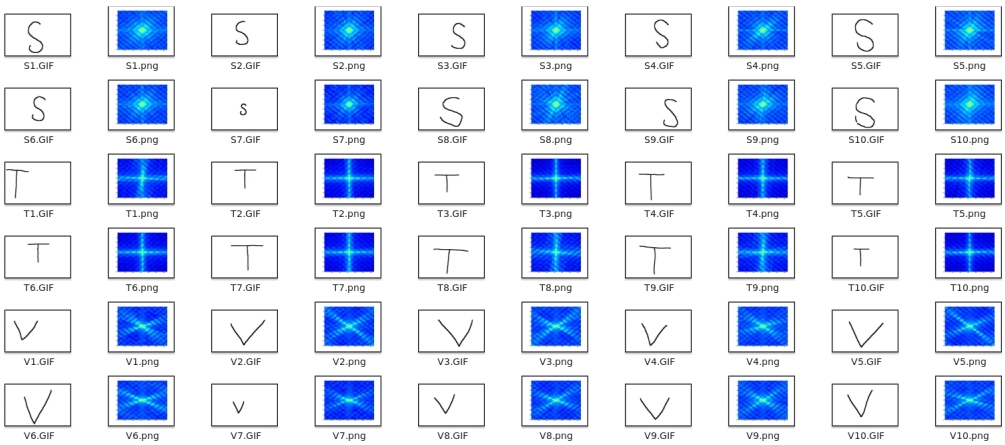
Symbols, Patterns and Signals CW2 – Hakeem Kushoro

Introduction

The aim of this Coursework is to create a classifier which will take a hand-drawn letter (Either S, T or V) and utilise feature extraction to sort them into the relevant letter. I shall analyse the character data and select key features to decide on what letter has been drawn. These key features shall all be obtained from the Fourier Space.

Approach to analysis in the Fourier domain

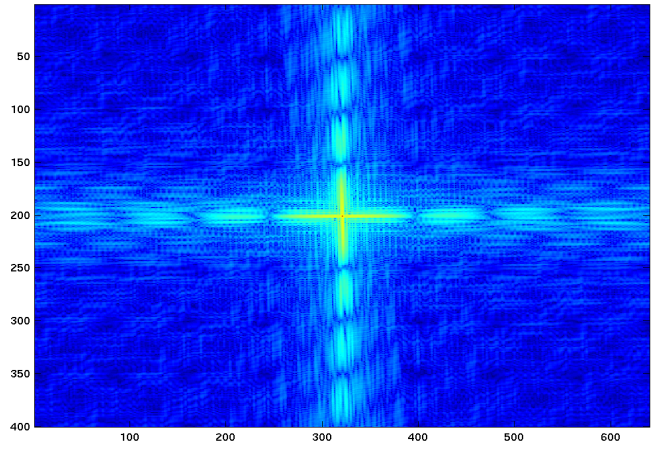
To begin with, I was given 30 hand-drawn letters – This was to be used as my training data. 10 of them were drawings of the letter 'V', 10 were drawings of the letter 'T' and 10 were drawings of the letter 'S'. Using the code supplied from the lecture notes, I produced a shifted Fourier Matrix and then produced a colour graph to visualise the Fourier space shift from each letter and the results looked like this:



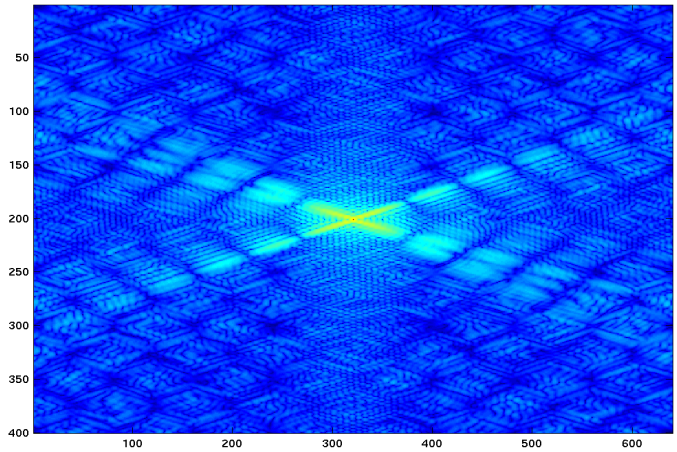
This gave me a visualisation of the Fourier space for each letter. Each pixel in the visualisation plot represented a value in the matrix (With each value representing a certain shade of a colour – Colorbar to the side).

Upon obtaining the Fourier Space for every letter, I noticed similarities between Fourier Spaces of the same letter and differences between Fourier Spaces of different letters. This was my basis for the features I chose to distinguish the handwritten letters from each other.

To withdraw distinguishing key features, I analysed multiple Fourier shifts from all 3 letters but to bring about a specific case just for the purpose of this report, I'll compare the Fourier Shift visualisations for the files 'T7.GIF' and 'V7.GIF' (Shown below):



T7.GIF

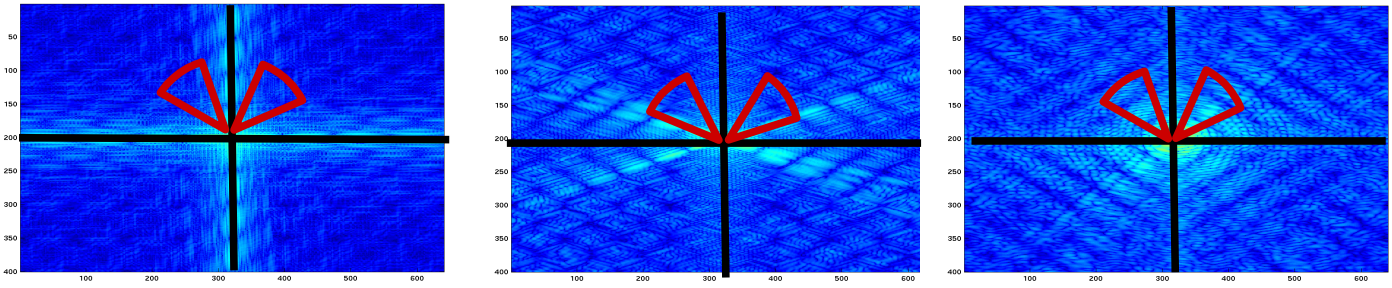


V7.GIF

Checking each and every Fourier Space, I'd already made the assumption that every Fourier Space of the same letter was extremely similar based on my comparison of all of them. Hence, I had a good idea of a rough 'Base shape' for each of the letters. For 'T', this was a plus. For 'V', this was a cross. For 'S', this was a small diamond with alternating outward diamond pulses.

At this point, I decided to look for areas of both Fourier spaces that were very different. Since I could fairly use both these Fourier spaces as a base case for the other instances where the letter was the same, if I could find areas where the colours on the map/values in the matrix were different, that would be a good indication of identifying the letter.

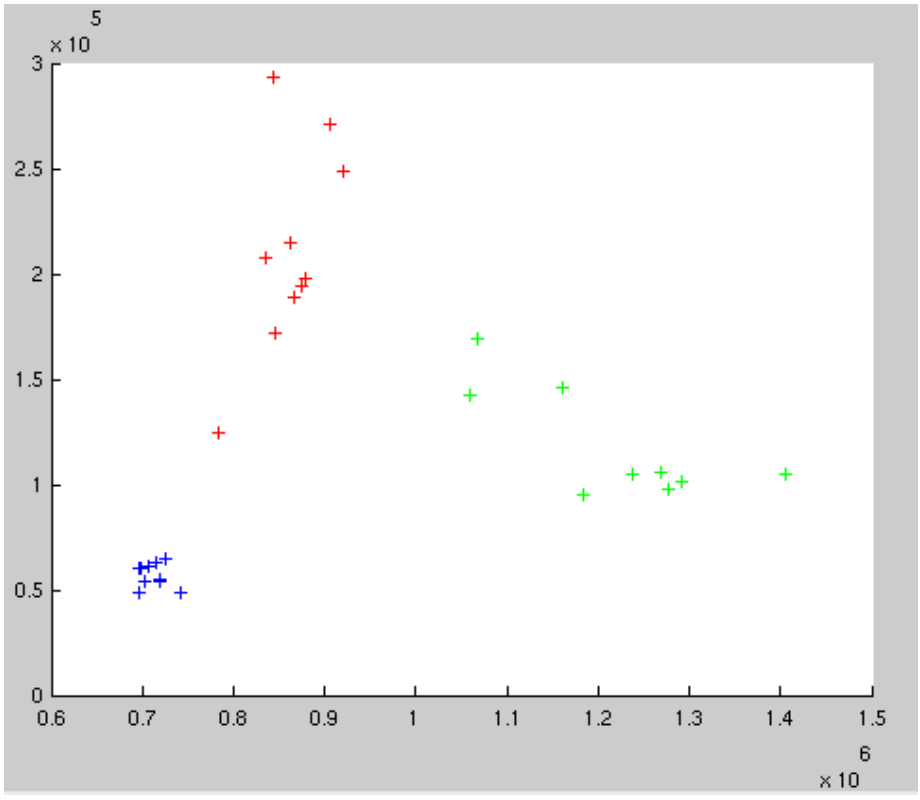
There were multiple choices I could have taken by in the end, I decided to use the areas designated below as indication features:



I extracted two features from each Fourier space. One feature came from the sum of all of the Matrix values in both red circle sectors (The circles have centres in the middle of the circle and the sectors are at an angle between 55 and 80 degrees) and the other feature came from the sum of all of the Fourier Matrix values in both black lines. The reason these areas were chosen were due to the diversity of intensity between the three letters. For example, if we take the black crosses, we can see the blue intensity is predominantly light in the 'T' Fourier space at every point in the cross while there are some points in the 'V' Fourier space black cross where the intensity is darker. And, because we've made the claim that each of these specific Fourier spaces is very similar to and represents the base case of their own letters, we can assume that this is the case for all 30 Handwritten letters.

Using this, we produce 2 variables for each Training data letter. We can use this to plot every letter on a scatter graph with the black cross variable on the x-axis and the red circle sectors variable on the y-axis.

The graph ended up looking like this:



Red point = 'S'
Blue point = 'V'
Green point = 'T'

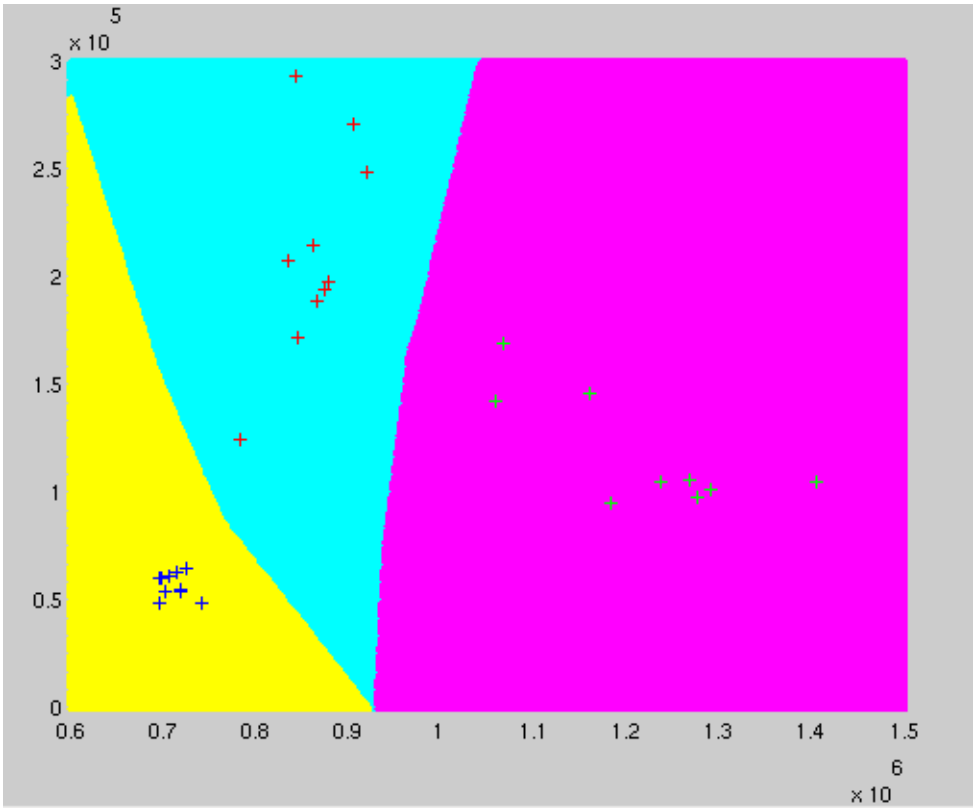
As you can see, the points are all grouped up into classes based on their letter. The letters recognised as a 'V' and the letters recognised as an 'S' are highly correlated (as Variable 1 increases, Variable 2 increases by a large amount). This trend continues until it passes a threshold (Around 1x10^6 on the Variable 1 axis) and after that, it is deemed to be a 'T'. However, my test data will most likely plot onto other points of the graph where the training data has not been plotted (There are so many combinations so (x, y) pairs that it would be extremely unlikely one of my test points would take on the same (x, y) value as one of the training points). So, I need to produce a plot on this graph by looking at each individual point (or pixel) on the graph and using an effective classifier in order to decide what letter that point would associate with if a point was plotted there.

Producing a plot for the Decision Boundaries

There are multiple classifiers I could use but at this point, I'm going to use the nearest neighbour classifier. This classifier will take in a point and find the closest point to it (finding the point with the shortest Euclidean distance). The point being used for the nearest neighbour classifier would then associate itself with the same class as the closest point it picked up.

To produce a plot, I have to consider every point on the graph, as any one of those points could be plotted. Hence, I consider every point in the entire range of the graph and use the nearest neighbour classifier on each point to decide where it would be classified as a 'T', 'V' or 'S'.

Once I had a collection of what each individual point was classified as, I plotted it amongst the training data and this was the result:



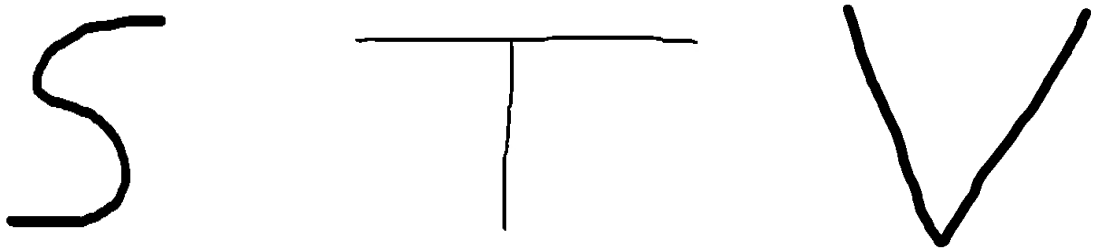
Colours of the crosses link to the same letter as the last figure.
Yellow background = 'V' Cyan background: 'S' Magenta background: 'T'

Now, we have a graph with a letter for each plotted scenario and we can test it using my own test data.

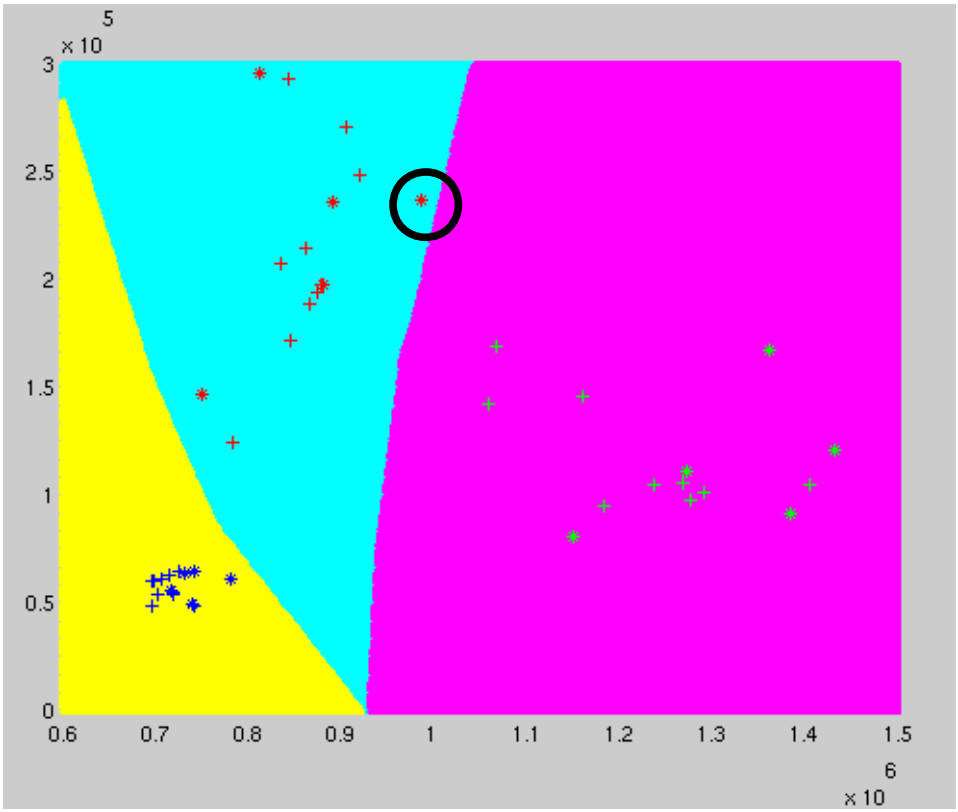
Test data

Now that we have used the training data to define our classes and decision boundaries, we can move onto the test data. While the training data is responsible for forming the classes, the test data is responsible for testing the classes and boundaries are correct by producing data that the code has not been given information for (A.K.A – What letter a certain piece of test data is).

I produced 15 samples of my own drawn letters (5 for each of the letters 'S', 'V' and 'T') to test the class with. I used varying pen thicknesses to see if the classier and features could handle different writing styles. Here are 3 samples out of the 5 (1 for each letter):



I used the nearest neighbour classier to decide what class they are in and plotted them on the graph, alongside the training data. I plotted them in asterisk form (*) to make them look distinct and got this:



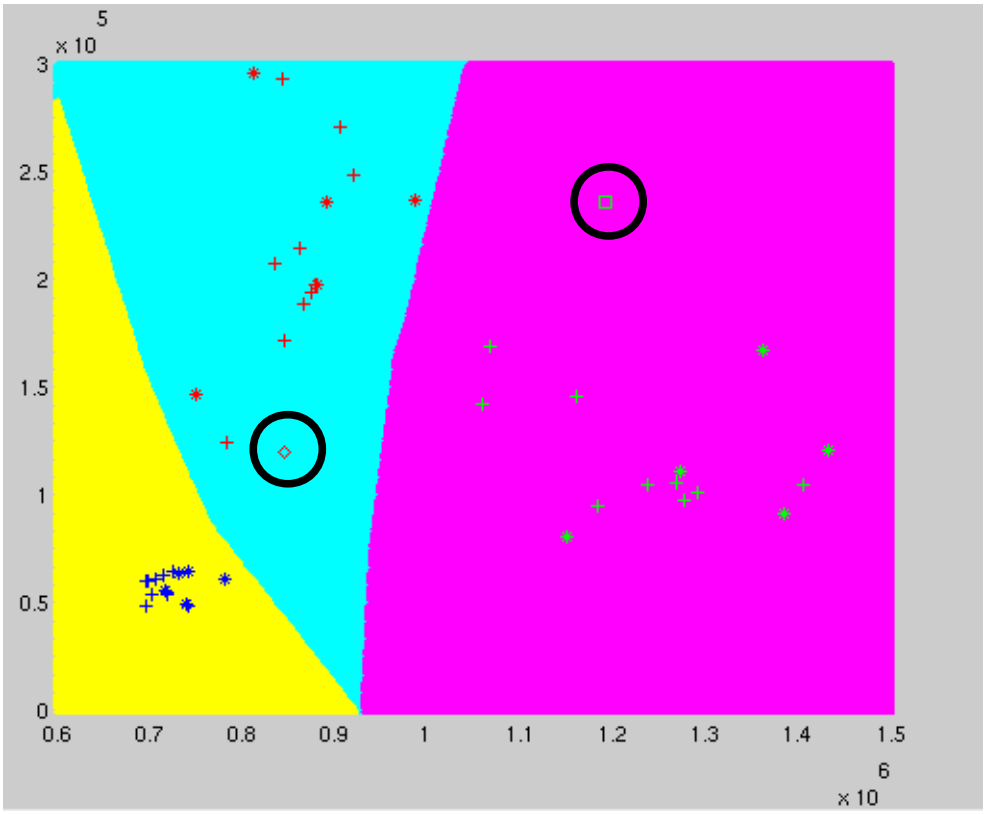
All letters seem to fit correctly so the classier was correct in this instance. However, take note of the point circled above. You can clearly see that the letter that was drawn as an 'S' was almost classified as a 'T'. The circled point is the file named 'XS4.GIF'. This may be because the bottom of the 'S' in the file is a straight line and hence, may have been partially mistaken for the top straight line in the letter 'T' as both have similar frequency composition.

The accuracy of this makes it very suitable for a mobile/tablet application that needs to recognise hand-written letters. However, a downside of this is the amount of time that the code takes to load up (Takes about 5-10 seconds to produce the graph) which is due to the plotting of the decision boundaries (Every point on the graph is classified). This might not be great if the phone/tablet application needs to keep up to speed.

Attempts with 'A' and 'B'

These classifiers were built for the letters 'V', 'S' and 'T' so I used the provided 'A' and 'B' to check what would happen if I used the nearest neighbour classier on the two letters using the same two features I used on the training/test data.

I put the two files through the code to be classified. The plotted graph is just below. The 'A' file is the plotted diamond and the 'B' file is the plotted square. As you can see, the 'A' has been recognised as a 'S' and the 'B' has been recognised as a 'T' (Both circled). On top of that, both letters seem to be in a comfortable position in both sections. In other words, were you to slightly change some features about the letters A and B, there is a great chance that they will still be in the same class.

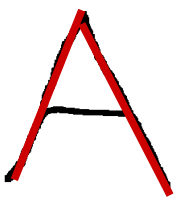


A = V?

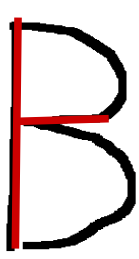
If you take a look at A, you can see that it follows a similar structure to 'V'. The outside base of 'A' ('A' without the middle line) is basically 'V' turned upside down. Hence, the point where the diagonal lines of 'A' meet is extremely similar to the same way the diagonal lines of 'V' meet which contributes to the Fourier Space being similar to 'V', as well as the shape of both letters (Both 'A' and 'V' have two straight lines coming off of each other at about 45 degrees).

B = T?

A lot of T's seem to be grouped together because they relate to each other through the relatively straight lines being at right angles to one another (The top part of 'T' is a straight line roughly perpendicular to the vertical straight line of 'T'. The letter 'B' follows this trend as the vertical line of 'B' as three more horizontal straight lines coming off as right angles to it (Roughly). In fact, if you look closely at 'B', you can see a small subset of 'T'. The vertical straight line combined with the middle line trailing off (Before splitting into the two curves to complete the letter 'B'.

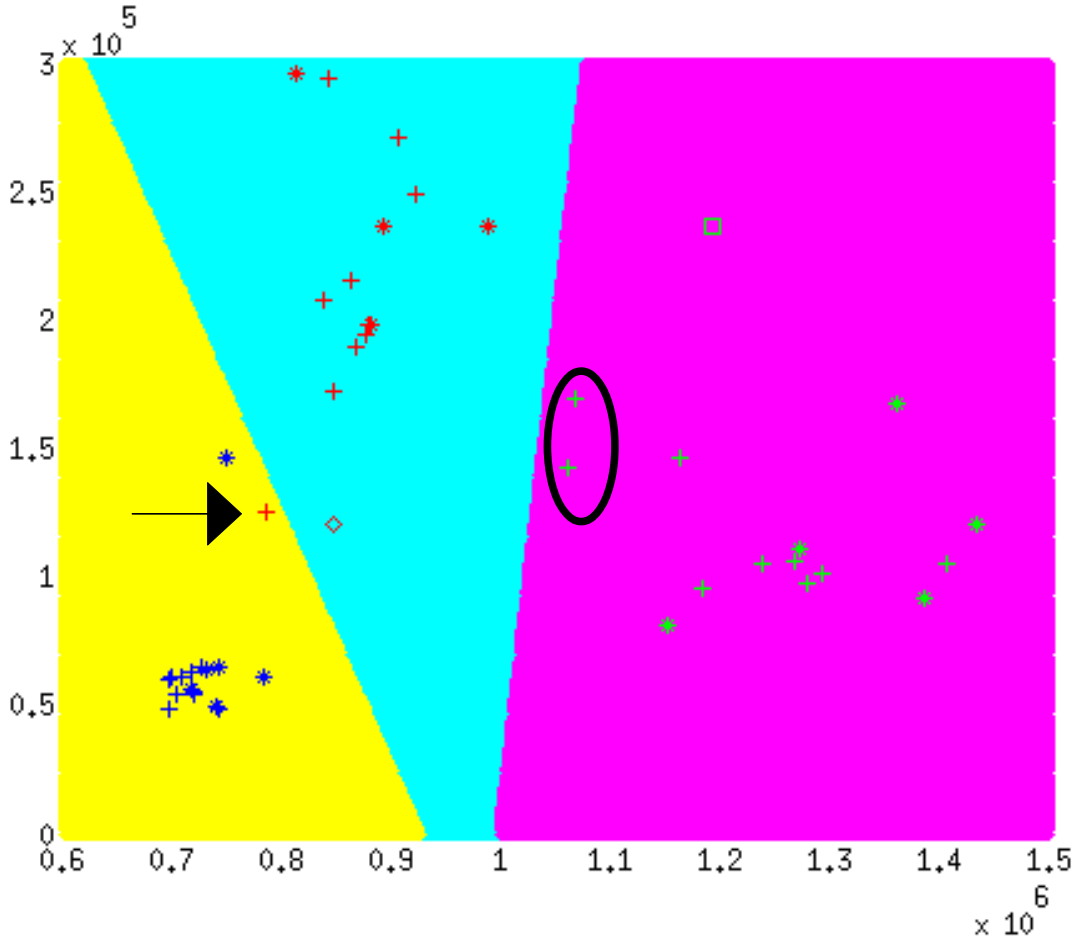


A and B GIF files with the V and T subsets highlighted. These both contribute to the Fourier Space and hence contribute to their classification.



Alternative classifier – Nearest Centroid

Now, I am going to explore what results I yield when using the Nearest Centroid classifier instead of the nearest neighbour. I chose this because the 'S' and 'T' class are producing an observant trend, however, they are not as clustered as they could be and this affects the nearest neighbour classifier. For example, had one of the 'T' letters dragged to the left by a great amount, this one anomaly would affect the decision boundary of the 'S' class. The nearest centroid classifier prevents the boundaries from being dramatically changed by just one plot. To adjust to this, I shall change the Nearest neighbour classifier to the Nearest Centroid classifier (But, the nearest neighbour shall remain in my code – Just commented out but marked in the code):



Upon comparison between this graph and the graph using the nearest neighbour classifier, you can instantly tell that the boundary lines between classes have gone from slightly curved to instantly straight. This is because in the nearest centroid classifier, we only had 3 points to compare all the plots to (The mean of each class) and since you can put a straight line between each pair of mean points, you can construct a perpendicular bisector. This perpendicular bisector is the straight line such that all points on that line are equal distance from each of the points. In other words, these lines form the class boundaries.

You can also notice the red cross (Training data) in the yellow section (Pointed to by the black arrow). This is due to the 'S' being mistaken as a 'V'. This may be due to the diverse range of the 'S' letters which may result in some letters being too far from the mean of the 'S' class. In this scenario, this indicated 'S' just happened to be closer to the mean of the letter 'V' and was, hence, placed in the class 'V'. Also notice the blue star just a little above it (The Test data). The 'V' data is usually very compressed and clustered in a small area while this one is so far from the rest. It is much more likely that this was also an 'S' classed as a 'V'.

Overall, I would select the nearest Centroid classifier over the Nearest Neighbour as it is able to select clearer, more linear and more discrete boundaries in comparison, which may defect a little if dealing with anomalies/classes with a huge range but this is only by a small margin (The two wrongly classed letters were only *just* mis-classed) but works very well with other classes, especially in a general sense (Which is needed if this is being used for mobile/tablet applications).