International Journal of Data Science and Analytics (2023) 16:435–440
https://doi.org/10.1007/s41060-023-00409-5

REGULAR PAPER

Applications of the discrete-time Fourier transform to data analysis

Dayne Sorvisto[1]

Received: 26 September 2022 / Accepted: 15 June 2023 / Published online: 8 July 2023

Abstract
We define a discrete analogue of the characteristic function for discrete random variable and develop numerical procedures for computing the discrete characteristic function for several well-known discrete random variables. We rigorously define what is meant by the Fourier transform of a probability mass function and also show how to reverse the process to recover the probability mass function of a discrete random variable, given a procedure for computing the characteristic function. Unlike previous work on the subject, our approach is novel in that we are not computing closed-form solutions of a continuous random variable but are focused on applying the methods to real-world data sets.

Keywords Estimating the empirical distribution for discrete random variables using discrete-time Fourier Transforms · Fourier transform · Discrete Fourier transform · Characteristic function

1 Introduction

This paper introduces a novel methodology for computing the empirical distribution of real-world data using discrete-time Fourier transforms. We rigorously define what is meant by the Fourier transform of a probability distribution and compute the characteristic functions for an arbitrary random variable, encoded as a vector. Unlike previous work on this subject, our approach is novel in we do not explicitly compute a closed-form solution from a continuous probability distribution but instead have adapted to work in the discrete case for a probability mass function. We also explicitly show how to reverse the process through the inverse Fourier transform and define a high-order function that accepts, as an argument a procedure for computing the characteristic function and returns an estimate to the empirical probability distribution.

1.1 Motivation

The motivation for writing this paper was twofold. Fourier analysis is a very well-studied concept, and its importance in fields like engineering, applied mathematics and statistics cannot be overstated. However, outside of signal processing applications and pure statistical applications, the use of Fourier transform as a tool to represent real-world data sets is not very well defined. The idea of using characteristic functions as a fundamental tool for data analysis, on real data sets where the distribution is unknown or can only be approximated is the goal of this paper. Secondly, it is the hope that a tool like Fourier transforms having more wide-spread adoption in the field of data science could make some analysis more rigorous as the characteristic function provides a kind of mapping between real-world data sets and the world of mathematics.

1.2 Mathematical foundations

We compute characteristic functions from probability mass functions of discrete data sets. We build on top of these techniques to introduce machinery for estimating the characteristic function where the probability distribution is unknown using Bayesian optimization and gradient descent, an approach that has far reaching use cases where the where standard maximum likelihood estimation techniques fail.

Throughout this work, it is our goal to connect theory with practice, to show how algebraic invariants can be used for statistical inference in a practical data analysis setting but also hint at how a more rigorous form of data analysis can be practiced by using characteristic functions to transform structured and unstructured data sets into mathematical objects that can be manipulated by way of algebra. We will start by laying the groundwork for defining characteristic functions in the next section.

✉ Dayne Sorvisto
daynesorvisto@gmail.com

[1] Department of Mathematics, University of Calgary, Calgary, AB, Canada

1.3 Discrete-time Fourier transform

The Fourier transform is ubiquitous in engineering and mathematics to decompose a periodic signal into sines and cosines. In a probabilistic context, the Fourier transform can be applied to give an equivalent view of a probability distribution, by perfectly encoding information about the distribution in the representation of a characteristic function of a single complex variable in the case of a univariate distribution.

We examine the mathematical foundations of this and show through practical example how we can build a practical methodology for applying algebraic invariants of probability distributions to data analysis,

1.4 Probabilistic treatment of Fourier transforms

The reader is likely already familiar with Fourier transforms as Fourier transform has applications across a wide-range of engineering and scientific disciplines. As has been stated previously, in the context of a probability distribution or probability mass function, which is not a periodic function, it is treated slightly differently. We are interested in statistical estimation of the probability mass function instead of a continuous density function so we also need to use a discrete version of the Fourier transform. With these considerations in mind, the discrete-time Fourier transform (DTFT) is chosen as the fundamental tool for algebraic data analysis.

The DTFT is used to analyze samples of a continuous functions [2]. The DTFT also has the useful property that under mild theoretical assumptions the original continuous function can be re-constructed from the DTFT and so is ideal for preserving information in a non-destructive way. We give a precise definition of the DTFT:

Given an input dataset $x(n)$ with N samples the DTFT is defined as

Definition 1

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad k = 0, \ldots, N-1$$

N-th primitive root of unity $\mathrm{i} = \sqrt{-1}\, e^{\mathrm{i}x} = \cos(x) + \mathrm{i}\sin(x)$

While it is unusual to see complex numbers in data analysis, we will explore the usefulness of this definition by applying it to represent probability distributions and their discrete analogues as algebraic objects called characteristic functions.

1.5 Characteristic function of a probability distribution

In terms of Fourier transforms, we can define the characteristic function.

However, in probability theory a more convenient definition is to define the characteristic function in terms of expectation. We state the definition as follows.

Definition 2 The characteristic function $\phi(t)$ of a random variable X is defined as expectation value of the random variable e^{itX} and is given by $\phi(t) = E\left[e^{itX}\right] = \int_\Omega e^{itx} d\mu$, where μ is the distribution of X. We can recover the original distribution via the integral equation:

$$p(x) = \frac{1}{2\pi} \int_0^{2\pi} e^{-ixt} \phi_x(t) dt$$

In can generalize this to the discrete case, we have the following definition for the characteristic function of a discrete random variable.

Definition 3

$$\phi_x(t) = E\left[e^{itX}\right] = \sum_k e^{itx_k} p\,(x_k) \tag{1.1}$$

And by similar logic [1], the discrete inverse transform implies we can re-cover the probability distribution via the inverse transform by evaluating ϕ along a sequence of equally spaced, discrete intervals X_k that exist within the range $[0, 2\pi]$:

$$p\,(x_k) = \frac{1}{N} \sum_{k=0}^{N-1} \phi(X_k) \cdot e^{-iX_k k} \tag{1.2}$$

Now that we have a formal definition for the characteristic function of a discrete random variable, we can go on to compute it and give explicit examples. Our code for computing characteristic functions using a naive approach is provided in Appendix A. We also provide code to re-construct the underlying probability mass function of a discrete random variable, given its characteristic function evaluated at select data points that can be thought of as algebraic invariants of the function.

We provide two novel optimizations of this naive approach to estimating the characteristic function using Bayesian optimization with certain mathematical conditions and a more flexible approach that considers discrete Fourier transforms within the context of gradient descent. We implement this last approach as a neural network architecture.

1.5.1 Existence of a characteristic function

Before we can explicitly evaluate equation 1.1 and 1.2 to compute characteristic functions and their underlying probability mass function by numerical simulation, we need to prove the existence of the characteristic function. We can

think of this proof of existence as a kind of fundamental theorem in algebraic data analysis and we present it as follows.

Theorem 1 *A characteristic function of a continuous random variable X exists.*

Proof By definition of a probability distribution, $\left|e^{itX}\right| = 1$. It follows by the law of total expectation that

$$\left|E\left[e^{itX}\right]\right| \leq E\left[\left|e^{itX}\right|\right] = 1$$

Hence, $E\left[e^{itX}\right] < \infty$ for any random variable X. Thus, characteristic function $\phi(t)$ exists. □

2 Methods

In this section, we present the mathematical building blocks for the foundations of algebraic data analysis, expanding on the idea of a characteristic function and the definitions we agreed upon for the definition of a characteristic function of. discrete random variable.

The novel insight that enabled us to apply the theory of characteristic functions to data analysis was the following observation: For a real-world data set where the distribution is unknown, we can feed the data through an algorithm to learn the empirical probability density function, pass these weights through a Fourier transform layer and then use an inverse transform layer to re-construct the original data. By comparing the original data to the re-constructed data, we can construct a loss signal that can inform the weights on the learned empirical distribution, say via gradient descent. This can ultimately be implemented in an autoencoder with Fourier transform layers (who chose to implement this in PyTorch), extending on the idea that the Fourier transform can be represented as a neural network.

2.1 Experimental design

In order to illustrate this connection between the mathematical world of characteristic functions and data analysis as well as test the feasibility of our approach in concrete terms, we designed a classic Bernoulli trial of a fair coin (probability of heads is 0.5) to illustrate how we can estimate the characteristic function using some algebraic invariants and used the estimated characteristic function to re-construct the probability mass function.

We wrote code to re-construct the hyper-parameter p (probability of heads) for a fair coin toss over N independent trials for several values of N. We recorded our results including sample size for $N = 10$, 100, 1000 and 10,000 tosses. We recovered the probability of heads in each case using the inverse Fourier transform. The outcome of each toss, heads or tails were encoded as 'H' and 'T,' respectively.

The open-source code for our experiment is bin Appendix A. In particular, the pseudo-code for our heuristic algorithm, which relies on probability density estimation, is stated below and also given in Appendix A as a Python function.

Algorithm 1 The pseudo-code for numerical approximation of characteristic function estimation

1: Initiate the algorithm with vector X and n bins as input
2: $l \leftarrow$ length of X
3: Group X into n bins and compute the histogram call this x_k
4: Initial estimate of the empirical distribution $p(x_k)$ of X given by $softmax(\hat{X})$
5: $\phi(t) == \sum_k e^{i\pi wk} p(x_k)$
6: Emit ϕ as characteristic function of X

We note that the above algorithm is sufficient for estimating broad bounds on parameters that can then be refined by Bayesian search, but also depends on the initial choice of probability density estimate.

We describe such an algorithm in the next section. Although we do not have a formal proof convergence for this algorithm, we note for the continuous case and under specific mathematical conditions on the random variable, in particular for stable random variables this methodology approximates the empirical distribution (Figs. 1, 2).

2.2 Reversing the process

Our goal in this section is to create a novel algorithm for dimensionality reduction from the ground up. We borrow from several modern elements of data analysis. We note that out method is particularly interesting and novel because unlike $T - SNE$ which uses topological invariants, we use algebraic invariants. The field of algebraic geometry is a gigantic body of research and having the ability to view the data set as a triangular mesh of cloud data points, where those data points encode statistical and algebraic information about the underlying empirical distribution can have profound impact on data analysis.

We describe our dimensionality reduction algorithm in the next steps.

k < N: return 1/N * np.sum(char(2*m*math.pi/N) * np.exp((-1j)*2*math.pi*k*m/N) for m in np.arange(N)).real

Since i is confusing as the characteristic function is a function of a single complex variable, we use m for the index.

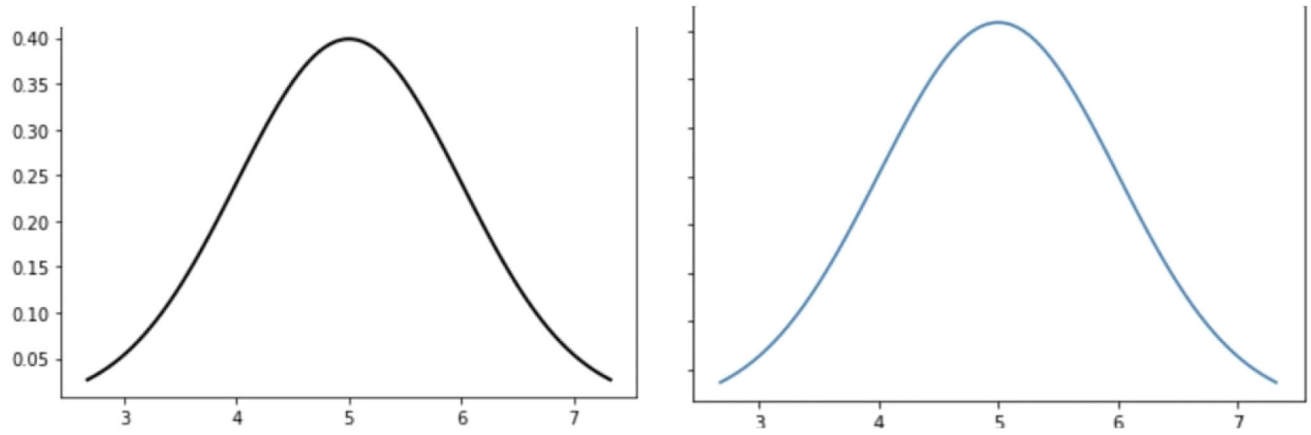

Fig. 1 Right probability density estimation is re-constructed from simulation

Algorithm 2 The pseudo-code for recovering the discrete probability mass function (PMF) given a characteristic function f

1: Input a procedure for computing the characteristic function $f(x)$, k and N where k < N
2: Define the integer function $w(m) \leftarrow 2 * \pi * \frac{m}{n}$ for convenience
3: Define an invariant vector v to be f evaluated at distinct points in the compact set $[0, 2\pi]$, formally, $f(w(m))$ for $0 <= m < N$ (Note f(0) = 1)
4: Compute another vector $u(k)$ defined by $e^{-i*w(m)*k}$ for $0 <= m < N$ these are points on the complex unit circle
5: Set $p(k) \leftarrow \sum_i v \cdot u$ where sum is taken over the ith component of the resultant dot product
6: Repeat above steps to compute $p(k)$ for each k < N

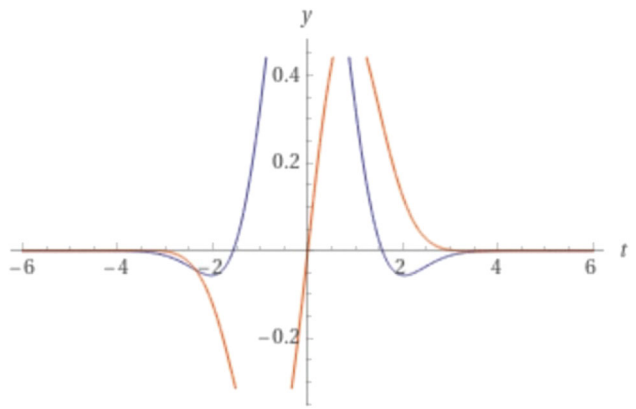

Fig. 2 Visualization of computed characteristic function for a standard normal

3 Results

3.1 Algorithm 1 evaluation

We computed the characteristic function and reported the mean squared error (MSE) for a normally distributed random variable as this was the simplest random variable we could think to compute.

Our results are summarized in this table for varying sample sizes. Note that MSE in this case was computed using the following closed-form solution for comparison:

$$e^{it\mu - \frac{1}{2}\sigma^2 t^2}.$$

We used this well-known closed-form solution to generate the test data to compute our MSE stated in the following table for several different sample sizes. It is worth noting that Python can natively handle complex numbers, so no special libraries were required.

Sample size (N)	MSE	Sigma	Mu
10	0.0066	1	0
100	0.0140	1	0
1000	0.0150	1	0

3.2 Algorithm 2 evaluation

We put our algorithm to the test by designing a simple numerical experiment to see if we can recover the probability distribution for a discrete random variable. The simplest example we could think of that had a practical interpretation was a fair coin toss.

We simulated a Bernoulli random variable with probability of heads being 0.5 for varying sample sizes and were able to recover the probability of heads by first computing the characteristic function of the random variable using the algorithm presented in the previous section. Our result is given in the following table. The code for reproducing the simulation is in Appendix A.

Sample size (N)	Estimated Pr[H]	Estimated Pr[T]
10	0.2000	0.8000
100	0.5400	0.4600
1000	0.4930	0.5070
10,000	0.4947	0.5050

4 Discussion

We presented a heuristic framework for practical data analysis that builds on the mathematical foundations of several different areas including Fourier transforms and probability density estimation.

We rigorously defined the characteristic function of a probability distribution for discrete random variables and created procedures for computing this function when no closed-form solution is known or the mathematics is too complex to work out. We also explicitly defined an algorithm to reverse the process and evaluated the results.

Specifically, we were able to evaluate algorithm 1 by computing the MSE for several sample sizes. For $N = 1000$, the MSE was 0.015 where the MSE was computed over $[0, 2*\pi]$ at uniform intervals.

In the case of algorithm 2, we were able to recover the probability of heads for a fair coin toss experiment and the results were reasonable. The true probability of heads was 0.5, but our algorithm was able to estimate this at varying number of trials. For $N = 10{,}000$, we estimated $Pr[Heads]$ to be 0.5050 reported to 4 significant figures. This is a mean absolute error of 0.0050. We noted less accuracy for smaller values of N which indicates our algorithm's accuracy depends not only on a reasonable estimate for the empirical distribution but also on the number of data points and the requirement that those data points available.

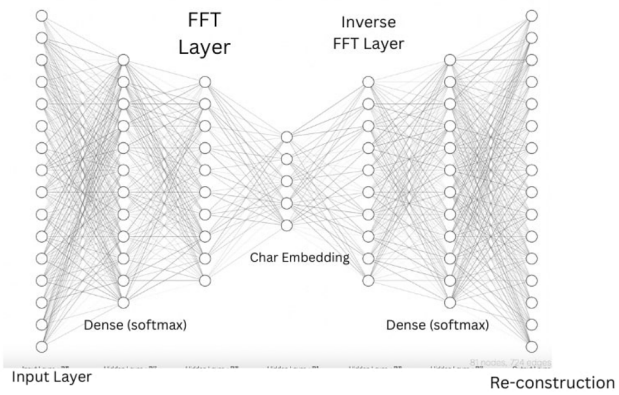

Fig. 3 Architectural diagram of FFT autoencoder inspired by LiNet-5

4.1 Future work

The Fourier transform is very important in engineering and not frequently used in data science or machine learning. Instead, convolutional neural networks are more popular. In the future, we would like to experiment with neural network architectures that use Fourier transform layers instead of convolutional nets and build a network architecture inspired by LiNet-5, one of the first convolutions neural networks that uses Fourier layers instead of convolutions (Fig. 3).

Appendix A: Library for simulating characteristic function

```
import random

#set random seed
random.seed(234)

#simulate 10,000 trials
N = 10000

def toss_simulation():
  """
  Simulate coin toss for a fair coin
  """
  return random.choice(['H', 'T'])

# training data size = N
vector = np.array([toss_simulation() \
for _ in np.arange(N)])

def compute_char(vector):
  """
  Higher-order function
  to generate discrete
  characteristic
  """
  def char(w):
      """
      Compute characteristic function.
      """
      pmf = estimate_pmf(vector)
      l = len(pmf)
      weights = pmf[:l]
      phi = [np.exp((1j)*(math.pi)*w*k)
      for\
      k in range(l)]
      return np.dot(phi, pmf[:l])
  return char
```

Author Contributions This manuscript and all contents in its entirety were created by the author Dayne Sorvisto.

Funding Not applicable.

Declarations

Conflict of interest I hereby confirm there are no conflicts of interest associated with this manuscript.

References

1. Lukacs, E.: Characteristic Functions. Griffin, London (1970)
2. Pinsky, M.: Introduction to Fourier Analysis and Wavelets. Brooks/Cole. ISBN 978-0-534-37660-4 (2002)