Lab 6
November 1, 2016
Vector-Space Signal Processing

INSTRUCTIONS:
All lab submissions include a written report and source code in the form of an m-file. The report contains all plots, images, and figures specified within the lab. All figures should be labeled appropriately. Answers to questions given in the lab document should be answered in the written report. ***The written report must be in PDF format***. Submissions are done electronically through **my.ECE**.

1 Convolution Revisited

If a system satisfies linearity and shift-invariance, its output $y(n)$ can be expressed as a convolution between $x(n)$ and the system impulse response $h(n)$.

$$y(n) = \sum_{m=0}^{N-1} x(m)h(n-m) \tag{1}$$

Recall that convolution is a linear operation, and linear operations can be expressed as a matrix equation of the form $\mathbf{y} = \mathbf{Cx}$. For convolutions, $\mathbf{C}$ is called the convolution matrix and it has a special structure.

Report Item: Given

$$x(n) = \{1, 4, -4, -3, 2, 5, -6\}$$
$$h(n) = \{1, 2, 3, 4, 5\}$$

Find the convolution between $x(n)$ and $h(n)$ by generating the convolution matrix using **convmtx**. Plot the resulting matrix using **imagesc**. What do you notice about the structure of the **convmtx**? Verify that **Cx** produces the same result as **conv**. Plot the results for both using **stem** on the same plot.

2 A Linear Algebra Primer

In all of our labs and in MATLAB, we treat the signals as N dimensional vectors. This section provides a brief overview of some useful linear algebra concepts and tools that when applied to signal processing, could give us rather interesting results. Rigourous mathematical understanding of this section is not required but will be helpful.

2.1 Properties of Matrices

We call a matrix $\mathbf{U}$ orthornormal if:

$$\mathbf{U}^H\mathbf{U} = \mathbf{U}\mathbf{U}^H = \mathbf{I} \tag{2}$$

where ^{H} denotes the conjugate transpose[1].

This also implies that for square $\mathbf{U}$,

$$\mathbf{U}^{-1} = \mathbf{U}^H \tag{3}$$

2.2 Eigenvalues and Eigenvectors

Given a matrix A, we can find vectors and scalars that satisfy the equation:

$$\mathbf{A}\mathbf{x}_n = \lambda_n\mathbf{x}_n \tag{4}$$

The vectors x and the corresponding λ values that satisfy the equation above are called eigenvectors and their corresponding eigenvalues respectively. Note that for an eigen value, there is an infinite number of eigenvectors as we can simply scale $\mathbf{x}$ on both sides of the equation. In the following section, we will make use of this flexibility in choice by fixing the magnitude of the eigenvector $\mathbf{x}$ to 1 (this is the result MATLAB will return when you use the function eig). (i.e $\mathbf{x}^H\mathbf{x} = 1$) Since magnitude can be changed arbitarily, we are only interested in eigenvectors that have a different direction. We call these *distinct eigenvectors*.

We can then collect these distinct eigenvectors and their corresponding eigenvalues into matrices as follows:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N \end{bmatrix} \tag{5}$$

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & \\ 0 & \lambda_2 & & \\ \vdots & & \ddots & \\ & & & \lambda_N \end{bmatrix} \tag{6}$$

So that[2]

$$\mathbf{A}\mathbf{X} = \mathbf{X}\mathbf{\Lambda} \tag{7}$$

2.3 SVD and Matrix Factorization

Given a matrix, we often want to find this matrix as a product of 2 or more matrices satisfying certain conditions. This is called matrix factorization or matrix decomposition. While there are many possible ways to factorize a matrix, the Singular Value Decompositionis of interest to us in this case. It is a factorization that satisfy the following conditions[3]:

[1] The conjugate transpose is also called the Hermitian transpose. Therefore the ^{H} notation.

[2] This is also known as eigen-decomposition

[3] In general, SVD is defined as products of non-square matrices for a arbitarily shaped $\mathbf{A}$, but we are restricting ourselves to only square matrices in this lab for simplicity.

Given $\mathbf{A} \in \mathbb{C}^{\mathbf{N}\times\mathbf{N}}$ an $N \times N$ square matrix

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H \tag{8}$$

where $\mathbf{U}, \mathbf{V}, \mathbf{\Sigma} \in \mathbb{C}^{N\times N}$ are also square matrices of the same size as $\mathbf{A}$. Furthermore, $\mathbf{\Sigma}$ is a diagonal matrix with, and $\mathbf{U}$,$\mathbf{V}$ are orthornomal.

In general, even with the above conditions the choice of $\mathbf{U}, \mathbf{V}, \mathbf{\Sigma}$ are not unique. One way of obtaining these matrices is as follows. Taking (8) and multiplying it with its conjugate transpose, we have

$$\begin{aligned}\mathbf{A}\mathbf{A}^H &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H\mathbf{V}\mathbf{\Sigma}^H\mathbf{U}^H \\ &= \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^H\mathbf{U}^H\end{aligned}$$

The second equality makes use of the fact that $\mathbf{V}$ is orthonormal. We can then multiply both sides by $\mathbf{U}$ and apply the orthonormal property again to obtain

$$\mathbf{A}\mathbf{A}^H\mathbf{U} = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^H \tag{9}$$

Observe that (7) and (9), have the same form. Now we have that $\mathbf{U}$ is the eigenvector matrix[4] of $\mathbf{A}\mathbf{A}^H$ and since Σ and Λ are diagonal, Σ is simply the square root of the elements of Λ.

Report Item: Repeat the above process for $\mathbf{A}^H\mathbf{A}$ to obtain an expression for $\mathbf{V}$. Include your derivation in your lab writeup.

Report Item: Given

$$\mathbf{A} = \begin{bmatrix} 1 & 4 & -2 \\ 3 & 11 & 5 \\ 7 & 7 & 7 \end{bmatrix}$$

Determine the eigenvalues and eigenvector of $\mathbf{A}\mathbf{A}^H$ and $\mathbf{A}^H\mathbf{A}$ using **eig**. Then, take the singular value decomposition of $\mathbf{A}$ using **svd** and verify that $\mathbf{U}$ are the eigenvectors of $\mathbf{A}\mathbf{A}^H$ and $\mathbf{V}$ are the eigenvectors of $\mathbf{A}^H\mathbf{A}$.

3 Vector Space Representation of Signals

Labs 1 and 2, saw the matrix representation of the DFT operation.

$$\mathbf{X} = \mathbf{F}\mathbf{x}$$

where $\mathbf{F}$ is the DFT matrix and its elements are

$$\mathbf{F}_{mn} = e^{-i2\pi mn/N} \tag{10}$$

We often call $\mathbf{x}$ the time domain representation and $\mathbf{X}$ the frequency domain representation of the signal. Recall from any linear algebra class, if we wish to obtain $\mathbf{x}$

[4] In general, the eigen-decomposition of a matrix will not be unitary. However, AA^H has a unique structure

from $\mathbf{X}$, we need to find the inverse of the matrix $\mathbf{F}$. Owing to the special properties of the DFT matrix, the inverse is simply $\frac{1}{N}F^H$.

Report Item: Given

$$x(n) = \{1, 1, 4, -4, -3, 2, 5, -6, 3, 2, 4, -2, 5, 9, -8, 4\}$$

Generate the DFT matrix $\mathbf{F}$ corresponding to an input $\mathbf{x}$ using **dftmtx**. Plot the real and imaginary parts of this matrix using **subplot** and **imagesc**. Comment on what you see. In particular, what waveform do you notice in each row and how does the frequency vary with higher row numbers? Use the term *dot product* to describe the relationship between DFT coefficients and $\mathbf{x}$.

One way to look at the DFT operation is finding the representation of the signal as a linear combination of the columns of the IDFT matrix. We call these columns the basis vectors of the DFT.

Report Item: (1) Find the inverse DFT matrix of the previous question, do not use any of MATLAB's built-in matrix inverse functions or operators. (2) Compute the inner product (dot product for discrete vectors) of each of these columns. This can be done by generating a matrix $\mathbf{A} = \mathbf{B}^H \cdot \mathbf{B}$ where $\mathbf{A}$ is a matrix containing the results of the inner products. What can you say about the relationship between the columns in this matrix (are they orthogonal)?

The time domain signal (of length N) is a vector in $\mathbb{R}^N$ using the standard basis, while the Fourier transform is the representation of the same signal in the Fourier Transform basis. While useful in a large number of situations, we need not restrict ourselves to this basis. A natural question is that, if we know what kind of signals we are dealing with, can we come up with a basis that's well suited to representing them?

4 Principal Component Analysis

In the signals that we have dealt with so far, we had assumed that they varied with time, and each sample that we had taken is a small step in time. When a signal is of such a nature, the use of the Fourier Transform Basis is usually a good fit for the problem.

Now consider data samples of this nature. Everytime you take a measurement you obtain a vector of length N. After M measurements, you obtain M vectors of length N. One possible way to analyize such a signal is to place them side by side, forming a very long vector of length MN. However, suppose you know that every n-th sample in each measurement is somehow related the n-th sample in other measurements, can we take advantage of this knowledge?

One example of such a type of signal is videos or sequences of images. At every frame, millions of pixels were measured, and multiple samples were made across time. Each of these pixels in every frame are very much related to the same pixel in the next or previous frame. This is even more so if every frame is an image of a similar item/object.

With such a structure, we can measure the mean and variance of each sample point and also covariances between sample points since the location of where they are in the vector matters. We can then use this knowledge to learn(generate) a basis that can represent these most of the variances without using too many components. This method is known as Principal Component Analysis (PCA). PCA is one of the simple, yet powerful techniques often used in machine learning.

Instead of placing the samples side by side, we can stack these samples so that we form a $M \times N$ matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_M^T \end{bmatrix} \tag{11}$$

where $\mathbf{x}_m^T$ are row vectors (or the transpose of column vectors). Each row $\mathbf{X}$ is called a sample or an example, especially in machine learning literature. We can then compute the mean of each element of $\mathbf{x}$ and define a vector as

$$\boldsymbol{\mu}_X = \frac{1}{M} \sum_{n=1}^{M} \mathbf{x}_n^T \tag{12}$$

Next we can recenter our data so that they have zero mean.

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{x}_1^T - \boldsymbol{\mu}_X \\ \mathbf{x}_2^T - \boldsymbol{\mu}_X \\ \vdots \\ \mathbf{x}_M^T - \boldsymbol{\mu}_X \end{bmatrix} \tag{13}$$

With mean zero, we can now obtain the covariance matrix, $\mathbf{R}$, simply as

$$\mathbf{R} = \tilde{\mathbf{X}}^H \tilde{\mathbf{X}} \tag{14}$$

Notice that in doing so, the covariance matrix is always square and also symmetric. Since it's symmetric, the SVD of $\mathbf{R}$ will be of the form

$$\mathbf{R} = \mathbf{V}\mathbf{D}\mathbf{V}^H \tag{15}$$

where $\mathbf{V}$ is the eigenvector matrix of $\mathbf{R}$ and $\mathbf{D}$ is a diagonal matrix of singular values. Recall that in Section 2 we merely specify that $\mathbf{D}$ is the collection of the eigenvalues. In PCA, we sort this matrix in the order of *decreasing* eigenvalues (i.e. $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_N$), MATLAB does this for you automatically in its implementation of SVD. When we do this, the first eigenvector is then the direction of greatest variance in the dataset. Furthermore, each subsequent eigenvector is next largest direction of variance. Together with the mean vector, these eigenvectors form the PCA basis.

Report Item: Use the provided function loadImages to load the images provided in *yalefaces* into $\mathbf{X}$ (read what's provided in the function and make sure you understand what the function is doing). Implement the function compMeanVec to compute the mean vector of a matrix as described above. Reshape this mean vector into a 2D image and display it using imagesc using a gray color map. What do you see?

Report Item: Compute its covariance matrix of $\mathbf{X}$ and perform SVD on the covariance matrix to obtain the first 100 eigenvectors and eigenvalues.

1. Plot a graph of these eigenvalues (the values on the diagonal of the matrix $\mathbf{D}$). Comment on the magnitude first 20 eigenvalues as compared to the rest of the eigenvalues.

2. Reshape and display the first 4 eigenvectors using imagesc. Can you describe what you see?

3. Reshape and display the 50th and 100th eigenvector. What do you see now? What do you think the eigenvectors v_n for $n \geq 50$ look like?

Now that we have a set of basis vectors, what remains is to find a way to represent our signals in this new basis, and also a way to return to our original representation.

Recall that $\mathbf{V}$, the matrix of the eigenvectors is a collection of the basis vectors. This means that any signal x can be represented as a linear combination of its columns. That is:

$$[\tilde{\mathbf{x}}]_{orig} = \mathbf{V}[\mathbf{x}]_{pca} \tag{16}$$

However, remember that we had subtracted away the mean from the signal, Therefore, the original signal can be recovered as

$$[\mathbf{x}]_{orig} = \mathbf{V}[\tilde{\mathbf{x}}]_{pca} + \boldsymbol{\mu}_x \tag{17}$$

Which gives us the inverse transform.

As for the forward transform,

$$\begin{aligned} [\mathbf{x}]_{pca} &= \mathbf{V}^{-1}([\mathbf{x}]_{orig} - \boldsymbol{\mu}_x) \\ &= \mathbf{V}^{-1}[\tilde{\mathbf{x}}]_{orig} \\ &= \mathbf{V}^{H}[\tilde{\mathbf{x}}]_{orig} \end{aligned} \tag{18}$$

since $\mathbf{V}$ is unitary.

In the previous exercises, we only compute a small number of eigenvectors and ignored the rest since their eigenvalues are small. Verify that the equations given above are still valid when we have smaller matrices for $\mathbf{V}, \mathbf{D}$ (check that the dimensions agree). If we were to compute every eigenvector, we will be able to represent any signal just like using the Fourier Transform basis. However, since we generated this basis from our set of example data, we claim[5] that we will be able to represent these faces using very few components with little loss in signal accuracy.

[5]Those interested in a proof can do it as an exercise if you have sufficient knowledge in linear algebra.

Report Item:

1. Implement the function PCAtransform that takes in a mean vector, the matrix $\mathbf{V}$ of eigenvectors and signal $[\mathbf{x}]_{orig}$. It should return a vector in the PCA basis. The length of the vector should be equal to the *number* of eigenvectors provided. (Equation (18))

2. Implement the function invPCAtransform that takes in a mean vector, the matrix $\mathbf{V}$ of eigenvectors and signal $[\mathbf{x}]_{pca}$. It should return a vector in the original basis. The length of the vector should be equal to the *length* of the eigenvectors provided. (Equation (17))

Report Item: PCA Noise Reduction

Load the image noisy_face.png. Reshape it into the correct shape. Using the PCAtransform function that you had implemented, obtain the representation of this image in the PCA basis.
Now, use the invPCAtransform function to transform it back into the original basis. Display this image using imagesc.