Assessed Coursework 2020-2021 - Programming Foundations for Medical Imaging Analysis (MPHY0030)

Report- Part 2

Emmanuel Bijaoui (S/N: 14034511)

February 2021

Contents

1 Introduction

The contents of this section outline the Report for Part 2 of the 2020-2021 assessed coursework, titled 'Spatial Data Augmentation using a Gaussian Spline'. For reference, both parts of this coursework along with all relevant documentation, functions and toolboxes can be found **here**.

This exploration is itself divided into two sections (*Gaussian Spline Class* and *Free-form Deformation Class*, before concluding with an actual implementation through the *Implementation* section. In the first two sections, we aim to form and establish certain programming 'Classes' and 'Functions' to help streamline the execution of the implementation phase. These details and considerations will be expanded on in the below sections, and where relevant, an indication to the report question will be provided (e.g. **Q[1]** for Question 1).

The work in this section is based primarily off of the provided paper by Fornefett et al [REF], describing the implementation of Radial Basis Functions directly to Medical Images.

The entirety of this section has been programmed in MATLAB, using MATLAB 2017b.

2 Gaussian Spline Class

Here, the first action was to create an RBFSpline class, consisting of 3 constituent functions to help execute any Gaussian Splines. Details of the functions and their inputs are available as comments in the code proper.

a) Question 1

In Fornefett et al, the authors describe the many approaches and implementations of Radial Basis Functions (RBFs) in the case of medical image registration. Medical image registration largely focuses on the mapping of one set of coordinates to a new image, and thus allow us to infer insights on the image provided. Two main applications involve the ability to detect differences in a similar image (e.g. presence of tumor, or shrinkage of tumor from same patient following treatment) and the comparison between different imaging modalities (i.e. MRI of the brain and PET). The purpose therefore is largely to determine a mapping (or transformation) that when a applied to a certain image produces a certain output and then use this mapping to identify, predict, and evaluate similar other use cases. This is outlined by our FIT and EVALUATE functions respectively.

The approximation function used is given by Fornefett et al as:

$$\begin{pmatrix} \boldsymbol{\phi} & \mathbf{P} \\ \mathbf{P}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \mathbf{q}_k \\ 0 \end{pmatrix}$$

Or, otherwise expressed:

$$\boldsymbol{\phi} + \mathbf{P}\boldsymbol{\beta} = \mathbf{q}_k \tag{1}$$

Where $\boldsymbol{\phi}$ represents the matrix applied RBF to a set of initial Points, $\mathbf{P}$ represents a matrix of those initial points and $\mathbf{q}_k$ the target or resultant points of the transformation. $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ represent the weighting functions and the coefficients of the derived spline function.

As mentioned, using a polynomial component has some shortcomings, namely in the effect they have on the global solution of our result. However, not all of the RBFs explored can allow for this elimination, as it follows that in order to obtain a result and solve the above solution, we need to obtain a non-singular matrix.

The purpose of this exploration focuses on the features of the Gaussian RBF in particular. This RBF is defined as:

$$RBF_{Gauss.} = e^{\frac{-r^2}{2\sigma^2}} \tag{2}$$

Whereby 'r' represents the euclidean distance between the initial points (p) and target points (q).

The Gaussian solution therefore is positive semi-definite, allowing for us to obtain a non-singular matrix without the use of an additional polynomial component [**Q1**].

b) Question 2 & 3

Expanding on this, we can then isolate our solution to solving for $\boldsymbol{\alpha}$ as $\boldsymbol{\beta}$ relates directly to the expansion of the polynomials in the RBF spline.

The equation to be solved can additionally be expressed as:

$$(\boldsymbol{\phi} + \lambda \cdot \mathbf{W}^{-1})\boldsymbol{\alpha} = \mathbf{q}_k \tag{3}$$

Where $\mathbf{W}$ represents a matrix that consists of the weightings of the individual landmarks in the image, and λ is an overall weighting parameter to be tuned and selected.

In the first instance (i.e. function FIT) we are attempting to solve for the spline function that provides a mapping between the original points and a set of target points. Therefore mathematically, we are aiming to solve α as depicted above.

If we represent $(\boldsymbol{\phi} + \lambda$

Normally, a simple multiplication by the inverse would allow us to then make alpha the subject and solve the equation algorithmically. However, in this case, we cannot assume that Q is invertible. Rightly so, as the image loaded from *'example_image.mat'* does not have square dimensions. One solution to this would be to reduce the matrix using Singular Value decomposition into constituent U and V matrices, and leverage their orthogonality. However, in this instance, it is more efficient to apply the psuedoinverse (denoted by a super-scripted '+'), which allows us to manipulate the system of matrices in a similar way to an inverse. That is:

$$\begin{aligned} \mathbf{Q} \cdot \boldsymbol{\alpha} &= \mathbf{q}_k \\ \mathbf{Q}^{+}\mathbf{Q}\boldsymbol{\alpha} &= \mathbf{Q}^{+}\mathbf{q}_k \\ \mathbf{I}\boldsymbol{\alpha} &= \mathbf{Q}^{+}\mathbf{q}_k \\ \therefore \boldsymbol{\alpha} &= \mathbf{Q}^{+}\mathbf{q}_k \end{aligned} \tag{4}$$

By applying $\mathbf{Q}^{+}$ to $\mathbf{q}_k$, we can therefore solve fot the spline coefficients $\boldsymbol{\alpha}$.

c) Question 4

Next, in function 'EVALUATE', we seek to work backwards on what we implemented in 'FIT'. That is, now that we haver determined the spline values, we want to apply these spline coefficients to a new set of points and assess the outcomes. In this case therefore, the 'query' points represent the initial points of this new image for example, whereas the solution (*'predict'* in the code) are the newly transformed image voxels. The control values in this case allow us to compare the result to an initially formed 'fit' function. That is, it allows us to implement the 'r' value in the Gaussian function $\boldsymbol{\phi}$, by applying it as the euclidean distance between the query points and the control points. This allows us to essentially assess how different to the control points our transformed values are. This therefore suggests that the control points must be points that exist within the set of initial points which were passed through our 'FIT' function.

d) Question 5

In the 'evaluate' section as mentioned, we are applying a defined mapping ('splice') to a set of new points. In this instance therefore, the function lambda becomes less relevant as an explicit mapping already exists, rather than the general function of lambda.

e) Question 6

One method, particular to MATLAB, involves the vectorisation via predefined matrices. This significantly increases the compute time as compared to an iterative loop, as each individual cell in the matrix executes itself in a one-to-one method, as opposed to a loops potential one-to-many. Additional aspects such as pre-instantiating vectors to be populated as opposed to an iterative population, as well as monitoring of the hyperparameters and iterations also aid in managing large data point sets.

f) Question 7

The variable σ is a tunable hyperparameter in our Gaussian RBF which represents the variance. The principle of a RBF kernel is that we select nearby points in an image, and allow them to be clustered as 'nearest neighbours' in the calculation of our solution. Essentially, this variable modulates the 'width' of our Gaussian distribution, and therefore increases the region of 'neighbourhood' for our selected point. A larger σ would therefore encourage a wider range of values to be included as 'similar' values; increasing the speed of the solution, but ultimately compromising on the accuracy. The inverse is also true; a σ that is too low, can be 'too precise' in its variance, and therefore miss out on results in a form of 'overfitting'.

3 Free Form Deformation Class

In this section, the focus lies in applying various random affine transformations to a set of points, and use these outcomes to then warp an initial image. This is based on the principle of matrix transformation on a 3D point, represented by the expression:

$$\begin{pmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \\ 1 \end{pmatrix}$$

Where the solution provides us with a set of transformed 3D points. Most notably, the values t_x, t_y, t_z represent the translation values in a matrix, present in a null matrix with a diagonal of ones. A transformation with values solely along the diagonal (a, e, i) represents a scaling of an image. It is also important to note that subsequent transformations are just the matrix product of individual transformation matricies.

a) Question 8

In my approach, I decided to restrain the affine transformations to the translation and scale transformations. This would then require the multiplication of both a randomised translation function as well as a randomised scale function. However, the parameters indicate that a random value between 0 and 1 must be used as a strength value, and generate an identity matrix if the value is 0. To do this, I decided to randomly select a value from 0 to 1, and subsequently multiply it by a randomly selected value of either -1 or +1. This would then be multiplied to the strength value and summed to 1. This ensures that the resultant scaling matrix would be an identity matrix should the strength value be 0, and additionally maintains the scaling values within the sphere of expectancy for this kind of image; a maximum 2x scaling of the image ensures that subsequent training of the RBF does not become too convoluted.

Additionally, the translation values are capped at a displacement no greater than half the respective dimension (i.e. if the image is 240 units wide, the maximum translation to randomise across would be 120 units). The aim of this is to restrict the chances of a displacement of a point outside our frame of reference. An additional stop-measure implemented was to assess each translated point, and if this value was larger than the limitations of the image, it would be re-positioned to the edges of the image frame. For example, if the image was bounded by dimensions 240 x 240 x 30, and the x-value is translated so that it exists at 350, it would automatically be bounded to 240. Additionally, shearing transformations were omitted from this process, as this represents a 3 point transformation that is deemed biophysically impossible, or at the very least improbable. Scaling on the other hand, is possible (inflammation, growths etc), and therefore was retained. This scaling matrix is then pre-multiplied to the translation matrix to provide us with an overall transformation for this point.

b) Question 9

While the transformations selected may be biophysically possible in general, there is no discrimination as to which voxel represents what body part. Therefore, a scaling of a 'bone' voxel, is not realistic, and therefore additionally requires an external party to assess the image properly. The notion of image registration again is to highlight regions of change, to then derive further conclusions.

c) Question 10

In my implementation, the method used to create a warped image, involves a series of lower-level functions, all existing in the class 'FreeFormDeformation'. Largely however, we obtain a set of initial 'control' points, which we then randomly transform as detailed above. This then serves to train our 'fit' function in the RBF class, and will aim to output the coefficient of the spline determined. Once this is done, we attempt to solve it forward, by taking the values provided in the image itself, and applying the coefficients as well as a reference to the original control points to obtain a new set of warped images. Essentially, this maps the processes detailed in the RBF class, primarily the 'Fit' and 'Evaluate' functions.

d) Question 11

e) Question 12

While the results have not been displayed, it is possible to infer one aspect of this last question regarding the tuning of different hyperparameters. Firstly, as mentioned above, we know that changing the Gaussian paramter will cause an effect on the 'nearest neighbours' scenario. In an imaging example, it can be hypothesised that increasing this value will increase the number of constituent voxels on which any transform or function is applied. This essentially aggregates them as one unit, and therefore may suggest that if this value is too high, there may be increased regions of blur. An comparative example would be the reduction of pixel size of an image. The strength parameter as mentioned, modulates the scaling of the image. Since this is moderated by the randomness of our function, we would be unable to say whether this specifically enlarges or shrinks our relative image locations. However, we can say that an absolute larger strength parameter will cause the most drastic changes to a single set of localised points, similar to an 'inflation'. Lastly, the number of control points can be hypothesised to enhance the relative accuracy of the plot. With more control points, the trained 'FIT' function may develop a more accurate mapping, and therefore result in clearer images. However, there surely will exist a tradeoff with the number of control points, as at a higher level, these may overfit and magnify even small 'landmarks' or aberrations in the image.