Evridiki Christodoulou, Stelios Stylianou
ec15441 ,ss15293

Symbols, Patterns and Signals, Coursework 2
Fourier Feature Extraction

1. Introduction

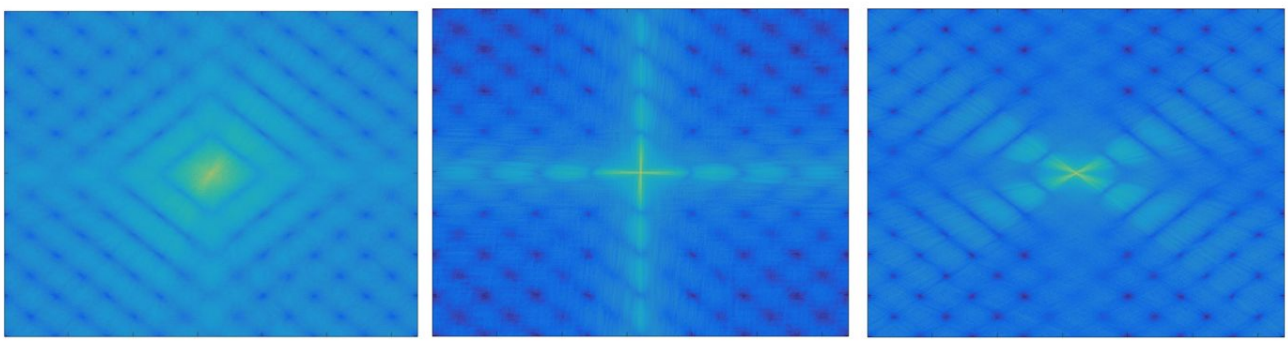
Our report demonstrates the process of feature extraction from a data set, by selecting spectral features in the Fourier domain, and then the creation of a classifier that will be able to classify future data according to these features. Additionally, the classifier's accuracy is tested using some test data.

2. Analysis in the Fourier Domain

In order to analyze an image and extract information about its geometric characteristics, we had to move from the spatial domain to the frequency domain using the Fourier transform. The Fourier transform decomposes an image into its constituent sinusoidal components and the resulting image is the image in the Fourier domain. The central point in the frequency domain is the DC point which represents the average value of the image and it is usually the brightest point in the Fourier domain. Brighter points represent frequencies with higher magnitudes and thus contain more information. Points that are close to the DC point represent low frequencies and the further away a point is from the centre, the higher its corresponding frequency is. Lower frequencies, which appear close to the DC point, represent the contrast of the image while higher frequencies represent the details of an image. Furthermore, edges in the spatial domain appear as components perpendicular to that edge in the frequency domain. Thus horizontal lines in the spatial domain appear as vertical lines in the frequency domain and vice versa.

Because we were mostly interested in the geometric structure of an image, we only used the magnitude information retrieved from the Fourier transform. The phase information was discarded as it is only useful for reconstructing an image after processing it in the Fourier space. In order to get an overall idea of what the frequency domain of each letter looked like, we computed the average Fourier spectrum for each character class. This was done by averaging the magnitudes of the Fourier transform of each character in our training set.

However when trying to plot the Fourier spectrum in order to visually analyse it, the only thing visible was a white dot in the middle, while everything else was black. Due to the large dynamic range of the Fourier coefficients, the only value that stands out when plotting the Fourier spectrum, is the DC point which has the largest magnitude by far. In order to make smaller values more visible we took the logarithm of the magnitudes (plus one to avoid having the *log 0*). The results of the three Fourier spectrums are illustrated in *Figure 1*.



Average Fourier spectrum of "S" Average Fourier spectrum of "T" Average Fourier spectrum of "V"

Figure 1: Average Fourier spectrums of the 3 letters.

3. Feature Selection in the Transformed Space

After computing the Fourier spectrum of the three letters we then went to analyse it in order to extract information from the frequency domain about the shape of each letter. By using features selected from the frequency domain, we reduce the effect of variations in size and position in our test data as the Fourier transform is not be affected.

As shown in *Figure 1*, in the Fourier space of "T" there is a strong presence of horizontal and vertical lines passing through the centre. These lines originate from the two orthogonal lines that form the T shape in the spatial domain. The Fourier space of "V" is similar to the one in T. The two diagonal lines that form the V shape in the spatial domain are also represented as strong diagonal lines in the Fourier space. We can also notice that in both Fourier spaces, higher frequencies have low magnitudes. However, when looking at the Fourier space of "S", it is apparent that it is completely different than the previous two as it contains both high and low frequencies.

After analysing the average Fourier spectrum of each letter, we could now find spectral features from spectral regions that could be used to create a classifier that would classify future data. For our classifier to be accurate, our features had to be selected carefully in order to achieve the best separation between the classes. Furthermore, the number of features selected, had to be as small as possible in order to avoid analysing data in high dimensional spaces. Therefore, we decided to choose only two features for our classifier thus avoiding the Curse of Dimensionality. Firstly, we decided to use a rectangular region just above the centre as our first feature. By having a small offset from the central point we were able to avoid including the DC point in our feature. The reason for choosing this feature is that it includes the vertical lines present in the frequency spectrum of T as well as the high magnitude frequencies present in S. Thus both S and T will have high values whereas V will have a low value. As our second feature, we decided to choose a right-angled triangle near to the centre. This feature tries to include the high magnitudes of S that represent the curves in the shape of the character. In this feature, S will have a high value followed by V and then T. The position of the triangle was chosen in order to avoid having redundant data by including any lines of T, which is what the first feature will try to capture. The selected features are shown in *Figure 2*.

As a metric for our features we used the power of the selected regions shown in *Figure 2*. The power is simply the squared magnitude. By summing up the squared magnitudes in a region we can get a value of how much power is present and thus find out how strong are the frequencies in that region. However, the value that resulted summing all squared magnitudes was too large and became a computational bottleneck. To solve this we decided to rescale our training features using a process called feature standardisation. Feature standardisation rescales the features in our training set to have $\mu = 0$ and $\sigma = 1$, like a Standard Normal Distribution. This is done by firstly calculating the mean and the variance of each feature and then calculating the standard scores for all training features using the following formula: $z = \frac{x - \mu}{\sigma}$. By standardising our features we are also making sure that both features contribute equally to the classification.

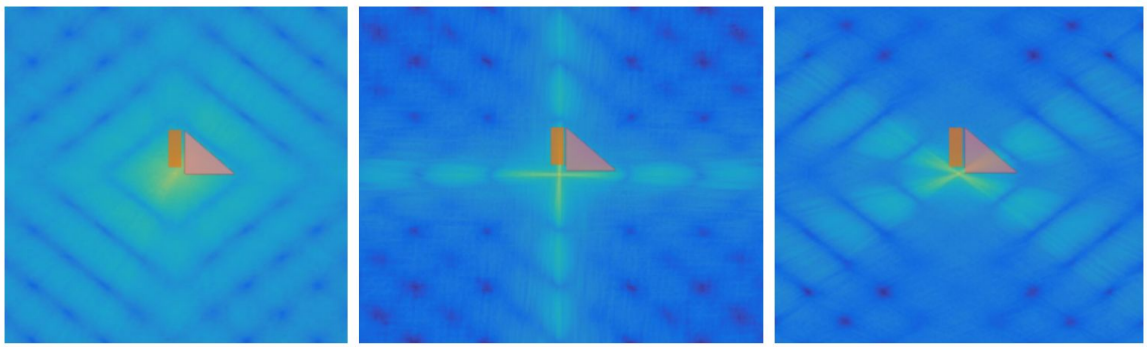


Figure 2: Feature selection (Feature 1: Rectangle, Feature 2: Triangle)

4. K- nearest neighbour classifier

The k-nearest neighbour (k-NN) classifier can be used in order to classify future data to the class of their k-nearest neighbours. The k-NN classifier works by taking a set of n points (train data), whose classes are known and then classifies future data (test data) in those classes. The assignment of classes of future data is done by examining the k-closest neighbours of a point, and then deciding its class based on what class the majority of its neighbours belong to. The choice of the parameter k in the k-NN classifier is highly training data dependent but in general higher k-values can suppress noise as more neighbours are considered[6]. Furthermore, in machine learning a good rule of thumb is to choose k as the $\sqrt{n}$ where n is the number of instances of samples in our training data[2][3]. For our classifier we experimented with different values of k but there was little variation when classifying future data and thus decided on using k = 5 as it is roughly the square root of our training data set. To create our classifier, we used the standardised features we extracted earlier from our training set, along with their corresponding classes.

Then we proceeded to visualise the decision boundaries of the k-NN classifier. Firstly a 2-dimensional grid of points was created, with the distance between each point being 0.001 so that the decision boundaries appear more defined (lower distance between points leads to "pixelated" boundaries). Then using the k-NN classifier, we assigned a class to each point in the grid and colored each point with a different color according to its class. Furthermore, we plotted the training data onto the plot with the decision boundaries in order to observe how the boundaries were formed as shown in *Figure 3*. Because of the choice of k=5 we can see that the presence of an outlier, like the red point in the middle of *Figure 3*, does not greatly affect the boundaries. Instead, if we opted for k=1 then the red boundary would be pushed to the left because only one neighbour is taken into consideration. Furthermore, from this figure we can deduce that the features chosen offer really good separation of our data. We can clearly see that S has high values for both features, as predicted earlier, while T and V have one feature with high value and another with low.

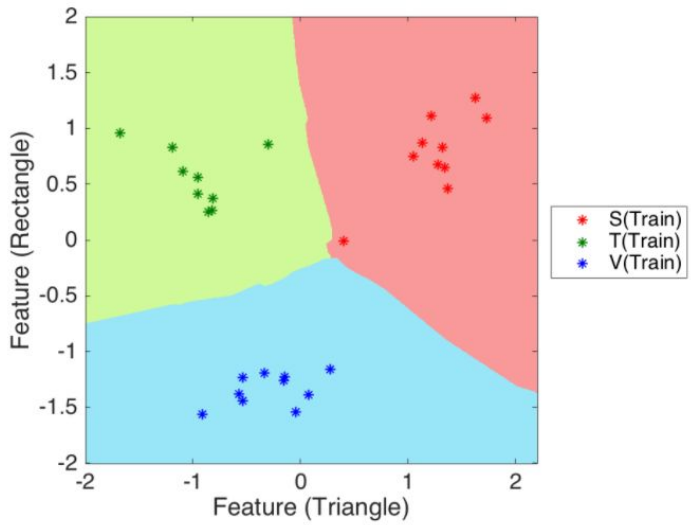


Figure 3: Decision Boundaries of K-Nearest Neighbour

5. Testing the k-NN Classifier

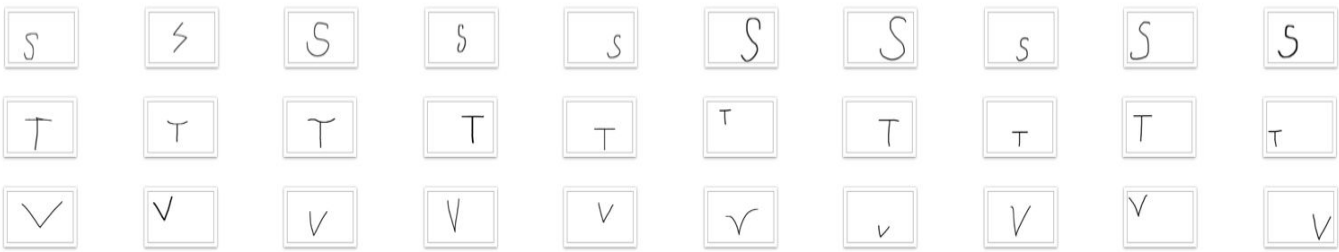


Figure 4 : Test data set

In order to test how well our classifier classifies data based on the chosen features, we produced ten images for each character. The new test data includes letters of various sizes, and positions. It captures letters that differ in many ways such as shape, position and size. Some characters appear to be written in a more calligraphic way or having a thicker line. Additionally, some others tend to be more edgy or more curvy. Generally we tried to include all kind of character variations in order to check the classifier's accuracy on data that vary a lot from each other. The test data are shown in *Figure 4* above.

The results of running the classifier on the test data are displayed in *Figure 5* below. It is clear that the classifier has correctly classified most of our test data. There are, however, some letters that were incorrectly classified. Recall

that the rectangular feature tries to capture the presence of horizontal lines whereas the triangle feature captures the presence of curves. We can see from the figure that two letters that should have been classified as an S were instead classified as a T. The two misclassified letters are the second and the last S's, from *Figure 4*. By viewing their Fourier spectrums we were able to detect why the misclassification occurred. Both of them contained strong horizontal lines and that is why they were not classified as a V. The first one has a more edgy appearance which is the main reason why it has been classified as a T. It is positioned to the far left side of the figure because of its lack of curves, which is what the triangle feature tries to capture. The second one though, is located towards the middle as it has a mixture of edges and curves.

In addition to the two S's mentioned above, there were two more misclassifications. The second T and the sixth V from *Figure 4* are both classified as S's. The V has a strong presence of curves, similar to the ones in an S, and that is why it is incorrectly classified as one. On the other hand T is classified as an S but is actually on the boundary of S and T. This is because the upper part of this T is a curved line whereas its lower part is a straight vertical line. However it seems like the curved top contributed more to the classification and that is the reason why it was barely classified as an S .

Even though the classifier did not correctly classify the letters mentioned above, it did manage to classify letters that were more calligraphic like the third T and eighth V. All in all, we were really satisfied by the performance of our k-NN classifier as it had an accuracy rate of about 86%.

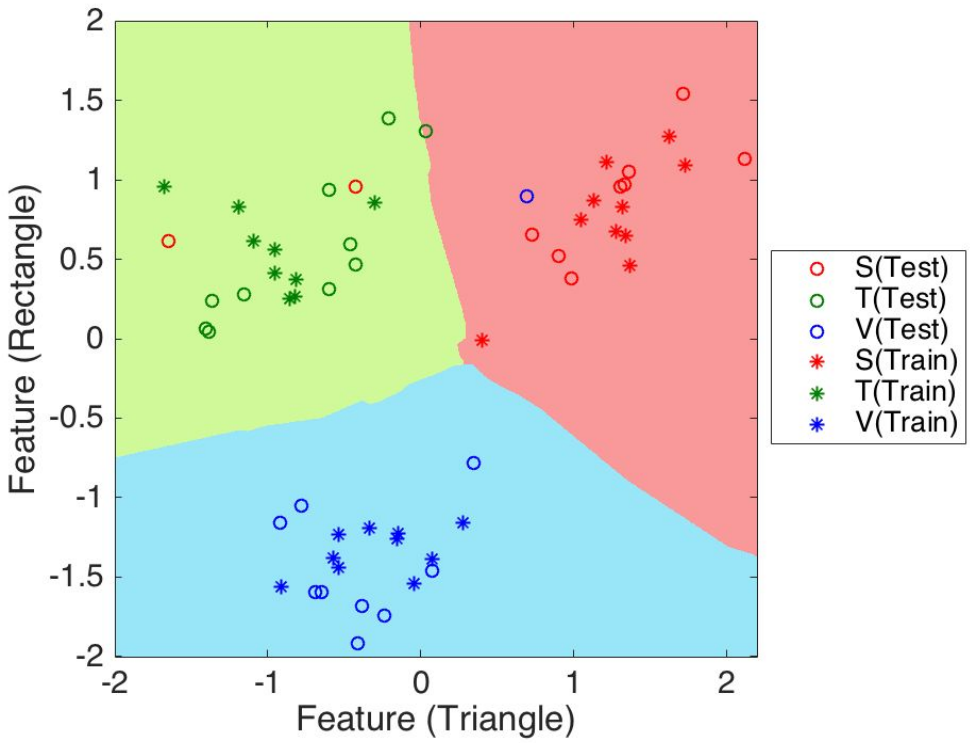


Figure 5: Plot with both train and test data

6. Classifying A and B characters

In this section we will use the k-NN classifier that was created earlier, in order to classify an A and a B and then reason about their classification.

First of all, we classified the two letters using our classifier; A was classified as a T and B was classified as an S. In order to fully understand the reason behind this classification, we decided to view the Fourier spectrum of both letters in order to observe which features were prominent in the Fourier domain, as shown in *Figure 6* . However, even after looking at the Fourier domain of A and B, we couldn't fully understand why the classifier recognised the two characters in that way.

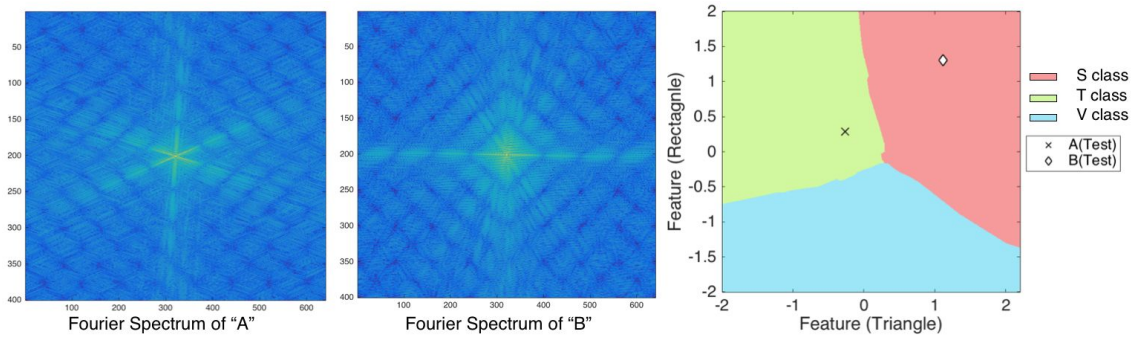


Figure 6 : Fourier Spectrums of A and B along with a plot showing the classification of each letter

To tackle this problem we decided to decompose the shape of these two letters in simpler shapes. We then proceeded to view the Fourier domain of all shapes in order to see which geometric characteristics contributed more to the classification of A as a T and B as an S. Starting with A, we splitted the test image into three different ones, each representing a part of the character. These images are shown in *Figure 7* along with their Fourier spectrums. The first and third images show the left and right diagonal lines of A and the second one represents its middle line. Then, for B, two images were created with the fourth one representing a vertical line and the last one, two semicircles.

By splitting A into simpler shapes and then computing the Fourier transform of each decomposed shape, we can clearly see why the Fourier spectrum of A shown above, is composed of three lines. The two diagonal lines in the Fourier domain represent the two diagonal lines in the spatial domain whereas the horizontal line in the spatial domain appears as a vertical line in the Fourier domain. Because A's Fourier spectrum includes strong vertical lines, representing the horizontal line that connects the two diagonal ones, our classifier classifies A as being a T. For B, the vertical line in the spatial domain is represented as a horizontal one in the Fourier domain whereas the two semicircles are represented in the Fourier domain as multiple lines passing through the centre. Because of the roundness of B's semicircles, B is classified as an S.

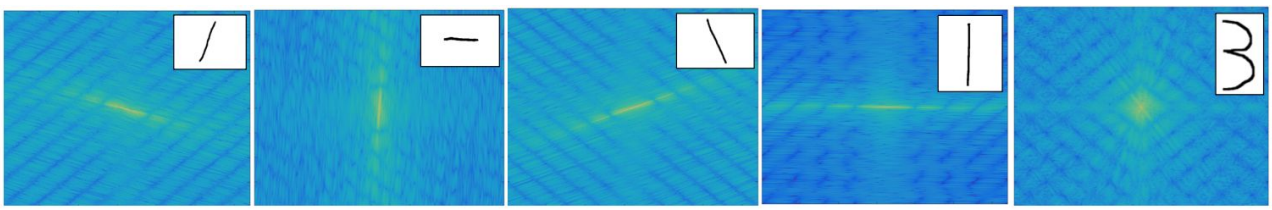


Figure 7: Fourier spectrums of the parts that compose A and B

7. Using Alternative Classifiers

In the previous sections of this report, we used the k-NN classifier to classify characters according to some spectral features. In this section we will examine the use of two alternative classifiers, namely the Nearest Centroid classifier (NCC) and the Maximum-Likelihood classifier (MLC).

Firstly, we will investigate the use of the Nearest Centroid classifier. The NCC, assigns classes to data point by calculating the Euclidean Distance between the point and the centroids of each class and then assigning the data

point to the class whose centroid has the minimum distance. The centroids were found by calculating the mean of each class in our training data. Using these centroids we could now build a NCC that would assign to any future data the class that each point belongs to. After creating this classifier, we loaded our test data in order to test the classifier's performance and accuracy. The speed of the NCC was far slower than the k-NN due to the time needed to compute the distance of all points to each centroid. However this classifier was slightly more accurate than k-NN as it only misclassified 3 letters, instead of 4 that the k-NN did. The NCC managed to correctly classify the T that is close to the the boundary of T and S. The decision boundaries of the NCC along with the test and train data are shown in *Figure 8*.

The second classifier that will investigate is the Maximum-Likelihood classifier. To create this classifier we model each class as being generated from a 2-D normal distribution with μ and Σ being the mean and covariance of the training data for each class respectively. From here, we can calculate the probability of each point belonging to a class by using the Probability Density Function (PDF) for Multivariate Gaussian which is given by the formula:

$$P(x|\mu_n, \Sigma_n) = \frac{1}{\sqrt{2\mu_n \pi}} e^{-\frac{1}{2}(x-\mu_n)^T \Sigma^{-1} (x-\mu_n)}$$

Assigning classes to future data involves calculating the PDF of a test point belonging to each one of the three distributions and then assigning the point to the class that has the highest PDF. As now we are dealing with a 2-D Gaussian-Distribution with covariances, the decision boundaries are not straight lines like in the NCC but appear to be bended. The decision boundaries of the three classes using MLC are shown in *Figure 8* along their respective 95% confidence level ellipses. The MLC had the best performance from all three classifiers. This is due to the fact that there is less amount of computation needed; we only need to compute three probabilities for each data point and then find the minimum. Accuracy-wise, the MLC performed better than the k-NN classifier making only 3 misclassifications, the same as in the NCC. Moreover, we can see that the MLC has more accurate decision-boundaries than the NCC and did a better job overall in classifying our test data as there is only one point near the boundaries compared to 3 in k-NN and the NCC. This was because the MLC takes into consideration the covariance of each class.

For our data sets, the MLC appeared to be the best performing classifier in both efficiency and accuracy. However there wasn't sufficient train and test data to make a certain conclusion as to which classifier performs better.

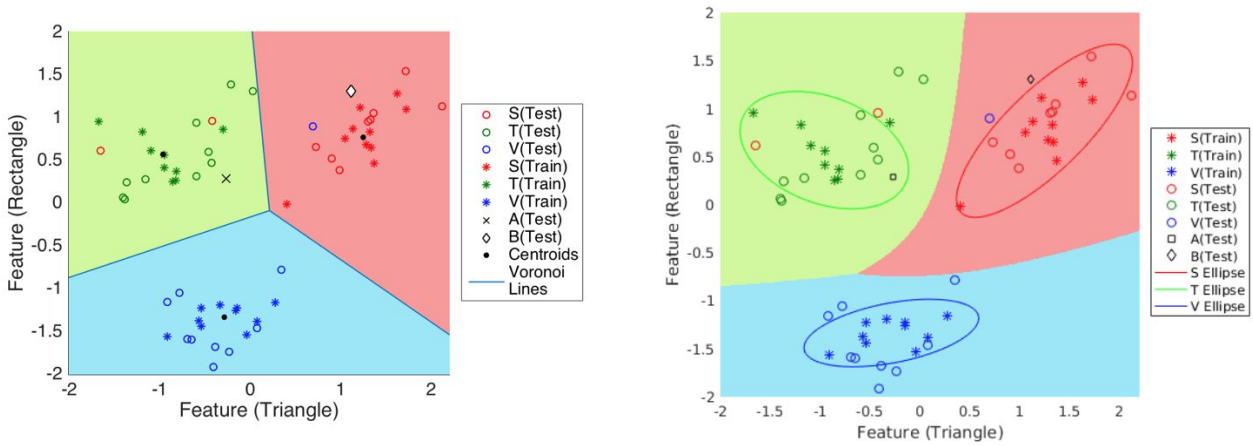


Figure 8: Decision boundaries using the Nearest Centroid Classifier (Left) and the Maximum-Likelihood Classifier (Right)

[1]. http://homepages.inf.ed.ac.uk/rbf/HIPR2/Fourier.htm
[2]. Pattern Classification, 2nd Edition,Richard O. Duda, Peter E. Hart, David G. Stork
[3]. https://www3.nd.edu/~steve/computing_with_data/17_Refining_kNN/refining_knn.html
[4]. http://sebastianraschka.com/Articles/2014_about_feature_scaling.html
[5]. http://www.visiondummy.com/2014/04/draw-error-ellipse-representing-covariance-matrix/
[6]. http://scikit-learn.org/stable/modules/neighbors.html