BACHELOR OF SCIENCE IN
COMPUTER SCIENCE AND ENGINEERING

Inspiring Excellence

Brain Image fMRI Data Classification and Graphical Reprentation of Visual Object

AUTHORS

Indrani Datta Tithi

Ummay Sadia Khanum Shuchi

Nazifa Afroza Tasneem

SUPERVISOR
Dr. Md. Ashraful Alam
Assistant Professor
Department of CSE

A thesis submitted to the Department of CSE
in partial fulfillment of the requirements for the degree of
B.Sc. Engineering in CSE

Department of Computer Science and Engineering
BRAC University, Dhaka - 1212, Bangladesh

December 2018

We would like to dedicate this thesis to our loving parents …

Declaration

It is hereby declared that this thesis /project report or any part of it has not been submitted elsewhere for the award of any Degree or Diploma.

Authors:

Nazifa Afroza Tasneem
Student ID: 18241028

Indrani Datta Tithi
Student ID: 18241030

Ummay Sadia Khanum Shuchi
Student ID: 16101121

Supervisor:

Dr. Md. Ashraful Alam
Assistant Professor, Department of Computer Science and Engineering
BRAC University

December 2018

The thesis titled Brain Image fMRI Data Classification and Graphical Representation of Visual Object
Submitted by:
Author Name -Ummay Sadia Khanum Shuchi ID: 16101121
Author Name -Nazifa Afroza Tasneem ID: 18241028
Author Name -Indrani Datta Tithi ID: 18241030
of Academic Year 2018 has been found as satisfactory and accepted as partial fulfillment of the requirement for the Degree of Computer Science and Engineering

1.	Dr. Md. Ashraful Alam Assistant Professor
2.	Dr.Iftekharul Mobin Assistant Professor
3.	Dr. Md. Abdul Mottalib Chairperson of Department of CSE
4.	Ummay Sadia Khanum Shuchi Author
5.	Nazifa Afroza Tasneem Author
6.	Indrani Datta Tithi Author

Acknowledgements

This work would not have been possible without the immense support of our supervisor Dr. Md. Ashraful Alam who provided us all the necessary facilities for the research. We also give sincere thanks to our co-supervisor Dr. Md. Iftekharul Mobin for giving the right direction and suggestion to complete the research. We would also like to thank our family members and seniors who directly or indirectly helped us to fulfill this work. We had them on our side every time we faced any difficulty. Also credit goes to the departments because without it the thesis would not even happen in the first place. Sincere thanks goes to all the member associated with CVIS lab , specially Dr. Md. Ashraful Alam who used to visit the lab regularly and inspired us to progress at a consistent pace .The equipment of CVIS lab advanced enough to provide us a smooth work flow. Lastly, thanks to the almighty who kept us in a good health throughout all this time during our thesis work and give us the strength to finish it successfully.

Abstract

Analyzing neuroimaging data has become a research of interest these days because of their several applications starting from analysis of brain region connectivity to analysis of ventral streams and visual stimuli. In this paper, we propose a model that explains what image a human brain visually perceives based on the neuroimaging information from the ventral temporal cortex (VT) portion. In the model, we used the nilearn library from python repository along with the haxby data set which includes a set of functional MRI from 6 subjects viewing images that contains a grid of black and white pictures of some certain figures. Firstly, the haxby data set was collected and few pre-processing steps such as masking, scaling and smoothing was done in order to reduce the complexity, noise and to standardize the data. Then, the entire data set was cross validated into 80 percent of training example and 20 percent of test example. After the splitting was done, the training examples were passed through a set of learning frameworks such as 'Nearest Neighbors', 'Linear SVM', 'RBF SVM', 'Gaussian Process', 'Decision Tree', 'Random Forest', 'Neural Net', 'AdaBoost', 'Naive Bayes' and 'QDA' algorithms. Completing the training, the accuracy of the frameworks was tested and on an average the most accuracy of 95 percent was found with Neural Network and Support Vector Machine (SVM) across all the subjects.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Neuroimaging in medical science are the techniques of directly or indirectly imaging the function of the nervous system. In many tasks, we need to visualize the nervous system, especially the brain to determine the reason of many physical sickness symptoms. For example, to detect brain tumor or to examine how a brain function during various different tasks and states. For certain types of tasks there is a certain pattern of blood flow in different parts of our brain. Therefore, we can mention these patterns as phonological codes [1] which has individual entity for each different types of tasks. For example, there is phonological code of brain activity with substantial difference when we read a word versus when we hear the same word [2]. We record these events of blood flowing in the brain in form of fMRI.FMRI is usually a series of 3D MRI image taken at different points of time, which indicates what type of change is happening in the brain depending on what type of task a subject is doing in different points of time. The power of magnetic field is used to construct MRI which will be discussed in more detail at chapter 2 of this paper.

The ventral object vision pathway can create virtually infinite amount of object representations in our brain (3). The representation of seeing different objects is widely distributed and largely overlapping [4-6] and it happens in the ventral temporal cortex of the brain which contains a limited number of areas that are specialized for representing specific categories of stimuli [7,8]. The visual stimuli of seeing a cat will be different in our brain than the visual stimuli of seeing a piece of chair. This visual stimuli mapping in our brain can be very helpful. For example, understanding which brain portion works better for some specific tasks may open up lots of new possibilities. There are many disable person in the world who are unable to do some specific visual and cognitive task. Knowing what brain portion is responsible for the lack of the specific visual and cognitive sense may open up new ways or recovery. In addition to that, if in the future brain simulation may become a viable approach to to create virtual reality experience, the study of brain visual mapping will be very helpful in terms of implementing the technology. As a result, trying to determine what a person is seeing based on the visual stimulus in the brain has become an important application.

Specific dataset of understanding visual stimuli from MRI has been done rarely so far. The haxby dataset is one of the first dataset in this regards that was produced during 2002. It is a block-design fMRI dataset from a study on face and object representation in human ventral temporal cortex which consists of 6 subjects with 12 runs per subject. The ventral temporal cortex is the portion of the brain where all the visual processing happens so that is why is it was prioritized in the haxby dataset. Each image was shown for 500ms and was followed by a 1500ms inter-stimulus interval. The inter-stimulus interval is known as the 'rest period'. Full-brain fMRI data were recorded with a volume repetition time of 2.5s, thus, a stimulus block was covered by roughly 9 volumes. For out experiment, we shall use the same dataset for now.

1.1 Thesis Outline

We have segmented our thesis work in several parts, depending on the areas we have focused and also the working process. In each part there are several sub parts relevant to the main parts which describe the processes in details. To understand our work properly we have added figures, tables and graphical representation in each chapter. Here, Chapter 2 focuses on the background study and all the literature information needed for this paper. Chapter 3 describes about the proposed model and working tools, soft wares and libraries used. Then, Chapter 4 represents the experimental set up, dataset preprocessing, algorithms, result and analysis of result along with comparisons of different classifiers. Finally, Chapter 5 concludes our whole paper and gives the idea of possibilities of future work.

Chapter 2

Literature Review

2.1 Related Work

According to Raheel Zafar,Sarat C. Daas(2017), for object recognition convolution neural network has progressive performance. Functional magnetic resonance in neuroimaging analysis is best technique because based on structural or functional information; data is extracted from specific brain region. In this study, during ROI analysis, t-contrast of the design matrix is obtained from which significant voxels are obtained. Convolution neural network is applied on these significant voxels. In their work, data is taken against two conditions.1000 significant voxels with highest absolute values are taken for each of the conditions. In their proposed method, convolution neural network and ROI analysis are used for analysis and for both the methods, support vector machine is used. From their work, we can find that features can be extracted from convolution neural network which can provide significant results in comparison to others [17**?**].

According to Young fan, Christos Davatzikos(2006), there are some difficulties in detection of a particular cognitive state across different subjects using FMRI images. One of the major difficulties in building classifiers for this purpose is high inter-subject functional variability in brain activation patterns. In this study, the authors have tried to overcome this difficulty; they firstly worked on detecting the brain regions that are relevant to the problem from training data. Then regional features are formed by extracting statistical information from each brain region. Finally they used PCA technique to reduce the regional feature statistical variations across different samples.They further worked on the improvement of the generalization ability and efficiency of the classification. From the extracted regional features, they selected the most discriminative features using a hybrid feature selection method. These discriminative features are used for decoding brain states from FMRI images by training

SVM classifier [18]. From their study and work, we learn that, their proposed method gives better results compared to other FMRI image classification methods.

Chi Zhang, Kai Qiao(2018) believe that, constraint-free natural image reconstruction from brain activity is a challenge till now and the existing research addresses the problem by using semantic prior information or just reconstructing simple images, including letters and digital. The authors wanted to work without semantic prior information. Thus they proposed a method to reconstruct natural images from the fMRI signals of human visual cortex based on the computation model of convolution neural network (CNN). Firstly, they extracted the unit output of viewed natural images in each layer of a pre-trained CNN as CNN features. Secondly, they transformed image reconstruction from fMRI signals into the problem of CNN feature visualization. For this purpose, they trained a sparse linear regression to map from the fMRI patterns to CNN features. They optimized iterative to find matched image as the CNN unit features of the matched image is most similar to those predicted from brain activity. From this study we find that, the authors finally achieved their desired results for constraint-free natural image processing by iterative optimization [2].

Daniel D. Leeds, Darren A. Seibert(2013) did a research on how the human brain encodes object information along the ventral pathway€"the neural real estate associated with visual object processing. Most studies of the visual properties that lead to the recruitment of these class-specific, functionally defined brain regions have focused majorly on two things. Those are- the effects of spatial transformations and of the altering of simple domain-specific features. In their study they tried to find out how well the human cortical visual pathway is measured using FMRI. Different computer vision models are responsible for neural object encoding. The authors employed several models of visual representation drawn from machine vision. Each of them provides a putative hypothesis regarding the features used in object perception. These representations incorporate diverse linear and nonlinear operations on image properties. These results in maximization of machine performance for detecting and recognizing objects. With the help of this set of models, they collected data on human object processing using fMRI. Then the dissimilarity matrices predicted by each computer vision model were correlated and compared with the resultant neural data. As a result, they could establish a correspondence between each model patterns of neural activity in specific areas of the brain. From this study it is found that different models were associated with neural object encoding in different cortical locations [20]. Visual categorization usually happens in the human ventral temporal cortex (VTC). Visual perception is a rapid task and human can recognize object in only 1/10 seconds. Ventral temporal cortex contains regions that are selective for objects, faces, bodies and places. In addition to that, it contains information about color, eccentricity bias, visual field maps, specific domains, expertise, object categories, concepts, semantics and real-world object size. In order to understand the role in visual categorization, the authors took the approach of understanding information- processing system which came from David Marr. According to Marr computation (goal), representation (way to achieve goal) and neural implementation (application level) are the three levels of system that are needed to be studied [15, 8].

The computational theory says that VTC should be able to efficiently generalize across examples of a category while being able to distinguish examples from different categories. One way to achieve this efficiently is to have representations that are linearly separable if the spikes in the neuron are unique for each different object, the categorization can be done easily‘[22, 18].

The representation theory confirms the achievement of target from computational theory. After experimenting the authors saw that the region that responds to a set of particular image, those same regions respond to the images of same type compared to the other regions. VTC responses are primarily driven by the shape and content rather than less intuitive physical property such as the contrast. Thus, regions dedicated for each type of visual stimuli manages to react correctly because of the clearly distinguishable features they observe‘[26, 5].

Finally, in the neural implementation the authors saw that the neurons with similar properties are clustered together. Cluster of neurons with similar property react with the visual stimuli of portions of object, multiple clusters overlap one another for a more complex visual stimuli in the brain. In this article the author talked about the validity of GLM approach in the task of statistically analyzing fMRI data. In 2011, the author said that while the GLM approaches was one of the most flexible tools to analyze fMRI time series back then, the model had some issues. The problem is not the bias of the model but the variance of the model. The author also mentioned that checking of the model€™s assumptions is virtually absent because of the sheer amount of data making it difficult for the learning approach. For inference purpose, the author suggested other approaches that do not necessarily make use of the GLM for prediction. The writer also proposed Bayesian methods as a potential alternative [23]. Trying to determine which different brain region performs during different cognitive operation has been an ongoing task for a while. This is especially so in pre-frontal cortex whose sub regions, such as the dorsolateral pre-frontal (DLPFC), anterior cingulate (ACC) and orbitofrontal (OFC) cortices are known to have differential roles in cognition. The authors discriminated the different task stages by using multivariate test statistics with bootstrap-based procedures where they observed the fMRI blood oxygenation level dependent signal pattern. They also developed an algorithm that can select any relevant voxel in the brain. Using both the multivariate statistics approach and the algorithm that searches for maximally informative voxels the authors showed that DLPFC and ACC contribute more during the

Jumping to Conclusions task while the OFC is more involved in choice evaluation and uncertainty feedback. This experiment shows how fMRI data can be analyzed to distinguish the role and functionality of brain [7, 23, 20].

James V. Haxby, Maura L. Furey (2001), believe that, FMRI has revealed a large scale spatial organization within the ventral object vision pathway for specialization. These are represented by different patterns of response i.e., increase of neural activity indicated by increase of blood oxygenation in ventral temporal cortex, for faces and other types of objects. They talked about three classes which are models for this functional architecture. In one model it is proposed that ventral temporal cortex has limited number of areas. The areas which are specialized for demonstration of categories of stimuli in specific terms, as a result two specialized area are found- the fusion form face area (FFA) and Para hippocampal placer area(PPA). Another model proposes that the FFA is specialized for expert visual recognition any type of objects, not only faces [24, 25, 21].

In the third model, the representation of faces and different categories of objects are widely distributed and overlapped. According to the third model, ventral temporal cortex has a organized representation of attributes topographically which has a form to recognize faces and objects. In this work, the model object form topography is used which predicts the way of all categories that might evoke distinct patterns of response in ventral temporal cortex and gives the idea about how this cortex can produce unique representations for different categories of objects. They tested their model by investigating the patterns of response evoked in ventral temporal cortex by faces and multiple categories of objects. Studying this paper, we find that, this model predicts that each category elicits a distinct pattern of response in ventral temporal cortex that responds maximally to other categories [19, 11, 10, 17, 18, 2, 16, 8, 14].

Modular and Distributed hypothesis studies have been done and evidences for both have been found in the application of novel analyses of the patterns of neural activity in viewing and recognition of objects while studying ventral temporal cortex (VT) to objects in neuroimaging [1, 2, 14].

Researchers found diverging conclusions about the distributed versus modular nature of object representations in VT cortex. A standard for determining the degree to which individual patterns vary is obtained from the precise definition of distributed vs. modular patterns of activation. However, it does not give the answer that why certain areas are more or less modular/distributed or how object category representations are organized. In this paper work, author Alice J. O'Toole argues that the structure of brain activity patterns is based on computational analysis of the structure of object categories. The representations that may underlie the neural response patterns have hypotheses. These hypotheses about these kinds of representations can be constrained by the representational parameters of this analysis. Modular and distributed are qualitatively discrete type of representations. Voxel information content is the variable that connects modular with distribution. Each voxel carries information for only one category of object in the modular case and all voxels carry information for all categories in the distributed extreme [1, 3, 4, 6].In this paper the authors proposed that, the key to linking neural responses with the represented physical world understands partially distributed codes of stimulus parameters. This is because, how objects are represented and recognized neutrally is known from the clues found from modular and distributed patterns of neural responses. In this article, the first goal of the authors was to employ a pattern based classification analysis that can generate data on the confusability of the neural response profiles for pairs of object categories [9, 10, 12, 13]. A measure of shared neural resources among the categories can be obtained from the pattern confusability data. Secondly they evaluated the relationship between the neural responses and stimulus structure by implementing a computational analysis of object recognition [9**?**]. This computational analysis of object recognition operated on the stimuli from the FMRI experiment. They combined the analysis of stimulus structure with an analysis of neural activation patterns from a functional neuroimaging study to bring the previously divergent neuro-imaging results in a single framework. This accumulates the previous findings and gives new knowledge of why responses of object categories vary in the extent to which they are distributed [20, 15, 23].

Inspired by these related works, we worked further in this paper and got ideas from different models and their functions. The background knowledge about different approaches, tools and processes used in our thesis work are briefly described below:

2.2 Different Forms of Imaging

With the currently available tools, there are few elements that are used for imaging some certain body parts. In terms of imaging the property of radiation, electricity and magnetic fields come in handy. MRI is one type of imaging technique that utilizes the property of magnetic and electrical sources of energy. It is used to visualize the structure of the brain in the forms of a 3D image consisting of multiple slices. It is helpful for discovering unnoticed anatomical anomalies caused by a disease process or traumatic event. With some modification, a sister machine was created from MRI machines which is called as fMRI machine. While in MRI images we see the structural property of the brain, in fMRI we see the behavioral property of the brain. In fMRI, the blood flow or the amount of oxygen in the blood of the brain is measured to achieve the functional image of the brain. So, an MRI studies the property of hydrogen molecule in a nuclei but fMRI calculates the level of oxygen. Even though, it all started from the invention of MRI, fMRI has slowly taken a significant part in medical science because of the diseases that affect the type of human behavior and activity. In addition to that, study of human behavior and the reaction of brain on specific behavior has become important for statistical application. As a result, in statistical and big data analysis it has started to play a big role too.

2.3 FMRI Acquiring Process

FMRI is acquired using the property of oxygenated blood in human brain. The neurons of a human brain works like muscles, when it functions it expands and requires more oxygenated blood. Therefore, the oxygenated blood flow increases in the part that are more active and this property is used to take fMRI.

When a brain region becomes more active, the region gains more water molecule because of the amount of oxygenated blood in that region. The water molecule has a magnetic property which is known as the spin. The spin works like a compass, means it tries to realign with the external magnetic field. Spins are not as reactive as a compass so the magnetic pole of the earth does not affect them. However, an MRI machine can create magnetic fields that is much higher than the magnetic field of earth, a general MRI machine can create magnetic field that is almost 10,000 times stronger than the magnetic field of earth. This strong magnetic field causes the spin of the water molecules in brain to align towards the direction of water molecule. Because of the alignment of all the water molecules towards the exact direction, it creates the very strong magnetic signal. When the water molecules in the brain are aligned at the same direction, the external magnetic field created from the MRI machine is then removed and the spin of the water molecules start to de-align with each other. This is where the amount of oxygen in each area of the brain starts to play a role. The spin de-aligns more quickly in the places where there is less amount of oxygen in the blood and the spin stays aligned for more time in the places where there is more oxygen. We know that there is more oxygen in the more active part of a brain, so the more active part of the brain loses its magnetic signal much slowly compared to the less active part of the brain. When the magnetic field of the MRI machine is removed, after a small amount of time the magnetic signal coming out of the spins in brain is examined per slice of the brain. A slice is a thin horizontal segment of a brain structure and multiple slices are used to construct a full fMRI image. The amount of magnetic field is recorded in the form of voxels. A voxel is a 3D cell that records the activity of a tiny segment in the brain. Multiple voxels are used to construct a particular slice of the functional MRI. Now, when a particular region of a brain is more active, there is more oxygen and after measurement we get stronger magnetic field in those region. Therefore, the voxels in those regions are recorded brighter (more intensity value of the voxels). On the other hand the voxels in the other region are less bright (less intensity value per voxel) because of the weaker magnetic signal. These variations of brightness of voxels are recorded in a slice and many slices combined build up a single fMRI shot. A single fMRI shot represent the activity level of a brain during a specific moment. A complete package of fMRI contains a series of multiple fMRI that records the activity level of the brain across a long span of time.

2.4 FMRI Analyzing Approach

As mentioned in segment 2.2, fMRI is a set of functional magnetic resonance images where we observe the change of intensity of each of the voxels in a brain while doing a certain task. Therefore, while creating an fMRI the data information of blood flow gets encoded into voxels as intensity and when decoded, each voxels represents a time series where we get the change of blood flow in a tiny portion of the brain with time. The X-axis of the time series represents a certain period of time and the Y-axis represents voxel intensity. As a result, the dataset is a huge collection of time series resulting from a large number of voxels and each of the time series looks something like the following,

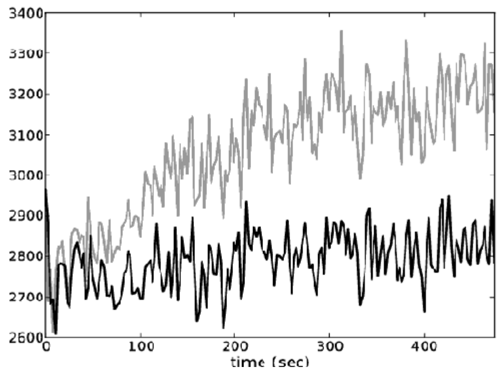

Fig. 2.1 Time series of voxel intensity

This time series is then used to fit machine learning algorithms and then apply for different statistical analysis.

2.5 Haxby Data set

6 subjects are shown gray scale images of different objects which are represented in grid format. All the different objects are,

- Cars
- Bottles
- Houses
- Faces
- Scissors
- Shoes
- Chairs

These object pictures are represented in 8 x 6 sized grids, the grids look like the following Fig 2.2,

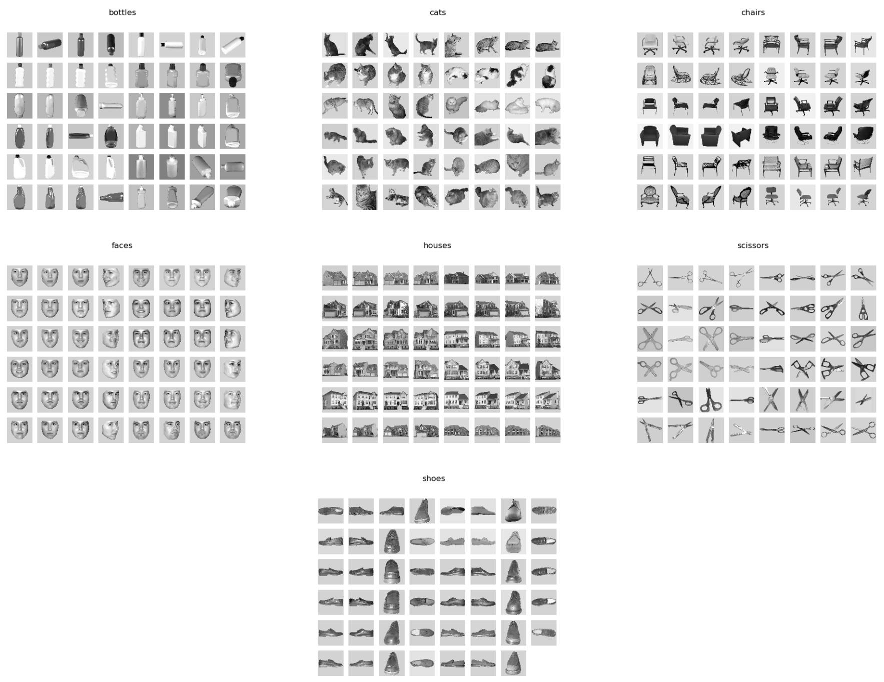

Fig. 2.2 Haxby dataset objects

These pictures are showed to the subjects for a constant amount of time that are separated by rest period in between them. According to the dataset manual, each image was shown for 500ms and was followed by a 1500ms inter-stimulus interval. Meaning, 5 seconds of viewing image with 15 seconds of rest period before viewing the next one. When all the 7 images are shown separated by rest period, a single run is complete. There is 12 runs per subject, means each subject saw all the images 12 times.

From haxby dataset, we get the voxel intensity of brains at a specific moment as the feature set X, and name of the object the subject is seeing at that particular moment as the label set Y. This dataset then can be used for statistical application. Haxby dataset can be downloaded from the PyMVPA repository.

2.6 Extracting useful data from raw dataset

The total package for one single subject in the haxby dataset collection is,

- An anatomical MRI of the subject. This is a high resolution image that represents the anatomical structure of the brain of subject
- A series of fMRI time series images. An fMRI is a 3D image, a whole series of those 3D images are usually referred as a 4D image series. There are 1452 images in the series with 40 x 64 x 64 voxels. A voxel has a size of 3.5 x 3.75 x 3.75 mm. This time series contains information of all the 12 runs.
- A masker to mask the ventral temporal cortex portion
- A label file containing information of what a particular subject is seeing at a specific moment

To extract useful data from haxby dataset, few pre-processing steps has to be done. Firstly the ventral temporal cortex has to be masked to reduce the complexity and then the data has to be taken to numerical format and pre-processed so that machine learning approach can be taken. These details will be discussed thoroughly in the next segment.

Chapter 3

Proposed Model

In our model, we have followed some methods to complete our goal. To visualize the object more accurately, we tried our best to find a good accuracy. That is why we tried to solve the same problem in different ways. Some approach gave us quite good results and some did not. At first we trained the pre-processed data, keeping first 30 percent data aside, and the rest for training the machine, so that the machine can learn about those data. After training these data, when we tested rest of the data. The accuracy rate found was very low. For this reason, we did train and test our data set in the ratio 80-20 percent respectively, as in machine learning, most of the data sets trained and tested in this ratio gives most accurate results. We got tremendously better accuracy after training in this ratio over the classifiers. Among all the classifiers we got best scores in SVM and Neural Net classifier. So, we started working on these two classifiers to get more accurate results and determine best parameters. We fixed the parameter "kernel" as "RBF" and parameter "c=100" in SVM. In Neural net also we fixed parameters "hidden layer sizes=464" and "activation" in "tanh" to get better results. Afterwards, we used the process grid search to optimize the parameters and determine the best parameters. This cross validation process also enlightens us about the parameters for which the accuracy rate lowers. Grid search is a process of hyper parameter optimization through which data scanning is done which is used in configuration of optimal parameters in a given model. Certain parameters are necessary in Grid search to identify for which parameters the classifiers work the best. Grid search builds a model in each parameter combination and iterates over every parameter combination to store a model. Grid search is types of cross validation which go through multiple combinations of parameter, cross validate each other. Then the result gives the best parameters and also identifies for which parameters the result becomes inefficient. The function of cross validation is to separate the test and training data and then to check training results with test data. Cross Validation involves reserving a particular sample of a data set, on which the model is not trained. The model is to be tested for finalization. Manually set bounds and discontinuation may be necessary before applying grid search; because, the parameter space of a machine learner may include real-valued or unbounded value spaces for certain parameters.

3.1 Workflow

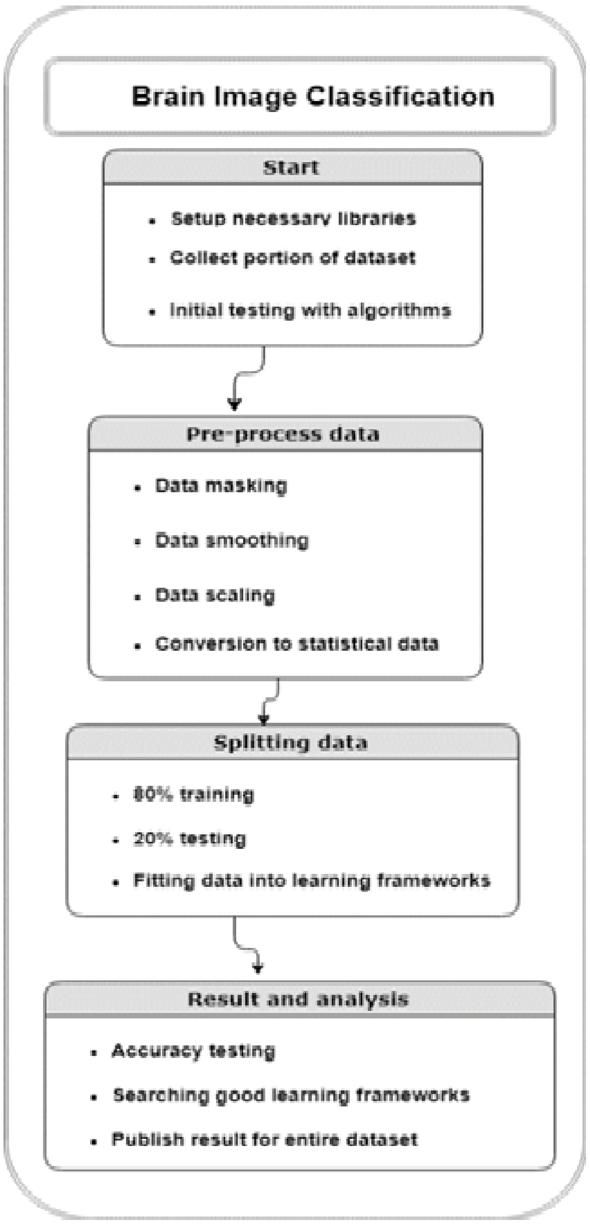
Brain Image Classification
Start
Setup necessary libraries
Collect portion of dataset
Initial testing with algorithms
Pre-process data
Data masking
Data smoothing
Data scaling
Conversion to statistical data
Splitting data
80% training
20% testing
Fitting data into learning frameworks
Result and analysis
Accuracy testing
Searching good learning frameworks
Publish result for entire dataset

Fig. 3.1 The workflow of entire process

3.2 Softwares, Tools and Libraries

We used different software, tools and libraries in this thesis work to accomplish our desired goal.

3.2.1 PyCharm

It is an integrated development environment (IDE) computer programming, specifically for the python language. The Czech company Jet Brains developed this IDE. Code analysis, graphical debugger, integrated unit tester, integration with version control systems etc. are provided by py-charm. It is a cross platform with windows, mac OS and Linux versions. The Apache Version releases the community Edition. There is also Professional Edition with extra features. Some features of py-charm are coding assistance and analysis, with completion of code, syntax and error highlighting,integration, and quick fixes, project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages, re factoring of python including rename, extract method, introduce variable, introduce constant, pull up, push down and others, support for web frameworks ,integrated unit testing, with line-by-line coverage of codes, python developments by goggle app engine, version control integration etc.

3.2.2 Jupyter Notebook

Jupyter notebook creates notebook documents on web based interactive computational environment. The Jupyter web application, Jupyter Python web server, or Jupyter document format etc can be referred by Jupyter Notebook. A Jupyter Notebook document contains an ordered list of input/output cells which can contain code, text, mathematics, plots and rich media, usually ending with the ".ipynb" extension. The documents of Jupyter notebook can be converted to a number of standard output formats such as HTML, presentation slides, Latex, PDF, Markdown, Python via the nbconvert library or jupyter nbconvert command line interface in a shell. Many kernels can be connected via Jupyter notebook. Browser-based Interactive computer programming environment that takes single user inputs or termed as “language shell” are provided by Jupyter notebook. These are based on open source libraries such as IPython, Tornado, jQuery, Bootstrap, MathJax etc. A machine learner may include real-valued or unbounded value spaces for certain parameters.

3.2.3 Numpy

Numpy: Numpy is a package in Python. It is used for scientific computing. A powerful N-dimensional array, sophisticated functions, tools for integrating C/C++ etc. are available in numpy package. It's a multidimensional array which stores values of same data type. These arrays are indexed in sequences and it starts with zero. Numpy is a numerical package of python which is generally used for data analysis.

3.2.4 Pandas

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis or manipulation tool available in any language. It is already well on its way toward this goal. Pandas is well suited for many different kinds of data such as tabular data with heterogeneously-typed columns, as in an SQL table or excel spreadsheet. The two primary data structures of pandas, Series (1-dimensional) and data frame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, data frame provides everything that R's data frame provides and much more. Pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries. The performances and functions of pandas are-easy handling of missing data, size mutability, automatic and explicit data, intelligent label-based slicing, fancy indexing, and sub setting of large data sets, intuitive merging and joining data sets, flexible reshaping and pivoting of data sets, hierarchical labeling of axes, time series-specific functionality etc.

3.2.5 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hard copy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface tool kits. Matplotlib is also a massive library, and getting a plot to look just right is often achieved through trial and error. Using one-liners to generate basic plots in matplotlib is fairly simple, but skillfully commanding the remaining 98 percent of the library can be daunting.

3.2.6 Nilearn

Nilearn is a python module that makes the usability of many advanced machine learning, pattern recognition and multivariate statistical techniques on neuro imaging data easier. These are used for multi-voxel pattern analysis, decoding, predictive modeling, functional connectivity, brain connectivity functions etc. It helps fast and easy learning on neuro-imaging data. Different machine learning applications such as predictive learning, classification, decoding, connectivity analysis etc. are done by nilearn using scikit- learns python toolbox. Nilearn can perform tasks on FMRI, resting state, VBM data and so on.

Chapter 4

Experimental setup and Result

4.1 Data set Details

As mentioned in the previous section, the data set for a single subject contains 4 files. They are, an anatomical image, a 4D fMRI series, a label file and a masker. The anatomical image is in a file name called ‘anat.nii’ which is a NIFTI file. It is a single 3D MRI which shows the anatomical image from the brain of a particular subject. However, the main interest is the

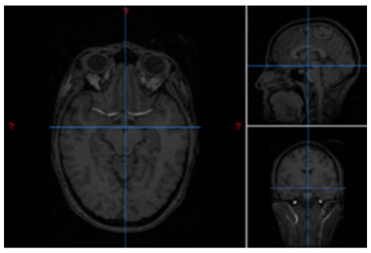

Fig. 4.1 The anatomical MRI ‘anat.nii’ of subject 2

file named ‘bold.nii’ which contains the series of 3D fMRI with respect to time. There are overall 1452 fMRIs for each particular subject. Partial information was taken from the series to create the feature set X.

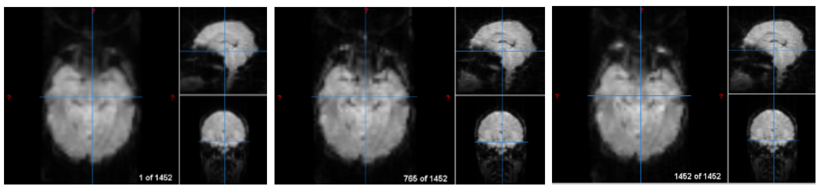

Fig. 4.2 The first, 765th and last fMRI image of subject 2
(Orthogonal main view)

Finally, from the text file we take the label set Y for particular subjects.

4.2 Masking

The aforementioned fMRI series that was mentioned in section A contains fMRIs of the entire brain. This results in an abundance of unnecessary data from the portion of brains which is used for other tasks beside ventral streams. In the fMRI we only observe the ventral object vision pathway and the all of these happens in the VT (ventral temporal cortex) portion of the brain. As a result, the ROI (Region Of Interest) is only the voxels in VT portion of the fMRI images and rest of the data was treated as noise. Therefore, the entire fMRI series was masked to keep only the VT portion of the fMRI series. The location of the VT is given in fig 3.3 where the VT was shown in the average of all the 1452 fMRIs of a subject 2.

After the masking is done we get a reduced amount of voxels, the masked fMRI contains only 464 voxels in each of the fMRI which significantly reduce the volume of the data while keeping the useful portion untouched.

After masking the ROI of fMRIs with the maskers, now it was time to convert it into statistical data. The conversion was done with the niftimasker library of nilearn. After conversion, it was transformed into a 2D numpy array of time series for each voxel. There were 464 voxels in the each filtered MRI for 1452 seconds so it was a numpy array with a dimension of 1452 x 464 that was the feature set X. Now the array was combined with the 'labels.txt' file and thus the label set Y was formed for X. Now the data set is in a proper state for the machine learning algorithms to learn and evaluate.

4.3 Scaling

While there were fewer amounts of data after masking, the voxel intensities vary a lot in-between different ranges using Label Encoding and MinMax Scaling.

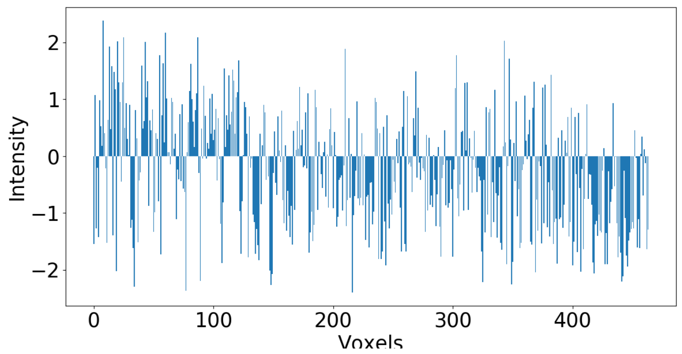

Fig. 4.3 Model example of value range

In order to make the learning algorithms to learn and converge faster, the entire voxel intensity matrix values is scaled and standardized in the range of 0-1.

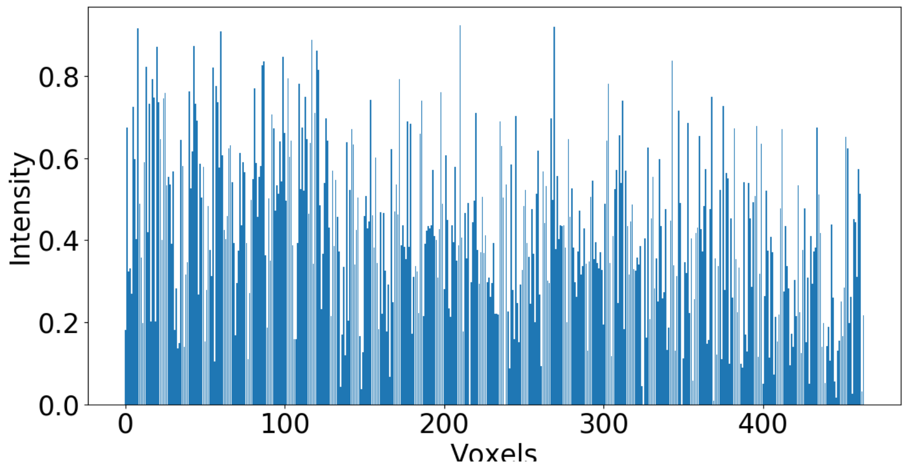

Fig. 4.4 Model example of fig 4.3 after scaling

Standardization of data sets is a common requirement for many machine learning estimators implemented in scikit-learn; they might behave badly if the individual features do not more or less look like standard normally distributed data: Gaussian with zero mean and unit variance. Standardization involves shaping of the distribution and transformation of data to center it by removing the mean value of each feature, then scaling it by dividing non-constant features by their standard deviation. An alternative standardization is scaling features to lie between a given minimum and maximum value, often between zeros and one, or so that the maximum absolute value of each feature is scaled to unit size. This can be achieved using MinMaxScaler or MaxAbsScaler, respectively. The motivation to use this scaling includes robustness to very small standard deviations of features and preserving zero entries in sparse data.

4.3.1 MinMax Scalar

MinMax Scalar works in a very similar fashion, but scales in a way that the training data lies within the range [-1, 1] by dividing through the largest maximum value in each feature. It is meant for data that is already centered at zero or sparse data. MinMax Scalar transforms features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.

4.3.2 Normalization and Level Encoding

Normalization is the process of scaling individual samples to have unit norm. This process can be useful if you plan to use a quadratic form such as the dot-product or any other kernel to quantify the similarity of any pair of samples. The pre-processing module further provides a utility class normalizer that implements the same operation using the Transformer API (even though the fit method is useless in this case: the class is stateless as this operation treats samples independently). We labeled the types of data in numeric values within range; label encoder is a utility class to help normalize labels such that they contain only values between 0 and n classes-1. We labeled the data in numeric value range (0-2) for avoiding string comparison.

4.4 Smoothing

At this step, the data was further smoothed with Gaussian filtering function

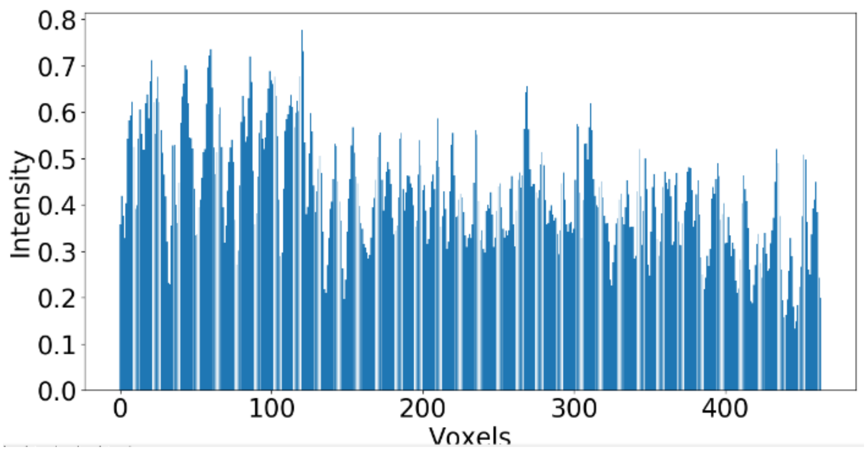

Fig. 4.5 Model example of fig 4.4 after smoothing

After the steps of scaling and smoothing, the data set had far less noise than before with reduced amount of detail for the learning algorithms to learn well.

4.5 Algorithms Used

4.5.1 KNN

The first algorithm we have used is Knn which is not only non-parametric and lazy learning algorithm. We can use it to predict the classification of a new sample point based on the database in which the data points are separated into several classes. The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point along with predicting the labels. Here distance can be generalized in any metric measures

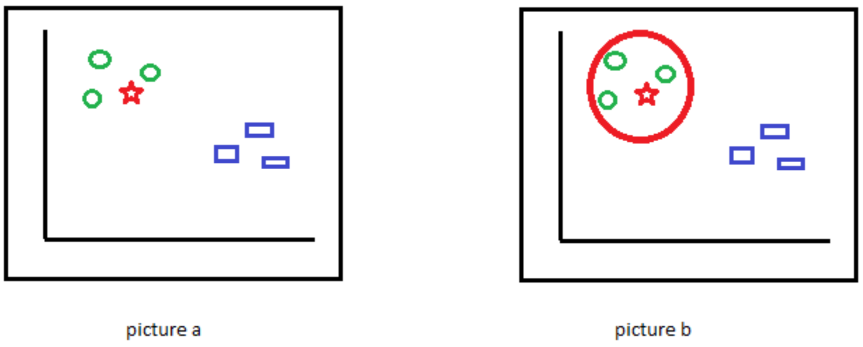

Fig. 4.6 how KNN algorithm works

In the figure 4.6 the picture defines that there are some green circles and blue squares along with a red star. So our work is to identify the class of the red star. By using Knn we can find out the best class of red star whereas defining the value of k determines the approximate values of the algorithm. If we assume the value of k is 3 that means there will be a circle around the red star and green circles as it means 3 data points are in the plane (picture b). These 3 closest points to red star are all green circles so it's obvious that red star belongs to the class of green circles. Here the value of k determines how smooth data can be classified by segregating between circles and squares.

In k-nearest neighbor learning the number of samples which would be used can be a user-defined constant whereas radius-based neighbor learning depends on the local density of points. It's a type of non-generalizing machine learning methods of which nearest neighbors has been successful in a large number of classification and regression problems. Despite of being decision boundary is very irregular it has been often successful in classification situations. The optimal choice of the value is highly data-dependent: in general a larger suppresses the effects of noise, but makes the classification boundaries less distinct. Rather than discrete variables the data labels are continuous whenever we can use Neighbors-based regression. The label assigned to a query point is computed based on the mean of the labels of its nearest neighbors. Another noticeable point is KNN performs better with a lower number of features than a large number of features. In case of over fitting, the needed data will need to grow exponentially as we increase the number of dimensions. According to the researchers no optimal number of neighbors suits all kind of data sets. Each data set has it's own requirements. In the case of a small number of neighbors, the noise will have a higher influence on the result, and a large number of neighbors make it computationally expensive. Researchers has also shown that a small amount of neighbors are most flexible fit which will have low bias but high variance and a large number of neighbors will have a smoother decision boundary which means lower variance but higher bias. One of the advantages of KNN is the training phase of K-nearest neighbor classification is much faster compared to other classification algorithms. Because of this KNN is known as the simple and instance-based learning algorithm. Besides that in case of nonlinear data KNN can be used with the regression problem. Here another point is to specified that the testing phase of K-nearest neighbor classification is slower and costlier in terms of time and memory also it requires large memory for storing the entire training data set for prediction. Additionally, KNN requires scaling of data because it uses the Euclidean distance between two data points to find nearest neighbors. Euclidean distance is sensitive to magnitudes. The features with high magnitudes will weigh more than features with low magnitudes though KNN is also not suitable for large dimensional data.

4.5.2 SVM

A Support Vector Machine (SVM) is a discriminating classifier formally defined by a separating hyper plane. This algorithm outputs an optimal hyper plane which categorizes new examples. Besides that in two dimensional space this hyper plane is a line dividing a plane in two parts where in each class lay in either side.

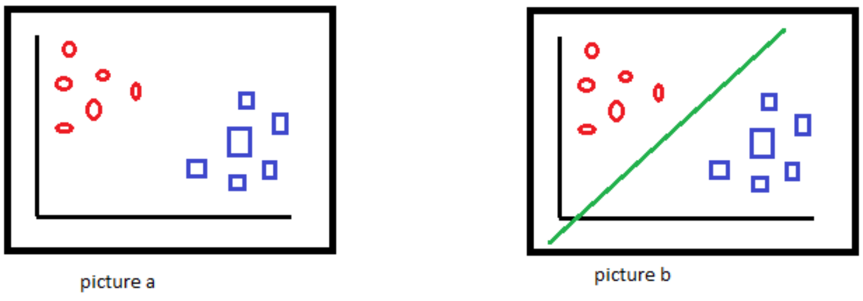

Fig. 4.7 how SVM (example 1) algorithm works

Here are the pictures(figure 4.7) of red circles and blue squares where our task is to separate them by using svm algorithm. With the hyper plane line we can easily separate the two different classes that actually what svm does according to picture b.

In figure 4.8 here blue circles and green squares (picture a) have to be separated by using svm algorithm. But here we cannot use just one straight line to differentiate them whereas

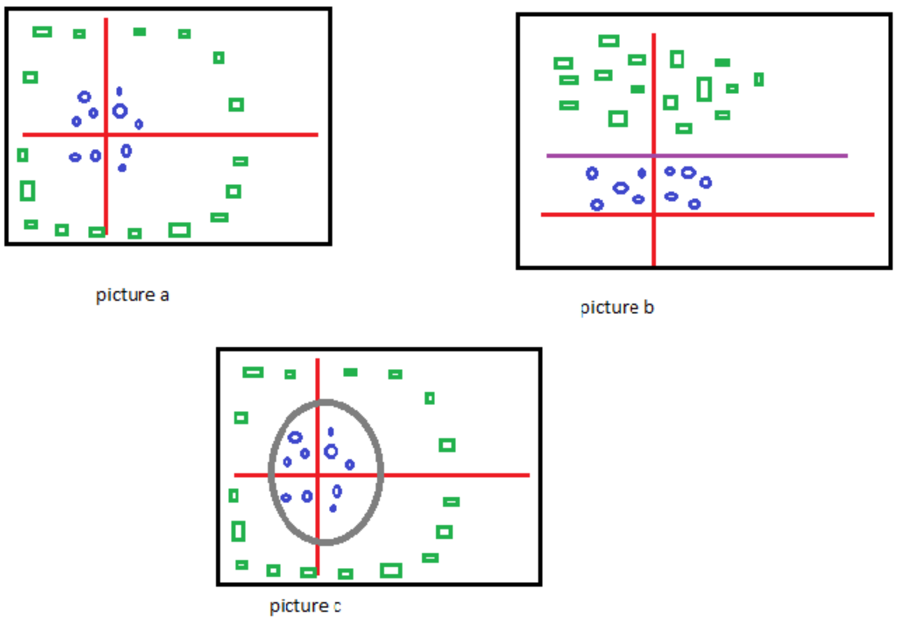

Fig. 4.8 how SVM (example 2) algorithm works

we can add one more dimension using transformation process of z axis according to picture b and then we can separate them by a single straight line. When we transform back this line to original plane, it maps to circular boundary as shown in picture c. These transformations are called kernels. Furthermore by using svm process we can solve much type of difficult problems by using kernels along with parameter tuning. From the several advantages of svm one of them is highly appreciated for being effective in high dimensional spaces which is useful in cases where number of dimensions is greater than the number of samples. Besides that it uses a subset of training points in the decision function or support vectors. That's why it is also memory efficient. Besides that another noticeable part is versatility for which different Kernel functions can be specified for the decision function. Here common kernels are provided, but it is also possible to specify custom kernels. Though Linear SVM is the newest extremely fast machine learning (data mining) algorithm for solving multiclass classification problems from ultra large data sets that implements an original proprietary version of a cutting plane algorithm for designing a linear support vector machine. The radial basis function kernel, or RBF kernel, is a popular kernel function used in various kernelled learning algorithms. In particular, it is commonly used in support vector machine classification. Radial Basis Function is a commonly used kernel in SVC. SVC classifier using an RBF kernel has two parameters: gamma and C. gamma is a parameter of the RBF kernel and can be thought of as the 'spread' of the kernel and therefore the decision region. When gamma is low, the 'curve' of the decision boundary is very low and thus the decision region is very broad. When gamma is high, the 'curve' of the decision boundary is high, which creates space of decision-boundaries around data points. C is a parameter of the SVC learner and is the penalty for non classifying a data point. When C is small, the classifier is okay with non classified data points (high bias, low variance). Whenever C is large, the classifier is heavily penalized for non classified data and therefore bends over backwards avoid any non classified data points (low bias, high variance).

4.5.3 Gaussian Process

Gaussian Processes (GP) are a generic supervised learning method which is designed to solve regression and probabilistic classification problems. A Gaussian process is a stochastic process where a collection of random variables indexed by time or space. For that every finite collection of those random variables has a multivariate normal distribution of its own. Here every finite linear combination is normally distributed. The distribution of a Gaussian process is the joint distribution of all those random variables. A machine-learning algorithm which involves a Gaussian process uses lazy learning and a measure of the similarity between points (the kernel function) to predict the value for an unseen point from training data. The prediction is not just an estimate for that point, but also has uncertainty information. For some kernel functions, matrix algebra can be used to calculate the predictions using the technique of kriging. When a parameterized kernel is used, optimization software is typically used to fit a Gaussian process model. The Gaussian Process Regressor implements Gaussian processes (GP) for regression purposes. For this, the prior of the GP needs to be specified. The prior mean is assumed to be constant and zero (for normalize=False) or the training data's mean (for normalize=True). The prior's co variance is specified by passing a kernel object. The hyper parameters of the kernel are optimized during fitting of Gaussian Process Regressor by maximizing the log-marginal likelihood (LML) based on the passed optimizer.

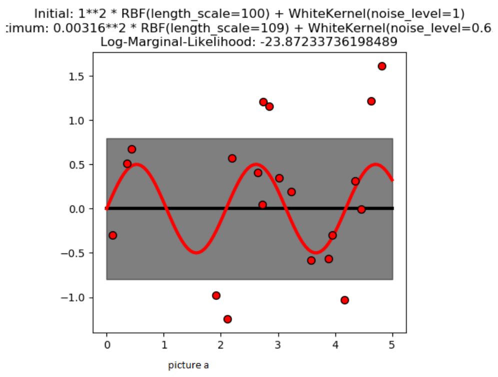

Fig. 4.9 GPR with noise-level estimation (picture a)

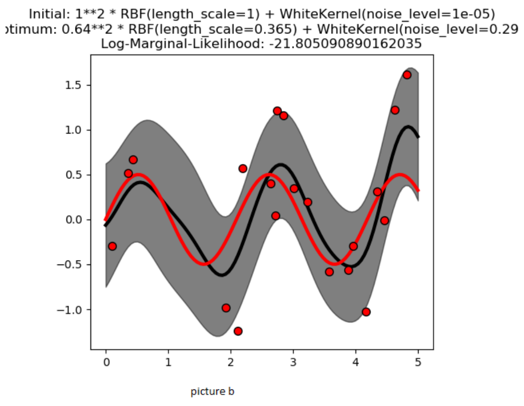

Fig. 4.10 GPR with noise-level estimation (picture b)

In the figure 4.9 and 4.10 there shows that GPR with a sum-kernel including a White Kernel can estimate the noise level of data. An illustration of the log-marginal-likelihood (LML) landscape shows that there exist two local maxima of LML. Picture a corresponds to a model with a high noise level and a large length scale, which explains all variations in the data by noise. Whereas picture b has a smaller noise level and shorter length scale, which explains most of the variation by the noise-free functional relationship. The second picture has a higher likelihood. However, depending on the initial value for the hyper parameters, the gradient-based optimization might also converge to the high-noise solution. It is thus important to repeat the optimization several times for different initializations.

4.5.4 Decision Tree

Decision tree learning uses a decision tree (as a predictive model) to observe an item (represented in the branches) to have conclusions about the item's target value (represented in the leaves). However it is one of the predictive modeling approaches used in statistics, data mining and machine learning. Decision trees used in data mining are of two main types- classification tree, regression tree. Decision tree learning is the construction of a decision tree from class-labeled training tuples. A decision tree is a flow-chart-like structure, where each internal (non-leaf) node defines a test on an attribute, each branch shows the outcome of a test, and each leaf (or terminal) node have a class label. The topmost node in a tree represents the root node.

In the figure 4.11 denotes that decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. When the tree is deeper, the more complex will be the decision rules and the fitter the model. The decision trees are used to fit a sine curve with addition noisy observation. As a result, it learns local linear regressions approximating the sine curve. When the maximum depth of the tree (controlled by the max depth parameter) is

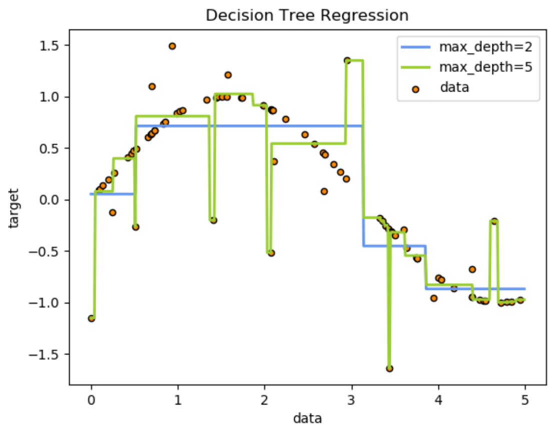

Fig. 4.11 How decision tree regression works

set too high, the decision trees learn too fine details of the training data and learn from the noise, otherwise they over fit.

4.5.5 Random Forest

Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms. Because it is simplicity and the fact that it can be used for both classification and regression tasks. It is also a supervised learning algorithm. Random Forest has nearly the same hyper parameters as a decision tree or a bagging classifier. Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. The Hyper parameters in random forest are either used to increase the predictive power of the model or to make the model faster.

The image of figure 4.12 denotes that how random forests try to use most suitable option among other options by voting and make a prediction. Firstly, there is the “estimators “hyper parameter, which is just the number of trees the algorithm builds before taking the maximum voting or taking averages of predictions. In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation. Another important hyper parameter is “max features“, which is the maximum number of features Random Forest is allowed to try in an individual tree. The last important hyper-parameter is “mins ample leaf “. This determines, the minimum number of leafs that are required to split an internal node. The “njobs“hyperparameter tells the engine how many processors it is allowed to use. If it has a value of 1, it can only use one processor. A value of

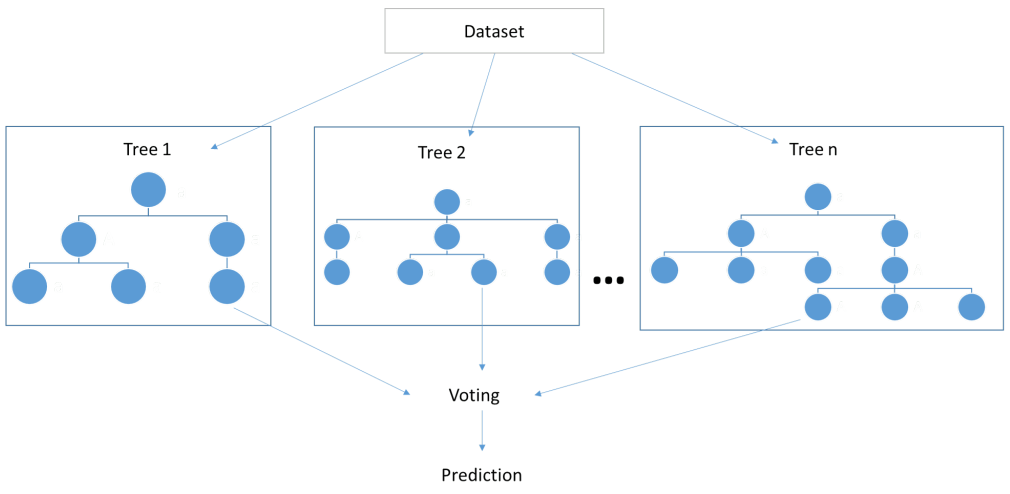

Fig. 4.12 How Random Forest works

"-1" means that there is no limit. Random state"makes the model's output replicable. The model will always produce the same results when it has a definite value of random state and if it has been given the same hyper parameters and the same training data. Lastly, there is the "oob score"(also called oob sampling), which is a random forest cross validation method. In this sampling, about one-third of the data is not used to train the model and can be used to evaluate its performance. These samples are called the out of bag samples. It is very similar to the leave-one-out cross-validation method, but almost no additional computational burden goes along with it.

4.5.6 Neural Net

Neural network is a series of algorithms that attempts to recognize original relationships in a set of data through a process that imitators the way the human brain operates. Neural networks can adapt to changing input so the network produces the best possible result without requiring for redesigning the output criteria. The conception of neural networks is rapidly gaining popularity in the area of trading system development. Neural networks, in the world of finance, support in the development of such process as time-series forecasting, algorithmic trading, securities classification, credit risk modelling and erecting proprietary indicators and price derivatives. A neural network mechanism is similarly to the human brain's neural network. A "neuron" in a neural network is a mathematical function that gathers and classifies information according to a specific architecture. The network allows a strong resemblance to statistical methods such as curve fitting and regression analysis. A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression.

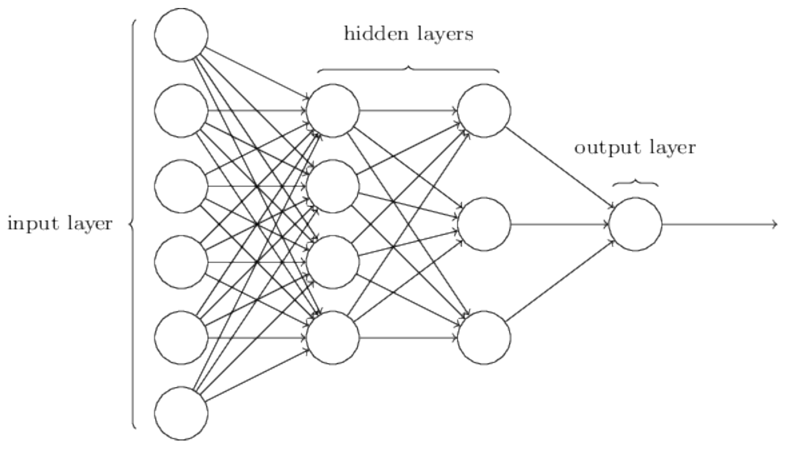

Fig. 4.13 How Neural Network works

In the figure 4.13 we can see the leftmost layer in this network is called the input layer, and the neurons within the layer are called input neurons. The rightmost or output layer contains the output neurons, or as in this case, a single output neuron. The middle layer is called a hidden layer; meanwhile the neurons in this layer are neither inputs nor outputs. The image of figure 4.6 has two hidden layers. In these cases such multiple layer networks are sometimes called multi layer perceptron or MLPs. In a multi-layered perceptron (MLP), perceptions are arranged in interconnected layers. The input layer gathers input patterns. The output layer has classifications or output signals to which input patterns may map. It is hypothesized that hidden layers generalize salient features in the input data that have predictive power regarding the outputs. This defines feature extraction, which achieves a utility similar to statistical techniques such as principal component analysis.

4.5.7 Adaboost

Ada-boost classifier syndicates weak classifier algorithm to form strong classifier. A single algorithm may classify the objects poorly. But if we associate multiple classifiers with selection of training set at every iteration and assigning right amount of weight in final voting, we can have good accuracy score for overall classifier. It re-instructs the algorithm iterative by picking the training set based on accuracy of previous training. The weight-age of each trained classifier at any iteration rest on the accuracy achieved. After training a classifier at any level, ada-boost allocates weight to each training item. Non classified item is allotted higher weight so that it appears in the training subset of next classifier with higher probability. After each classifier is trained, the weight is assigned to the classifier as well based on accuracy. More accurate classifier is assigned higher weight so that it will have more impact in final outcome. Ada-boost like random forest classifier gives more accurate results since it depends upon many weak classifier for final decision. One of the applications to Ada-boost is for face recognition systems. Ada-Boost is the first successful boosting algorithm for binary classification problems[1].

4.5.8 Naive Bayes

The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly matched when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often overtake more sophisticated classification methods. Naive Bayes classifiers are highly scalable, demanding a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression which takes linear time, sooner than by expensive iterative approximation as used for many other types of classifiers. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible associations between the color, roundness, and diameter features. For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervise learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

4.5.9 QDA (Quadric classifier)

A quadratic classifier is used in machine learning and statistical classification to separate measurements of two or more classes of objects or events by a quadratic surface. It is a more general version of the linear classifier. Quadratic discriminant analysis (QDA) is diligently related to linear discriminant analysis (LDA), where it is expected that the measurements from each class are normally distributed. Unlike LDA however, in QDA there is no assumption that the co-variance of each of the classes is indistinguishable. When the normality assumption is true, the best possible test for the hypothesis that a given measurement is from a given class is the probability of ratio test. While QDA is the most frequently used method for obtaining a classifier, other methods are also possible. One such method is to create a longer measurement vector from the old one by adding all pairwise products of individual measurements. The default solver is ‘svd’. It can perform both classification and transform, and it does not rely on the calculation of the co-variance matrix. This can be an advantage in situations where the number of features is large. However, the ‘svd’ solver cannot be used with shrinkage. The ‘lsqr’ solver is a well-organized algorithm that only works for classification. It supports shrinkage. The ‘eigen’ solver is based on the optimization of the between class scatter to within class scatter ratio. It can be used for both classification and transform, and also supports shrinkage. However, the ‘eigen’ solver needs to compute the co-variance matrix, so it might not be suitable for situations with a high number of features.

4.6 Result and Analysis

4.6.1 Data Training and Testing

Keeping first 30 data aside during preprocessing, and the rest for training we trained the machine so that the machine can learn about those data. After training these data, when we tested the data keeping first 30 aside, the accuracy found over the classifiers in this phase are k-Nearest neighbours-36.67 percent, Linear SVM- 63.33 percent, RBF SVM-30.30 percent, Gaussian Process-30.00 percent, Decision Tree-30.00 percent, Random Forest-23.33 percent, Neural Net-70.00 percent, Ad boost- 36.67 percent, Naïve Bayes-40.00 percent, QDA-26.67 percent which was very low accuracy rate.

4.6.2 Splitting into Training and Test Sample

As mentioned earlier in section B, the data set was a numpy array with the dimension of 1452 x 464. The entire data set was divided into 80 percent of training examples and 20 percent of test examples. The split was done randomly in several times to test with different samples in training and test examples. The average accuracy of all those split was calculated. After completing the whole machine learning processes from our experiment we have found three types of results. That ten types of classifiers that we have used on our prepossessed data while training all the data except 30 from the beginning gave a result that did not satisfied much because of the low accuracy.

Furthermore we have found tremendous score on each algorithm while splitting the data into 80:20 ratios. That actually gave a satisfactory result to work on further process. From this process nearest neighbor and neural net gave the best accuracy among them that are 90.77 percent and 92.31 percent. Here is given a chart representing the accuracy differences for each algorithm.

Name of Algorithm	Accuracy rate in percentage
Nearest Neighbors	36.67
Linear SVM	63.33
RBF SVM	30
Gaussian Process	30
Decision Tree	30
Random Forest	23.33
Neural Net	70
Ada-boost	36.67
Naive Bayes	47
QDA	26.67

Table 4.1 Accuracy rates based on reducing 30

Name of Algorithm	Accuracy rate in percentage
Nearest Neighbors	90.77
Linear SVM	75.38
RBF SVM	24.62
Gaussian Process	25.52
Decision Tree	61.54
Random Forest	53.85
Neural Net	92.31
Ada-boost	56.92
Naive Bayes	50.77
QDA	32.31

Table 4.2 Accuracy rates based on 80:20 splitting

The algorithms with the top 3 performance are Neural Net with 92.31 percent, SVM with 75.38 percent and nearest neighbor with 90.77 percent. Consequently, these algorithms were pushed further by pre-processing the data and by finding the best optimizing parameters for the algorithms through exhaustive searches. In this experiment, nearest neighbor almost always lagged considerably behind compared to other two because of over fitting issue and both neural net along with SVM performed very close to each other. After all the pre-processing such as smoothing, scaling, label encoding applied, the performance for both neural net and SVM got good amount of accuracy.

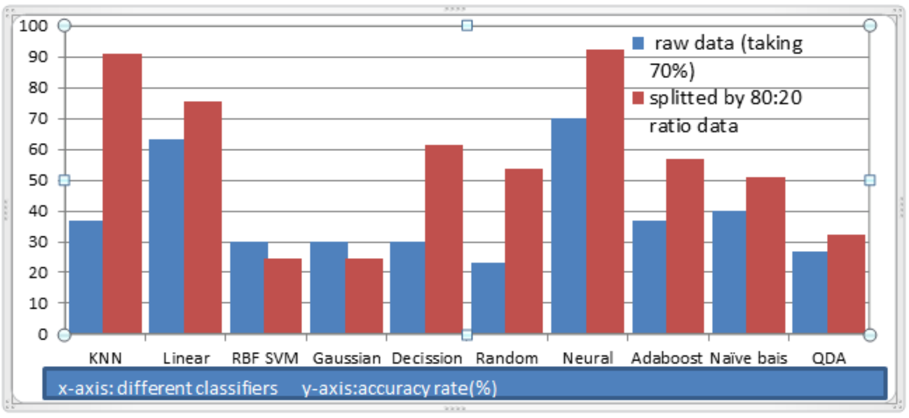

Fig. 4.14 Different classifiers rates on two types testing

With 80-20 percent training and test data split, among 65 test example Neural net manages to predict 63 correctly and only 2 wrong. Meanwhile, among the same number of test example SVM also correctly predicts 63 instances and make mistake in case of 2 instances.

In figure 4.16 and 4.17 here is two confusion matrix that defines how our algorithms predicted scissors, face, cat in terms of true label vs predicted label. Here true label and predicted label for SVM shows that it could successfully identify them by representing 21 scissors, 27 faces and 15 cats. Additionally from the neural net 21of scissors, 26 faces and 16 cats we could identify successfully. Here we can also see that identification of wrong objects are so much minimal. That’s really a noticeable point for further identifying processes.

4.6.3 Reason Behind Different Accuracy

We have found different accuracy on each classifier because of training data set into two parts. By these we could understand that by splitting data into 80:20 ratio that is more effective on training data set. Because by splitting data into this specific ratio most of the time it is proven best for training machine. Giving 80 percent data for training into machine could

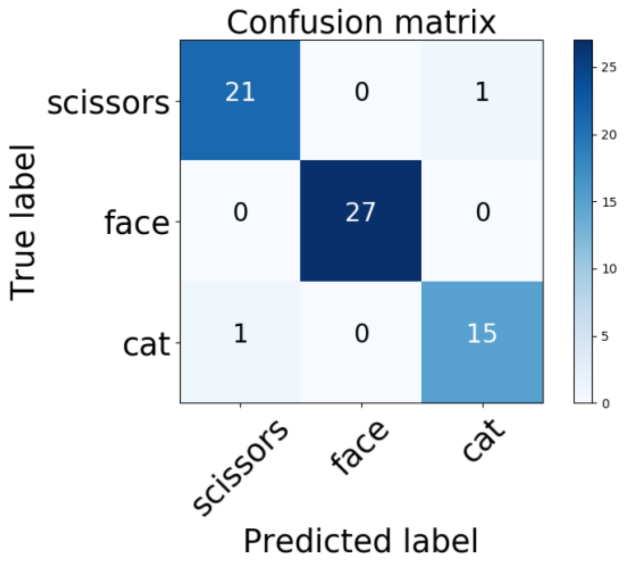

Fig. 4.15 Prediction correctness of SVM

properly learn the data .By this after giving 20 percent data the machine can easily predict on the data and gives a good accuracy .By this method we can also avoid over fitting of the data. Besides that training the data by avoiding 30 from the beginning is actually sometimes confuse the machine to predict for which thing. It gives us accuracy at a low rate because machine could not learn the data properly in this process. We think that's the main difference on getting different results on each training.

4.6.4 Parameter Fixation And Grid Search

After splitting the data and finding accuracy we could identify linear SVM, nearest neighbor and neural net gave a good amount of accuracy. Meanwhile setting the fix parameter for SVM in kernel =RBF and c=100 we have got 96.92 percent accuracy which so much high from all other processes. After that we have done grid search on setting different linear and nonlinear parameter also with different values of c starting from 1-5000. From the grid search we have got 93.82 percent accuracy for the best parameter kernel-poly, c=1000. The reason behind giving a great amount of accuracy on fixing a parameter is because of nonlinear parameter for c=100. At this rate the classifier gave the best result for fitting the curve properly. First in RBF we have used gamma function which is the inverse of the standard deviation of the RBF kernel (Gaussian function), which is used as similarity measure between two points. Intuitively, a small gamma value defines a Gaussian function with a large variance. Here C is a trade-off between training error and the flatness of the solution. The larger C is the

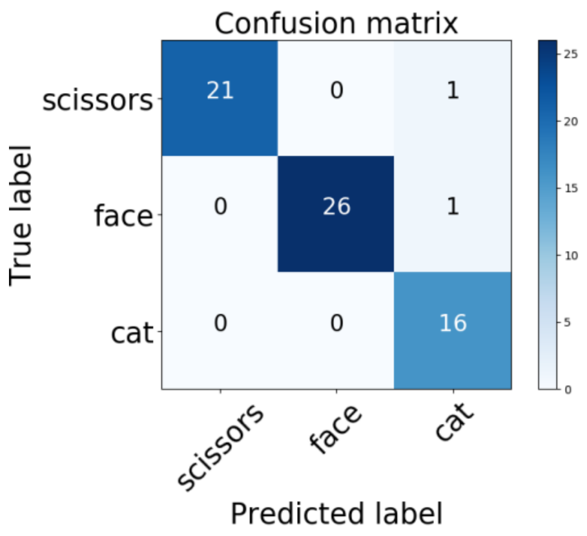

Fig. 4.16 Prediction correctness of Neural Net

less the final training error will be. But if we increase C too much we might risk losing the generalization properties of the classifier, because it will try to fit as best as possible all the training points (including the possible errors of our data set). In addition a large C usually increases the time needed for training. If C is small, then the classifier is flat (meaning that its derivatives are small - close to zero, at least for the Gaussian rbf kernel this is substantiated theoretically). We have to find a C that keeps the training error small, but also generalizes well. That's why by several testing we found the value 100 of c in RBF kernel which gives the best accuracy in fixing parameter. But in grid search the accuracy rate have become low from the fixed parameter because of using linear and nonlinear kernels .By randomizing values of c of different kernels we have got approximately 93 percent which is pretty good according to us.

On the contrary we have also used fixed parameter on neural net to find how much accuracy it can give by fixing parameter method. And with surprisingly it also gave such good result that is accuracy of 96.92 percent.We has used mlp classifier for neural net whereas hidden layer sizes are 464 and for activation we have used tanh. Here hidden layer sizes 464 have been used because in most of the cases for mlp classifier 464 values is like a standard value for tuning the parameters. Activation functions are really important for Artificial Neural Network to learn and make sense of something really complicated and Non-linear complex functional mappings between the inputs and response variable. And for the tanh value it gave such a good amount of accuracy because it is a hyperbolic tangent function which also has a mathematical formula. Its output is zero centered because its range in between -1 to 1. Hence optimization is easier in this method so for practice it is always preferred over

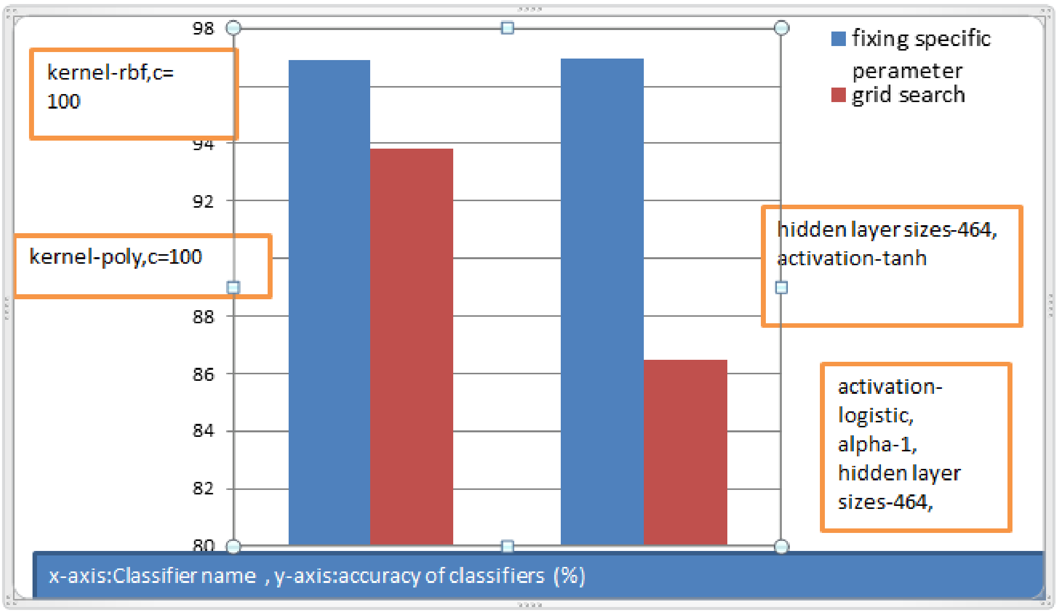

Fig. 4.17 Difference between fixed parameter and grid grid search on two types of classifiers

sigmoid function. On the grid search part we have used 464 for hidden layer sizes and also used different types of parameters such as identity, logistic, tanh, relu and for a fixed range of alpha that is(.0001-1) and for solver sgd ,adam we have got 86.48 percent accuracy for activation-logistic, alpha -1,hidden layer sizes-464,solver-adam.

Here we could find that foe grid search accuracy rate has been decreased due to using randomized parameter. For activation though relu avoids and rectifies vanishing gradient problem its limitation is that it should only be used within Hidden layers of a Neural Network Model. Another problem with ReLu is that some gradients can be fragile during training and can die. It can cause a weight update which will makes it never activate on any data point again. Furthermore sigmoid has vanishing gradient problem. Secondly, its output isn't zero centered. It makes the gradient updates go too far in different directions. $0 <$ output < 1, and it makes optimization harder. Sigmoid saturate and kill gradients also have slow convergence. Also here different value of alpha is set with two types of solver. Solver increases the parameter accuracy due to working on different activations.

At the end we could find a conclusion that for nonlinear SVM and neural net we have best accuracy for fixing parameter. So we can say that by optimizing different values for object detection we can have best accuracy from neural net and nonlinear SVM algorithm process. With a decent amount of accuracy in case of both Neural Net and SVM, the two algorithms were tested with different samples in the training and test splits. Furthermore, so far every experimentation was being done on the data of the second subject. Now that the best algorithm with optimal parameters was found, the algorithm was also applied for rest of

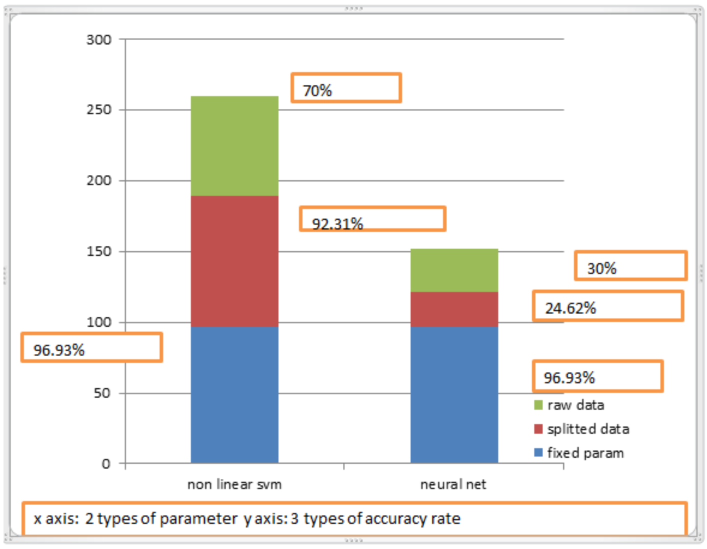

Fig. 4.18 Three types of accuracy rates on two parameters

the 5 subjects. The accuracy of neural network very slightly edges the accuracy of SVM in case of subject 4, 5 and 6 while SVM surpasses neural network in case of subject 1, 2 and 3. However, in all the cases both the algorithm managed to hit around 93-96 percent accuracy mark with 95.1 percent of average accuracy from Neural Net and 94.8 percent of average accuracy from SVM.

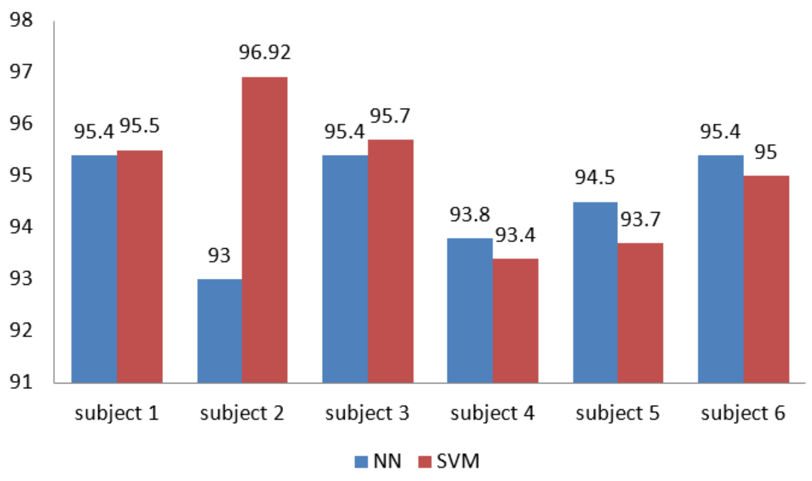

Fig. 4.19 Results across the entire data set

Chapter 5

Conclusion

Though several studies showed different techniques of brain decoding for visualizing any object, in our thesis paper we have tried to represent how easily visualized object can be determined from decoded data. We tried to search for the best machine learning algorithm that gives us the most accurate result and we have presented those results in graphical format. We also tried to compare the results of different algorithms and showed how they varied from one another. We believe that our work can be used as a way to differentiate different algorithms in an easier way for determining visualized object from brain decoding as we have represented our work in a consistent topographic arrangement. The proposed model in this paper extracts necessary data from fMRIs that was taken while visualizing certain objects and tries to find unique patterns of visualizing different objects in the brain. Neural net and SVM were the most helpful for finding patterns and over 93 percent of samples were managed to be uniquely identified. That being said, haxby data set (2001) has become quite old and future works can be done by manually creating more data set on other objects beside the 8 objects of the given data set in this paper. Additionally, for our future work we are working on categorizing different objects by creating different data sets manually in such a way to predict what type of object it is. Furthermore, if we can prepare machine learning algorithms in a way so that they can determine the brain activity from certain visual stimuli, it can be used to treat or recover any anomaly that may happen in terms of the visual perception of the brain. In the distant future it may also become possible to provide artificial brain simulation with the help of machine learning techniques. Therefore, as we can see here, there are lots of possibilities in the proposed idea.

References

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

[2] Chao, L. L., Haxby, J. V., and Martin, A. (1999a). Attribute-based neural substrates in temporal cortex for perceiving and knowing about objects. *Nature neuroscience*, 2(10):913.

[3] Chao, L. L., Martin, A., and Haxby, J. V. (1999b). Are face-responsive regions selective only for faces? *Neuroreport*, 10(14):2945–2950.

[4] Chen, C.-C., Tyler, C. W., and Baseler, H. A. (2003). Statistical properties of bold magnetic resonance activity in the human brain. *NeuroImage*, 20(2):1096–1109.

[5] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

[6] Cox, D. D. and Savoy, R. L. (2003). Functional magnetic resonance imaging (fmri)"brain reading": detecting and classifying distributed patterns of fmri activity in human visual cortex. *Neuroimage*, 19(2):261–270.

[7] Demanuele, C., Kirsch, P., Esslinger, C., Zink, M., Meyer-Lindenberg, A., and Durstewitz, D. (2015). Area-specific information processing in prefrontal cortex during a probabilistic inference task: a multivariate fmri bold time series analysis. *PloS one*, 10(8):e0135424.

[8] Edelman, S., Grill-Spector, K., Kushnir, T., and Malach, R. (1998). Toward direct visualization of the internal shape representation space by fmri. *Psychobiology*, 26(4):309–321.

[9] Friston, K. J., Rotshtein, P., Geng, J. J., Sterzer, P., and Henson, R. N. (2006). A critique of functional localisers. *Neuroimage*, 30(4):1077–1087.

[10] Gauthier, I., Skudlarski, P., Gore, J. C., and Anderson, A. W. (2000a). Expertise for cars and birds recruits brain areas involved in face recognition. *Nature neuroscience*, 3(2):191.

[11] Gauthier, I., Tarr, M. J., Anderson, A. W., Skudlarski, P., and Gore, J. C. (1999). Activation of the middle fusiform'face area'increases with expertise in recognizing novel objects. *Nature neuroscience*, 2(6):568.

[12] Gauthier, I., Tarr, M. J., Moylan, J., Skudlarski, P., Gore, J. C., and Anderson, A. W. (2000b). The fusiform "face area" is part of a network that processes faces at the individual level. *Journal of cognitive neuroscience*, 12(3):495–504.

[13] Grill-Spector, K. and Kanwisher, N. (2005). Visual recognition: As soon as you know it is there, you know what it is. *Psychological Science*, 16(2):152–160.

[14] Grill-Spector, K., Kushnir, T., Edelman, S., Avidan, G., Itzchak, Y., and Malach, R. (1999). Differential processing of objects under various viewing conditions in the human lateral occipital complex. *Neuron*, 24(1):187–203.

[15] Grill-Spector, K. and Weiner, K. S. (2014). The functional architecture of the ventral temporal cortex and its role in categorization. *Nature Reviews Neuroscience*, 15(8):536.

[16] Haxby, J. V., Ishai, A., Chao, L. L., Ungerleider, L. G., and Martin, A. (2000). Object-form topology in the ventral temporal lobe: response to i. gauthier (2000). *Trends in cognitive sciences*, 4(1):3–4.

[17] Haxby, J. V., Ungerleider, L. G., Clark, V. P., Schouten, J. L., Hoffman, E. A., and Martin, A. (1999). The effect of face inversion on activity in human neural systems for face and object perception. *Neuron*, 22(1):189–199.

[18] Ishai, A., Ungerleider, L. G., Martin, A., Schouten, J. L., and Haxby, J. V. (1999). Distributed representation of objects in the human ventral visual pathway. *Proceedings of the National Academy of Sciences*, 96(16):9379–9384.

[19] Kanwisher, N., McDermott, J., and Chun, M. M. (1997). The fusiform face area: a module in human extrastriate cortex specialized for face perception. *Journal of neuroscience*, 17(11):4302–4311.

[20] Leeds, D. D., Seibert, D. A., Pyles, J. A., and Tarr, M. J. (2013). Comparing visual representations across human fmri and computational vision. *Journal of vision*, 13(13):25–25.

[21] Logothetis, N. K. and Sheinberg, D. L. (1996). Visual object recognition. *Annual review of neuroscience*, 19(1):577–621.

[22] McCarthy, G., Puce, A., Gore, J. C., and Allison, T. (1997). Face-specific processing in the human fusiform gyrus. *Journal of cognitive neuroscience*, 9(5):605–610.

[23] Monti, M. M. (2011). Statistical analysis of fmri time-series: a critical review of the glm approach. *Frontiers in human neuroscience*, 5:28.

[24] Rocha-Miranda, C., Bender, D., Gross, C., and Mishkin, M. (1975). Visual activation of neurons in inferotemporal cortex depends on striate cortex and forebrain commissures. *Journal of Neurophysiology*, 38(3):475–491.

[25] Tanaka, K. (1996). Inferotemporal cortex and object vision. *Annual review of neuroscience*, 19(1):109–139.

[26] Zhu, J., Rosset, S., Tibshirani, R., and Hastie, T. J. (2004). 1-norm support vector machines. In *Advances in neural information processing systems*, pages 49–56.