PSYCH-GA.2211/NEURL-GA.2201 – Fall 2024
Mathematical Tools for Neural and Cognitive Science

Homework 3

Due: 29 Oct 2024
(late homeworks penalized 10% per day)

See the course web site for submission details. Reminder: rather than using the functions `pinv()` and `norm()`, use the linear algebra tools we learned in class. Do yourself a favor, and don't wait until the day before the due date... *start now* !

1. **LSI system characterization.** You are trying to experimentally characterize three auditory neurons, in terms of their responses to sounds. For purposes of this problem, the responses of these neurons are embodied in compiled matlab functions `unknownSystemX.p` with X=1, 2, 3. If you are using Python, import the `unknown_systems` module from the obfuscated Python file. Each takes an input column vector of length $N = 64$ whose elements represent sound pressure over time. In Python the response of each is a column vector (of the same length) representing the mean spike count over time. For each neuron,

   (a) "Kick the tires" by measuring the response to an impulse in the first position of an input vector. Check that the system is consistent with shift-invariance by comparing this to the response to an impulse at positions $n = 2, 4, 8$. Check that the system is consistent with linearity by testing whether the response to a sum of any two of these impulses is equal to the sum of their individual responses. Also examine respones to impulses at different n to determine how the system handles inputs near the boundary (i.e., whether the system does circular boundary-handling). Describe your findings.

   (b) If the previous tests succeeded, examine the response of the system to sinusoids with frequencies $\{2\pi/N, 4\pi/N, 8\pi/N, 16\pi/N\}$, and random phases, and check whether the outputs are sinusoids of the same frequency (i.e., verify that the output vector lies completely in the subspace containing all the sinusoids of that frequency). [Note: make all elements of the the input stimuli positive, by adding one to each sinusoid. The responses will then also be positive (mean spike counts).]

   (c) If the previous tests succeeded, verify that the change in amplitude and phase from input to output is predicted by the amplitude (`abs`) and phase (`angle`) of the corresponding terms of the Fourier transform of the impulse response. If not, explain which property (linearity, or shift-invariance, or both) seems to be violated by the system. If so, does the combination of all of your tests *guarantee* that the system is linear and shift-invariant? What set of tests would provide such a guarantee?

2. **Retinal and LGN neurons.** The response properties of ganglion cells in the retina or neurons in the lateral geniculate nucleus (LGN) are often described using linear filters. We'll examine a one-dimensional cross-section of a common choice, known as the difference-of-Gaussians (or DoG) filter.

   (a) Create a one-dimensional linear filter that is a difference of two Gaussians, $\exp\left(-\frac{n^2}{2\sigma^2}\right)$, (each normalized to sum to 1), and with standard deviations $\sigma = 1.5$ and $\sigma = 3.5$

samples. The filter should contain 15 samples, with both Gaussians centered on the middle (8th) sample. Plot the filter to verify that it looks like what you'd expect. Plot the amplitude of the Fourier transform of this filter, sampled at 64 locations (MATLAB's `fft` function takes an optional additional argument). What kind of filter is this? Why does it have this shape, and how is the shape related to the choice of parameters (σ's)? Specifically, how does the Fourier amplitude change if you alter each of these parameters?

(b) If you were to convolve this filter with sinusoids of different frequencies, which of them would produce a response with the largest amplitude? Obtain this answer by reasoning about the equation defining the filter (above), and also by finding the maximum of the computed Fourier amplitudes (using the `max` function), and verify that the answers are the same. Compute the *period* of this sinusoid, measured in units of sample spacing (hint: this is the inverse of its frequency, in cycles/sample), and verify by eye that this is roughly matched to the oscillations in the graph of the filter itself. Which two sinusoids would produce responses with about 25% of this maximal amplitude?

(c) Create three unit-amplitude 64-sample sinusoidal signals at the three frequencies (low, mid, high) that you found in part (b). Convolve the filter with each, and verify that the amplitude of the response is approximately consistent with the answers you gave in part (b). (hint: to estimate amplitude, you'll either need to project the response onto sine and cosine of the appropriate frequency, or compute the DFT of the response and measure the amplitude at the appropriate frequency).

(d) Verify the convolution theorem. Apply the Fourier transform to each of your three stimuli. Multiply each by the Fourier transform of the Gabor filter. Inverse Fourier transform the results and verify that the imaginary part is zero, and the real part is equal to the result you obtain from the convolution.

3. **Deconvolution of the Haemodynamic Response.** Neuronal activity causes local changes in deoxyhemoglobin concentration in the blood, which can be measured using functional magnetic resonance imaging (fMRI). One drawback of fMRI is that the haemodynamic response (blood flow in response to neural activity) is much slower than the underlying neural responses. We can model the delay and spread of the measurements relative to the neural signals using a linear shift-invariant system:

$$r(n) = \sum_k x(n-k)h(k), \tag{1}$$

where $x(n)$ is an input signal delivered over time (for example, a sequence of light intensities), $h(k)$ is the haemodynamic response to a single light flash at time $k = 0$ (i.e., the impulse response of the MRI measurement), and $r(n)$ is the MRI response to the full input signal.

In the file `hrfDeconv.mat`, you will find a response vector r and an input vector x containing a sequence of impulses (indicating flashes of light). Your goal is to estimate the HRF, h, from the data. Each of these signals are sampled at 1 Hz. Plot vectors r and x versus time to get a sense for the data. Use the `stem` command (or `plt.stem` in Python) for x, and label the x-axis.

(a) Convolution is linear, and thus we can re-write the equation above as a matrix multiplication, $r = Xh$, where h is a vector of length M, N is the length of the input x, and X is an $[N + M - 1] \times M$ matrix. Write a matlab function `createConvMat`, that takes as arguments an input vector x and M (the dimensionality of h) and generates a matrix X such that the response $r = Xh$ is as defined in Eq. (1) for any h. Verify that the matrix

generated by your function produces the same response as Matlab's `conv` function when applied to a few random h vectors of length $M = 15$. Visualize the matrix X as an image (evaluate `imagesc(X)` in MATLAB or `plt.imshow` in Python), and describe its structure.

(b) Now, given the X generated by your function for $M = 15$, solve for h by formulating a least-squares regression problem:

$$h_{\text{opt}} = \arg\min_h ||r - Xh||^2$$

Plot h_{opt} as a function of time (label your x-axis, including units). How would you describe it? How long does it last?

(c) It's often easier to understand an LSI system by viewing it in the frequency domain. Plot the power-spectrum of the HRF (i.e. $|\mathcal{F}(h)|^2$, where $\mathcal{F}(h)$ is the Fourier transform of the HRF). Plot this with the zero frequency (DC) in the middle (in Matlab you can use a built-in function called `fftshift`), and label the x axis, in Hz. Based on this plot, what kind of filter is the HRF? Specifically, which frequencies does it allow to pass, and which does it block?

(d) Use the convolution theorem to now find h_{opt} by working in the fourier domain. You will need to use the matlab functions `fft` and `ifft`. Remember to be careful about how many samples you choose to have in your fft. Based on the operations you have done, what can you say about when this method will fail? On the same graph, plot the HRF impulse response you recovered from parts (b) and (d).

4. **Pyramid representations and aliasing.** In 1983, Peter Burt and Ted Adelson introduced two "multiscale" representations of images: the Gaussian and the Laplacian pyramid. This provided an efficient method for representing an image in multiple frequency bands. For lower frequencies, subsampling was imposed so that you used fewer numbers to represent smoothly varying, low-frequency images. Here, you will work with a one-dimensional version.

(a) Create a 128-sample signal g_0 (i.e., a vector with 128 entries) that is the sum of three sine waves that have frequencies of 3, 13, and 60 cycles per 128 samples and amplitudes equal to .2, .3 and 1, respectively. Subsample this signal to create vector g_s by throwing away half the samples, maintaining the samples at positions $0, 2, \cdots$, resulting in a vector of length 64. Plot g_0 and s on the same axes using a line graph for each vector (i.e., connect the samples), where the samples from s are plotted in the same positions as where they came from $(0, 2, \cdots)$. Explain what you see.

(b) Write a function `downsample` that takes a vector as input, convolves it with kernel $\left[\frac{1}{16}\frac{1}{4}\frac{3}{8}\frac{1}{4}\frac{1}{16}\right]$, where the impulse response is symmetric, so that the convolution output at sample j applies the central value $\frac{3}{8}$ to the input sample at the same position j. The result of this convolution is then subsampled (usually called "down-sampling"), throwing away every other sample (keeping the samples at positions $0, 2, \cdots$), resulting in a vector with one-half the length of the input. Compute g_1 as the down-sampled version of g_0. Add this blurred-and-subsampled vector as a curve on the above plot. Explain how g_1 differs from s and from g_0.

(c) Next, rather than plotting the subsampled vector along with the original signal, let's try to approximate the input vector as best we can from the subsampled one. The operation is called "up-sampling", that is, to take a vector and create a new vector with twice as

many samples using interpolation. Write a function `upsample` that is given an input vector and (i) first adds samples with value zero between every sample of g_i (plus an extra one at the end), and then (ii) blurs the resulting vector using the same kernel above that we used when down-sampling. You will need to multiply all the values of the kernel by 2 to yield an up-sampled signal with approximately the same values as you started with. Plot `upsample`(g_1) on the same axes and note and explain any differences from what you had before.

(d) Next, we can visualize what information was lost as a result of blurring and subsampling by computing $l_0 = g_0 - upsample(g_1)$. Plot l_0 to see what information it contains (compared to the components of your original signal). Explain.

(e) This process can be iterated. For $i = 1, \cdots, 4$, compute $g_{i+1} = downsample(g_i)$ and $l_i = g_i - upsample(g_{i+1})$. This results in two "pyramid" representations of your signal, the "Gaussian pyramid" (the g_i) and the "Laplacian pyramid" (the l_i), which are both a series of vectors of length 128, 64, 32, ...

(f) Repeatedly up-sample each of your pyramid levels g_i until you have vectors for each with the original 128 samples. Plot all of them and explain what you see.

(g) Up-sample all the l_i to have 128 samples. Plot them along with the up-sampled g_5 and explain what you see.

(h) Finally, sum the upsampled l_i plus the upsampled g_5. What was the result? Why?