The DFT and Sparsity

EE 20 Spring 2014
University of California, Berkeley

1 Introduction

In this note, we will look at the Discrete Fourier Transform (DFT) from a different angle than in lecture. After developing the DFT as a tool for spectral analysis, we will then introduce the concept of sparsity and how it can be used to reduce the sample complexity of acquiring a signal. Familiarity with finite-dimensional linear algebra as taught in Math 54 is assumed to be known.

2 Vector Spaces and the DFT

In this section, we briefly review the concepts necessary to view the DFT as a *change of basis*, and then develop the DFT via *projections*.

2.1 Review

Recall from Math 54 that a *basis* for a finite-dimensional vector space V is a set of vectors that *spans* V, i.e. any vector that lives in V can be written as a linear combination of the basis vectors (we will refer to this as a *basis expansion*). Keep in mind that bases are not unique! In other words, if $\mathcal{B} = \{v_1, \ldots, v_N\}$ is a basis for V where $\dim(V) = N < \infty$, for every $u \in V$, there exist scalars $\alpha_1, \ldots, \alpha_N$ such that

$$u = \sum_{k=1}^{N} \alpha_k v_k.$$

For each $k \in \{1, \ldots, N\}$, the corresponding weight in the linear combination α_k is found by *projecting* the vector u onto the basis vector v_k. That is,

$$\alpha_k = \langle u, v_k \rangle,$$

where $\langle \cdot, \cdot \rangle$ is the inner product defined in V. Since bases are not unique, the above holds for any valid choice of $\mathcal{B}$. In the finite dimensional case, recall that a "valid choice" of $\mathcal{B}$ is any set of N linearly independent vectors. Recall a set of vectors $v_1, \ldots, v_N$ is *linearly independent* if none of the vectors can be represented as linear combinations of the others. Another way of putting this is

to say that the set $v_1, \ldots, v_N$ is linearly independent if, for scalars $\beta_1, \ldots, \beta_N$, the only way to satisfy

$$\sum_{k=1}^{N} \beta_k v_k = 0$$

is to set $\beta_k = 0$ for each $k \in \{1, \ldots, N\}$. An example of a set of vectors that is always linearly independent is a set of *orthogonal* vectors (as an exercise, verify this). Recall that two vectors are orthogonal if their inner product is zero. The set $v_1, \ldots, v_N$ is then orthogonal if the following is true:

$$\langle v_j, v_k \rangle = \|v_j\|^2 \delta[j-k] = \|v_k\|^2 \delta[j-k]$$

$\|v_j\|$ denotes the *norm* of the vector v_j, which can be thought of as the "length" of v_j. In many vector spaces we work in, the norm is *induced* by the inner product, i.e. $\|v_j\| = \sqrt{\langle v_j, v_j \rangle}$. Similarly, a set of vectors is said to be *orthonormal* if it is orthogonal and each vector has unit norm, i.e.

$$\langle v_j, v_k \rangle = \delta[j-k].$$

2.2 Signals as Vectors

You may have noticed that we have not explicitly defined what a vector is. In general, given a vector space V, *vectors* are all the elements contained in V. Since V is a vector space, these elements must satisfy the properties of a vector space:

1. The sum of any two vectors in V must also be a vector in V.
2. Every scalar multiple of a vector in V must also be a vector in V.

For a complete definition of a vector space, refer to Wikipedia. [1] With these properties in mind, we are now ready to look at signals as vectors. In Math 54, you often looked at vectors as lists of numbers represented by "column vectors". In EE 20, we view lists of numbers as discrete time signals, therefore we should be able to define a vector space of signals. Consider the vector space $V = \mathbb{R}^N$, which is defined as all length-N lists of real numbers. In EE 20 terminology, this means $\mathbb{R}^N$ consists of all real-valued discrete time signals of length N given by $x[n] = \begin{bmatrix} x[0] & \ldots & x[N-1] \end{bmatrix}^\intercal$, where $\intercal$ denotes the matrix transpose operation (we will also use the notation $\{x[n]\}_{n=0}^{N-1}$ as a more compact representation of finite length signals). An alternative definition of $\mathbb{R}^N$ would be all *periodic* real-valued discrete time signals with period N, since any periodic signal is completely determined by one period.

Let's verify that $\mathbb{R}^N$ is a vector space, according to the properties given above.

[1] `http://en.wikipedia.org/wiki/Vector_space#Definition`

Consider two arbitrary length-N real-valued discrete time signals $x_1[n] = \begin{bmatrix} x_1[0] & \dots & x_1[N-1] \end{bmatrix}^\intercal$ and $x[n] = \begin{bmatrix} x_2[0] & \dots & x_2[N-1] \end{bmatrix}^\intercal$. Their sum is then given by

$$\begin{aligned} x_1[n] + x_2[n] &= \begin{bmatrix} x_1[0] \\ \vdots \\ x_1[N-1] \end{bmatrix} + \begin{bmatrix} x_2[0] \\ \vdots \\ x_2[N-1] \end{bmatrix} \\ &= \begin{bmatrix} x_1[0] + x_2[0] \\ \vdots \\ x_1[0] + x_2[N-1] \end{bmatrix}. \end{aligned}$$

Since $x_1[n]$ and $x_2[n]$ are real valued, their sum is real valued (a sum of real numbers is a real number), and thus the first property is satisfied. Recall that a scalar in $\mathbb{R}^N$ is any $\alpha \in \mathbb{R}$. Then for any $x[n] \in \mathbb{R}^N$ and $\alpha \in \mathbb{R}$, the signal $\alpha x[n]$ is given by

$$\begin{aligned} \alpha x[n] &= \alpha \begin{bmatrix} x[0] \\ \vdots \\ x[N-1] \end{bmatrix} \\ &= \begin{bmatrix} \alpha x[0] \\ \vdots \\ \alpha x[N-1] \end{bmatrix}. \end{aligned}$$

Since $x[n]$ is real valued and α is a real number, the signal $\alpha x[n]$ is real valued (a product of real numbers is a real number), and thus the second property is satisfied, and we are convinced that $\mathbb{R}^N$ is indeed a vector space.

2.2.1 Exercise:

Another vector space of interest is $V = \mathbb{C}^N$, which is defined as all length-N lists of complex numbers. Recall that scalars in $\mathbb{C}^N$ are given by any $\alpha \in \mathbb{C}$. Verify that $\mathbb{C}^N$ is a vector space, according to the properties given above.

2.2.2 Exercise:

A very practical vector space in signal processing is $\ell_2(\mathbb{Z})$, which is defined as all discrete time signals with *finite energy*. In other words,

$$\ell_2(\mathbb{Z}) = \{x[n] : \|x[n]\|_2^2 = \langle x[n], x[n] \rangle = \sum_{n=-\infty}^{\infty} |x[n]|^2 < \infty\},$$

where $\|x[n]\|_2$ is the norm defined in $\ell_2(\mathbb{Z})$. Note that signals in $\ell_2(\mathbb{Z})$ are not of fixed length N! In fact, the dimension of $\ell_2(\mathbb{Z})$ is *countably infinite*. We won't rigorously look at infinite dimensional vector spaces in this course, but you can still verify that $\ell_2(\mathbb{Z})$ is a vector space, according to the properties given above (hint: triangle inequality).

2.3 The DFT as a Change of Basis

If you're convinced that all length-N complex-valued discrete time signals form the vector space $\mathbb{C}^N$, let's look at basis sets for this space. The canonical basis is given by

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \right\},$$

which we can write in signal processing terms as $\{\delta[n-k]\}_{k=0}^{N-1}$. Recall the inner product in $\mathbb{C}^N$ is

$$\langle x[n], y[n] \rangle = \sum_{n=0}^{N-1} x[n] y^\star[n],$$

where $\star$ denotes the complex conjugate. We can then write the basis expansion (as defined earlier) of any $x[n] \in \mathbb{C}^N$ with respect to the canonical basis by finding the weights α_k and then plugging into the sum:

$$\begin{aligned} \alpha_k &= \langle x[n], \delta[n-k] \rangle \\ &= \sum_{n=0}^{N-1} x[n]\delta[n-k] \\ &= x[k] \\ \Rightarrow x[n] &= \sum_{k=0}^{N-1} \alpha_k \delta[n-k] \\ &= \sum_{k=0}^{N-1} x[k]\delta[n-k] \end{aligned}$$

Unfortunately, this representation is not very interesting, and does not tell us anything about the signal $x[n]$ that we cannot see by simply looking at its N values. In signal processing, we are often interested in finding alternate representations of signals that are more meaningful. This is the idea behind the DFT: can we find a basis in which expansions of signals tell us about "frequency content"?

2.3.1 Exercise:

Let $\phi_k[n] = \frac{1}{\sqrt{N}} e^{i\frac{2\pi kn}{N}} = \frac{1}{\sqrt{N}} \begin{bmatrix} e^{i\frac{2\pi k \cdot 0}{N}} & e^{i\frac{2\pi k \cdot 1}{N}} & \cdots & e^{i\frac{2\pi k \cdot (N-1)}{N}} \end{bmatrix}^\intercal$. Show that the set $\{\phi_k[n]\}_{k=0}^{N-1}$ is an orthogonal basis for $\mathbb{C}^N$. Is it orthonormal?

2.3.2 Exercise:

Consider the basis expansion of an arbitrary $x[n] \in \mathbb{C}^N$ with respect to the basis $\{\phi_k[n]\}_{k=0}^{N-1}$, with $\phi_k[n]$ as defined above. Give the expression for the weights α_k, then plug them into the basis expansion formula. What is the relationship between this basis expansion and the DFT? What's different?

3 Sparse Signal Processing

In many signal processing applications, we are interested in *sparse representations* of signals. We say a signal is sparse if there exists an alternate representation of the signal in which most of its samples are zero. For now, let us define a signal $x[n] \in \mathbb{C}^N$ to be *K-sparse* if only K of its DFT coefficients are nonzero, but keep in mind there are many transforms other than the Fourier Transform which a signal may have a sparse representation in. [2] In general, we are interested in sparse representations of signals because they give us the information contained in a signal in the most compact, efficient format, which can help speed up computations and cut down on storage space.

3.1 Fourier Perspective on Sparsity

The following exercises were adapted from Professor Kannan Ramchandran's EE 225A HW 3, Spring 2014.

Consider a K-sparse signal $x[n] \in \mathbb{C}^N$ which exists as an array in your MATLAB workspace, and suppose you want to send a copy of $x[n]$ to your friend who lives very far away. This is easy – save it as a `.mat` file, attach it to an email, and send it off. However, what if $N = 50 \cdot 10^6$ and each sample is represented by 8 bits (1 byte)? $x[n]$ would then take up 50 MB of space, which is two times the size of Gmail's attachment limit! [3] Suppose also you have a terrible computer which won't let you compress things to `.zip` files. What would you do? Now suppose you observe that only the first $K << N$ DFT coefficients are nonzero, i.e. $X[k] = 0 \; \forall k \geq K$, where

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i\frac{2\pi kn}{N}}, \; k = 0, \ldots, N-1$$

[2] As we saw earlier, the Fourier Transform is just a change of basis from the canonical basis to the Fourier basis (the complex exponentials). Since bases are not unique, there are many other bases in which a signal may look sparse. The class of transforms which perform a change of basis are called *orthogonal transforms* (can you guess why?) and are ubiquitous in engineering.

[3] Fun fact: have you ever used the phrase "find attached" in an email, but forgotten to actually attach a file? Try it in Gmail and you'll be pleasantly surprised by a popup reminding you to attach the file! This feature was implemented by EE-20-lab-writer and former head TA Jon Kotkter! Thanks, Jon!

is the DFT of $x[n]$. We've now seen the DFT as nothing more than a change of basis – this means *all of the information in* $x[n]$ *is present in* $X[k]$! As a result, instead of sending all $N = 50 \cdot 10^6$ time domain samples of $x[n]$ to your friend, you could simply take the DFT to yield $X[k]$ and send the first $K << N$, say $K = 1000$, nonzero samples along with a note that reads "Please find attached $K = 1000$ samples of the DFT of $x[n]$. Zero-pad the signal to be of length $N = 50 \cdot 10^6$ and take an IDFT to decode my message."

However, you can do better. This method involves taking an $N = 50 \cdot 10^6$-point DFT, which might take a while in MATLAB! The intuition behind why you can do better comes from what you know about solving systems of equations. There are only K unknown values that completely determine $x[n]$ (since all of the information in $x[n]$ is contained in $X[k]$, which only has K nonzero values): if you want to solve for K unknowns, you need only K distinct equations. So, what you'd like to be able to do is send only K time domain samples of $x[n]$ to your friend and let him solve for the K unknown DFT coefficients $\{X[k]\}_{k=0}^{K-1}$. He can then put those values in an array, zero-pad it, and take an IDFT (which will not take nearly as long as taking a DFT, since $K << N$). The only question that remains is which K samples of $x[n]$ should you send?

3.1.1 Exercise:

Let $N = 60$ and $K = 4$. Show that your friend can recover the nonzero DFT coefficients (and thus reconstruct the entire signal $x[n]$) from from four uniform samples, $x[0]$, $x[15]$, $x[30]$, and $x[45]$ (hint: how can you turn this problem into one of solving four equations in four unknowns, i.e. how can you relate the four unknowns $X[0]$, $X[1]$, $X[2]$, and $X[3]$ to the four known samples $x[0]$, $x[15]$, $x[30]$, and $x[45]$?).

3.1.2 Exercise:

MATLAB exercise: coming soon...

3.1.3 Exercise:

Argue that your friend can, in fact, recover the entire signal $x[n]$ from *any four* samples of $x[n]$, i.e. random sampling works as well (hint: Vandermonde matrices [4]).

4 Closing Thoughts

Hopefully you have gained some intuition as to how the DFT works, and hopefully you are now curious about sparse signal processing! However, there is one thing that should bother you about the example problem given above: there was

[4] http://en.wikipedia.org/wiki/Vandermonde_matrix

really nice structure in the DFT. How often do you think we see that structure (the nonzero coefficients nicely arranged in the first K DFT bins) in the real world? Additionally, how do we even know how many nonzero coefficients there actually are without computing a full N-point DFT (which completely defeats the purpose of sparse signal processing)? And even worse, what's the point in knowing K if you don't know *where* the nonzero values are located in the DFT?

Per the intuition we used earlier, it seems reasonable that we should be able to determine the locations and values of the nonzero DFT coefficients from $2K$ time domain samples of the signal ($2K$ unknowns, location and value in this case, should require only $2K$ distinct equations to solve), but this is an open problem in signal processing! The best we've been able to do so far is $4K$ samples, [5] and that result was only achieved one year ago! Sparse signal processing is an example of *compressive sensing*, which deals with recovering signals from sparse measurements. For those interested in this topic, speak with Professor Miki Lustig and consider taking courses like EE 121, EE 123, EE 127, and EE 225A.

[5] http://arxiv.org/abs/1305.0870