Lab 7
April 17, 2018
Multirate Theory and Msc. Topics in Adaptive Filters

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 **Compass 2g**.

If you're not that experienced with up and down sampling, you may want to do the problems in the following order 4,5,6,7,2,1,3.

1 Discussion

Sample rate conversion is, by definition, changing the sample rate of a discrete signal. Upsampling is the addition of samples to a signal to effectively increase the sample rate of the signal. Conversely, downsampling is the decimation of a signal to effectively decrease the sample rate. In either case, the total length of the signal (in time) does not change. When the number of samples increases, the sample rate decreases such that the total time remains the same. For example, music can be sampled at a low bitrate is the same length as music sampled at a high bitrate. A higher bitrate just corresponds to more samples per second which corresponds to higher frequencies captured.

2 Upsampling and Downsampling

A discrete signal must be band limited in order for the signal to be upsampled. A signal which is not bandlimited can still be upsampled, but the results will be undesirable. Upsampling is a 2 step process. For example, upsampling by U corresponds to inserting $U-1$ zeros between each sample of the original signal. More precisely, given that $x(n)$ is the original signal with length N and $x_u(n)$ is the upsampled signal,

$$x_u(m) = \begin{cases} x(n) & m = Un \\ 0 & else \end{cases} \tag{1}$$

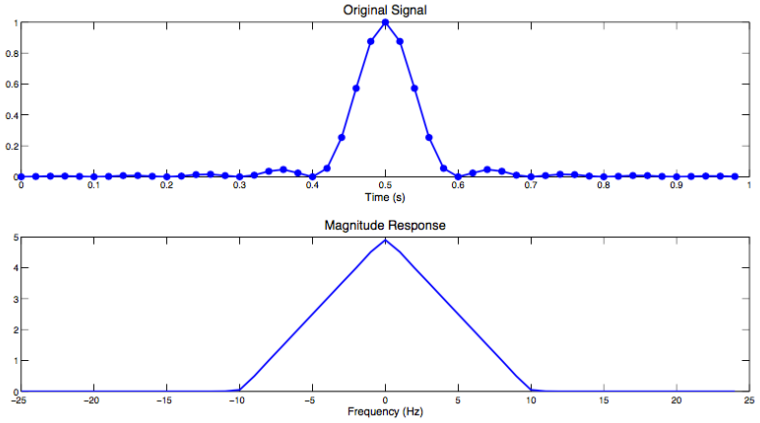

Figure 1: The original signal $x(n)$ and its magnitude response

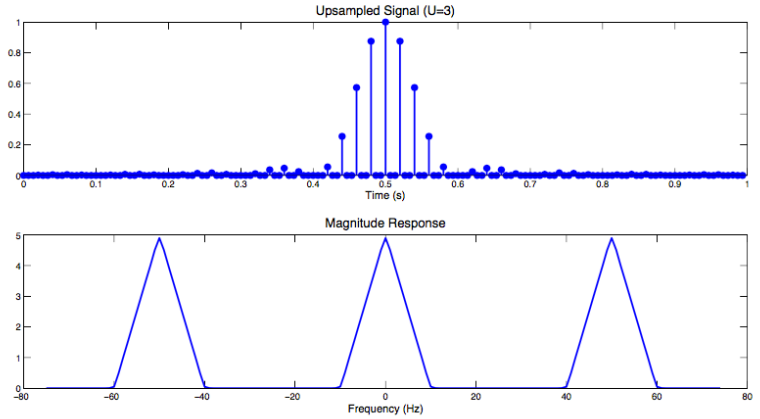

Figure 2: $x(n)$ upsampled by $U = 3$. This increases the sampling frequency by 3 and also introduces copies in the frequency domain due to the scrunching of the frequency axis.

The insertion of $U-1$ zeros between samples of $x(n)$ causes the frequency axis to scrunch by a factor of U. However, this scrunching is only apparent in the DTFT, where $-\pi < \omega < \pi$. In the Ω domain, there is no scrunching. That's because $\Omega = \omega \cdot f_s'$ where $f_s' = f_s U$. So the scrunching in the ω domain comes from the fact that f_s' for $x_u(n)$ is larger than f_s. Pay special attention to the time axis of each graph. The duration of the signal from a time perspective does not change.

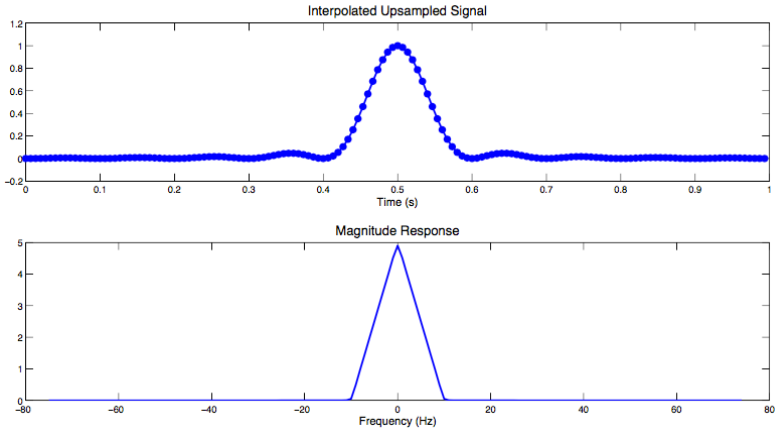

Figure 3: Applying an ideal LPF in the frequency domain to remove the copies.

The spectrum of $x_u(n)$ has the same shape as the spectrum of $x(n)$, barring the additional copies. To remove the additional copies, we can apply an ideal LPF. In MATLAB, this can be done by letting $X_u(\omega) = 0$ for $|\omega| > f_s/2$ where $X_u(\omega)$ is the DTFT of x(n). The result is shown in Figure 3. Figure 4 superimposes $x(t)$ and $x_u(t)$ to show that one is the upsampled version of the other.

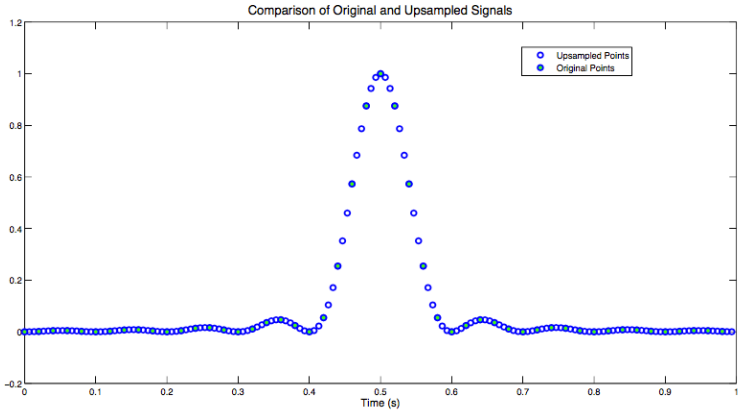

Figure 4: Comparison of the original signal $x(n)$ to its upsampled version $x_u(n)$.

By comparison, downsampling is much simpler than upsampling. The downsampling operation can be expressed as

$$x_d(n) = x(nD) \tag{2}$$

where D is an integer greater than 1. The choice of D is restricted by Nyquist sampling rate, which can be violated if D is chosen to be too large. Upsampling and downsampling can be combined to achieve the desired sample rate. The final sample rate can be expressed as $f_s' = \frac{U}{D} f_s$.

Report Item 1: An audio technician accidentally set the sample rate to 66,150 Hz during an important recording session. This is 1.5 times the standard 44,100 Hz. Having already said goodbye to the musicians, he happens to come across an ECE311 student who understands sample rate conversion. Read the sound **audioclip.wav** using **audioread**. What values of U and D can you choose such that the final sample rate is 44,100 Hz. Plot the magnitude spectrum (in dB) of the upsampled signal before applying the lowpass filter, after applying the lowpass filter, and post downsampling. Be sure to scale the frequency axis Ω **in Hz** with respect to the proper sampling frequency. Use **sound** to play the original sound and the sound after sample rate conversion **at the new sampling rate, 44,100 Hz**. Does the original sound faster or slower compared to the new sound?

Report Item 2: Luckily, the audio technician's dad owns the record company, and therefore he did not lose his job. Not too long after, the same audio technician was asked to record the newest electronic music album "The DSPain for life" from Ben and Yu-Jeh. While he was eating nachos during the studio sessions, he opened up the door to catch some fresh air. Surprisingly, Bruce Wayne lives just nearby, and one of the bats was curious enough to leave the bat cave and flew right into the studio without anyone noticing. Bats are known to communicate with ultra-high frequency sound waves, and so the hit song from Ben and Yu-Jeh was corrupted.

(a) Load **song1_corrupt.mat** and plot the FFT frequency response. Here we have the song recorded at a sampling rate of 44100 Hz. Can you tell which could be the bat's sound ? What is the frequency for that ?

(b) Listen to it. Can you hear the bat? Why or why not? **HINT:** human hearing has a typical lower bound and upper bound for the frequency of a sound.

(c) Without a LPF, downsample the song by $D = 2$. Now, you have the song with a sampling rate of 22050 Hz. Can you now hear the bat? Plot the FFT frequency response and explain.

(d) Now, we know that without a LPF, the downsampling could cause some problem. Look at the frequency response obtained in part (a) and design an LPF for downsampling by $D = 2$

(e) Apply the LPF designed in part (d) and downsample the original corrupted sound signal by $D = 2$. Can you hear the bat? Plot the FFT frequency response and explain.

3 Spotify Data Transfer Issue

You're a Graduate of the ECE program from the University of Illinois, working as an audio engineer at Spotify. On March 10, 2019, a hit song (to be remain unnamed), goes on the radio. It goes viral. Suddenly, at the Spotify Server center, they realize that the data transfer rate has reached its limit. Your boss, Mr. Kuo, believes that you can downsample the data and have the software at the receiver end do upsam-

pling and interpolation. This may fix the data transfer issue.

Report Item 3:
Load the file 'songz.mat', a variable named `good_news` should appear on the screen. This contains a song that has not been upsampled or downsampled; it's the original. The sampling rate for this song is 48000 Hz. Determine an acceptable upsampling and downsampling rate for this problem. At the minimum, include a plot of the magnitude spectrum of the downsampled spectra, upsampled spectra, the magnitude response of your interpolation filter, and the magnitude response of your filtered signal. Listen to your final song to ensure if the quality is acceptable. Briefly explain the process of determining these rates.

4 Image Filtering

An image can be filtered using 2-D convolution, which can be expressed as

$$Y(m,n) = \sum_{m'} \sum_{n'} X(m',n')h(m-m',n-n') \tag{3}$$

where $h(m,n)$ is the filtering kernel. An example of a low-pass filter is

$$h(m,n) = \begin{bmatrix} 1/8 & 1/16 & 1/8 \\ 1/16 & 1/4 & 1/16 \\ 1/8 & 1/16 & 1/8 \end{bmatrix} \tag{4}$$

There are several ways to recognize that this is a lowpass filter. First, using the fact that the DC component of the Fourier transform is proportional to the sum of the coefficients in $h(m,n)$, it can be seen that $\sum_m \sum_n h(m,n) = 1$. Since the DC component is non-zero, it is likely a lowpass lter. Another clue is the fact that the filter looks Gaussian. Since a Gaussian in the spatial domain is also a Gaussian in the frequency domain, the filter is a lowpass.

Report Item 4: Load *image1.jpg* using **imread**. This image contains what is known as salt and pepper noise. Filter this image using the lowpass filter given in (8) and plot the result using **imagesc**. Use **conv2** to apply the filter.

In contrast, the Laplacian filter is a highpass filter. This can be deduced from the fact that the sum of its coefficients equals zero, which suggests that it cannot be a lowpass. A highpass filter can be used for edge detection in an image.

$$h(m,n) = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad (\textit{Laplacian Filter}) \tag{5}$$

Report Item 5: Load *image2.jpg* using **imread**. Apply the Laplacian filter and plot the resulting figure using **imagesc** with **colormap('gray')**. Use **conv2** to apply the filter.

5 Least Mean Squares Filter (LMS)

In this class thus far, we have worked on filter design and spectral analysis. Typically, the problems encountered have you design some filter h to augment the signal characteristics of some input x. For examples, you may be asked to design a low pass filter to remove high frequency interference. You have employed spectral analysis to design such filters. Filter design addresses this type of problem.

Another type of problem you may be interested in solving is given some output y and input x, determine a filter h such that $y = h*x+\epsilon$ where $*$ denotes a convolution and ϵ represents noise and uncertainty. The method that one may use to find this h is known as the Least Mean Squares Filter algorithm (LMS). This technique is iterative and data driven.

5.1 Signal Model

Consider the following linear FIR system

$$y[n] = h[0]x[n] + h[1]x[n-1] + \dots h[N-1]x[n-(N-1)] \tag{6}$$

This expression simply means that the current output is just a weighted sum time lagged samples of $x[n]$. In this case, the output depends on the current samples and the previous $N-1$ samples, weighted by the impulse response h. We can also write the output $y[n]$ as the convolution of $x[n] * h[n]$.

Suppose that we are not given h, the impulse response. If we are given instead, $x[n]$, the input, and $y[n]$, the output, how can we determine the impulse response $h[n]$?

We could use a least mean squares technique. In other words, can we find $h[n]$ that minimizes the cost function

$$C(h) = ||E||_2^2 = ||y[n] - h[n] * x[n]||_2^2 \tag{7}$$

In matrix notation, we can reform this convolution as a matrix operation.

$$E = Y - X^T h \tag{8}$$

$$\begin{bmatrix} e[0] \\ e[1] \\ \vdots \\ e[T-1] \end{bmatrix} = \begin{bmatrix} y[0] \\ y[1] \\ \vdots \\ y[T-1] \end{bmatrix} - \begin{bmatrix} x[0], x[-1], \dots, x[-(N-1)] \\ x[1], x[0], \dots, x[-(N-1)+1] \\ \vdots \\ x[T-1], x[T-2], \dots, x[T-1-(N-1)] \end{bmatrix} \begin{bmatrix} h[0] \\ h[1] \\ \vdots \\ h[N-1] \end{bmatrix}$$

Y is a vectorized form of $y[n]$. X^T is the convolution matrix formed from $x[n]$ by lagging the $N-1$ samples in the rows. The h vector contains the filter coefficients.

This simply means find $h[n]$, such that $h[n] * x[n]$ has a minimum mean square error with $y[n]$. This can be done by setting the gradient of the cost to zero or $\nabla C(h) = 0$. More concretely:

$$\nabla C(h) = 2X(X^T h - Y) = 0 \tag{9}$$

By some playful algebra, we can directly solve for h

$$h = (XX^T)^{-1}(XY); \tag{10}$$

For large matrices X, the inversion of XX^T in equation 10 may not be computationally tractable. For modern, data-driven optimization problems, we will use an algorithm called gradient or steepest descent to solve the LMS filtering problem. One should note that this LMS formulation is very general; it can be applied to regression problems and training neural networks.

For exact derivation of the gradient and a more complete formulation of the LMS refer to the ECE 310 notes. Below, we outline the algorithm for determining the filter $h[n]$.

5.2 Algorithm

Here, we attach pseudo-code for the LMS algorithm. Read these steps carefully.

Data: X, an input array, Y, an output array, N, filter length, μ, learning rate
Result: filter coefficients h
$h = [0, 0 \dots 0]$ // Initialize h with N zeros;
for *n = 0,1,2 ... T-1* **do**
  $x_n = [x[n], x[n-1], x[n-2], \dots, x[n-(N-1)]$;
  $e_n = y[n] - h^T x_n$;
  $h = h + \mu e_n x_n$;
end

Algorithm 1: LMS Algorithm

One should note that the algorithm above is a one step update, where one iterates sample by sample. Other methods can iterate over ensembles or batches of samples to minimize the same cost; this is known as stochastic gradient descent (SGD).

Report Item 6: For this problem, load the files *sig.mat*. This file contains variables x and y. Using the LMS algorithm described above, determine the filter h. Estimate h and plot the error as a function of iteration. The length of the filter is 5. Use $\mu = 0.01$. Repeat this process for $\mu = 0.001$ *and* 6. This problem will result in four error plots, and four estimates of h.

6 Convolutional Neural Networks

More recently in the field of signal processing, there has been much attention paid to machine learning. It is difficult to define what machine learning is; the definition varies from person to person. To some members of the signal processing community, machine learning is just a generalization of adaptive filtering.Generally, machine learning is the study of algorithms that uses statistical techniques to model systems from data. Many times, these systems perform tasks like classification, regression, and prediction. At a high level, these techniques have applications to speech recognition, autonomous cars, image captioning. The list is endless. This section will have you play around with an already built neural network.

One should note that the study of neural networks is covered in more details in other signal processing classes like ECE 417, CS 446, and ECE 544NA.

In the following report item, we will perform a few experiments with using a convolutional neural network to perform digit classification with the MNIST dataset. At a high level, a neural network is a collection of nodes connected by links similar to neurons and synapses in biology. These neural networks transform data, perhaps an image, to so target value. Perhaps, you are interested in classifying dogs and cats. One would input an image of a picture of a cat into the network and through a series convolutions, regressions, and non-linear operations, the network can output the probability that input into the image is a dog or a cat. Similar to a filter, these networks are parameterized by weights. In large networks, the number of weights can be in the hundreds of thousands. To compute these weights, we can use least

squares techniques like in the previous section to fit a set of data. This process is known as training.

Report Item 7:

For this problem, read through the file traincnnmnist.m . In addition, make sure to run the command "module load matlab/R2017b" before running matlab. This file contains all the components for a neural network.

This code is taken from the website: https://www.mathworks.com/help/nnet/examples/create-simple-deep-learning-network-for-classification.html

1.) Read through the code and answer the following questions. One of these questions will be directly on the quiz.
a.) What is training and validation?
b.) What do activation, convolutional, and fully connected layers do?
c.) What is the input of this network? Output?

2.) Run this code. Comment on the loss function and validation accuracy?
3.) Redo 2.) but set the number of training samples to 250? comment on the performance (loss and accuracy)

4.) Redo 2.) but set the learning rate to 0.5. Comment on the performance (loss and accuracy)