Final Degree Work for
Computer Engineering Degree and Mathematics Degree

Stacked Sequential Multi-class Discriminative Dictionary Learning for Brain MRI Segmentation.

Javier Noguera Ropero

Director: Laura Igual Muñoz
Universitat de Barcelona
Made in: Department of Applied
Mathematics and Analysis
January 2015

Acknowledgements

I would like to thank Laura Igual for her great work as director of labour, her support and assistance in carrying out the work in addition to the motivation that she has given me. I also want to thank Oualid for the great help he has provided to me. Finally I want to thank my family for their support during the development of this work.

Contents

List of Tables

List of Figures

Abstract

The segmentation of brain structures in Magnetic Resonance Imaging is a challenging problem due to the low contrast and resolution of the structures and the noisy images. The Discriminative Dictionary Learning Segmentation is a classification technique which has been applied for different image processing problems such as compression, image denoising and recently in Magnetic Resonance Imaging segmentation. We consider the segmentation problem as a classification problem and apply Discriminative Dictionary Learning Segmentation to solve it using a patch-based representation and minimising the reconstruction error. The main limitation of this method is that the classification is performed independently for each voxel. We propose to add contextual information for the classification of the image voxels using Stacked Sequential Learning as a second stage. We define a feature vector from the classification results of Multi-class Discriminative Dictionary Learning and apply a decision tree classifier. We validate the proposal using a public database presented in the SATA Challenge. Using the two stages Stacked Sequential Multi-class Discriminative Dictionary Learning Segmentation method, we obtain an improvement of X% with respect to Multi-class Discriminative Dictionary Learning.

Chapter 1

Introduction

Magnetic resonance imaging (MRI) is a medical imaging technique widely used in applications for medical diagnosis. MRI provides a useful visualisation of the different brain tissues. The segmentation of brain structures in MRI is a challenging problem due to the low contrast and resolution of the structures and the noisy images [6]. Several methods for segmentation of brain structures in MRI have been proposed such as atlas and multi-atlas based methods or methods based in single voxel intensity information or patches, both using or not learning and based on similarity or reconstruction errors.

In this work, we focus on Sparse Representation Classification (SRC) and Discriminative Dictionary Learning (DDL) classification strategies. These strategies have been applied for different image processing problems, as image compression [7] and image denoising [8]. We review these techniques applied for MRI segmentation and its multi-class version using label consistency (MDDLS) [1]. In MDDLS strategy voxels are labelled independently using a learned sparse representation and linear classifier by minimising a patch reconstruction error. The main limitation of these methods is that the classification is performed independently for each voxel. Thus, this method is not exploiting the inherent sequential relationships present in neighbour data classification.

Here, we propose to add contextual information for the classification of image voxels by adding neighbour information. We follow an strategy inspired by Stacked Sequential Learning. Sequential learning algorithms take benefit of the sequential relationships of neighbour data classification in order to improve generalisation [13]. We present the Stacked Sequential Discriminative Dictionary Learning Segmentation (SS-MDDLS) as a two stage method which first apply MDDLS and second learn a decision tree classifier using contextual information from MDDLS classification result. Then, we apply the trained classifier to label image voxel from the test MRI images. In this way, we capture the possible confusion areas and try to learn the potential patterns of MDDLS systematic errors to avoid them.

The proposed method SS-MDDLS is tested to segment the four Basal Ganglia sub-structures (Caudate, Putamen, Pallidum, Accumbens) on a public

databases presented in the SATA Challenge [14]. It shows a performance improvement compared to one stage MDDLS approach.

Chapter 2

Approaching the problem

2.1 Magnetic resonance imaging

Nowadays it has been a huge increase on the importance of imaging in research and clinical fields. That fact is due to the existence of several non-invasive methods for obtaining images from patients. Those images enables researchers to see how a process or a disease develops and to check if therapies applied are working in a correct way. Images help to diagnose too by allowing doctors to make an early recognition of diseases or abnormal events [2].

There exist many modalities of imaging, one of them is magnetic resonance imaging (MRI) also known as nuclear magnetic resonance imaging (NMRI) or magnetic resonance tomography (MRT) [6]. This method is used in radiology and widely applied in hospitals for medical purposes such as diagnosis and staging or follow-up of diseases. MRI has been improved and has become a volume imaging technique which provides a good contrast among different soft tissues. This fact is useful for example in the brain imaging due to its properties.

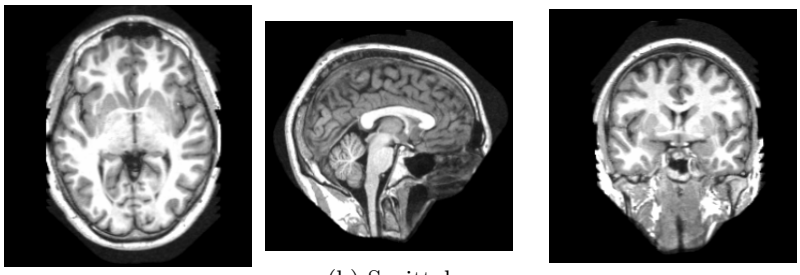
We can obtain three different types of images using MRI.

- T1-weighted: spin-latice relaxation [9].
- T2-weighted: spin-spin relaxation time [10].
- PD-weighted: proton density.

From which the most interesting for us are T1-weighted and T2-weighted because they are the primary determinants of signal intensity and contrast.

Nowadays we can obtain volume images so that we can get three different views from it:

- Axial plane.
- Sagittal plane.
- Coronal plane.



(a) Axial (b) Sagittal (c) Coronal

Figure 2.1: Three different views from MR images.

Many interesting properties of those image mentioned in [2] are:

- Excellent capability for soft tissue structures.
- High resolution.
- High signal to noise ratio.
- Using different pulses we can get multi-channel images with variable contrast; what can be used for segment and classify structures.

One of the problems we can find in MRI is the difficulty of obtaining uniform image quality. There are other problems such as being expensive and the need of spending time to take them.

2.2 Sub-cortical structures

Our brain is formed by several anatomical structures, cortical and sub-cortical structures [12].

The cortical structures are frontal lobe, parietal lobe, occipital lobe, temporal lobe.

The Sub-cortical structures are hippocampus, cerebellum, amygdala, basal ganglia.

In this project we have been working with basal ganglia structure [11] whose main components are caudate nucleus and putamen (both of them form the structure called striatum), the globus pallidus, the subtantia nigra, the nucleus accumbens and the subthalamic nucleus. From those substructures we are performing the segmentation of the most important: caudate, putamen, globus pallidus and accumbens 2.2.

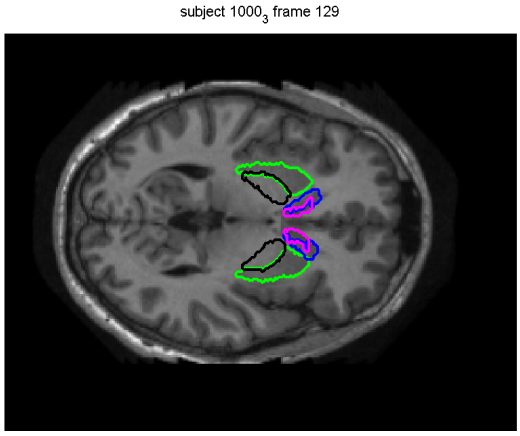


Figure 2.2: The four basal ganglia main structures manually segmented.: Caudate, Putamen, Pallidum, Accumbens

2.3 Segmentation methods

Since manual labelling is a laborious task liable to inter and intra- variability depending on the clinical expert who does it and differences among subjects, it is desirable an automated technique to MRI segmentation There is still a challenge in developing faster and more accurate automatic segmentation despite the fact that many methods which can do that task exist.

In this section some methods are briefly explained in order to give an approach for the state-of-the-art. This section is focused on magnetic resonance imaging segmentation methods giving more importance to those that are related with our proposal.

2.3.1 Atlas based

As it is said in [2] atlas based methods have outperformed other algorithms and those methods have had an increasing popularity. Unlike manual segmentation, atlas-based algorithms, as its name says, use atlases.

An atlas consist of two image volumes 2.3:

- Template which is an intensity image
- Labelled image which is a segmented image

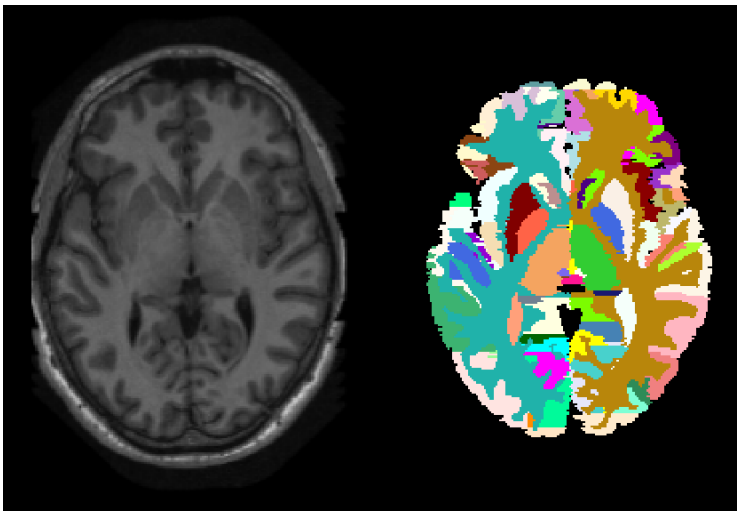


Figure 2.3: On the left side we show an MR image and on the right side the corresponding labelled atlas [2].

The target image is registered with the template and then labels are *propagated* to obtain the segmentation. Hence we have a registration problem.

The process is performed in two steps. A global registration for an initial alignment and a local registration for specific deformations. Registration has a high computation cost because it involves a complex discrete optimisation problem [2]. To carry out the segmentation process of the target image these label propagation is performed which consists of transforming the manual labelling of the atlas using the mapping determined during registration. This process might fail if the target volume differs too much from the atlas resulting in a poor segmentation.

The paper ***BrainGraph: tissue segmentation using the Geodesic Information Flows framework.*** *M. Jorge Cardoso, Marc Modat, and Sebastien Ourselin. UCL Centre for Medical Image Computing, London, UK* included in the SATA Challenge [14] shows a new framework for tissue segmentation. Population tissue priors are very important for an accurate segmentation. However, local brain different morphologies, algorithmic limitations in registration and mapping from each subject to a group-wise space is error prone.

In this framework [14], morphological similarity between pairs of images in a population it is used as a Markov Random Field constraint in the segmentation algorithm. This can be interpreted as an iterative generation of highly adapted subject specific priors from the locally most similar images in the database so that a population smoothness in segmentation procedure is introduced without the use of group-wise priors. This performance allows to segment subjects morphologically very different from the training set.

2.3.2 Multi-atlas label system

Segmentation errors produced by atlas-based methods are classified into systematic errors and random errors [3].

Systematic errors are those which occur consistently, they usually describe a systematic pattern between automatic and manual segmentation. Systematic errors might be caused by errors in the registration process, partial volume effects or bias in manual labelling of atlases. One example of systematic error could be under-segmentation which consists of making the segmented structure a bit smaller than the manually segmented, for example due to a different criterion while assigning labels to a voxel.

Random errors are those which might be caused by image noise or subject variation. To reduce random errors we can use multiple atlases or selecting the most similar atlases for a given image. While using multiple atlases we can capture the variability of target regions better than if we only use one single atlas. When performing multi-atlas methods, a set of atlases is registered pairwise to the target image in order to being able to perform label propagation, after that, all propagated atlas labels are fused to generate the segmentation result. This method corrects a high amount of random errors which appear during registration resulting in a more accurate segmentation. Some of multi-atlas based methods are based on similarity among registered atlases for a target image.

There are other performing strategies such as combination of several segmentations and refine it iteratively, those strategies do not consider the quality of the image registration. Atlas selection strategies depending on the information an atlas can contain and stopping when no improvement is expected and so on [2]. The problem of atlas selection strategies is that requires atlases to be registered first.

Label fusion strategies have shown an improvement in results too. That fusion takes part at the voxel level using one of the strategies: nearest neighbours, linear interpolation, majority voting, etc. A limitation of label fusing methods is that weights are computed independently for each atlas. This approach is sensitive to registration error too. To solve that it has been proposed local weighting voting.

We can conclude that multi atlas segmentation requires a pairwise accurate registration between atlas and target which is computationally expensive.

In fusion of multiple segmentation results, such as multi-atlas based algorithms, a majority vote is used as the basis of comparison of segmentation accuracy [14]. In the paper which appears in [14] a tailored majority method is shown. It improves the majority vote by choosing a threshold value that deviates from 50%, a number which is usually used in majority voting. At least 50% of voxels of the contributing segmentations indicates the class of the target voxel. This technique is applied in multi-atlas based methods.

This method it is also called "flexible majority" where "x% or more of the results agree" in the fused label vote where "x" is called "tailored majority vote value" and this performance is shown to out performs the traditional majority

vote. This value can be found by using the Dice similarity Coefficient of the overlap between ground truth and target image [14].

2.3.3 Intensity based: Combined global-local intensity mixture model

Intensity based methods which use classifiers, rely on contrast between tissue types in feature space and adequate signal compared to image noise [14]. Statistically an optimal boundary is identified to separate two tissue classes.

Many methods are proposed combining multi-atlas and intensity based in order to increase segmentation accuracy due to the fact that systematic errors occur consistently and, for some of them, is relatively easy to capture patterns correlated to them.

A method which works in that way is proposed in a paper found in [14]. That method consists of combining the patient global intensity with a population local intensity model.

2.3.4 Learning based

In order to reduce systematic errors some proposed works combine multi-atlas segmentation and learning-based methods [2]. If we suppose that the majority of systematic errors in segmentation occur consistently from subject to subject, then we can apply a wrapper method which try to learn intensity, spatial and contextual patterns associated to those errors. The wrapper method attempts to correct these systematic errors.

2.3.5 Patch based

The accuracy of non-rigid registration, fusion rules, selection of labelled images and labelling errors in manual segmentation are many key-points of registration-based label propagation [2].

Different from multi-atlas based, patch-based methods use local similar candidates, called image patches, to estimate labels. As it is explained in [3] these methods obtain a label for every voxel by using similar image patches from coarsely aligned atlases. Image patches are extracted in a predefined neighbourhood around a voxel. According to the similarity of target patch and atlas patches, is given a weight to that patch. The final label of the target voxel is given by fusing the labels of central voxels of every patch. Using these non-local patch-based segmentation we can avoid the need of accurate non-rigid registration so that we can get more computational efficiency.

However, image similarities over small image patches might not be an optimal estimator [3].

2.3.6 Reconstruction based

In [3] a segmentation method based on image patch reconstruction using discriminative dictionary learning [4] is proposed. Different from the idea of comparing the similarity between patches it uses a dictionary and a linear classifier which is learned from a template patch library for every voxel in the target image.

The surrounding patch to the target voxel can be reconstructed thanks to the dictionary and the label of the voxel is estimated by the classifier. As it is said in [3] the dictionaries can be learned offline and segment online.

Note that non-local assumption means that central voxel of similar patches belong to the same structure so that similar patches contribute to a better result.

2.3.7 Registration methods

ANTs System

Since many methods need an accurate registration of templates, this process has become an important challenge. In [14] the ANTs System is proposed for image registration. These registrations consist of an initial transformation, the identity if it is possible, of the image followed by several registration stages with increasing degrees of freedom. Each stage begins with k similarity metric definitions with many multi-resolution strategy parameters. Then subsequent stages are added in the same way. Finally many optimisation options and pre/post-processing details are taken by ANTs.

Discrete optimisation

Discrete optimisation is a technique used in registration-based segmentation propagation.

In the paper ***Uncertainty Estimates for Improved Accuracy of Registration-Based Segmentation Propagation using Discrete Optimisation***. *Mattias P. Heinrich, Ivor J.A. Simpson, Mark Jenkinson, Sir Michael Brady, and Julia A.Schnabel. Institute of Biomedical Engineering, Department of Engineering, University of Oxford, UK. Oxford University Centre for Functional MRI of the Brain, UK. Centre for Medical Image Computing, University College London, UK. Department of Oncology, University of Oxford, UK* contained in the SATA Challenge [14] it is described a method which incorporates uncertainty estimates which are evaluated over the space of possible transformations. Optimal marginals distributions for a large range of local displacements are calculated which can be converted into probabilities. These probabilities, information of uncertainty, can be used to improve segmentation accuracy.

Chapter 3

Multi-Class Discriminative Dictionary Learning Segmentation and Stacked Learning

3.1 Sparse representation classification and discriminative dictionary learning

The searching for sparse coding of signals has been increasing in recent years. It has been shown that sparse representation provides a high performance in several applications such as image denoising, image painting and image compression [2]. This idea has also been used for methods in pattern classification such as in the Support Vector Machine where sparsity can be related to learnability of an estimator.

The aim of sparse representation is to reconstruct an input signal, for example an image patch, as a linear combination of a reduced number of signals taken from a dictionary.

Let $y \in \mathbb{R}^n$ be signal. Let $D \in \mathbb{R}^{n \times k}$ an over-complete dictionary ($k > n$) where each column is called *atom*.

We can obtain a representation of y by the following linear system:

$$y = D\alpha \tag{3.1}$$

Where $\alpha \in \mathbb{R}^K$ is the sparse code of the signal y.

$$\alpha = argmin_\alpha \|y - D\alpha\|_2^2 \qquad subject\ to\ \|\alpha\|_0 \leq T \tag{3.2}$$

Being T a sparsity constraint and $\|y - D\alpha\|_2^2 \leq \epsilon$ reconstruction error and ϵ an error tolerance.

As a mesure of sparsity they suggest the ℓ^0 pseudo-norm [1] [2, 5].

Hence we have a minimisation problem which is NP-hard but we can find suboptimals solutions by iterative methods. Since ℓ^0 is not convex, the ℓ^1 is de closest convex function to carry out that minimisation, it was shown that both are equivalent as it is said in [2] if they are sufficiently sparse.

$$\hat{\alpha} = argmin_{\alpha} \|y - D\alpha\|_2^2 \qquad subject\ to\ \|\alpha\|_1 \leq T \tag{3.3}$$

Using the Lagrangian method, we can rewrite the problem as follows:

$$\hat{y} = min_{\alpha}\ \frac{1}{2}\ \|y - D\alpha\|_2^2\ +\ \lambda\|\alpha\|_1 \tag{3.4}$$

where $\lambda\|\alpha\|_1$ is the sparsity-inducing regularisation and $\lambda > 0$ is the Lagrangian multiplier which balances the trade-off between reconstruction error and sparsity.

This equation (4) can be solved by several sparse coding methods [2]. Equation (4) is solved using the Lasso [2] method [2] in this case.

Two issues to be aware of in the sparse coding model are the sparsity constraint and whether the squared loss term is effective enough to characterize the signal fidelity [2].

Although the sparse representation is a good way for an accurately reconstruction of a signal y such as denoising and image inpainting, for classification is more important if it is discriminative or not for the given signal classes than a small reconstruction error. Recently it has been shown that sparse representation classification (SRC) has been successful in image processing and texture classification and face recognition [2, 4, 5].

In SRC, the target sample is represented as a sparse linear combination of the training samples [2]. Similar samples can be combined to make approximately a sample from the same class due to the fact that, in classification terms, samples from a single class lie on a linear subspace approximately meanwhile the rest cannot offer a linear representation as compact as the ones from the target sample class. That is why it can be said that SRC can be discriminative.

The sparse code α can be used as a feature for classification [5].

Given a set of signals $Y = [y_1, ..., y_N]$ we assume that a dictionary D that gave rise to the given signal examples via sparse representation exists and solves (1) for each signal y_i given its sparse coding α_i. Now we could wonder how can D be recovered? There exist many algorithm to solve that such as *method of optimal directions* [2] and K-SVD algorithm [2, 4, 5]. Both methods are iterative approaches designed to minimise

$$min_{\alpha,D}\ \|y - D\alpha\|_2^2 \qquad subject\ to\ \|\alpha\|_0 \leq T, \tag{3.5}$$

and find the optimal dictionary.

First of all a dictionary D is initialised, then we have a loop composed of two stages [2]:

[1] The ℓ^0 pseudo-norm of a vector is the number of nonzero coefficients of that vector.

[2] Least absolute shrinkage and selection operator [15]

1. Sparse coding: Fixed D we fin the best sparse decomposition of each signal as we saw before. 2. Dictionary update: here we find a difference between K-SVD and MOD. In MOD, the decompositions α_i are fixed and a least square problem solved updating all atoms simultaneously. In K-SVD, values of non-zero coefficients in the α_i are not fixed and are updated at the same time as D. K-SVD receive its name from K-means.

The problem of K-SVD is that is not suitable for classification due to the fact that dictionaries must be representative and discriminative.

It has been developed a D-KSVD (discriminative K-SVD) [4] that uses labels of training data to incorporate a classifier into K-SVD and extends it by incorporating the classification error into the objective function. The complexity of the method is bounded to K-SVD complexity.

D-KSVD solves the next problem:

$$\begin{aligned}\langle D, W, \alpha\rangle = argmin_{D,W,\alpha}\|Y - D\alpha\|_2^2 + \ \lambda\|H - W\alpha\|_2^2 \\ + \beta\|W\|_2^2 \text{ subject to } \|\alpha\|_0 \ \leq T,\end{aligned} \tag{3.6}$$

where $W \in \mathbb{R}^{c\times k}$ are parameters for a linear classifier $W = H * \alpha$. Each column of $H \in \mathbb{R}^{c\times N}$ is a $h_i = [0, ..., 1, ...0]$, where non-zero position indicates the class, being c the number of classes. The term involving H is the classification error and $\|W\|_2$ is the regularisation penalty. Those two terms should be minimised, hence we have a multivariate ridge regression problem [2, 4].

General steps for the Baseline Algorithm [4]:

1. Initialise D and α using K-SVD solving (5).
2. Calculate W when D and α are fixed.
3. Calculate α when D and W are fixed.
4. Calculate D when α and W are fixed.
5. Iterate 2 to 4 until some criterion are met.

But this Baseline algorithm can only find an approximate solution to (6) because in each step it finds a solution for every subproblem in the equation.

The problem can be rewritten as follow [3]:

$$\begin{aligned}\langle D, W, \alpha\rangle = argmin_{D,W,\alpha}\| \begin{pmatrix} P_L \\ \sqrt{\beta_1} H \end{pmatrix} - \begin{pmatrix} D \\ \sqrt{\beta_1} W \end{pmatrix} \alpha\|_2^2 \\ + \beta_2\|W\|_2^2 \text{ subject to } \|\alpha\|_0 \ \leq T\end{aligned} \tag{3.7}$$

where $P_L = Y$ which corresponds to patches (patch library, what we called set of signals before) and scalars β_1 and β_2 are parameters controlling the contribution of each term.

Given the dictionary, sparse coding using Lasso is computed:

$$\hat{\alpha}_t = argmin_{\alpha_t} \|p_t - \hat{D}_t \alpha_t\|_2^2 + \lambda_2 \|\alpha\|_1 \tag{3.8}$$

where p_t is the target patch (y_i signal), $\hat{D}_t$ dictionary for the target patch and α_t the target patch sparse decomposition.

Classification of target patch [3, 4, 5]:

$$h_t = \hat{W}_t \hat{\alpha}_t \tag{3.9}$$

where $h_t = [0, ..., 1, ..., 0]$ should have ideally one none-zero element which indicates voxels class.

Labelling:

$$v_t = argmax_j h_t(j), \qquad j = 1, ..., C \tag{3.10}$$

3.2 Segmentation by using multi-class discriminative dictionary learning with label consistency

In multi-class dictionary learning we focus in reconstruction [2], it is assumed that the target patch (the one that is being labelled) can be represented by a few template patches from the same structure as it is explained in sparse representation, represented by a few representative atoms from the dictionary. Two phases for labelling a voxel: coding and classification.

In discriminative dictionary learning, each voxel could be classified in two different classes: structure or background (not in the desired structure). In multi-class discriminative dictionary learning the number of classes to which a voxel can belong are more than two, several structures and the background (any desired structure). Hence H matrix has more rows than before, one for each class.

What is important now is the new constraint added, the label consistency, called discriminative sparse code error which is added to the objective function:

$$\begin{aligned} \langle D, W, A, \alpha \rangle = argmin_{D,W,A,\alpha} \|P_L - D\alpha\|_2 + \ \beta_1 \|H - W\alpha\|_2 \\ + \beta_2 \|Q - A\alpha\|_2^2 + \beta_3 \|W\|_2 \quad \text{subject to } \|\alpha\|_0 \ \leq T, \end{aligned} \tag{3.11}$$

where $Q \in \mathbb{R}^{k \times N}$ allows to specialise some atoms in the patches of a particular class due to the fact that they are the "discriminative" sparse codes for the signals. It happens if the non-zero values for every $q_i \in \mathbb{R}^k$ occur where the input signal y_i and the dictionary items d_k share the same label.

Lets see an example [5]:

$$
Q_{K\times n} = \begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{array}
\begin{array}{c}
\begin{array}{cccccccccc} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} \end{array} \\
\left[\begin{array}{cccccccccc}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{array}\right]
\end{array}
$$

Figure 3.1: Q matrix: Atoms d_1 and d_2 specialised in patches p_1 and p_2 of one class, atoms d_3 and d_4 specialised in patches $p_3 - p_7$ of another class and atoms d_5 and d_6 specialised in patches $p_8 - p_{10}$ of the last class:

Matrix A is a linear transformation matrix which transforms the original sparse codes α_i to be most discriminative in feature space $\mathbb{R}^k$.Hence the term which involve Q and A represents de discriminative sparse-code error which enforce the sparse codes α since it forces the signals from same class to have similar sparse codes [5].

Therefore the unbalance problem is smoothed.

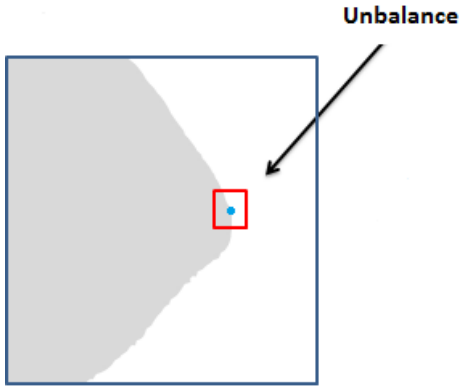


Figure 3.2: Unbalance problem: We can find it in boundary voxels where the number of structure patches and background patches in the patch library for the target voxel is unbalanced.

Learning and labelling proceeds as they do in DDLS. In the paper [5] another way of performing the label consistent K-SVD is shown where a linear predictive classifier is used $f(\alpha; W) = W\alpha$ which we mentioned before. Hence, both classifier and label consistency are applied in order to obtain a better performance.

3.3 Stacked Sequential Learning

In the MDDLS method we classify each voxel independently of its neighbours. In other words, we are not considering the classification of neighbours for classifying

the target voxel. In order to add this valuable information we consider a second stage inspired in Stacked Sequential Learning (SSL).

In this section we are reviewing briefly to Stacked Sequential Learning (SSL) [13, 16].

Despite of their values, neighbour data labels have inherent relationships in many classification problems. Sequential learning takes profit of this fact [13]. Contextual information is useful to solve ambiguous cases in classification.

Sequential learning is a meta-learning method [3] in which an arbitrary base learner is augmented, in this case by making it aware of its neighbourhood labels [16]. Standard classification assumes that samples are independently and identically drawn from a distribution of samples and their labels. Despite this fact, problems in real world can break this assumption.

Stacked sequential learning scheme is based in a two layers classifier. First of all a classifier is trained and tested with the original data set, then it is created an extended data set using the original data with predicted labels from the classifier added. A second classifier is training with this data which gives the final predicted labels. This approach shouldn't be used for problems with a long range sequential relationship because the size of the extended data set increases exponentially. Sequential learning has been addressed from different perspectives such as graphical models, Hidden Markov Models or Conditional Random Fields read [13, 16] for more information about those points of view since we are focusing on meta-learning view which also includes sliding windows and recurrent windows techniques [13, 16].

As it is explained in [16], sequential learning can be applied to multi-class problems. Hence it is needed that classifiers have to be able to deal with multiple classes. Another approach could be decomposing the problem in several binary problems and combine results. Here we find the problem on how can we decompose it in an efficient way and how can we combine the results we obtain.

There is also a generalised version of stacked sequential learning which includes a new block in its pipeline. As before, first of all a classifier is trained. Now appears the difference, the new block defines how the neighbourhood model of predicted labels is created. This block consists of a function which catches the data interaction with a parametrised model in a neighbourhood. Its output is added to the original data to create the extended features which are used to train the second classifier [13, 16].

Four more extensions for multi-scale stacked sequential learning (MSSL) are shown in [16], extension by likelihoods, learning objects at multiple scales, multi-class MSSL and extended data set grouping.

[3] A meta-learning technique uses a combination of different classifiers in order to predict a test example

Chapter 4

Stacked Sequential Multi-class Discriminative Dictionary Learning for MRI Segmentation

In this work, the objective is to segment the basal ganglia structure and four of its sub-structures [11, 12]. We apply MDDLS (multi-class DDLS) method as a first stage, then we incorporate more information obtained from MDDLS to classify each voxel applying Stacked Sequential Learning as a second stage (SS-MDDLS).

4.1 First stage: MDDLS

The first stage is formed by several steps in order to perform a multi-class discriminative dictionary learning (MDDLS) for magnetic resonance images. Sparse representation code (SRC) and binary discriminative dictionary learning (DDLS) methods for segment those images have been performed too [2]. Those steps are:

1. Initialisation and set default parameters.
2. Preprocessing, which consists of four sub-steps [23, 24].
3. Building target crop.
4. Library and reduced library construction.
5. Perform segmentation.
6. Merge cropped segmentation.

7. Inverting registration

Except the segmentation step (step 5), the other steps are common for each method SRC, DDLS and MDDLS.

4.1.1 Initialisation

In this first step the initialisation is performed, default parameters are set, structures are created and tools are included.

We define parameters such as structures and sub-structures to segment with their labels value. The template for the registration process is chosen. So are the number of best atlases, patch and windows size, sampling step, neighbourhood size parameters which is explained in the corresponding section. All those parameters are used by the subsequent pipeline steps.

4.1.2 Preprocessing pipeline

In this step, images are prepared for their segmentation. This step is composed by four sub-steps [23, 24]:

- Denoising
- Non-Uniformity correction
- Registration
- Intensity standardisation

This preprocessing step is performed in order to solve, or at least to reduce, many problems in MRI such as variability caused by image formation.

In order to perform these operations properly, first of all, images should be aligned resetting coordinates.

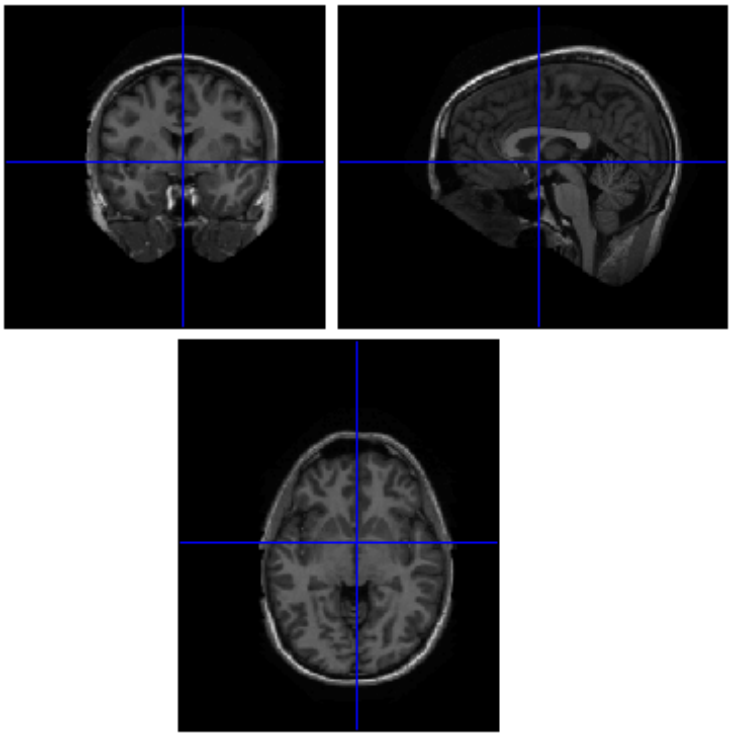


Figure 4.1: Example of resetting coordinates to the origin [2].

Denoising

The problem of noise is closely related to MRI system and it is know that it has a Rician distribution [2, 24].

So that, the denoising step is very important. This step increases image quality to improve subsequent steps performance for quantitative analysis.

Denoising methods applied in this performance are 3D block-wise non-local means filter [2]. To reduce the noise in the image, redundant information is used by the filter despite being more difficult to carry out in 3D images than 2D. For reducing the noise a method based on Median Absolute Deviation estimator for Rician noise [2] is used.

Once this step is done, we obtain denoised images which are used by the next step, the non-uniformity correction.

Non-Uniformity correction

Non-uniformity correction tries to reduce the intensity inhomogeneity problem also called bias field or shading artifact. In radio frequency, the non-uniformity when we obtain data might produce a shading effect. This effect is harmful due to the fact that affects to the qualitative and quantitative analysis of images.

The main feature of this effect is the variability of intensity value for voxels from the same tissue. That is why this issue has become an important problem to be aware of and to solve. This effect can appear for many reasons such as regular calibration or several subject properties. This effect can appear inter- and intra-slice.

To ensure that each tissue has the same intensity within a single image it is used the N3 (non parametric, non uniform, normalisation) intensity non-uniformity correction of Sled [2].

These images are used for registration step.

Registration

First of all, register two images means align them in order to make to see overlapping or differences among their common features. In this process a geometric transformation is performed. This problem is composed by four main components [2]:

- Feature space. Determine what is registered. Algorithm used is feature dependent.
- Search space. Consider two images (the two that we want to register) as two functions $f(x)$ and $g(x)$ in $\mathbb{R}^n$ where $n = 1, 2$, to align them a transformation $T(y)$ has to be found, this transformation must accomplish the next condition: $f(x) = T(g(x))\ \forall x$ supposing that $g(x) = y$. There exist rigid transformation such as translation and rotation or affine transformation for example to consider different scales. So the search space is the type of transformation we choose.
- Search strategy. Once the search space is chosen a transformation and their subsequent transformations have to be chosen.
- Metric similarity. Measure how good is $f(x)$ compared to $T(g(x))$. Mean-square error, correlation, etc.

Two categories for registration, rigid and non-rigid (affine) as it was said before. In this case an affine registration is applied for all our subjects [2] registered to the MNI-ICBM152 template. Same registration is applied for ground-truth (expert manually segmented images). This template is provided by Montreal Neurological Institute created by using data from ICBM project [2].

These resulting images are used by the next step.

Intensity standardisation

The grey scale for MR images is not standard, this problem might cause a lack of tissue specification within the MRI protocol, for the same body region, same patient, etc. Therefore intensity measures cannot be associated to an anatomical

meaning. So that a standardisation process has to be preformed. However, only few works have addressed this problem [2].

Those methods try to correct inter-subject intensity variations standardising the grey scale in this case a 0 to 255 scale using a 1-D histogram matching approach [2]. first by calculating percentiles on the template and the reference histogram then they are matched by linear interpolation between locations. So that, similar tissues obtain a similar value.

Finally, images are ready to be segmented.

4.1.3 Building Image crop

We are only extracting a part from the image, that part consists of the region where we can find the structure to segment, so that we do not need to segment the whole image. This process is carried out by calculating masks and bounding box. Once we have them, we can extract a sub-volume for every subject.

Masks and Bounding Box creation

During this step masks and bounding boxes for every structure and sub-structure are created.

Using the ground truth images (images manually segmented by an expert) we obtain the boundaries for every structure for the subjects of training. We can also apply this step to test subjects due to the fact that we have their respective ground truth images and this information is used later for analysis. Once we have them, they are merged into a bigger mask which delimits a region of the image where the structure (or substructure) is contained. We can ensure this fact because of the use of many ground truth images. As more subjects are used, better is that region.

The process which is used is simple, the union of all structure (or sub-structure) regions creates the final mask [2]. From these masks (one for every structure and substructure) we obtain the bounding box which is formed by coordinates which delimit the mask: initial x coordinate x_1, final x coordinate x_2, initial y coordinate y_1, final y coordinate y_2, initial z coordinate z_1, final z coordinate z_2. This information is saved matching every structure and substructure with their respective bounding box (BB).

These information is used later to analyse and show results.

4.1.4 Library construction

Centroids generation

Centroids (or target voxels) are the voxels that are used during the segmentation process and during the second stage. Those voxels are chosen by a step sample. For instance, if the step sample is 3 means that one voxel is chosen for every three.

Those voxels are defined in cropped images obtained in the previous state, so that, their coordinates correspond to the cropped image region starting by

$(1, 1, 1)$. Of course, it is sure that the target voxel belongs to the area delimited by the bounding box, so that it has relevant information.

A patch of a determined size around the target voxel is a group of voxels around the target, or central, voxel. It is more understandable having a look at the image 4.2.

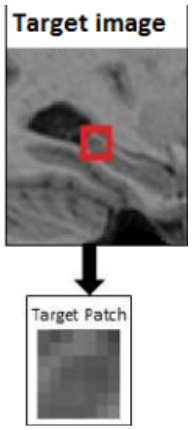


Figure 4.2: Patch creation from the cropped image (target image). The patch is marked by the red square. In this case is a $5 \times 5 \times 5$ voxel group around a target voxel (central voxel) [3].

Patch library creation

We have to create a patch library for every target voxel from the target image (the image we are labelling). The library is constructed by using the N most similar images from the training set. To find which images are the most similar, the squared intensity differences (SSD) method is used on the cropped image.

Hence, the first step is getting the N most similar subjects using SSD in the template space.

The next step is patch extraction, the target patch is denoted p_t [2, 3]. For this extraction a *search volume* consisting of a cube centred around the target voxel. The search volume is extracted for every similar subject, the cube is of size $p \times p \times p$ what is called *patch size* around the target voxel. If, for instance, $p = 5$, the patches are of size 125. Those extracted patches form the patch library P_L.

We construct a patch library P_L for every target voxel in the target image. Therefore our patch library contains thousands of patches. It should be mentioned too that this patch library is atlas-wise what means that is created for every target voxel for each atlas [2].

The patch library P_L can be represented as follows:

$$P_L = [p_1, p_2, ..., p_n] \in \mathbb{R}^{m \times n},$$

where p_i is a patch, m the patch size and n the number of patches that form the library.

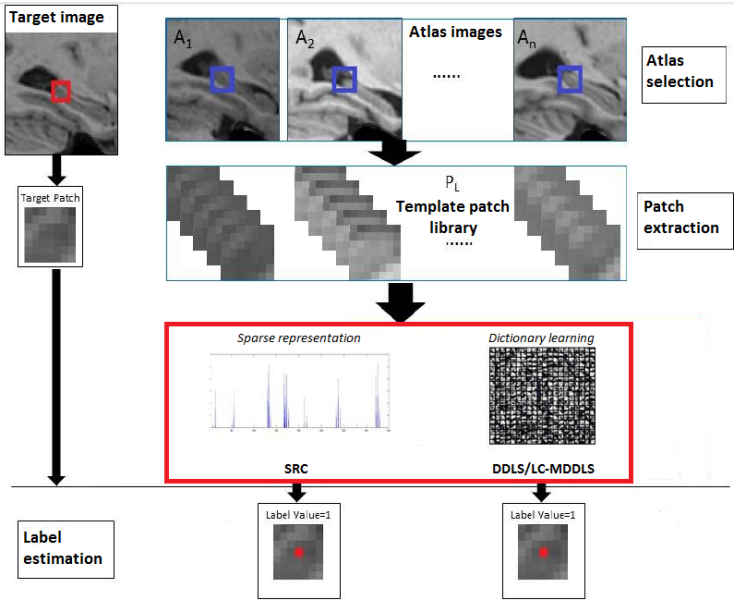


Figure 4.3: Flow chart for the labelling. Red square represents the patch and the blue square represents the search volume. The target image corresponds to the image to label and the target voxel the central voxel from the target patch. Atlas images correspond to the training set images. [3]

4.1.5 Dictionary construction

At this step we find the differences among the three methods: sparse representation classification, discriminative dictionary learning and label consistency.

Sparse representation based classification

This method uses the whole patch library P_L as a predefined dictionary for the sparse coding. So we can represent the target patch p_t, using elements from the library, approximately as follows:

$$p_t = \alpha_1 p_1 + \alpha_2 p_2 + ... + \alpha_n p_n,$$

since it is sparse, the majority of the coefficients α_i are zero. The problem is that this dictionary has too much redundancy and it makes the process take more time. The idea is that non-zero coefficients should concentrate on patches of the same class (more similar). That fact is due to the sparsity and the use of all classes while computing coefficients [2, 3]. Look at section 3.1 for more details.

Discriminative dictionary learning and label consistency

Learning a compact dictionary for each individual target patch is better than using the whole patch library as a predefined dictionary due to its drawbacks. Therefore, learning a new dictionary able to reconstruct and to be discriminative is a better option. In this case D-KSVD is used [4]. Hence, the objective function is:

$$\begin{aligned}\langle D, W, \alpha \rangle = argmin_{D,W,\alpha} \|P_L - D\alpha\|_2 + \ \beta_1 \|H - W\alpha\|_2 \\ + \beta_2 \|W\|_2 \quad \text{subject to } \|\alpha\|_0 \ \leq T,\end{aligned} \tag{4.1}$$

where each column of H is the label vector corresponding to a template patch, where the non-zero entry corresponds to the central voxel label from that patch.

As it was said before, D-KSVD uses K-SVD for finding an optimal solution simultaneously for all parameters [4]:

$$\begin{aligned}\langle D, W, \alpha \rangle = argmin_{D,W,\alpha} \| \begin{pmatrix} P_L \\ \sqrt{\beta_1}\, H \end{pmatrix} - \begin{pmatrix} D \\ \sqrt{\beta_1}\, W \end{pmatrix} \alpha \|_2^2 \\ + \beta_2 \|W\|_2^2 \quad \text{subject to } \|\alpha\|_0 \ \leq T\end{aligned} \tag{4.2}$$

We can rewrite it as follows:

$$\begin{aligned}\langle D, W, \alpha \rangle = \arg \min_{D,W,\alpha} \|\tilde{P}_L - \tilde{D}\alpha\|_2 \\ \text{subject to } \|\alpha\|_0 \ \leq T\end{aligned} \tag{4.3}$$

where $\tilde{D} = \left(D^t, \sqrt{\beta_1} W^t\right)^t$ is normalised column-wise and $\tilde{P}_L = \left(P_L^t, \sqrt{\beta_1} H^t\right)^t$ which includes the original patch and its label. We can drop the penalty term.

As it says in [2, 3] a online dictionary learning algorithm is used to solve (5).

Using $\tilde{D}$ a classifier $\hat{W}$ and a learned dictionary $\hat{D}$ are obtained [2, 4] computed as:

$$\hat{D} = \left\{ \frac{\tilde{d}_1}{\|\tilde{d}_1\|_2}, \frac{\tilde{d}_2}{\|\tilde{d}_2\|_2}, \cdots, \frac{\tilde{d}_k}{\|\tilde{d}_k\|_2} \right\} \tag{4.4}$$

$$\hat{W} = \left\{ \frac{\tilde{w}_1}{\|\tilde{w}_1\|_2}, \frac{\tilde{w}_2}{\|\tilde{w}_2\|_2}, \cdots, \frac{\tilde{w}_k}{\|\tilde{w}_k\|_2} \right\} \tag{4.5}$$

See [4] for proof.

4.1.6 Perform segmentation

Sparse representation classification

As it was said before, the whole patch library P_L is used as a dictionary. Hence the label value for the target voxel of the target patch p_t (the central voxel defines the class of the whole patch) is assigned as the class with less reconstruction error:

$$v_t = argmin_j r_j(p_t), \qquad \forall j = 1, ..., C, \tag{4.6}$$

where $r_j(p_t) = \|p_t - P_L^j \hat{\alpha^j}\|$ is the reconstruction error and α^j the sparse coefficient of the class j. These coefficients are obtained using Elastic Net method [2, 3]:

$$\hat{\alpha} = min_\alpha \frac{1}{2}\|p_t - P_L\alpha\|_2^2 + \lambda_1\|\alpha\|_1 + \frac{\lambda_2}{2}\|\alpha\|_2^2 \tag{4.7}$$

Discriminative dictionary learning

For labelling the target voxel it proceeds as we said in section 3.1. Using Lasso we obtain the sparse representation:

$$\hat{\alpha}_t = argmin_{\alpha_t}\|p_t - \hat{D}_t\alpha_t\|_2^2 + \lambda_2\|\alpha\|_1 \tag{4.8}$$

And we estimate the final label:

$$h_t = \hat{W}_t\hat{\alpha}_t \tag{4.9}$$

where h_t is the label vector of the central voxel from the target patch. The label v_t is the index of the largest element in that vector:

$$v_t = argmax_j\ h_t(j) \tag{4.10}$$

Label Consistent Multi-class DDLS

Due to the lack of handling unbalanced libraries that DDLS has, the label consistency multi-class discriminative dictionary learning is used as it is explained in 3.2. Therefore, matrix H has more rows due to the fact that now we have more classes, not only two.

Then, the function objective is the next one:

$$\begin{aligned}\langle D, W, A, \alpha\rangle = argmin_{D,W,A,\alpha}\|P_L - D\alpha\|_2 + \ \beta_1\|H - W\alpha\|_2 \\ + \beta_2\|Q - A\alpha\|_2^2 + \beta_3\|W\|_2 \quad \text{subject to } \|\alpha\|_0\ \leq T,\end{aligned} \tag{4.11}$$

were $\|Q - A\alpha\|_2^2$ is the label consistence regularisation term and Q the discriminative sparse codes of the input patches in the patch library P_L for classification [2, 5].

The learning process is the same as used in DDLS but adding the label consistency term:

$$\langle D, W, \alpha \rangle = argmin_{D,W,\alpha} \| \begin{pmatrix} P_L \\ \sqrt{\beta_1}\, H \\ \sqrt{\beta_2}\, Q \end{pmatrix} - \begin{pmatrix} D \\ \sqrt{\beta_1}\, W \\ \sqrt{\beta_2}\, A \end{pmatrix} \alpha \|_2 + \quad \text{subject to } \|\alpha\|_0 \ \leq T \tag{4.12}$$

And labelling is the same process as in DDLS [2].

4.2 Second stage: SS-MDDLS

In order to improve the performance we are proposing the next implementation which consists of a subsequent stage inspired by Stacked Sequential Learning [13, 16].

From the information we get from the linear classifier during the stage one, we are training another classifier for labelling the target voxels.

In DDLS and MDDLS we use the $h_t = W_t \alpha_t$ label vector to classify the target voxel where W is the linear classifier and α the sparse representation. So that we propose to use not only the target voxel vector but also the vectors of its nearest neighbours to classify it. Contextual information helps to classify the target voxel.For example a background voxel is surrounded by background voxels except for a voxel in a structure boundary, that is why contextual information can help to classify the target voxel.

So that we have a feature vector formed by seven different h_t vectors. One from the target voxel and the other six from its neighbours. We are getting the neighbours from right, left, up, down, front and rear having the target voxel as centre reference. Hence, the feature vector from the target voxel f_t is:

$$f_t = [h_t, h_1, \cdots, h_6],$$

where h_t is the label vector of the target voxel and $h_i \quad \forall\ i = 1, \cdots, 6$ the label vector of the neighbour voxels. In our case, each h vector is formed by C (number of classes) components, one for each class, hence, $f_t \in \mathbb{R}^{7*C}$.

4.2.1 Classification

Once we have computed the feature vectors for all the samples, we use this data to train a new classifier using decision trees [20, 21]. Once we have done it, we classify the testing subjects using that trained classifier and we obtain the predicted label for every voxel.

Decision Tree Learning

The decision trees are a bootstrap aggregation for ensemble of decision trees for either classification or regression. In our case we are using this strategy for classification [20, 22].

Given a training data set and their class labels, which corresponds to the ground truth segmentation in our case, the algorithm trains a group of classification trees. Then algorithm works as follows [20]: It generates in-bag (observations included in the decision tree) samples by oversampling classes with large miss-classification cost and under-sampling those with low miss-classification cost. So that out-of-bag (observations not included in the decision tree) samples have fewer observations from classes with large miss-classification cost and more observations from the other classes. Every tree is grown on an independently drawn bootstrap replica of input data. An average of predictions from every individual tree is taken in order to compute the prediction for unseen data.

The method of Bagging predictor [22] generates multiple versions of a predictor and uses them to get an aggregated predictor. When predicting a numerical outcome, the aggregation averages over the versions and does a majority vote to predict the final class. As it is said in [22], the multiple versions are formed by making bootstrap replicates of the learning set and using them as the new learning set.

The proof about why does Bagging predictor work can be found at [22] as well as a more mathematically oriented explanation of classification and regression using Bagging predictor.

4.2.2 Extended Feature Vector

In order to help to learn contextual patterns, we propose to use an extended feature vector.

In this case we add to f_t the sparse representation α. Hence, we create this new feature vector:

$$f'_t = [h_t, h_1, \cdots, h_6, \alpha_t, \alpha_1, \cdots, \alpha_6],$$

where α_t corresponds to the sparse representation for the target voxel and $\alpha_i \quad \forall\, i = 1, \cdots, 6$ are the sparse representations from its six neighbours.

Since we work on a crop from the image, we might find that a neighbour is out of the bounding box. We know that it is a background voxel, therefore, as we did before, we impose a feature for this type of voxels. So that, we add $[1, 0, \cdots, 0]$ as their α.

Each $\alpha \in \mathbb{R}^K$ (see section 3.1) where K is the number of atoms of the dictionary, then we add $7 * K$ features to the feature vector.. Hence, our new $f'_t \in \mathbb{R}^{7*(C+K)}$.

The big size of the new feature matrix $F \in \mathbb{R}^{n \times m}$ where n is the number of centroids (target voxels) and m the size of f'_t we decide to use a sub-sampling of voxels. This sub-sampling gets 1 for every 3 voxels in each dimension. Then, we

reduce the size of that matrix. It should be mentioned that despite having fewer voxels than before, we still have information about almost every voxel from the cropped image (target image) since we use information from neighbours. Therefore, we have less training observations but we have more information from them.

4.3 Matlab

Matlab [17] is a high level language which allows the user to explore and visualise data as well as image and signal processing, communication, control systems and computational finances.

This language has many useful features such as:

1. Numerical calculations: Matlab offers several numerical methods which allow to perform engineering and scientific operations whose functions are optimised in order to make vectorial and matricial calculus fast.

2. Data analysis and visualisation: Many tools are offered by matlab to obtain, analyse and visualise data to make those tasks faster.

3. Algorithm development and programming: since it is a high level language it allows to develop algorithms using many tools which help to do it faster. An important part is vector and matrices support.

4. Applications development and distribution: Many tools are given by matlab to make easier application sharing and distribution.

After this brief approach to Matlab we have chosen this development language due to the tools it offers which are very useful for imaging management because of its vector and matricial support. A part from that we also found useful the fact that many data analysis and visualisation tools which are really helpful for our purposes. Points 1, 2 and 3 are which we found more suitable for what we need.

4.4 Statistical parametric mapping

The Statistical Parametric Mapping (SPM) package is used for construction and assessment of spatially extended statistical processes used to test about imaging data [18]. It has been designed for the analysis of brain imaging data sequences as MRI.

More detailed information can be found in [18, 19].

It is voxel based. It allows to realign images, spatially normalise and smoothen them. Uses General Linear Model to describe data. SPM uses classical statistical inference where multiple comparisons problem is addressed using random field theory under many assumptions. It can also use Bayesian inference.

This package is helpful during the preprocessing period, when images are previously preprocessed in order to reduce noise, normalise intensity among subjects and image registration to a stereotaxic space.

4.5 Code details

An important structure is created during centroids generation process, a structure which contain the coordinates from every target voxel $n - centroids \times 3$, coordinates from a reduced number of voxels $m - centroids \times 3$ with their sequential indexes in the image $m - centroids \times 1$ and matrix containing the patches, their initial coordinates, for every centroid.

Patch library P_L is a matrix where patches are grouped column-wise. Hence, every column of P_L is a different patch.

To construct the feature vector we use a structure created during MDDLS segmentation. That structure contains coordinates of target voxels in the cropped image (all of them) $centroidMat \in \mathbb{R}^{n\times 3}$ where n is the number of target voxels, coordinates of a sampling of those voxels ($reducedMat \in \mathbb{R}^{l\times 3}$ where l is the number of sampling) in addition to their sequential index ($idxReduced \in \mathbb{R}^{l}$) in the cropped image and the patch library.

Steps:

1. Obtaining data. At first we obtain the h vector for every target voxel. This information is obtained while performing MDDLS segmentation.

2. Search neighbours. Then we look for the six neighbours for every target voxel and discern which ones are inside the bounding box and which ones are out of it.

3. Fill feature matrix. Feature matrix $F \in \mathbb{R}^{n\times m}$ where n is the number of target voxels and $m = f_t size$. Therefore, columns are the features for every voxel whose mapping is ith-row corresponds to the feature vector of the ith-target voxel in $centroidMat$ matrix. Hence we fill each row of F using the label vector from the target voxel and its neighbours label vectors.It should be mentioned that we have the label vector from every voxel, voxels out of the bounding box are, for sure, background voxels so that their label vector is $h_t = [1, 0, 0, 0, 0]$ and voxels label vector from the cropped image (target image) has been computed when performing MDDLS segmentation.

Once we have the feature matrix completely filled for every subject, we mix every feature matrix into a single feature matrix which we called $Xtrain \in \mathbb{R}^{(n*k)\times m}$ where n is the number of target voxels, k is the number of training subjects (in our case $k = 28$) and $m = sizef_t$ the size of the feature vector, in this case $m = 35$ since it contains the $h_t \in \mathbb{R}^5$ of seven voxels (target and six neighbours). Since it is the training set, we know their correspondent label so that we obtain them from the manually segmented image using coordinates given in $centroidMat$.

The extended feature vector $f'_t \in \mathbb{R}^{595}$; $595 = 7 * (5 + 80)$ where 7 is the number of subjects whose data used, the target plus its six neighbours, $C = 5$ the number of different classes and $K = 80$ the number of atoms of the dictionaries.

Chapter 5

Experiments

In this section we expose the performance of the different methods presented.

We replicate SRC, DDLS results applied to the four structures from the basal ganglia and a brief comparison between them[1, 2]. In addition we analyse the results obtained from MDDLS. To finish we are showing the performance of SS-MDDLS and a comparative analysis of the obtained results.

5.1 Dataset

For this performance we are using MR images from 35 different subjects. The same dataset as it is used in [2]. The 35 subjects and their corresponding segmentation were made public by the *MICCAI 2012 Grand Challenge and Workshop on Multi-Atlas Labelling*. The data-set images (corresponding to both training and test) for distinct human data consist of a de-faced T1-weighted structural MRI and its associated manually labelled volume with one label per voxel. Each volume (MRI and label) is stored in a separate 3D NiFTI[1] file.

The original MRI scans were obtained from the OASIS project `http://www.oasis-brains.org/`. The specific scans used were from a subset of the *Crossectional MRI Data in Young, Middle Aged, Non-demented and Demented Older Adult* called "reliability data". Here, 35 normal control volunteer subjects were each scanned twice. They were all right handed and include 13 males and 22 females. Their ages ranged from 19 to 90 with an average of 32.4 years old.

5.2 Methods and parameters

Four different methods have been used. From which the fourth is our proposal.

- SRC. See section 3.1.
- DDLS. See section 3.1.

[1] http://nifti.nimh.nih.gov

- MDDLS. See sections 3.2 and 4.1.
- SS-MDDLS. See section 4.2.

General parameters setting as used in [2]:

- Bounding box padding: 3
- Best subjects: 8
- Patch size: 5
- Window size: 3
- Sampling step: 3
- Dictionary atoms: 80
- Neighbours: 6
- C (number of classes) = 5

DDLS and MDDLS parameters;

- Lambda: 0.15
- Lambda2: 0
- Iterations: 40
- Mode: 5
- Positive constraint on coefficients: true
- Mode D: 0 (non-sparsity in dictionary atoms)
- Positive constraint on dictionary: true
- Batch size: 25
- Iteration update: 1
- Mode parameter: 0

SS-MDDLS parameters;

- Number of decision trees: 50, 100, 200
- Neighbours distance: 1, 6

5.3 Evaluation

The evaluation strategy is a nested cross validation. First, we split our dataset in two groups, training set and test set. We use the training set for learning dictionaries and classifiers for MDDLS and SS-MDDLS. Then we segment the test set using using those methods. Our dataset is formed by 35 subjects. Hence, our training set consists of 28 subjects and our test set consists of 7 subjects. We make five different combinations of this two subgroups in order to have all subjects as test subject once.

As an evaluation measure we have computed the Dice coefficient between manually segmented images and automatic segmentation. This overlap coefficient, also known as kappa coefficient, is computed as follows:

For two binary segmentation A and B, the Dice coefficient is computed as:

$$\kappa(A, B) = \frac{2|A \cap B|}{|A| + |B|} \tag{5.1}$$

5.4 Results

In this section, we show a comparative analysis of the obtained results. At first we introduce SRC and DDLS results and then we discuss about MDDLS and SS-MDDLS performance.

5.4.1 SRC and DDLS results

At first we replicate results in [2] just to have an initial approach of segmentation results applied to MRI and to briefly analyse SRC and DDLS performances.

As we can see in figures 5.1 5.2, DDLS method has a better performance than SRC since its overlap indexes are much higher. We can also notice that standard deviation is higher in SRC than in MDDLS so that, the overall accuracy is reduced.

It can be deducted that DDLS is more robust against subjects whose images are more different from the rest of the data-set. We can find this fact in the number of outliers. SRC has much more outlier subjects than DDLS what means, again, that SRC is outperformed by DDLS.

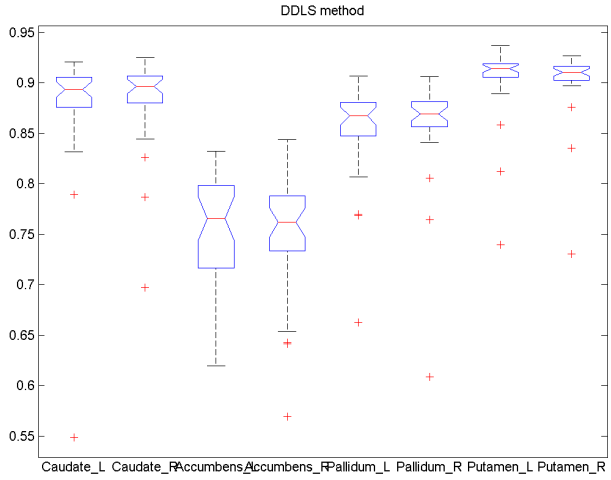


Figure 5.1: Boxplot showing overlap statistics for DDLS method for the four substructures from the Basal Ganglia.

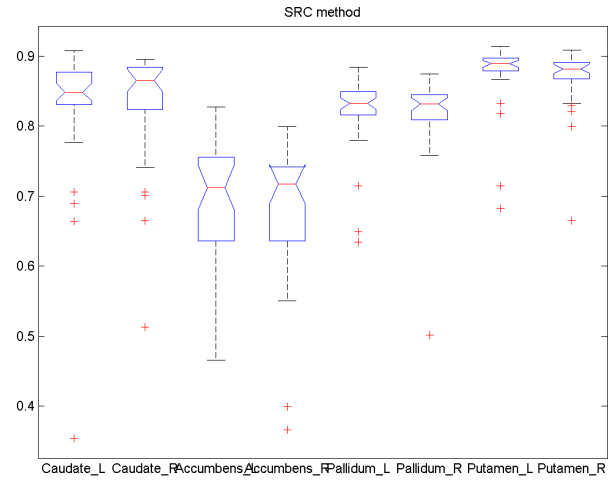


Figure 5.2: Boxplot showing overlap statistics for SRC method for the four substructures from the Basal Ganglia.

Let us see the results more accurately in the tables 5.1 and5.2. We split the results in two different tables to show left and right structures separately.

	Caudate L	Accumbens L	Pallidum L	Putamen L
SRC mean overlap	83%	69.7%	82.1%	87.6%
SRC deviation	±10%	±8.5%	±5.4%	±4.8%
DDLS mean overlap	87.8%	75.7%	85.5%	90.4%
DDLS deviation	±6.4%	±5.4%	±4.6%	±3.6%

Table 5.1: Average Dice overlaps for Basal Ganglia (BG) left structures.

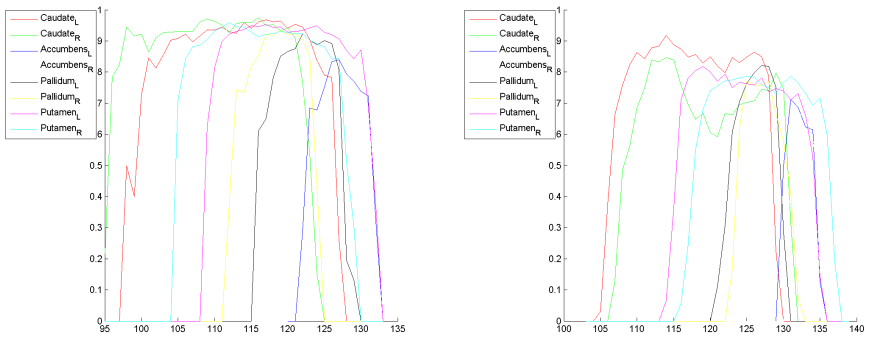
	Caudate R	Accumbens R	Pallidum R	Putamen R
SRC mean overlap	83.5%	67.8%	81.7%	87%
SRC deviation	8.1%	10%	6.1%	4.3%
DDLS mean overlap	88.5%	75.2%	85.8%	90.3%
DDLS deviation	4.4%	5.8%	5%	3.4%

Table 5.2: Average Dice overlaps for Basal Ganglia (BG) right structures.

As we deduce from the boxplot figures and tables, DDLS results are much better than SRC results. DDLS not only has higher average results but also has a lesser standard deviation than SRC. Remember that SRC uses the sparse representation from voxels as a dictionary, meanwhile DDLS learns dictionaries for every voxels from the most similar subjects in the training set. This fact has been proved to improve performance in MRI segmentation.

Since DDLS has shown a better performance we do a further analysis of its results.

Until now we show the average overlap among all the training subjects. Then we study the overlap index per subject and per slice in order to make a more accurate analysis. Let us see a couple of examples in the figure 5.3:



(a) Overlap per slice (b) Overlap per slice

Figure 5.3: Overlap per slice. The image in (a) shows the overlap for a standard subject and the image in (b) a subject which is usually classified as outlier since the age of this subject is 90 years old. We can observe that, despite having different indexes, both have their overlap reduced at start and end slices.

We have noticed that the overlap has a lower index at the starting and ending slices. The overlap ratio is considerably lower at those slices than at central slices. It might be caused by the substructure morphology because starting slices are those where the structure is beginning and ending slices are those where the structure is disappearing. Hence, because of the fact that we use a training set, for several subjects the structure appears before than for others. This fact means that slices where the structure appears or disappears (border slices) are confusion areas since it is hard to distinguish if there is structure in that location or not. Figure 5.4 shows a qualitative example.

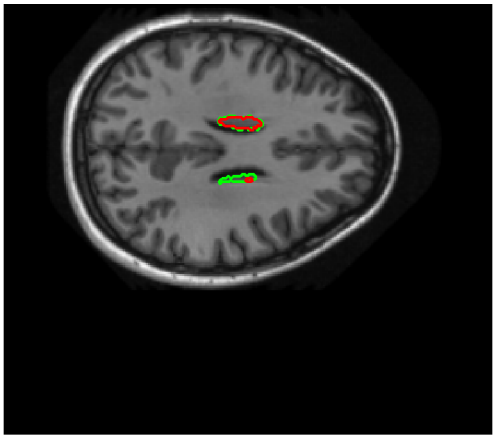


Figure 5.4: Axial view from a starting slice for the caudate segmentation where green is the manually segmented structure and red the automatic performing DDLS

We can describe this error as systematic since it is shown in every subject. These border slices are difficult to classify making the performance worse in those regions.

As it was said, central slices have much better results see figures 5.4 5.5.

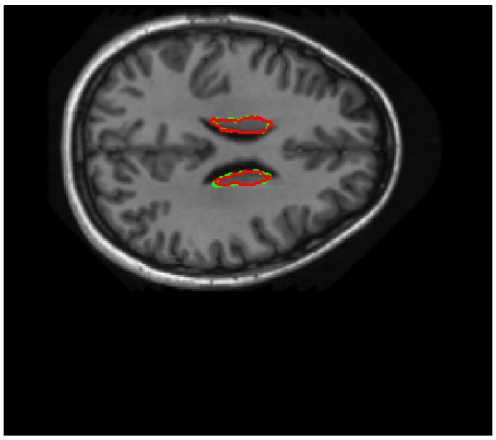


Figure 5.5: Axial view from a central slice for the caudate segmentation where green is the manually segmented structure and red the automatic performing DDLS

5.4.2 MDDLS results

In this new section we are showing more interesting results. MDDLS has a better performance than DDLS and SRC. So that we are studying these results more accurately.

First of all let us show an extended dice overlap table. Data has been split in two different tables due to the high amount of information. We show right hemisphere structures in 5.3 and left hemisphere structures in 5.4 substructures separately, as we did before. Averages are shown in 5.11.

	Putamen R	**Caudate R**	**Pallidum R**	**Accumbens R**
1000_3 R	0.911	0.901	0.846	0.774
1001_3 R	0.924	0.903	0.868	0.794
1002_3 R	0.906	0.891	0.863	0.711
1003_3 R	0.909	0.892	0.852	0.769
1004_3 R	0.922	0.899	0.856	0.769
1005_3 R	0.905	0.918	0.884	0.791
1006_3 R	0.914	0.914	0.907	0.785
1007_3 R	0.909	0.857	0.878	0.745
1008_3 R	0.923	0.897	0.873	0.783
1009_3 R	0.899	0.896	0.888	0.822
1010_3 R	0.920	0.871	0.863	0.688
1011_3 R	0.917	0.902	0.902	0.776
1012_3 R	0.881	0.882	0.856	0.805
1013_3 R	0.914	0.911	0.858	0.800
1014_3 R	0.874	0.898	0.868	0.712
1015_3 R	0.911	0.917	0.875	0.720
1017_3 R	0.798	0.781	0.740	0.479
1018_3 R	0.913	0.878	0.898	0.788
1019_3 R	0.913	0.884	0.875	0.791
1023_3 R	0.912	0.847	0.863	0.706
1024_3 R	0.925	0.904	0.863	0.821
1025_3 R	0.911	0.924	0.865	0.809
1036_3 R	0.899	0.898	0.843	0.787
1038_3 R	0.911	0.922	0.875	0.797
1039_3 R	0.911	0.886	0.882	0.747
1101_3 R	0.901	0.890	0.858	0.743
1104_3 R	0.916	0.899	0.864	0.767
1107_3 R	0.886	0.878	0.865	0.732
1110_3 R	0.903	0.878	0.885	0.785
1113_3 R	0.897	0.878	0.863	0.704
1116_3 R	0.896	0.856	0.856	0.614
1119_3 R	0.881	0.802	0.812	0.661
1122_3 R	0.897	0.817	0.873	0.658
1125_3 R	0.906	0.645	0.872	0.800
1128_3 R	0.678	0.539	0.614	0.530

Table 5.3: Dice overlaps for Basal Ganglia (BG) Right substructures using MDDLS.

	Putamen L	Caudate L	Pallidum L	Accumbens L
1000_3 L	0.918	0.890	0.877	0.727
1001_3 L	0.926	0.903	0.876	0.765
1002_3 L	0.908	0.886	0.869	0.720
1003_3 L	0.919	0.888	0.881	0.771
1004_3 L	0.934	0.913	0.863	0.799
1005_3 L	0.919	0.903	0.882	0.803
1006_3 L	0.921	0.900	0.876	0.760
1007_3 L	0.905	0.849	0.844	0.764
1008_3 L	0.921	0.899	0.875	0.804
1009_3 L	0.913	0.883	0.881	0.764
1010_3 L	0.923	0.881	0.882	0.798
1011_3 L	0.918	0.895	0.879	0.798
1012_3 L	0.905	0.910	0.869	0.822
1013_3 L	0.911	0.901	0.845	0.813
1014_3 L	0.908	0.894	0.893	0.705
1015_3 L	0.880	0.900	0.872	0.817
1017_3 L	0.875	0.815	0.859	0.645
1018_3 L	0.921	0.890	0.876	0.811
1019_3 L	0.919	0.896	0.885	0.831
1023_3 L	0.896	0.864	0.819	0.723
1024_3 L	0.912	0.910	0.865	0.833
1025_3 L	0.912	0.909	0.839	0.786
1036_3 L	0.913	0.906	0.846	0.714
1038_3 L	0.917	0.925	0.882	0.861
1039_3 L	0.909	0.913	0.883	0.816
1101_3 L	0.917	0.886	0.892	0.736
1104_3 L	0.919	0.892	0.889	0.767
1107_3 L	0.892	0.873	0.861	0.656
1110_3 L	0.912	0.864	0.889	0.770
1113_3 L	0.919	0.889	0.887	0.741
1116_3 L	0.875	0.822	0.846	0.696
1119_3 L	0.798	0.400	0.783	0.750
1122_3 L	0.902	0.833	0.814	0.787
1125_3 L	0.826	0.676	0.746	0.806
1128_3 L	0.732	0.812	0.703	0.677

Table 5.4: Dice overlaps for Basal Ganglia (BG) Left substructures using MDDLS.

As we can see, using MDDLS better results are obtained. Remember that it uses label consistency in order to reduce the unbalance problem. Hence, results have been slightly improved compared to DDLS.

5.4.3 Results analysis

Here we analyse the results visualising pseudo-probability maps and profiles of the first stage classification results.

While we perform MDDLS (sections 3.2, 4.1) in order to classify the target voxel (centroid) from a target patch p_t we use a label vector $h_t = W_t \alpha_t$ where W_t is the lineal classifier for the target voxel and α_t the sparse representation of the target voxel, sub-index t corresponds to *target*.

The final label of the target voxel, and consequently, the label of the target patch is obtained by:

$$v_t = argmax_j h_t(j), \qquad j = 1, ..., C \tag{5.2}$$

Since it is not an ideal $h_t = [0, ..., 1, ...0]$ with a one non-zero value which determines the label, we find interesting to study information given by h_t.

In order to visually analyse them we have defined the pseudo-probability maps as these h_t values of every voxel in an image crop.

For every subject we create this pseudo-probability map for each class, in this case five different probability maps. These maps allow us to visually compare h_t vector values.

First of all we normalise h_t vectors between 0 and 1 in order to have the same range for every map. For that we divide for the maximum of all h_t values in the image.

Our label vector has five different values. We have a h_t for every voxel in the target image (the crop where the structure is contained). Hence, we create a matrix $H \in \mathbb{R}^{5 \times n}$ where 5 is the number of classes (background, putamen, caudate, pallidum, accumbens) and n is the number of target voxels, for every subject.

Using H we create five different maps $M_i \in \mathbb{R}^{m \times k \times l} \quad \forall i = 1, \cdots, 5$ where m, k, l, are the sizes of the target image and $m * k * l = n$. Then we fill every M_i using data from every i-th row of H. In other words, the first value of every h_t from this subject fills M_1, the second fills M_2 and so on, making coordinates to match. See figure 5.6 for an illustration.

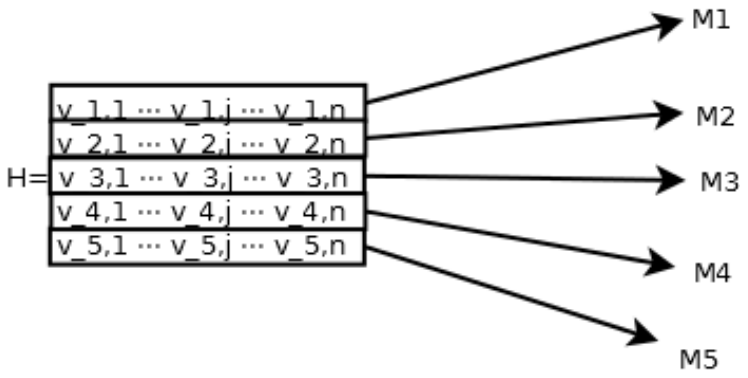
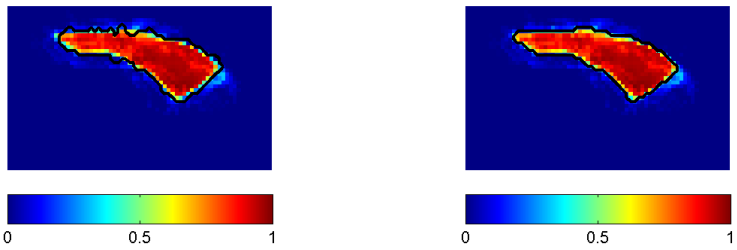


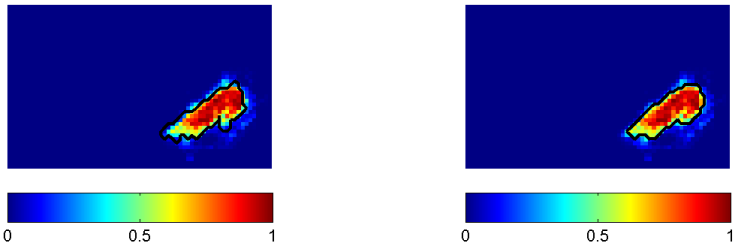
Figure 5.6: Schema about how H matrix is used to create probability maps. Making each row to be a different map.

As a result we obtain the following maps in figures 5.7 5.8 5.9 5.10 5.11:



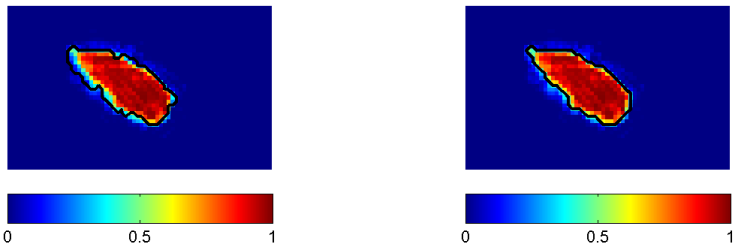
(a) Manual segmentation (b) MDDLS segmentation

Figure 5.7: Pseudo-probability maps where (a) has the manually segmented contour superposed and (b) the MDDLS contour superposed. Blue means the lowest probability for a voxel to belong to Putamen structure and Red the highest probability.



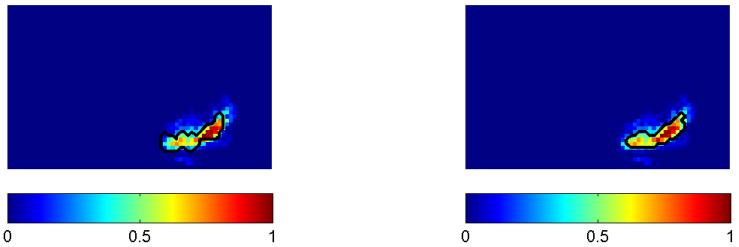
(a) Manual segmentation (b) MDDLS segmentation

Figure 5.8: Pseudo-probability maps where (a) has the manually segmented contour superposed and (b) the MDDLS contour superposed. Blue means the lowest probability for a voxel to belong to Caudate structure and Red the highest probability.



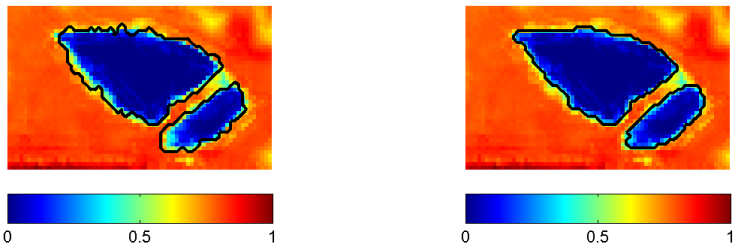
(a) Manual segmentation (b) MDDLS segmentation

Figure 5.9: Pseudo-probability maps where (a) has the manually segmented contour superposed and (b) the MDDLS contour superposed. Blue means the lowest probability for a voxel to belong to Pallidum structure and Red the highest probability.



(a) Manual segmentation (b) MDDLS segmentation

Figure 5.10: Pseudo-probability maps where (a) has the manually segmented contour superposed and (b) the MDDLS contour superposed. Blue means the lowest probability for a voxel to belong to Accumbens structure and Red the highest probability.



(a) Manual segmentation (b) MDDLS segmentation

Figure 5.11: Pseudo-probability maps where (a) has the manually segmented contour superposed and (b) the MDDLS contour superposed. Blue means the lowest probability for a voxel to belong to Background and Red the highest probability.

We can see that, obviously, the contour of MDDLS segmentation fits better than the manual segmentation one, since we are showing how does MDDLS classify voxels.

Important information can be obtained from these pseudo-probability maps. It is easy to see that MDDLS performs a under-segmentation, what means that MDDLS finds the sub structure smaller than it is in manual segmentation.

We can notice that MDDLS results n an under segmentation of structures compared to its manual segmentation.

It is also noticeable that frontier voxels (or bounding voxels) of the structure are a high confusion area. This fact makes the algorithm commit mistakes such as labelling structure voxels as background voxels or caudate as accumbens and vice versa.

We can also see that smaller structures are worse segmented. For example Accumbens, which is a small structure near the caudate [12], is a high confusion region. We can notice that caudate and accumbens voxels which are close to each other are becoming yellow or even light blue, something similar happens with pallidum and putamen bounding voxels.

In order to study this systematic problem, since it appears in every subject, we have built the graphics in figures 5.12, 5.13, 5.14, 5.15 where we show the h_t patterns.

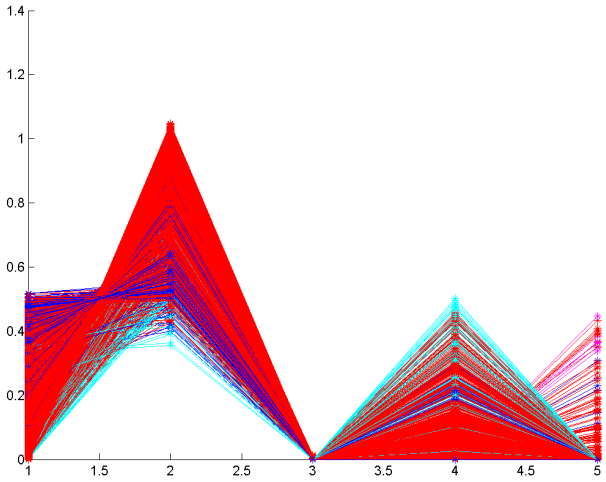


Figure 5.12: Confusion graphic of putamen. Every label vector classified as putamen is represented, its colour represents the manual segmentation classification being: blue=background, red=putamen, green=caudate, cyan=pallidum, magenta=accumbens. Horizontal axis: 1=background, 2=putamen, 3=caudate, 4=pallidum, 5=accumbens. Values of label vector $h_t = [background, putamen, caudate, pallidum, accumbens]$. Vertical axis: shows h_t value for the component

In this graphic 5.12 we can see that putamen classification has confusions with background and pallidum which are the neighbour structures. Every label vector classified as putamen is represented, its colour represents the manual segmentation classification being: blue=background, red=putamen, green=caudate, cyan=pallidum, magenta=accumbens. Horizontal axis: 1=background, 2=putamen, 3=caudate, 4=pallidum, 5=accumbens. Vertical axis: shows h_t value for the component. We can see several blue and cyan lines in the graphic. The interesting idea is to realise the different pattern of this vector for red, cyan and blue lines. Red lines have a clear maximum in position 2 (putamen) whereas the cyan lines have two similar relative maximums in 2 (putamen) and 4 (pallidum). We can also see that something similar happens to blue lines, they have two relative maximums at points 1 (background) and 2 (putamen).

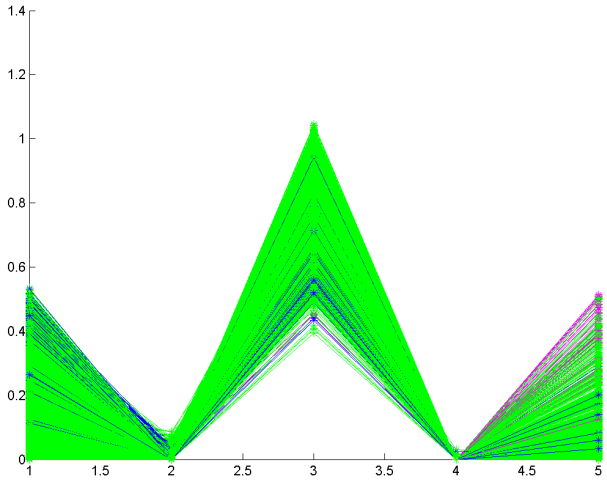


Figure 5.13: Confusion graphic of caudate. Every label vector classified as caudate is represented, its colour represents the manual segmentation classification being: blue=background, red=putamen, green=caudate, cyan=pallidum, magenta=accumbens. Horizontal axis: 1=background, 2=putamen, 3=caudate, 4=pallidum, 5=accumbens. Values of label vector $h_t = [background, putamen, caudate, pallidum, accumbens]$. Vertical axis: shows h_t value for the component

Something similar we can say looking at caudate confusion graphic 5.13. Every label vector classified as caudate is represented, its colour represents the manual segmentation classification being: blue=background, red=putamen, green=caudate, cyan=pallidum, magenta=accumbens. Horizontal axis: 1=background, 2=putamen, 3=caudate, 4=pallidum, 5=accumbens. Vertical axis: shows h_t value for the component. It is shown that this structure has several confusions with background and accumbens. In this case, again, we can observe that the green lines have a single maximum in 3 (caudate) whereas magenta lines have two relative maximums at points 3 (caudate) and 5 (accumbens). Blue lines also have two relative maximums, in this case in 1 (background) and 3 (accumbens).

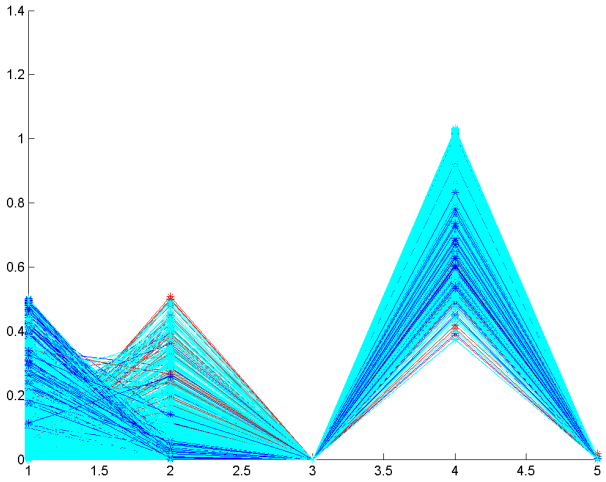


Figure 5.14: Confusion graphic of pallidum. Every label vector classified as pallidum is represented, its colour represents the manual segmentation classification being: blue=background, red=putamen, green=caudate, cyan=pallidum, magenta=accumbens. Horizontal axis: 1=background, 2=putamen, 3=caudate, 4=pallidum, 5=accumbens. Values of label vector $h_t = [background, putamen, caudate, pallidum, accumbens]$. Vertical axis: shows h_t value for the component

Again we can extract a pattern 5.14 for pallidum. Every label vector classified as pallidum is represented, its colour represents the manual segmentation classification being: blue=background, red=putamen, green=caudate, cyan=pallidum, magenta=accumbens. Horizontal axis: 1=background, 2=putamen, 3=caudate, 4=pallidum, 5=accumbens. Vertical axis: shows h_t value for the component. Now we can observe that cyan lines have a clear maximum in 4 (pallidum) whereas red lines have two relative maximums in 2 (putamen) and 4 (pallidum). Moreover, blue lines have two relative maximums in 1 (background) and 4 (pallidum).

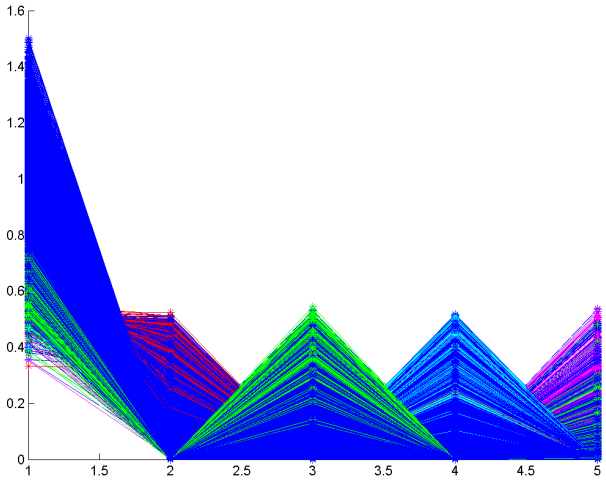


Figure 5.15: Confusion graphic of background. Every label vector classified as background is represented, its colour represents the manual segmentation classification being: blue=background, red=putamen, green=caudate, cyan=pallidum, magenta=accumbens. Horizontal axis: 1=background, 2=putamen, 3=caudate, 4=pallidum, 5=accumbens. Values of label vector $h_t = [background, putamen, caudate, pallidum, accumbens]$. Vertical axis: shows h_t value for the component

In the graphic 5.15 which shows the confusion of the algorithm where it has classified voxels belonging to a structure as background voxels.

We can see that it makes mistakes with every structure since every structure has voxels which delimit with background. That information is not as interesting as the given by the next graphics.

As it was expected, MDDLS has confusion areas in the limits of structures and background. Structures that are close to each other have similar patterns. Those patterns could *"complement"* each other. It is obvious the fact that adjacent structures have high values at the components corresponding to those structures in their delimiting voxels label vectors. SS-MDDLS obejtive is to exploit these different patterns in a second stage classification.

5.4.4 SS-MDDLS

As we have noticed in the previous section the results of automatic segmentation show a systematic error. This error consists of an under segmentation and a confusion among neighbouring structures compared to the ground truth (manual segmentation by an expert). So that we have concluded that boundary voxels are high confusion areas, where the algorithms assigns the wrong label to the target voxel. Therefore, we are exposing the results obtained by using contextual

information to classify each voxel obtained from segmentation methods applying Stacked learning.

Inspired by SSL (section 3.3) we train a new classifier with an extended data set using Tree Bagger from matlab..

Using the training data features we train our classifier, then we use it to classify our test set target voxels.

We know that MDDLS has problems while classifying boundary voxels. Hence, we tried at first to increase the distance of the neighbours in order to obtain more voxels with border information. Then we increase the number of decision trees for each distance parameter.

We have made several executions varying parameters of classifier and feature vectors information. Parameters that we change are the distances of neighbours to build feature vectors and the number of decision trees while we train the new classifier.

We use two main parameters in SS-MDDLS. We call d to the distance between the target voxel and its neighbours and n to the number of decision trees created to train the classifier. In tables 5.5 and 5.6 we show results for SS-MDDLS using n=50 decision trees and varying the parameter d. Then we can select the best d.

SS-MDDLS	Putamen R	Caudate R	Pallidum R	Accumbens R	BG
d= 1	**89.78%**	**87.07%**	**85.53%**	**73.69%**	**84.02%**
d= 6	89.30%	86.39%	84.96%	70.56%	82.80%

Table 5.5: Mean overlap for Basal Ganglia (BG) right hemisphere structures using SS-MDDLS with parameters n=50 and varying d in order to select which one gives better results.

SS-MDDLS	Putamen L	Caudate L	Pallidum L	Accumbens L	BG
d= 1	**89.97%**	**86.51%**	**83.64%**	**75.34%**	**83.87%**
d= 6	89.65%	85.82%	85.01%	73.12%	83.40%

Table 5.6: Mean overlap for Basal Ganglia (BG) left hemisphere structures using SS-MDDLS with parameters n=50 and varying d in order to select which one gives better results.

Once we choose a desirable d parameter, we look for the best n value. In tables 5.7 and 5.8 we can see average results for every substructure and the whole basal ganglia applying SS-MDDLS using different n values.

SS-MDDLS	Putamen R	Caudate R	Pallidum R	Accumbens R	BG
n= 50	89.78%	87.07%	85.53%	73.69%	84.02%
n= 100	89.81%	87.13%	85.62%	74.02%	84.14%
n= 200	**89.86%**	**87.23%**	**85.73%**	**74.09%**	**84.23%**

Table 5.7: Average dice overlaps for right hemisphere of Basal Ganglia (BG) structures using SS-MDDLS with different parameters. Where d is the neighbours distance and n the number of decision trees.

SS-MDDLS	Putamen L	Caudate L	Pallidum L	Accumbens L	BG
n= 50	89.97%	86.51%	83.64%	75.34%	83.87%
n= 100	90.04%	86.62%	85.72%	**75.76%**	84.53%
n= 200	**90.08%**	**86.65%**	**85.73%**	75.59%	**84.56%**

Table 5.8: Average dice overlaps for left hemisphere of Basal Ganglia (BG) structures using SS-MDDLS with different parameters. Where d is the neighbours distance and n the number of decision trees.

We finally choose $d = 1$ and $n = 200$ since they seem to perform the best segmentation using SS-MDDLS. The results of the best SS-MDDLS performance we have obtained are shown in the tables 5.9 and 5.10. Mean overlaps compared to MDDLS results in table 5.11.

	Putamen R	Caudate R	Pallidum R	Accumbens R
1000_3 R	0.912	0.908	0.845	0.767
1001_3 R	0.926	0.904	0.871	0.804
1002_3 R	0.906	0.892	0.864	0.680
1003_3 R	0.908	0.893	0.855	0.771
1004_3 R	0.923	0.901	0.850	0.761
1005_3 R	0.906	0.916	0.876	0.803
1006_3 R	0.919	0.914	0.912	0.774
1007_3 R	0.911	0.864	0.879	0.728
1008_3 R	0.924	0.907	0.876	0.770
1009_3 R	0.903	0.904	0.884	0.817
1010_3 R	0.918	0.882	0.858	0.676
1011_3 R	0.917	0.909	0.896	0.768
1012_3 R	0.886	0.886	0.855	0.784
1013_3 R	0.914	0.919	0.861	0.818
1014_3 R	0.879	0.906	0.872	0.716
1015_3 R	0.913	0.923	0.872	0.727
1017_3 R	0.801	0.784	0.744	0.489
1018_3 R	0.910	0.885	0.895	0.779
1019_3 R	0.914	0.889	0.874	0.808
1023_3 R	0.915	0.859	0.869	0.721
1024_3 R	0.929	0.909	0.869	0.825
1025_3 R	0.912	0.924	0.872	0.794
1036_3 R	0.900	0.901	0.846	0.794
1038_3 R	0.917	0.924	0.878	0.807
1039_3 R	0.909	0.896	0.885	0.750
1101_3 R	0.903	0.889	0.858	0.757
1104_3 R	0.915	0.900	0.862	0.777
1107_3 R	0.886	0.879	0.870	0.735
1110_3 R	0.906	0.869	0.888	0.762
1113_3 R	0.901	0.871	0.862	0.699
1116_3 R	0.901	0.852	0.854	0.588
1119_3 R	0.882	0.810	0.813	0.644
1122_3 R	0.897	0.817	0.873	0.672
1125_3 R	0.909	0.717	0.865	0.800
1128_3 R	0.677	0.528	0.604	0.565

Table 5.9: Dice overlaps for Basal Ganglia (BG) Right substructures using SS-MDDLS.

	Putamen L	**Caudate R**	**Pallidum L**	**Accumbens L**
1000_3 L	0.915	0.887	0.882	0.710
1001_3 L	0.926	0.907	0.875	0.747
1002_3 L	0.909	0.884	0.873	0.687
1003_3 L	0.921	0.885	0.881	0.734
1004_3 L	0.935	0.908	0.867	0.778
1005_3 L	0.918	0.903	0.890	0.808
1006_3 L	0.922	0.904	0.878	0.770
1007_3 L	0.901	0.852	0.838	0.746
1008_3 L	0.924	0.904	0.880	0.796
1009_3 L	0.915	0.891	0.881	0.764
1010_3 L	0.927	0.885	0.879	0.787
1011_3 L	0.919	0.894	0.880	0.781
1012_3 L	0.908	0.910	0.870	0.827
1013_3 L	0.913	0.909	0.844	0.815
1014_3 L	0.910	0.899	0.898	0.711
1015_3 L	0.886	0.907	0.880	0.814
1017_3 L	0.881	0.816	0.863	0.616
1018_3 L	0.921	0.891	0.874	0.802
1019_3 L	0.919	0.898	0.886	0.830
1023_3 L	0.896	0.869	0.813	0.708
1024_3 L	0.912	0.914	0.865	0.834
1025_3 L	0.913	0.908	0.839	0.783
1036_3 L	0.911	0.904	0.856	0.710
1038_3 L	0.918	0.926	0.881	0.861
1039_3 L	0.911	0.915	0.881	0.825
1101_3 L	0.918	0.886	0.889	0.743
1104_3 L	0.917	0.898	0.893	0.768
1107_3 L	0.898	0.873	0.859	0.632
1110_3 L	0.912	0.858	0.888	0.779
1113_3 L	0.919	0.883	0.879	0.740
1116_3 L	0.876	0.829	0.843	0.671
1119_3 L	0.798	0.401	0.779	0.720
1122_3 L	0.900	0.831	0.820	0.757
1125_3 L	0.824	0.683	0.726	0.811
1128_3 L	0.733	0.816	0.675	0.661

Table 5.10: Dice overlaps for Basal Ganglia (BG) Left substructures using SS-MDDLS.

	Putamen	Caudate	Pallidum	Accumbens	BG
MDDLS R	89.69%	86.73%	85.72%	**74.18%**	84.08%
L	89.99%	86.49%	**85.79%**	**76.67%**	**84.74%**
SS-MDDLS R	**89.86%**	**87.23%**	**85.73%**	74.09%	**84.23%**
L	**90.08%**	**86.65%**	85.73%	75.59%	84.56%

Table 5.11: Average dice overlaps for Basal Ganglia structures. SS-MDDLS results are obtained using parameters d=1 and n=200. We show in different rows right and left hemispheres.

As we can see MDDLS and SS-MDDLS results are pretty similar. We get a slight improvement using SS-MDDLS but sometimes, the overlap is lower for small structures (pallidum and accumbens).

In figures 5.16, 5.17, 5.18 and 5.19 we show some qualitative results of SS-MDDLS method.

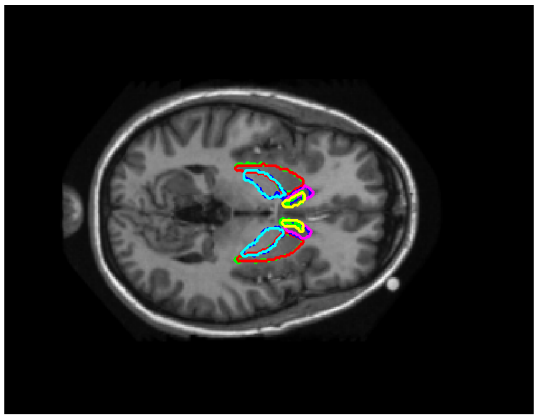


Figure 5.16: Original image with manual and SS-MDDLS segmentation contours superposed. Blue and green correspond to manual segmentation meanwhile red corresponds to putamen, magenta to caudate, cyan to pallidum and yellow to accumbens SS-MDDLS segmentation. Structures are over-segmented in central slices.

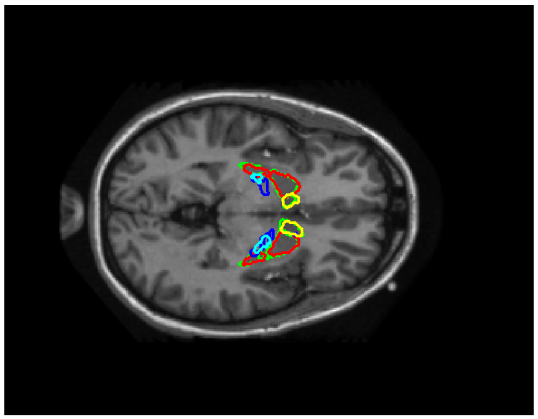


Figure 5.17: Original image with manual and SS-MDDLS segmentation contours superposed. Blue and green correspond to manual segmentation meanwhile red corresponds to putamen, magenta to caudate, cyan to pallidum and yellow to accumbens SS-MDDLS segmentation. The image corresponds to an ending slice. We can see how SS-MDDLS has troubles with pallidum and over-segments accumbens.

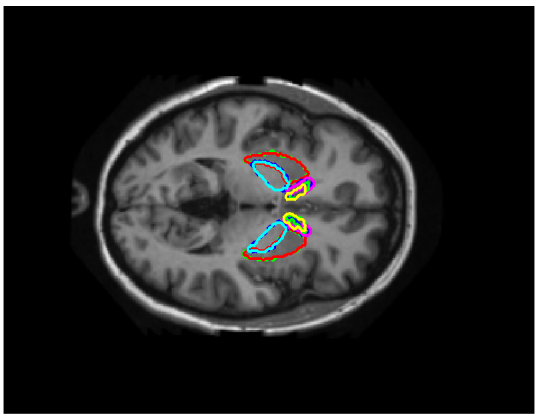


Figure 5.18: Original image with manual and SS-MDDLS segmentation contours superposed. Blue and green correspond to manual segmentation meanwhile red corresponds to putamen, magenta to caudate, cyan to pallidum and yellow to accumbens SS-MDDLS segmentation. Structures are over-segmented in a central slice.

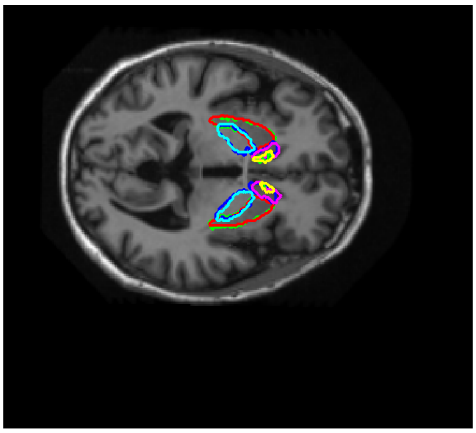


Figure 5.19: Original image with manual and SS-MDDLS segmentation contours superposed. Blue and green correspond to manual segmentation meanwhile red corresponds to putamen, magenta to caudate, cyan to pallidum and yellow to accumbens SS-MDDLS segmentation. Accumbens and pallidum are over-segmented meanwhile caudate and putamen fits the manual segmentations with high accuracy in a central slice.

As we see in figures 5.16, 5.17, 5.18 and 5.19, SS-MDDLS tends to over-segment the structures of pallidum and accumbens. We can also notice that this method has problems while segmenting starting and ending slices in the same way MDDLS has. Hence, we find an over-segmentation systematic error. On the other hand we find a better performance while segmenting putamen and caudate structures.

Chapter 6

Conclusions

In this document, we made a review of Sparse Representation Classification (SRC) and Discriminative Dictionary Learning Segmentation (DDLS) and its multi-class version using label consistency (LC-MDDLS). The main limitation of these methods was that the classification is performed independently for each voxel.

After a bit of research to learn about MRI segmentation and the state-of-the-art we focused on SRC, DDLS and LC-MDDLS methods. The way they work has been understood and their code studied in order to learn the execution flow better. After a replication of results has been done, we have analysed them by applying different visualisation strategies such as pseudo-probabily maps with MDDLS classification result and confusion graphics. Then a more exhaustive analysis about interesting data, such as label vectors or sparse representation, was performed. We noticed that there exist high confusion regions, for example boundary voxels and voxels between two structures which could be easily confused not only by algorithms but also by experts. In order to solve this problem we proposed a strategy inspired in Stacked Sequential Learning. We created feature vectors with resulting label vectors and sparse representation from the central voxel of the target patch and its neighbours. With this new data, we trained a decision tree classifier used to make the test classification. We called this strategy SS-MDDLS.

We validated the SS-MDDLS method for the segmentation of the Basal Ganglia of public databases presented in the SATA Challenge [14] using different parameters configuration.

We compared SRC, DDLS, MDDLS and SS-MDDLS. We obtained similar results using MDDLS and SS-MDDLS. On one hand, our proposed method over-segments small structures instead of under-segment them as MDDLS do. On the other hand, SS-MDDLS obtains a better segmentation for some structures according to manual segmentation.

In order to get better performance we might have to think about other features and use them for the feature vector. We could also try another ensemble classifier strategy such as adaboost or random forests. We would like to segment

other brain structures with SS-MDDLS to validate its usefulness.

Bibliography

[1] Oualid M. Benkarim, Petia Radeva, Laura Igual. Label Consistent Multiclass Discriminative Dictionary Learning for MRI Segmentation. VIII Conference on Articulated Motion and Deformable Objects (AMDO) 2014.

[2] Oualid M. Benkarim. *Multiclass Dictionary Learning for MRI Segmentation*. Tesis de Master en Intelligéncia Artificial de la UB-UPC-URV, gener 2014.

[3] T. Tong, R.Wolz, P. Coupe, J. V. Hajnal, and D. Rueckert. *Segmentation of MR images via discriminative dictionary learning and sparse coding: Application to hippocampus labeling*. NeuroImage, vol. 76, p. 11-23, 2013.

[4] Q. Zhang, and B. Li. *Discriminative k-svd for dictionary learning in face recognition*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2691-2698, 2010.

[5] Z. Jiang, Z. Lin, and L. Davis. *Learning a discriminative dictionary for sparse coding via label consistent k-svd*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1697-1704, 2011.

[6] `https://www.cis.rit.edu/htbooks/mri/`

[7] O. Bryt, and M. Elad. *Compression of facial images using the K-SVD algorithm*. IEEE Transactions on Image Processing, vol. 19, no. 4, pp. 270-282, 2008.

[8] M. Elad and M. Aharon.*Image denoising via sparse and redundant representations over learned dictionaries*. IEEE Transactions on Image Processing, vol. 15, no. 12, pp. 3736-3745, 2006.

[9] `http://en.wikipedia.org/wiki/Spin%E2%80%93lattice_relaxation`

[10] `http://en.wikipedia.org/wiki/Spin%E2%80%93spin_relaxation`

[11] `http://en.wikipedia.org/wiki/Basal_ganglia`

[12] `http://en.wikipedia.org/wiki/Neuroanatomy_of_memory`

[13] Eloi Puertas, Sergio Escalera, Oriol Pujol. *Generalized Multi-scale Stacked Sequential Learning for multi-class Classification.* Pattern Analysis and Applications, pp1-15,2013.

[14] MICCAI Challenge Workshop on Segmentation: Algorithms, Theory and Applications ("SATA")

[15] `http://en.wikipedia.org/wiki/Least_squares`

[16] Eloi Puertas.*Generalized Stacked Sequential Learning.* Doctoral thesis in mathematics and computer science. September 2014.

[17] `http://es.mathworks.com/products/matlab/`

[18] `http://www.fil.ion.ucl.ac.uk/spm/`

[19] `http://www.fil.ion.ucl.ac.uk/spm/doc/intro/intro.pdf`

[20] `http://es.mathworks.com/help/stats/treebagger.html`

[21] `http://es.mathworks.com/help/stats/treebagger-class.html`

[22] Leo Breiman. *Bagging Predictors.* Statistics Department, University of California, Berkley, CA 94720.

[23] `https://sites.google.com/site/pierrickcoupe/softwares`

[24] Pierrick Coupé, José Manjón, Elias Gedamu, Douglas Arnold, Montserrat Robles, D Louis Collins. *Robust Rician noise estimation for MR images.* Montreal Neurological Institute. Itaca. Medical Image Analysis, Elsevier, 2010, 14 (4), pp.483-93.