Age Regression from Brain MRI
Group: 57

James Dorricott, Dhruva Gowda,
{dg1119, jd3114}@imperial.ac.uk

1 Summary

This coursework is a part of the Machine Learning for Imaging module at Imperial. The task was to construct three regression models that use different techniques to ascertain a patient's "brain age" from a brain MRI scan. The three techniques that we were asked to implement, analyse and compare were:

- Segmenting the MRI volumes into four classes, calculating their volumes and running regression on the relative volumes of each class.
- Running PCA on the MRI data to reduce its dimensionality to its principle components, and doing regression on those components.
- Designing a convolution-based regression network that takes in MRI volumes and directly outputs a prediction for the brain's age.

Our models were expected to get an MAE of approximately 7 years for part A, 5 years for part B, while there was no requirement for part C as it is was an open-ended task.

2 Part A

Our goal in part A was to design and train and 3D CNN for brain segmentation as part of an age regression pipeline, where the features extracted from the segmentation are the input to the regression model. Following the advice given, we began by first implementing a simple architecture, observing the loss, and only adding complexity if required. This was done on the validation data set of 5 examples in order to speed up model development. The first (and most valuable) change was to increase the depth of the feature maps (channels) in order to increase the model capacity. This led to a large drop in the loss and increased segmentation ability. The number of layers were also experimented with, however, adding a fifth layer led to the model overfitting, so the layers were reduced back to four. Finally, batch normalisation and ReLU activations were applied to the intermediate layers to aid gradient flow and introduce non-linearity, respectively. The weights were also initalized with the Xavier method. The CNN was then re-trained on the set of 47 images for a total of 50 epochs, with batch size of 4 and using the Adam optimizer (learning rate 0.001).

In the next stage of the pipeline, we used the pre-trained CNN to generate segmentation predictions on a set of 500 subjects. To assess the quality of the segmentations, a Dice Similarity Coefficient score was computed for each class (see box and whisker plots in Figure 1). We then used the predicted tissue segmentations to extract features which were hypothesised to be able to generate a prediction of a subject's brain age. These features were the relative volumes between each tissue and the overall brain (see scatter plot in Figure 1).

The final stage of the pipeline was to experiment with different regression models, using the computed features as inputs, where each row in X corresponds to the features of one of the subjects. We looked at linear regression, support vector regression (SVR), linear regression with polynomial features and random forest regression. The results in terms of error metrics are show in Table 1. Our best (cross-validated) estimator was observed to be a linear model with polynomial features ($k = 3$).

The final stage was to use our pre-trained CNN to generate segmentations on an unseen hold-out set. Once (polynomial) features were subsequently extracted, and the linear model re-fitted on the full 500 subjects, we then generated brain age predictions for this test dataset (see section 5).

Estimator	Average R2	Average MAE
Linear	0.718	7.704
SVR	0.652	8.338
Linear+Polynomial	0.730	7.330
Random Forest	0.710	7.600

Table 1: Average regression error metrics using two-fold cross-validation.

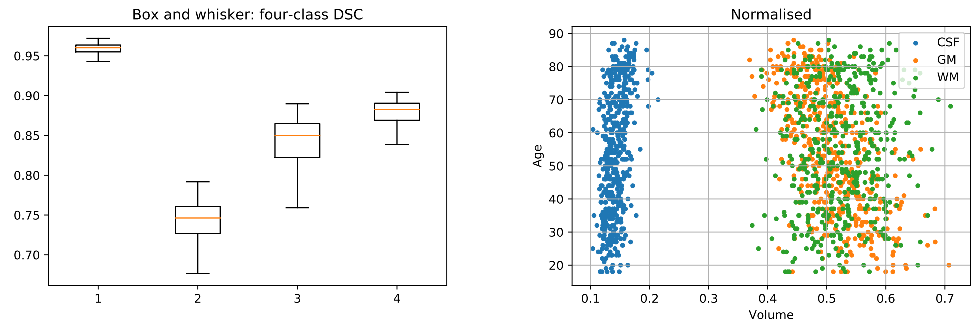

Figure 1: Left: Dice Similarity Coefficients for background (shown as class 1), CSF (class 2), GM (class 3), and WM (class 4). Right: Scatter plot of learned (normalised) features versus subject age.

3 Part B

Unlike part A, part B used the grey matter maps from the MRI data set that was provided. We found that the best results were achieved by our regressors when we first preprocessed the data with smoothing and downsampling, in that order. Smoothing was done using SimpleITK's GradientAnisotropicDiffusion. Downsampling was done by first smoothing, again with SimpleITK's discrete gaussian function, followed by selecting every second voxel in the volume (effectively reducing its size by half). The effect of this is shown in Figure 2.

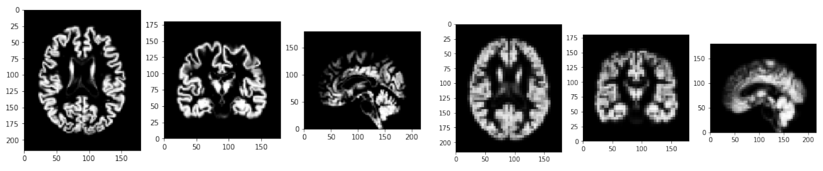

Figure 2: Preprocessing done for part B. The image on the left shows the original, and right shows the effect of preprocessing.

After preprocessing, the 500 data samples were split into two training and validation sets through 2-fold randomized cross validation. This was done using sklearn's kfold method with shuffle set to true, to ensure that global data statistics were preserved in each subset. Following this, two seperate PCA functions with n = 0.95 (to preserve 95% of data variance) were fitted on the test sets, and used to transform the test and validation sets of each fold. Finally, three regressors were trained on the dimensionally reduced datasets: linear regression, support vector regression, and gaussian process regression. We found that linear functions fit the PCA data best as the best regressor was found to be linear regression, followed by SVR with a linear kernel. Optimal hyperparameters were found using sklearn's GridsearchCV function. In the final stage, PCA was performed on all 500 datapoints and tested on the hold-out set.

4 Part C

Part C also used the grey matter maps from part B. The pipeline began with preprocessing the data. Here, we used DiscreteGaussian smoothing to first remove noise in the data, then we resampled using the resample helper function provided to us. Best results were found using the default image size of [64,64,64] and image spacing of [3,3,3]. This data was then split into two equal folds in a similar way to part A and B. The different processing steps can be seen in Figure 3 below.

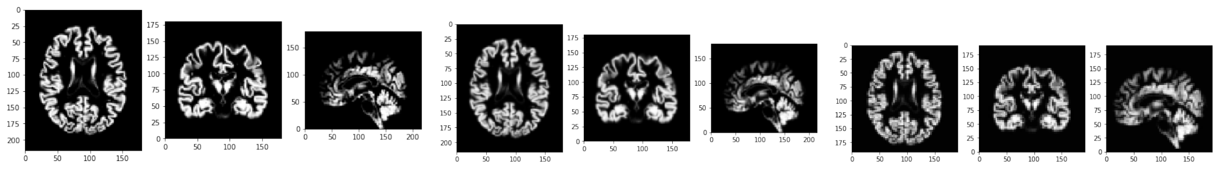

Figure 3: Preprocessing done for part C. From left to right: original, smoothed, resampled

The processed data was then fed into a convolutional neural network built on top of a standard LeNet architecture. The network has 5 3D convolutional layers, two 'pooling convolutional' layers, and three linear layers. The pooling layers were defined as 3D convolutional layers with kernel size 2 and stride 2, so the network could learn its own pooling function/downsampling. Batch normalization was applied after each convolutional layer, followed by ReLU activation functions. With the exception of the second convolutional layer which had kernel size 3, padding 1 and stride 1, all convolutional layers had kernel size 5, padding 0 and stride 1. The model was trained using a batch size of 64 and the Adam optimizer, with a learning rate of 0.001 and for 100 epochs. The loss function used was L1 loss (Mean Absolute Error (MAE)). The network was finally retrained on all 500 datapoints and tested on the hold-out set.

5 Age Regression Results

The final results on the hold-out test set were an MAE of 10.35 for part A, 5.34 for part B, and 5.23 for Part C. The scatter plots for each part can be seen in Figure 4. These results are good, but do not match the state of the art.

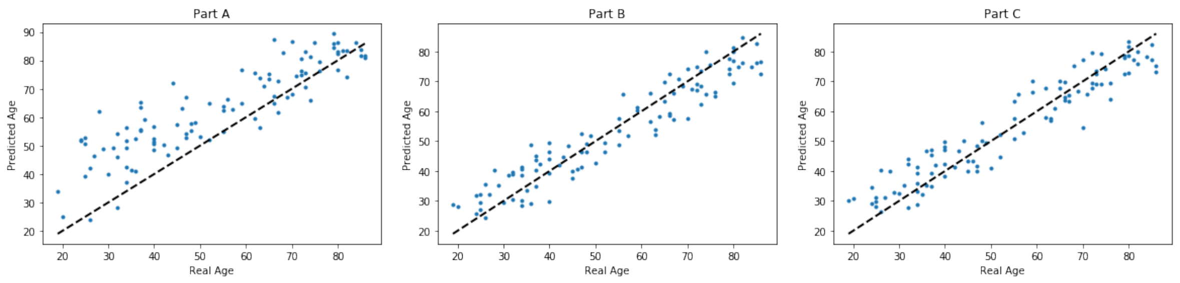

Figure 4: Scatter plots for age regression for three different approaches (part A, part B, and part C, from left to right).

Test results for parts B and C were better than validation results by a small amount. In part B, linear regression achieved a validation MAE of 5.55 averaged over both cross-validation folds. In part C, we got an average cross-validation score of 5.81. However, in part A, our cross-validation scores were significantly better than our result on the held-out test set. We believe this was due to the network struggling to segment cerebral fluid, as shown in Figure 1 (left). While our network in part A was less complex than in part C, part C's lower error metrics could point to the increased predictive power of convolutional regression networks over feature extractors. The agreement between validation and test scores for parts B and C could on the other hand suggest that these networks are more robust and handle unseen data well.