FAST FOURIER TRANSFORM (FFT) + K-NEAREST NEIGHBORS (KNN)

DISCRETE FOURIER TRANSFORM (DFT)

- In mathematics, DFT converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform (DTFT), which is a complex-valued function of frequency. The interval at which the DTFT is sampled is the reciprocal of the duration of the input sequence.
- The DFT is the most important discrete transform, used to perform Fourier analysis in many practical applications, such as on a time series.
- Since it deals with a finite amount of data, it can be implemented in computers by numerical algorithms or even dedicated hardware. These implementations usually employ efficient **Fast Fourier Transform (FFT)** algorithms.
- The discrete Fourier transform transforms a sequence of N complex numbers $\{\mathbf{x_n}\} := x_0, x_1, \ldots, x_{N-1}$ into another sequence of complex numbers, $\{\mathbf{X_k}\} := X_0, X_1, \ldots, X_{N-1}$ which is defined by

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn}$$
$$= \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N}kn\right) - i \cdot \sin\left(\frac{2\pi}{N}kn\right)\right]$$

- This operation is useful in many fields, but computing it directly from the definition is often too slow to be practical. An FFT rapidly computes such transformations.
- As a result, it manages to reduce the complexity of computing the DFT from O(N^2), which arises if one simply applies the definition of DFT, to O(N*log N), where N is the data size.
- **Cooley–Tukey Algorithm:** By far the most commonly used FFT is the Cooley–Tukey algorithm. This is a divide and conquer algorithm that recursively breaks down a DFT of any composite size N = N1* N2 into many smaller DFTs of sizes N1 and N2, along with O(N) multiplications by complex roots of unity.
- Suppose, we separated the Fourier Transform into even and odd indexed sub-sequences.

$$\begin{cases} n = 2r & \text{if } even \\ n = 2r+1 & \text{if } odd \end{cases}$$

where $r = 1, 2, ..., \frac{N}{2} - 1$

- After performing a bit of algebra, we end up with the summation of two terms. The advantage of this approach lies in the fact that the even and odd indexed sub-sequences can be computed concurrently.

$$x[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}}$$

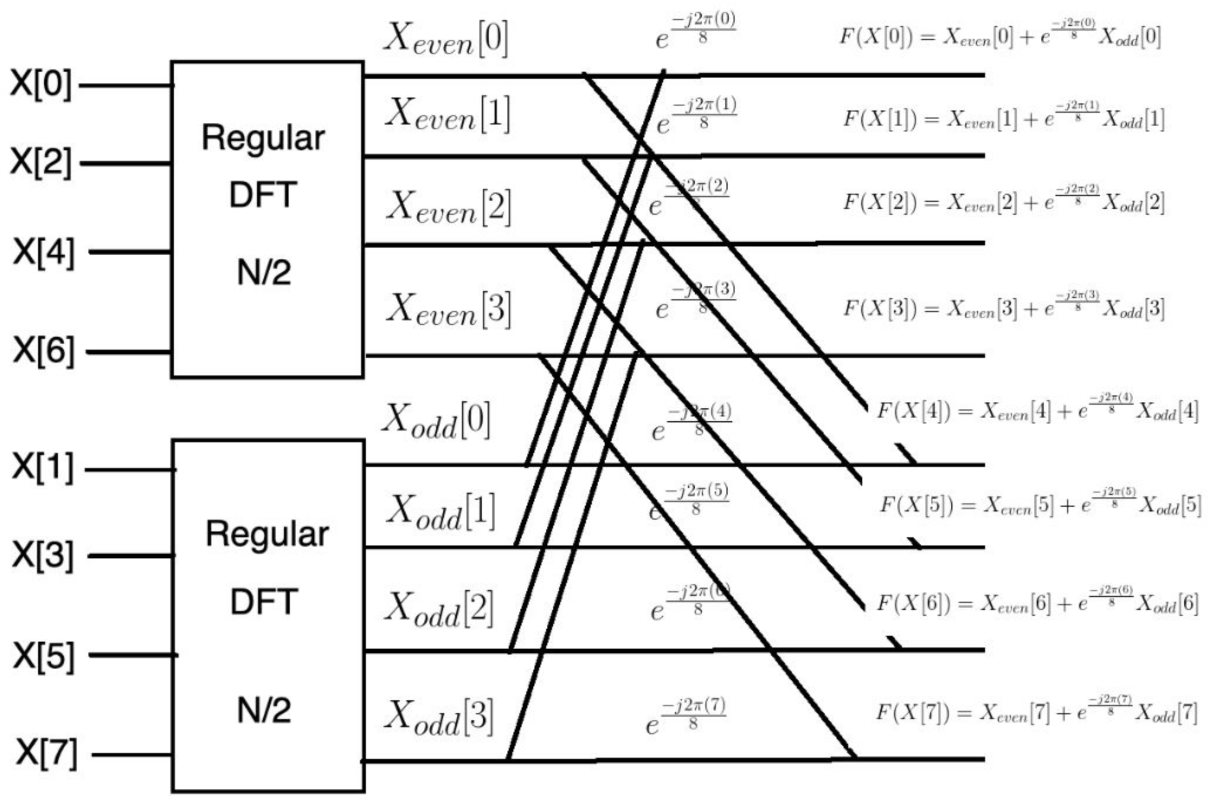
$$x[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] e^{\frac{-j2\pi k(2r)}{N}} \;+\; x[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] e^{\frac{-j2\pi k(2r+1)}{N}}$$

$$x[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] e^{\frac{-j2\pi k(2r)}{N}} \;+\; x[k] = e^{\frac{-j2\pi k}{N}} \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] e^{\frac{-j2\pi k(2r)}{N}}$$

$$x[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] e^{\frac{-j2\pi k(r)}{N/2}} \;+\; x[k] = e^{\frac{-j2\pi k}{N}} \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] e^{\frac{-j2\pi k(r)}{N/2}}$$

$$x[k] = x_{even}[k] \;+\; e^{\frac{-j2\pi k}{N}} x_{odd}[k]$$

- The below diagram illustrates how the above algorithm can be used to solve for when N = 8. We compute the Discrete Fourier Transform for the even and odd terms simultaneously. Then, we calculate x[k] using the formula from above.



X[0]
X[2]
X[4]
X[6]
X[1]
X[3]
X[5]
X[7]
Regular DFT N/2
Regular DFT N/2
$X_{even}[0]$
$X_{even}[1]$
$X_{even}[2]$
$X_{even}[3]$
$X_{odd}[0]$
$X_{odd}[1]$
$X_{odd}[2]$
$X_{odd}[3]$
$e^{\frac{-j2\pi(0)}{8}}$
$e^{\frac{-j2\pi(1)}{8}}$
$e^{\frac{-j2\pi(2)}{8}}$
$e^{\frac{-j2\pi(3)}{8}}$
$e^{\frac{-j2\pi(4)}{8}}$
$e^{\frac{-j2\pi(5)}{8}}$
$e^{\frac{-j2\pi(6)}{8}}$
$e^{\frac{-j2\pi(7)}{8}}$
$F(X[0]) = X_{even}[0] + e^{\frac{-j2\pi(0)}{8}} X_{odd}[0]$
$F(X[1]) = X_{even}[1] + e^{\frac{-j2\pi(1)}{8}} X_{odd}[1]$
$F(X[2]) = X_{even}[2] + e^{\frac{-j2\pi(2)}{8}} X_{odd}[2]$
$F(X[3]) = X_{even}[3] + e^{\frac{-j2\pi(3)}{8}} X_{odd}[3]$
$F(X[4]) = X_{even}[4] + e^{\frac{-j2\pi(4)}{8}} X_{odd}[4]$
$F(X[5]) = X_{even}[5] + e^{\frac{-j2\pi(5)}{8}} X_{odd}[5]$
$F(X[6]) = X_{even}[6] + e^{\frac{-j2\pi(6)}{8}} X_{odd}[6]$
$F(X[7]) = X_{even}[7] + e^{\frac{-j2\pi(7)}{8}} X_{odd}[7]$

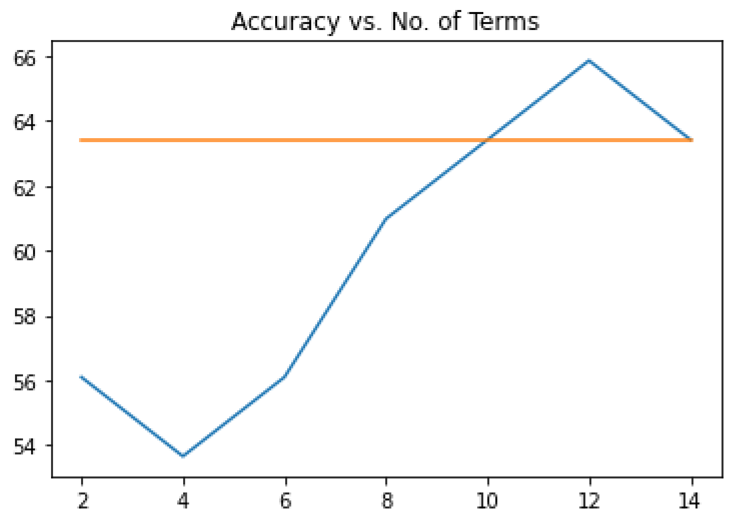
EXPERIMENTS

- A dataset of robot movements was used which is described by 6 time-series variables and the resulting consequences, to demonstrate multivariate time series classification. The dataset has the data for 93 movements of 15 time units. The dataset is available in the same folder as this report, by the name lp5.data.

- FFT was first applied on each time series and the first certain no. of terms were extracted. Let the no. of terms extracted be called k. Now, each term will have both a real and an imaginary part. So, in effect for each time series, we have 2*k parameters. Also, since each data point has 6 time series, we have 6*2*k parameters for each point.
- The number of dimensions was reduced using PCA, as we end up with a lot of dimensions. The no. of dimensions to reduce to was decided based on the explained variance plot.
- kNN was then used on the PCA-transformed data to get the final classification. The confusion matrix and the classification accuracy was calculated.
- The above steps were repeated for different values of k. Also, the above steps were done for the case where no Fourier Transform was applied on the data, to test whether a Fourier Transform helps in improving the classification, when compared to direct application of kNN.
- The resulting accuracies were plotted on a graph.

RESULTS

- The Accuracy vs No. of Fourier terms extracted (k) plot obtained is as shown below. The blue line represents the accuracies when Fourier Transform was applied, while the orange line represents the case when FFT was not applied.



- The graph shows a clear trend: The accuracy improves as more no. of Fourier terms are considered. This is intuitive because as more no. of terms are used, less amount of information is lost about the time series.

- The graph also shows that it is possible to achieve a higher accuracy by applying the Fourier Transform as opposed to not applying the transform. The only constraint is that a sufficient number of fourier terms should be used. For example, the above graph shows that at least 10 Fourier terms should be used to get a better accuracy.

CONCLUSION

- This report has given a theoretical introduction to Discrete Fourier Transform, and an efficient algorithm to implement the transform called the Fast Fourier Transform.
- The report has demonstrated a practical application of time series classification on a dataset of robot movements. The results demonstrate the power of FFT + kNN.
- The results have also demonstrated how accuracy improves when more Fourier terms are used.
- They also demonstrate how a Fourier Transform helps to improve the classification accuracy of a multivariate time series, as opposed to not using the transform.