracy as represented by equation (3.1) using $\mathcal{D}$ when voxels determined by ω is selected and learning algorithm Ψ is used. Procedure for calculating fitness of an individual will be described in depth in algorithm 2.

$$f_{\Psi,\mathcal{D}}(\omega) = 1 - \hat{\varepsilon}_{\mathcal{D}_\omega}(\Psi) \qquad \textbf{(3.1)}$$

Another issue for adapting genetic algorithm for this problem is inherent in the spatio-temporal structure of the data which means that the data spans across some spatial domain (i.e. voxels) as well as a temporal domain (i.e. 16 snapshots taken in 8 seconds). Since the problem is to select voxels we can consider a voxel as a temporal sequence of values bundled together, that is those 16 activation levels at successive snapshots. This means that when we are selecting a voxel, we are either selecting all the activation values in the temporal sequence or dropping them all. Equation (3.2) describes how an individual ω can be converted to the corresponding *expanded* binary vector ω' to implement this kind of feature selection routine.

$$\forall j \in \{1, \ldots, n\}; \forall s \in \{1, \ldots, 16\}; \quad \omega'_{(s-1)*n+j} := \omega_j \qquad \textbf{(3.2)}$$

So this kind of vector will actually be used where voxel selection with an individual ω is

applied to a classification-ready data, that is, first the individual ω will be expanded into a binary vector ω' then columns that correspond to a 1 will be selected.

There some possible methods to use as a convergence condition which will determine where the search will be terminated. We use to the *relative change in average fitness* method to define a convergence criterion. If the relative change is below some predefined threshold then we consider search as converged and no substantial change will happen. Note that the search might have been stuck in a local minimum. In that case results might be poorer with respect to global minimum and selected features may vary from run to run, even though we are searching within the same set of voxels.

With these definitions, one can simply integrate them to a genetic algorithm to create a feature selection method which finds useful voxels for the case of fMRI data which consists of sequence of snapshots. One final issue is to decide on how to utilize the individuals formed in the final generation. There are alternative methods here which will be described in section 3.2.1.

Combining Individuals in Final Generation

After the genetic algorithm finishes we have a set of individuals in the final population. At this point there might be different methods which can be applied to utilize this final set of individuals. Below, we describe the different methods we use.

Best:

This is the most commonly used method. The selection phase of the genetic algorithm already sorts the individuals in the population by their fitnesses, so one simply needs to pick the first individual from that order as indicated by (3.3).

$$\omega^* := \mathbf{\Omega}_1^{(t)} \tag{3.3}$$

Intersection:

Another possible way to exploit the final population is to consider the best performing individuals (M individuals) and select those voxels which appear in all of them. Since we can regard each individual as a set of voxels, this method basically takes the intersection of those sets. Equation (3.4) realizes this by simply multiplying corresponding bits in each individual.

$$\forall j;\ 1 \leq j \leq n, \quad \omega_j^* := \prod_{k=1}^{M} \mathbf{\Omega}_{k,j}^{(t)} \tag{3.4}$$

Average:

This method can be considered as a generalization over the intersection method. Equation (3.5) explains the procedure used in this method. It takes best performing M individuals, treats them as vectors, and averages each of the attributes (which are either 1 or 0) over those individuals, and then thresholds (θ) them to come up with a new binary vector. Intersection is a special case of this, where $\theta = 1$.

$$\forall j;\ 1 \leq j \leq n, \quad \omega_j^* := 1\{\frac{1}{M}\sum_{k=1}^{M} \mathbf{\Omega}_{k,j}^{(t)} \geq \theta\} \tag{3.5}$$

Spatio-Temporal Feature Selection

Up until now we treat a voxel as a single, integral, standalone entity. Nevertheless, the data has 16 activation values for each voxel that correspond to activations at different times, namely, snapshots. Instead of just searching in the spatial dimension, one could also use the temporal dimension, which is also available in the data as snapshots. In that case, one can simply extend the definition of individual in our current genetic algorithm problem instance as follows: Each bit of an individual now corresponds to the activation of a voxel at a single snapshot. Actually we have referred to this in section 3.2.1 as expanded form. Therefore the number of bits in those vectors became $n \times 16$. This is a simple extension that can easily be adapted and applied to the methods we use in this thesis. What one needs to do is to change two steps: First, population must be generated using the expanded form by modifying (2.25) accordingly (using $n \times 16$ instead of n should be sufficient) so that we can select both voxels and time instants (snapshots). Second, one needs to use the output of the genetic algorithm ω^* directly (select column j, if and only if $\omega_j^* = 1$), instead of first expanding to cover all snapshots equally, as explained in equation (3.2).

We also conducted experiments for this case, too. But since the results were not satisfying we do not include them in this thesis.

3.2.2 Random Subspaces

Random Subspaces is a kind of ensemble learning methods where various learners of the ensemble have all the same learning algorithm but different set of features, each set are established by randomly selecting among available features (**?**).

Stacking & Voting

When using an ensemble of classifiers simplest technique is to find majority voting of the classifiers. This is called *Voting*.

But instead of simply using the classifiers in a discrete manner (either class 1 or class 2) for each classifier, one can better utilize the classification results of individual classifiers if they are able to produce class posterior probabilities given the inputs. Even if that is not the case, one could have an estimate of trust for each of those probabilities by estimating validation error. If one weights all the outcomes from classifiers inversely with the classification error and averages it, a better classifier might be built. This type of aggregation of ensemble of classifiers is called *Stacking*.

Locally Clustered Random Selection

In locally Clustered Random Selection, rather than all features having equal chance to be chosen, there is simple rule to make them look spatially clustered. The rule is, if a feature is selected, then there is a high chance that nearby features are also selected. One way to achieve this rule is to determine some focal points, and randomly select features using a 2-dimensional Gaussian random variables having those focal points as their means.

The motivation behind this is, it is thought that the information in brain that are related to each other are processed in spatially close regions. This is why this method thought to be useful, and experimented on.

3.2.3 Functional Dependency Graph

Brain can be considered as a huge graph of neural connectivity. So modelling it as a graph makes perfect sense. In functional dependency graph, one analyses the regions in brain by how correlated they are with respect to neural activity within the some temporal course.

To implement this type of feature selection method, one first needs to define regions which are disjoint subsets of voxels that close to each other spatially. For each temporal activation instance for a region, an average is taken over activation values of voxels within each voxel set. Now we have $\#regions \times 16 \times \#examples$ values. One can now regard regions as random variables and calculate covariance among each of those variables. This operation needs to be done per class basis, so this yields two covariance matrices, and this completes the training phase.

When a new instance has come, one needs to apply similar operation over the newly acquired data, and obtain the covariance matrix. By matching the obtained matrix with those two matrices obtained from training phase, one can determine the predicted class of the test instance.

Graph matching is a uneasy topic. For that reason, a simple Jaccard matching is applied over the edges of the graphs. Let A be the set of edges in the first matrix, and B be the set of edges in the second matrix, then the Jaccard similarity can be found by equation (3.6).

$$J(A,B) = \frac{A \cap B}{A \cup B} \tag{3.6}$$

For each class, whichever has the most Jaccard value is used as the predicted class.

3.2.4 Other Methods

Some other simpler methods are also tried on some datasets. These trials were not directly aiming to increase performance of the classifier. Rather they aimed to have a better understanding of the data in terms of its temporal and spatial nature.

Mainly two methods are tried: Spatial averaging and temporal averaging.

Spatial Averaging

Since the data is distributed in a 3-dimensional space, one can easily find the voxels that lie in the 3-lattice with some size s. Partitioning the space with such lattices and averaging all the activations inside it, on can reduce the dimensionality exponential in s.

Temporal Averaging

The data have the temporal dimension, but if one wonders whether temporal activation course is really relevant, following method can be employed. Instead of using all the temporal activation instances of a voxel, one can average them out to get a single activation per voxel.

4. EXPERIMENTAL RESULTS & EVALUATIONS

In this chapter the experiments done will be explained and the results will be given.

4.1 GENETIC ALGORITHM

Algorithm 1 describes the method how the experiments are conducted for evaluating the performance of genetic algorithm and the active method.

```
Algorithm 1: Active vs. Genetic Algorithm Comparison
Input  : $X, Y, n, \Psi$
Output: $\varepsilon^{(GA)}, \varepsilon^{(Act)}$
$X \leftarrow$ SelectActiveVoxels($X$, $n$)
for $i \leftarrow 1$ to $N$ do
    Tst $\leftarrow \{i\}$ ;
    Trn $\leftarrow \{1, \dots, N\} \setminus$ Tst ;
    $\omega^* \leftarrow$ GeneticAlgorithm($X_{\text{Trn}}$, $Y_{\text{Trn}}$, $\Psi$) ;
    $\Theta^{(\text{GA})} \leftarrow$ Train$_\Psi$($X_{\text{Trn}, \omega^*}$, $Y_{\text{Trn}}$) ;
    $\Theta^{(\text{Act})} \leftarrow$ Train$_\Psi$($X_{\text{Trn}}$, $Y_{\text{Trn}}$);
    $\hat{Y}^{(\text{GA})} \leftarrow$ Predict$_\Psi$($\Theta^{(\text{GA})}$, $X_{\text{Tst}, \omega^*}$) ;
    $\hat{Y}^{(\text{Act})} \leftarrow$ Predict$_\Psi$($\Theta^{(\text{Act})}$, $X_{\text{Tst}}$) ;
    ${\epsilon^{(GA)}}_i \leftarrow 1\{\hat{Y}^{(\text{GA})} \neq Y_{\text{Tst}}\}$ ;
    ${\epsilon^{(Act)}}_i \leftarrow 1\{\hat{Y}^{(\text{Act})} \neq Y_{\text{Tst}}\}$
$\varepsilon^{(GA)} \leftarrow 1/N \sum_{i=1}^{N} {\epsilon^{(GA)}}_i$ ;
$\varepsilon^{(Act)} \leftarrow 1/N \sum_{i=1}^{N} {\epsilon^{(Act)}}_i$ ;
```

Algorithm 1 takes an fMRI data X with labels Y and takes the number of initial voxels n. First of all, line 1 applies active voxel selection to select n voxels. This is done prior to anything else since genetic algorithm will search for a better voxel subset within that initial active voxel set. This is also legitimate with respect to 'peeking', since active voxel selection method does not make use of the labels, there will be no flow from target variables. Algorithm then continues with a loop over the examples of the input dataset. This is done for the sake of leave-one-out cross-validation, so at each iteration one example is held out as test example and the rest is retained as training-validation purposes. At line 5 training-validation part is handed over to the function named $GeneticAlgorithm$ (appears in line 5) which is the modified genetic algorithm as described in section 3.2.1.

Lines 6 and 7 estimates the model parameters $\mathbf{\Theta}$ of learning algorithm Ψ using training data (X_{Trn,ω^*} for genetic algorithm, X_{Trn} for active method and Y_{Trn} for both). $X_{\mathcal{X},\omega}$ means that we first select the rows in set $\mathcal{X}$, then select the columns j for which $\omega_j = 1$. Lines 8–9 makes the predictions $\hat{Y}$ for the two methods, using the parameters $\mathbf{\Theta}$ that were just estimated. The predictions checked against expected labels Y_{Tst} and comparison results are stored in $\epsilon^{(\text{GA})}$ and $\epsilon^{(\text{Act})}$, respectively. These binary values are averaged over all examples and returned as $\varepsilon^{(\text{Act})}$ and $\varepsilon^{(\text{GA})}$ which are estimated error rates of the algorithm using active voxels and voxels found by the genetic algorithm within the active voxels, respectively.

To determine the fittest individual we need to be able to compare different sets of voxels that genetic algorithm gave us. For this purpose we come up with a cross-validation scheme for determining fitness of an individual as given in algorithm 2. In each iteration of genetic algorithm, when fitness of an individual is needed, algorithm 2 is invoked. What it does is very similar to algorithm 1 because both perform cross-validation. It basically holds out one example for validation at a time then tries to predict it correctly using the voxels imposed by input ω. Since we need to estimate a fitness, and since the more the fitness the better the system would likely to be, we can simply use the correct classification rate for it.

```
Algorithm 2: CalculateFitness
input  : $\omega$, $\Psi$, $X, Y$
output: $\varphi$
for $i \leftarrow 1$ to $N$ do
    $\text{Val} \leftarrow \{i\}$ ;
    $\text{Trn} \leftarrow \{1, \dots, N\} \setminus \text{Val}$;
    $\mathbf{\Theta} \leftarrow \text{Train}_\Psi\ (X_{\text{Trn},\ \omega}, Y_{\text{Trn}})$ ;
    $\hat{Y} \leftarrow \text{Predict}_\Psi\ (\mathbf{\Theta}, X_{\text{Val},\ \omega})$ ;
    $\epsilon_i \leftarrow 1\{\hat{Y} \neq Y_{\text{Val}}\}$ ;
$\varphi \leftarrow 1 - 1/N \sum_{i=1}^{N} \epsilon_i$
return $\varphi$
```

Experimental Results

We made our evaluations on a per-subject basis. In the graphs and tables, if no subject is mentioned, then it means those values are obtained through averaging over subjects. Subjects are mentioned by their subject numbers (e.g. 4748, 5680, etc.).

The experiments are done in a fashion that one can observe how much improvement is done by genetic algorithm approach over active voxel selection method. To achieve this, we first select subsets of voxels of different sizes by the active method, and then apply genetic algorithm to search within those subsets of voxels in order to find a reduced set of voxels whose performance is at least the same as that of the voxels selected by the active method.

We evaluate several learning algorithms that are known to be performing better on such high dimensional and very sparse data. We use Gaussian Naïve Bayes (GNB), Support Vector Machines (SVM) with linear kernel, and k-Nearest Neighbours (kNN) algorithms for measuring improvement over the active method. We use the same algorithms for evaluating the fitness of the individuals. For example, if we use kNN for evaluating the performance of genetic algorithm then we also use kNN to determine the fitness of the individuals in every iteration of the genetic algorithm.

Table 4.1: The values of the parameters used in the experiments.

$p = 50$	Size of the population
$\rho = 0.4$	Survival rate
$\tau = 0.5$	Initial rate of the ratio of 1s in an individual
$\kappa = 0.5$	Uniform crossover exchange ratio
$\mu = 0.02$	Mutation rate
$\theta = 0.3$	Threshold for the average method
$M = 10$	# best individuals to use
$n \in \{25k \mid k \leq 20, k \in \mathbb{N}\}$	# initial voxels selected by active method

The values of the parameters that are used in the experiments are given in Table 4.1. Our method uses a genetic algorithm approach to search in the space of voxels selected by the active method. As discussed earlier, this method is a promising way to find a subset of voxels which perform better then the voxels selected by active method. Figure 4.1 shows the performance of kNN, GNB, and SVM algorithms for different feature selection methods. The x-axis in these figures shows the number of voxels used by these algorithms to achieve the specified classification performance. The graphs shown in Figure 4.1 are smoothed by averaging the four surrounding points in order to better show the trends in the results. As can be seen the best classification performance is achieved by using SVM (about $\sim$90%). The classification performance of kNN and GNB are comparable (at about $\sim$80%). (These results are similar to those given by Mitchell et al. (2004)). A general pattern can be seen in these figures: all the curves have a, roughly, bowl shape. This means that the performance of the algorithms achieve their best values for a specific number of voxels (roughly between 150 and 300, depending on the learning algorithm) and gets worse when we increase or decrease the number of voxels used.

Intersection method for combining individuals within a population performs worst in all cases. This result is consistent with the findings mentioned in Pereira et al. (2009), where different parts of the training sets yield pretty much different sets of voxels when some voxel selection method like active method is applied. Since the intersection method intersects the voxel sets selected by top M individuals, there may be no common voxels in those sets, which results in a random guess. Such cases seem to increase the error rate significantly. On the other hand when the intersection set is nonempty its size is small and the voxels in those small intersection sets still has the potential of predicting the target class. That is why this method can still have some classification performance using a very small number of voxels.

We can see that the average and best methods let the algorithms to achieve similar classification performances by using substantially less number of voxels. Figure 4.1a, Figure 4.1b, and Figure 4.1c show that the GA based feature selection methods help the algorithms (kNN and GNB) to achieve similar classification performances using much less number of voxels. For example, in 5NN and GNB the best method achieves its low error rates using only about 70 voxels. The algorithm which gives the best classification performance using the active method is SVM. Figure 4.1d shows the results when the learning algorithm used is SVM. Here, we can see that the average method gives similar classification performance to the active method. However, it can bee seen that the average method achieves this performance using much less number of voxels. While SVM using active method achieves its best performance using approximately 300 voxels, SVM using the average method achieves similar accuracy levels using 150-200 voxels.

Table 4.2 reports the results in a per-subject manner. It shows the minimum classification error along with the number of voxels used for each subject and method. The last row in each table averages all those minimum values. Even though the classification performances of the algorithms for different subjects show different levels of errors, it is clearly

Figure 4.1: Comparison of classification performances with respect to the number of voxels selected of various learning algorithms using active method and GA based feature selection methods.

Figure 4.1a 3NN

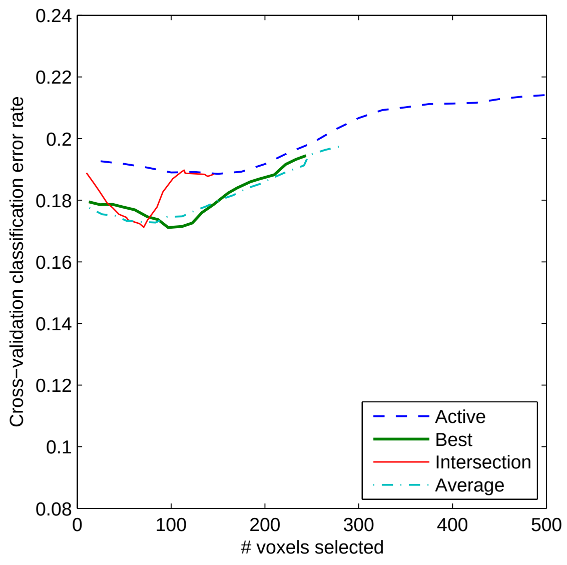


Figure 4.1b 5NN

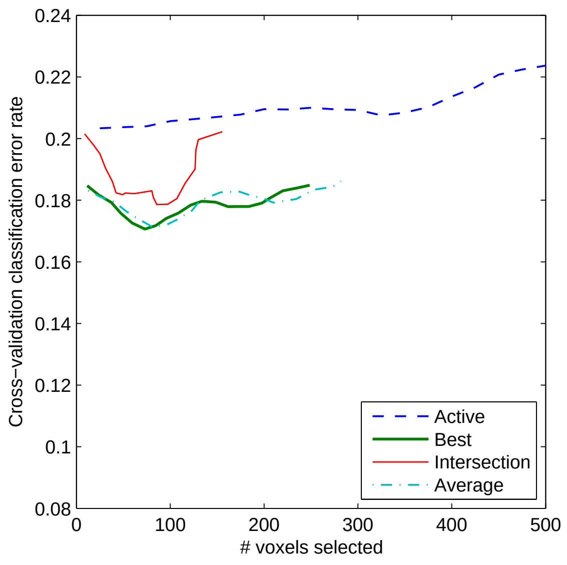


Figure 4.1c GNB

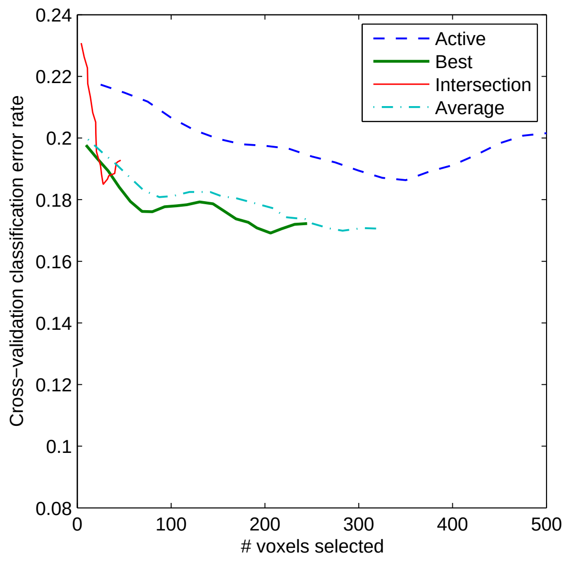
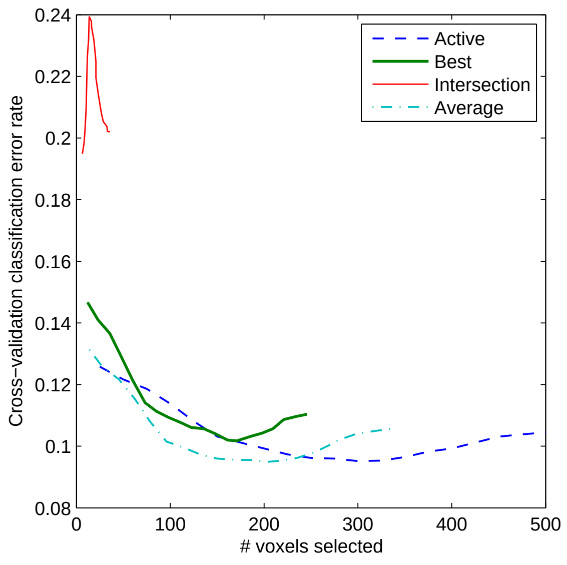


Figure 4.1d SVM



seen that in general the GA based feature selection methods let the algorithms to achieve similar levels of performance while using much less number of voxels.

4.2 RANDOM SUBSPACES

As explained in section 3.2.2, the methods available here can be considered as ensemble methods. Experimental results obtained using those methods will be given here.

Stacking & Voting

This section gives the experimental results obtained from stacking and voting methods. To do this experiment, first some number of voxels selected by Active method. Then as mentioned in section 3.2.2 many random subsets are selected from (for this experiment $L = 10$ subsets, therefore number of classifiers) available voxel set. Number of voxels in a single set is held constant $M = 50$. Leave-one out cross-validation is run 50 times, and results are averaged over them.

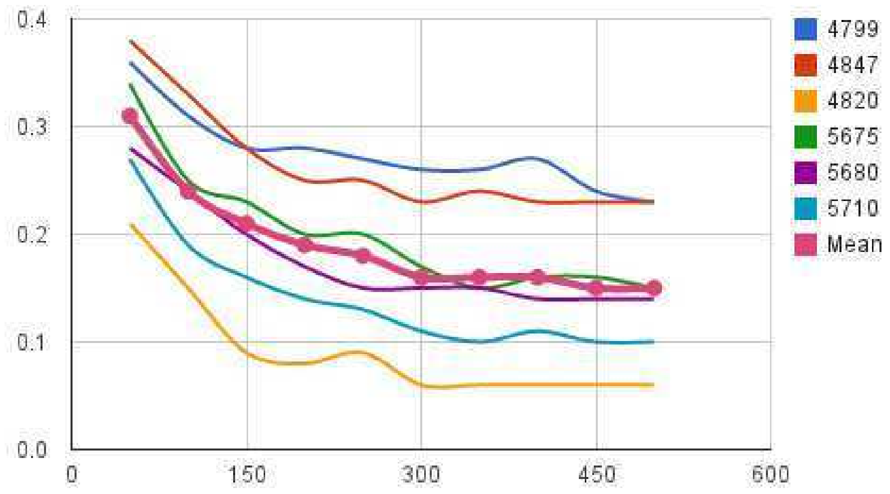
There is also another column in the results that has not been mentioned yet. One could also use squares of the weights when using the stacking method. This is referred to as Stacking^2 in the tables. This method is tried because it could assign much lower trusts for low accuracy classifiers so that the overall classification performance could benefit from this. Table 4.3 shows the experimental results with different base set sizes obtained from Active method.

Locally Clustered Random Selection

Throughout the experiments an ensemble of $L = 10$ classifiers are used and held fixed.

In Figure 4.2, one can observe how error falls as the number of voxels randomly selected M for each classifier is increased. For small number of voxels, possibly due to lack of information available, classifiers cannot do well. As the number increases error falls since the available information increases. But after some point it saturates at $0,15$. 300 voxels for each classifier looks like it is optimal from both classification and running-time efficiency.

Figure 4.2: Classification errors per subjects vs. local set size.



4.3 FUNCTIONAL DEPENDENCY GRAPH

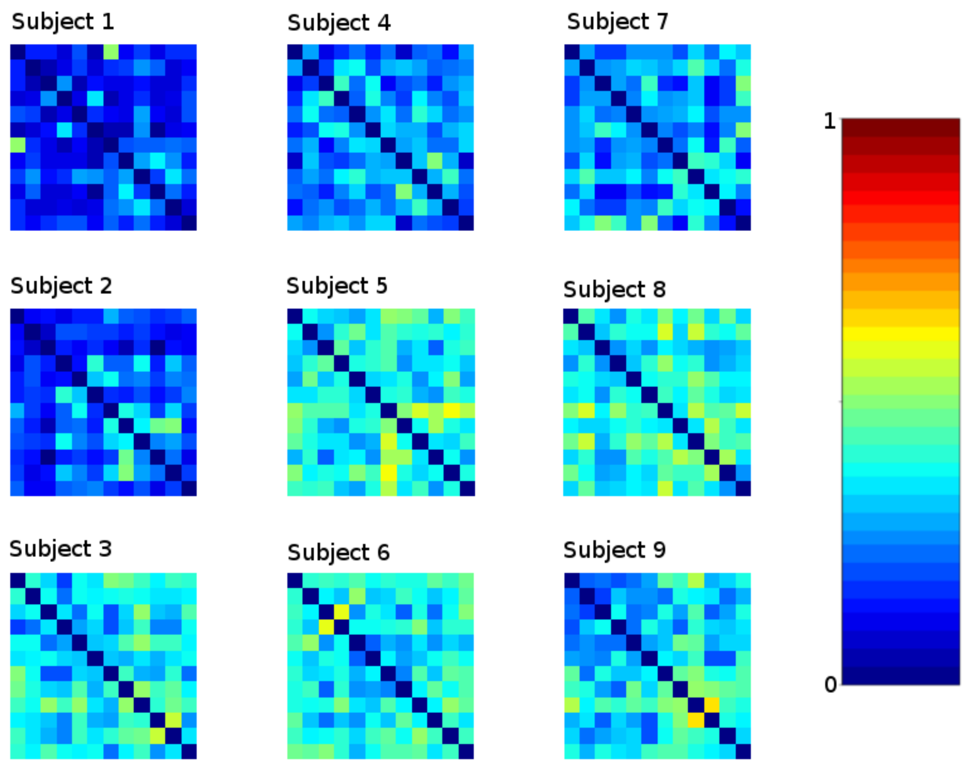
Up until now the StarPlus dataset has been used used in experiments. In this one, Science dataset is used. First of all, base error rates needs to be mentioned. Since this dataset contains 12 class labels, the expected classification error of a random classifier is $1/12 = 0,917$. To better understand the separability of the data, GNB is run for 12 classes over the data. Table 4.4 shows how well gnb does on this dataset with both leave-one-out and 6-fold cross validation schemas.

The mean errors is lower than the expected error of the random classifier but it is still high. This shows how hard it is to naively separate this dataset.

Table 4.5 shows the accuracy of classifiers when they are fed the functional dependency graphs extracted from input data. The classes are as follows:

Figure 4.3 visualizes this confusion matrix with a color map, for each subject.

Figure 4.3: Pairwise confusion matrices visualization for SVM classifier.



4.4 OTHER METHODS

In the remaining methods simple image-processing like manipulations are done to observe how classification performance will be affected.

Table 4.6 shows the results of averaging activation values of a voxel for each time instance. It also compares genetic algorithm with the active method under this basis. n represents the number of voxels selected by active method. GA method again starts with the active voxels selected.

Table 4.7 shows the results of neighbouring voxels averaging method. r is the radius which defines a cube with sides $2r + 1$ centered at the voxels. The experiment averages all the non-overlapping cubes and assigns single voxels to them.

Table 4.2: Tables showing minimum % errors achieved per subject along with the number of voxels used for that % error level.

Table 4.2a 3NN

Subj.	**Active**		**GA Best**		**GA Intrsct.**		**GA Avg.**	
	Error	#voxels	Error	#voxels	Error	#voxels	Error	#voxels
4799	36.36%	75	29.95%	37	31.50%	31	28.57%	47
4820	17.58%	100	17.49%	49	16.21%	75	14.93%	147
4847	2.38%	350	1.19%	173	1.19%	30	2.47%	105
5675	10.26%	50	7.69%	23	6.41%	78	10.26%	137
5680	9.98%	150	8.61%	74	9.98%	28	9.89%	168
5710	6.23%	100	6.23%	48	7.51%	14	6.41%	43
Mean	13.80%	138	11.86%	67	12.13%	42	12.09%	108

Table 4.2b 5NN

Subj.	**Active**		**GA Best**		**GA Intrsct.**		**GA Avg.**	
	Error	#voxels	Error	#voxels	Error	#voxels	Error	#voxels
4799	32.51%	375	30.22%	180	27.38%	47	27.38%	188
4820	22.53%	175	20.97%	86	21.34%	57	19.78%	188
4847	3.66%	400	3.75%	200	0.00%	41	3.66%	147
5675	12.64%	175	4.95%	80	8.70%	34	7.60%	190
5680	11.26%	100	12.36%	48	12.45%	26	12.36%	44
5710	4.95%	50	8.52%	26	7.33%	131	4.85%	41
Mean	14.59%	213	13.46%	103	12.87%	56	12.61%	133

Table 4.2c GNB

Subj.	**Active**		**GA Best**		**GA Intrsct.**		**GA Avg.**	
	Error	#voxels	Error	#voxels	Error	#voxels	Error	#voxels
4799	31.25%	400	28.75%	189	27.50%	15	28.75%	195
4820	18.75%	450	16.25%	215	16.25%	58	18.75%	169
4847	5.00%	350	2.50%	167	1.25%	11	1.25%	199
5675	12.50%	350	10.00%	163	8.75%	28	11.25%	151
5680	13.75%	25	12.50%	10	12.50%	6	12.50%	287
5710	10.00%	75	7.50%	34	10.00%	12	8.75%	160
Mean	15.21%	275	12.92%	130	12.71%	22	13.54%	193

Table 4.2d SVM

Subj.	**Active**		**GA Best**		**GA Intrsct.**		**GA Avg.**	
	Error	#voxels	Error	#voxels	Error	#voxels	Error	#voxels
4799	8.75%	150	6.25%	74	10.00%	22	7.50%	257
4820	15.00%	300	16.25%	146	21.25%	54	15.00%	256
4847	1.25%	150	0.00%	77	12.50%	2	1.25%	29
5675	3.75%	125	5.00%	61	11.25%	8	3.75%	291
5680	7.50%	250	6.25%	121	11.25%	20	6.25%	44
5710	7.50%	25	5.00%	11	8.75%	6	6.25%	28
Mean	7.29%	167	6.46%	81	12.50%	19	6.67%	151

Table 4.3: Classification errors of Voting and Stacking using $n = 250$, $n = 500$, $n = All$ initial voxels selected by Active method.

Table 4.3a 250 Active Voxels

Subject	**Voting**	**Stacking**	**Stacking^2**	**Single SVM**
4799	0,2625	0,1375	0,0875	0,0875
4847	0,3125	0,1125	0,0750	0,1875
4820	0,1375	0,2375	0,2000	0,0500
5675	0,3500	0,2375	0,1125	0,0250
5680	0,2625	0,1875	0,0625	0,0875
5710	0,1875	0,1750	0,1250	0,0750
Average	0,2521	0,1812	0,1104	0,0854

Table 4.3b 1000 Active Voxels

Subject	**Stacking**	**Stacking^2**
4799	0,1250	0,0875
4847	0,1000	0,0875
4820	0,3000	0,2875
5675	0,2875	0,1125
5680	0,2250	0,1375
5710	0,1375	0,1000
Average	0,1958	0,1354

Table 4.3c No Feature Selection

Subject	Voting	Stacking	Single SVM
4799	0,4625	0,2250	0,3750
4847	0,3625	0,2375	0,3125
4820	0,2625	0,3000	0,1125
5675	0,4500	0,3375	0,2750
5680	0,3125	0,2750	0,2250
5710	0,3375	0,2625	0,1750
Average	0,3646	0,2729	0,2458

Table 4.4: GNB error levels on Science dataset without .

	6-fold CV	**Leave-one-out CV**
Subject 1	0,753	0,756
Subject 2	0,858	0,856
Subject 3	0,897	0,864
Subject 4	0,753	0,747
Subject 5	0,897	0,886
Subject 6	0,911	0,903
Subject 7	0,908	0,908
Subject 8	0,886	0,881
Subject 9	0,900	0,881
Mean	0,863	0,853

Table 4.5: Average of pairwise confusion matrices over subjects.

Table 4.5a GNB

Versus	$\mathcal{C}$ **1**	$\mathcal{C}$ **2**	$\mathcal{C}$ **3**	$\mathcal{C}$ **4**	$\mathcal{C}$ **5**	$\mathcal{C}$ **6**	$\mathcal{C}$ **7**	$\mathcal{C}$ **8**	$\mathcal{C}$ **9**	$\mathcal{C}$ **10**	$\mathcal{C}$ **11**
$\mathcal{C}$ **0**	0,44	0,43	0,44	0,44	0,47	0,46	0,44	0,48	0,44	0,45	0,44
$\mathcal{C}$ **1**		0,36	0,41	0,42	0,41	0,39	0,42	0,44	0,47	0,40	0,42
$\mathcal{C}$ **2**			0,42	0,38	0,46	0,37	0,36	0,41	0,35	0,38	0,44
$\mathcal{C}$ **3**				0,39	0,45	0,41	0,39	0,46	0,39	0,38	0,48
$\mathcal{C}$ **4**					0,44	0,43	0,43	0,44	0,45	0,44	0,46
$\mathcal{C}$ **5**						0,37	0,40	0,43	0,42	0,42	0,45
$\mathcal{C}$ **6**							0,41	0,47	0,40	0,47	0,43
$\mathcal{C}$ **7**								0,45	0,53	0,44	0,42
$\mathcal{C}$ **8**									0,46	0,43	0,48
$\mathcal{C}$ **9**										0,46	0,41
$\mathcal{C}$ **10**											0,40

Table 4.5b SVM

Versus	$\mathcal{C}$ **1**	$\mathcal{C}$ **2**	$\mathcal{C}$ **3**	$\mathcal{C}$ **4**	$\mathcal{C}$ **5**	$\mathcal{C}$ **6**	$\mathcal{C}$ **7**	$\mathcal{C}$ **8**	$\mathcal{C}$ **9**	$\mathcal{C}$ **10**	$\mathcal{C}$ **11**
$\mathcal{C}$ **0**	0,30	0,25	0,30	0,29	0,31	0,41	0,38	0,36	0,28	0,34	0,34
$\mathcal{C}$ **1**		0,35	0,29	0,35	0,36	0,32	0,33	0,36	0,31	0,34	0,32
$\mathcal{C}$ **2**			0,36	0,27	0,31	0,25	0,27	0,32	0,29	0,34	0,35
$\mathcal{C}$ **3**				0,26	0,39	0,35	0,26	0,37	0,32	0,27	0,33
$\mathcal{C}$ **4**					0,28	0,32	0,32	0,38	0,29	0,39	0,29
$\mathcal{C}$ **5**						0,35	0,32	0,32	0,24	0,29	0,35
$\mathcal{C}$ **6**							0,35	0,40	0,31	0,40	0,36
$\mathcal{C}$ **7**								0,39	0,47	0,39	0,34
$\mathcal{C}$ **8**									0,37	0,41	0,36
$\mathcal{C}$ **9**										0,38	0,26
$\mathcal{C}$ **10**											0,35

$\mathcal{C}$ **0** = Animal, $\mathcal{C}$ **1** = Body part, $\mathcal{C}$ **2** = Building, $\mathcal{C}$ **3** = Build part, $\mathcal{C}$ **4** = Clothing, $\mathcal{C}$ **5** = Furniture,
$\mathcal{C}$ **6** = Insect, $\mathcal{C}$ **7** = Kitchen, $\mathcal{C}$ **8** = Man made, $\mathcal{C}$ **8** = Tool,
$\mathcal{C}$ **10** = Vegetable, $\mathcal{C}$ **11** = Vehicle

Table 4.6: Comparison of Best GA individual vs Active method using temporally averaged data.

Table 4.6a $n = 50$

Subj.	Active	GA Best
4799	0,4375	0,4500
4820	0,3375	0,3750
4847	0,1875	0,1500
5675	0,2625	0,2500
5680	0,3000	0,3375
5710	0,1875	0,1375
Mean	0,2854	0,2833

Table 4.6b $n = 100$

Subj.	Active	GA Best
4799	0,3500	0,2750
4820	0,3625	0,3500
4847	0,1500	0,1125
5675	0,1500	0,1500
5680	0,2750	0,2375
5710	0,1125	0,1000
Mean	0,2333	0,2042

Table 4.6c $n = 150$

Subj.	Active	GA Best
4799	0,4750	0,4000
4820	0,2750	0,2375
4847	0,0750	0,0500
5675	0,1125	0,1000
5680	0,3000	0,2875
5710	0,1375	0,1000
Mean	0,2292	0,1958

Table 4.6d $n = 240$

Subj.	Active	GA Best
4799	0,4125	0,4125
4820	0,3000	0,2625
4847	0,0875	0,1375
5675	0,1625	0,1500
5680	0,2875	0,2750
5710	0,1375	0,1125
Mean	0,2313	0,2250

Table 4.7: Classification errors of GA spatially (with radius r) averaged data with n initial number of voxels is selected.

Table 4.7a $n = 240$

	1NN	**3NN**	**5NN**	**9NN**	**SVM**	**GNB**	**Mean**
Original Data	0,1979	0,1792	0,1729	0,2104	0,0854	0,1813	0,1712
Averaged $(r = 1)$	0,2021	0,1812	0,1938	0,2250	0,0854	0,1979	0,1809
Averaged $(r = 2)$	0,1979	0,1917	0,1917	0,2250	0,0938	0,1958	0,1827

Table 4.7b $n = 500$

	1NN	**3NN**	**5NN**	**9NN**	**SVM**	**GNB**	**Mean**
Original Data	0,2083	0,2104	0,2125	0,2187	0,1063	0,2063	0,1938
Averaged $(r = 1)$	0,2167	0,2167	0,2104	0,2188	0,0958	0,2125	0,1952
Averaged $(r = 2)$	0,2104	0,2208	0,2125	0,2292	0,1042	0,2104	0,1979

Table 4.7c $n = 1000$

	1NN	**3NN**	**5NN**	**9NN**	**SVM**	**GNB**	**Mean**
Original Data	0,2708	0,2458	0,2396	0,2458	0,1438	0,2146	0,2267
Averaged $(r = 1)$	0,2542	0,2542	0,2521	0,2333	0,1271	0,2063	0,2212
Averaged $(r = 2)$	0,2521	0,2500	0,2417	0,2313	0,1271	0,2021	0,2174

5. CONCLUSION

Active method is one of the best performing voxel selection methods in the fMRI domain. In this method voxels are sorted with respect to their activity values and the most active voxels are selected. Although the level of activity of a voxel is an important indication of its relevance for the task being performed, the level of activation might not be the only parameter. So, in order to test this idea, we used active method to select sets of voxels of different sizes and used genetic algorithms to search for better voxels in this reduced space.

The results show that it is possible to further reduce the number of voxels substantially and still achieve comparable classification performances. Specifically, the empirical results show that the best classification performance is achieved using approximately 240 voxels for this fMRI dataset. We can achieve similar classification performances using much less number of voxels (between 100-150 voxels).

Using less number of voxels might be useful for achieving fast classification performances for time-critical tasks. Also it might lead to a better understanding of which voxels are more important for the classification task at hand. Researchers can focus on these small but informative subsets of voxels.

Our genetic algorithm based technique is not specific to this dataset or to the active method. It can be applied to other datasets and to other feature selection methods which produces an ordering of features.

REFERENCES

Books

Bishop, C. M., 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. New Jersey: Springer-Verlag, Inc.

Goldberg, D. E., 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.

Jain, A. K. & Li, S. Z., 2005. *Handbook of Face Recognition*. Secaucus: Springer-Verlag, Inc.

Mitchell, T. M., 1997. *Machine Learning*. New York: McGraw-Hill.

Royden, H. L., 1988. *Real analysis*. New York: Macmillan.

Russell, S. J. & Norvig, P., 2009. *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Vapnik, V. N., 1995. *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, Inc.

Periodicals

Boehm, O., Hardoon, D., & Manevitz, L., 2011. Classifying cognitive states of brain activity via one-class neural networks with feature selection by genetic algorithms. *International Journal of Machine Learning and Cybernetics* **2**, pp. 125–134.

Chávez, E., Navarro, G., Baeza-Yates, R., & Marroquín, J. L., 2001. Searching in metric spaces. *ACM Comput. Surv.* **33(3)**, pp. 273–321.

Collins, D., Neelin, P., Peters, T., & Evans, A., 1994. Automatic 3d intersubject registration of mr volumetric data in standardized talairach space. *Journal of Computer Assisted Tomography* **18**, pp. 192–205.

Davatzikos, C., Ruparel, K., Fan, Y., Shen, D., Acharyya, M., Loughead, J., Gur, R., & Langleben, D., 2005. Classifying spatial patterns of brain activity with machine learning methods: Application to lie detection. *NeuroImage* **28(3)**, pp. 663–668.

Formisano, E., Martino, F. D., & Valente, G., 2008. Multivariate analysis of fmri time series: classification and regression of brain responses using machine learning. *Magnetic Resonance Imaging* **26(7)**, pp. 921–934.

Guyon, I. & Elisseeff, A., 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research* **3**, pp. 1157–1182.

Hong, J.-H. & Cho, S.-B., 2006. Efficient huge-scale feature selection with speciated genetic algorithm. *Pattern Recognition Letters* **27(2)**, pp. 143–150.

Keller, T. A., Just, M. A., & Stenger, V. A., 2001. Reading span and the time-course of cortical activation in sentence-picture verification. *Annual Convention of the Psychonomic Society, Orlando, FL* .

Kudo, M. & Sklansky, J., 2000. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition* **33(1)**, pp. 25–41.

Liu, H., Liu, L., & Zhang, H., 2009. Boosting feature selection using information metric for classification. *Neurocomputing* **73(1-3)**, pp. 295–303.

Mitchell, T. M., Hutchinson, R., Niculescu, R. S., Pereira, F., Wang, X., Just, M., & Newman, S., 2004. Learning to decode cognitive states from brain images. *Machine Learning* **57**, pp. 145–175.

Mitchell, T. M., Shinareva, S. V., Carlson, A., Chang, K., Malave, V. L., Mason, R. A., & J.T., M., 2008. Predicting human brain activity associated with the meanings of nouns. *Science* **320**, pp. 1191–1195.

Mourão-Miranda, J., Bokde, A. L., Born, C., Hampel, H., & Stetter, M., 2005. Classifying brain states and determining the discriminating activation patterns: Support vector machine on functional mri data. *NeuroImage* **28(4)**, pp. 980–995.

Pereira, F., Mitchell, T., & Botvinick, M., 2009. Machine learning classifiers and fmri: A tutorial overview. *NeuroImage* **45(1, Supplement 1)**, pp. 199–209.

Ramirez, R. & Puiggros, M., 2007. A genetic programming approach to feature selection and classification of instantaneous cognitive states. In Giacobini, M., ed., *Applications of Evolutionary Computing*, vol. 4448 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, pp. 311–319.

Shinkareva, S. V., Mason, R. A., Malave, V. L., Wang, W., Mitchell, T. M., & Just, M. A., 2008. Using fmri brain activation to identify cognitive states associated with perception of tools and dwellings. *PLoS ONE* **3(1)**, p. 1394.

Siedlecki, W. & Sklansky, J., 1989. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters* **10(5)**, pp. 335–347.

Yoshida, W. & Ishii, S., 2005. Model-based reinforcement learning: a computational model and an fmri study. *Neurocomputing* **63(0)**, pp. 253–269.

Zhang, Q. & Lee, M., 2009. Analysis of positive and negative emotions in natural scene using brain activity and gist. *Neurocomputing* **72(4-6)**, pp. 1302–1306.

Zongker, D. & Jain, A., 1996. Algorithms for feature selection: An evaluation. *Proceedings of the 13th International Conference on Pattern Recognition* **2**, pp. 18–22.

CURRICULUM VITAE

Name Surname	:	Ceyhun Can ÜLKER
Address	:	Bahçeşehir Üniversitesi Mühendislik Fakültesi Çırağan Caddesi 34353 Beşiktaş/İSTANBUL
Date and Place of Birth	:	18.08.1988 ANKARA
Languages	:	Turkish (native), English (fluent)
B.S.	:	Bahçeşehir University
M.S.	:	Bahçeşehir University
Institute	:	The Graduate School of Natural and Applied Sciences
Program	:	Computer Engineering
Work Experience	:	Bahcesehir University Computer Engineering Department *Research and Teaching Assistant* (Istanbul, 2010 - today)