3D brain MRI classification: application to Alzheimer diagnosis.

Sagaf Youssouf

Université Paris-Est Créteil (UPEC)
International Master of Biometrics and Intelligent Vision
https://www.international-master-biometrics-intelligent-vision.org/

1 Introduction

Classification is an important task in computer vision. It consists of predicting a label given input features. It has been applied for natural images and medical images. In the medical field, several classification methods have been proposed to discriminate patients with Alzheimer's disease from patients with no disease based on T1-weighted MRI. However, most of the proposed methods were assessed in different populations. Here, we present a 3D brain MRI classification pipeline applied to Alzheimer's diagnosis. Our dataset consists of 10 nifty files from 4 participants cognitively normal adults and 5 individuals at various stages of cognitive decline aged from 45 to 72 years old. We extract the gray matter from the brain tissue and the result is converted to meshes. The next step consists of calculating some morphologic features before running a binary classifier. In the following sections, we presented step by step the processes used to accomplish the objective of the lab.

2 Loading nifty files

Loading NIFTI files format is done using `ninabel` library. We used `OS` module to list all the NIFTI files stored in a directory. The result is saved in a python list.

We created a function to extract the image and matrix from nibabel loaded NIFTI files for each file by looping over the python list containing the path of files using `nib.load()` function. Below is the code of the function used to load NIFTI files:

```
def getNiftiData(img_loc):
    im= nib.load(str(img_loc))
    data = im.get_fdata()
    return data, im

# Looping over the python list containing path of NIFTI files
data = [getNiftiData(path+"/"+s) for s in MRI_images]
```

Each NumPy array has a shape of `160x256x256`. It means there are 160 2D arrays (slice) of `256x256`. Examples of slices in grayscale are given in figure 1.

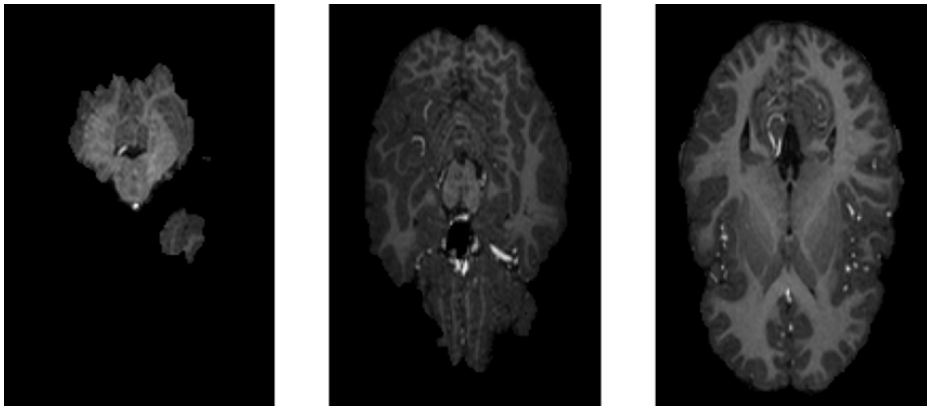

Fig. 1: Examples of slices of the 3D brain MRI

3 Gray matter segmentation

Having the pixels range of the gray matter `180, 210`, segmentation of the brain image is done by creating a mask using the following function :

```
def getMask(imgData, pxlRanges):
    binaryMask=(imgData>=pxlRanges[0]) & (imgData<=pxlRanges[1])
    grayMatter=np.where(binaryMask, imgData, 0)
    return binaryMask, grayMatter
```

Above function is applied for each image to extract gray matter from the image. Figure 2 illustrate examples of the computed images. The result for each gray matter 3D array is saved as a `.nii` file.

Fig. 2: Examples of slices of the 3D brain MRI

To convert the result to meshes, we save the performed gray matter of the 10 files using the following two lines of code by looping over all of the gray mappter 3D array.

```
gmSave = nib.Nifti1Image(grayMatter[i], affine=img_nifti[i].affine)
nib.save(gmSave, 'gray_matter/img_'+str(i)+'.nii')
```

We try also labeling image components using following line of code :
`labels, nlabels = ndi.label(grayMatterCopy[f])`. The result is an integer `ndarray` where each unique feature of the gray matter has a unique label in the returned array.

4 Mesh generation from 3D imaging data

Mesh is a way to represent in a more realistic and accurate geometrical description of 3D image data. In medical imaging case, we use VTK opensource tool to create meshes. To do so, we convert the NIFTY file vtk file for each one and we calculate features : volume and surface to be used as input for our classification model. Following function illustrate step by step the method used :

```
def computeFeature(data):
    reader = vtk.vtkNIFTIImageReader()
    reader.SetFileName(data)
    reader.Update()

    contour = vtk.vtkContourFilter()
    contour.SetInputData(reader.GetOutput())yguf(jojçu

    contour.SetValue(0,0.1)
    contour.Update()

    # compute volume and surface
    polydata = contour.GetOutput()
    Mass = vtk.vtkMassProperties()
    Mass.SetInputConnection(reader.GetOutputPort())
    Mass.SetInputData(polydata)
    Mass.Update()

    print("Volume   = ", Mass.GetVolume())
    print("Surface  = ", Mass.GetSurfaceArea())

# Example of using the function
compute_feature("gray_matter/1-Brain-alzheimer-GM.nii")
```

An example using this function is given. The function takes as input the path of the gray matter computed file saved as `.nii` file. The file is loaded and some morphologic features are computed (volume and surface). The process is repeated for all of the files. The label `A` is given for Brain-alzheimer files and label `C` is given for Brain-Control files. We have 10 patients and for each one, we computed his gray matter volume and surface, so we have 10 lines and 2 columns of data in our csv file as shown in figure 3.

	VOLUME	SURFACE	LABEL
0	294167.917537	516443.563676	A
1	332987.713479	620796.352754	C
2	305295.795421	522164.017973	A
3	102289.588198	150885.185806	C
4	50038.050655	90737.533603	A
5	354336.883329	747442.823847	C
6	341964.803751	756014.873326	A
7	198017.287522	204864.375575	C
8	151415.615028	179809.354628	A
9	50038.050655	90737.533603	C

Fig. 3: Values of volume and surface area of the gray matter

5 Classification

We used SVM for classification and the followings are the steps processed from the loading of the dataset to the evaluation of the model :

- Loading the dataset and feature scaling for the independent variable;
- Splitting the data into train and test using `train_test_split` function. 70% of data have been used for training and 30% for the test set;
- Classification model : We have 2 features and 2 classes, we used svm with a linear kernel as it is the simplest classifier and faster to train;
- Prediction of the output for our SGD Linear Model with the test set;
- Evaluation of the model;
- Evaluation on the test set;

```
Traing set results
ACCURACY for train set 0.5714285714285714
F1 SCORE for train set 0.4155844155844156

Test set results
ACCURACY for test set 0.3333333333333333
F1 SCORE for test set 0.16666666666666666

Confusion matrix
[[1 0]
 [2 0]]
```

Fig. 4: Evaluation results of the SVM model

Evaluation results are given in figure 4. We got an accurayc of 57,14% with this configuration. We need tune again to improve the performance of the model as accuracy on the test set result is very low.

6 Conclusion

In this assignment, we worked on 3D MRI obtained from OASIS Database. We used 10 NIFTI files with 5 participants cognitively normal adults and 5 individuals at various stages of cognitive aged from 45 to 72 years old. Once each file is loaded, the segmentation of the gray matter part of the brain is performed. The results are converted into meshes in order to calculate some morphologic features. The results were used as input for our svm binary classifier. In this lab, we learn how to extract interesting features from 3D MRI for classification. The program could be improved for critical analysis for the model evaluation.

References

1. Automatic classification of patients with Alzheimer's disease from structural MRI: a comparison of ten methods using the ADNI database
2. Binary classification using svm, `https://www.kaggle.com/matgarate/learning-binary-classification-with-keras-nn`.
3. SVM Classification for heart disease `https://github.com/karthikeyanthanigai/SVM-Classification-for-Heart-Disease-Prediction`.