

9.29 Problem Set 2 (due Feb. 20)

More convolution and correlation

Feb. 13, 2003

1. Examples of convolution. In class, we studied the boxcar filter $p_i = (\rho_{i-1} + \rho_i + \rho_{i+1})/3$ as a way of estimating probability of firing from a spike train. This can be written as $p = \rho * h$, where $h_1 = h_0 = h_{-1} = 1/3$. For the following, fully specify the h such that $f = g * h$.
 - (a) $f_i = g_i - g_{i-1}$ (discrete approximation to the derivative)
 - (b) $f_i = g_{i+1} - g_i$ (another discrete approximation to the derivative)
 - (c) $f_i = g_{i-5}$ (5 step time delay)
 - (d) $f_i = g_{i+1} - 2g_i + g_{i-1}$ (discrete approximation to the second derivative). For this example, also write down a matrix H such that $f = gH$. Assume that f and g have lengths 5 and 3 respectively, regard them as row vectors (note that the matrix given in the lecture notes is for column vectors).

2. The `conv` function is built in to MATLAB, while the `xcorr` function is part of the Signal Processing Toolbox. Imagine that you are a starving grad student (a modern-day counterpart of Abe Lincoln), and cannot afford to purchase the Toolbox. Nevertheless, your fervent desire to study the art of signal processing drives you to write your own version of `xcorr`.

Type `help xcorr` in MATLAB, and read the description of the first invocation `C=XCORR(A,B)`. Duplicate this with your own code. It shouldn't take more than a few lines, if you make use of the `conv` function. (This exercise is supposed to teach you the relationship between correlation and convolution).

3. Show that the convolution

$$x(t) = \int_0^{\infty} dt' g(t-t')h(t')$$

is the solution of the linear first-order differential equation

$$\tau \frac{dx}{dt} + x = g$$

where $h(t) = \tau^{-1}e^{-t/\tau}$ for $t \geq 0$, and $g(t)$ is an arbitrary function of time. (Hint: integrate by parts). Now define $h(t) = 0$ for $t < 0$ (a kind of zero padding), and show that the convolutional formula above implies

$$\tau \frac{dh}{dt} + h = \delta$$

where $\delta(t)$ is the Dirac delta function. This is why h is regarded as the impulse response for the differential equation.

4. The optimal stimulus. Consider the linear filter

$$r_i = \sum_j d_{i-j} s_j$$

transforming stimulus s into response r using the kernel d . Suppose d is given, and consider a fixed i . The optimal stimulus is defined as the s that maximizes r_i , given the constraint that $\sum_j s_j^2 = 1$. The constraint

is necessary in the definition because the response can be made arbitrarily large by scaling up s by a constant factor.

Prove that the optimal stimulus is proportional to the kernel, $s_j \propto d_{i-j}$. This gives an interesting interpretation of the kernel as the stimulus that is most effective at producing a large response.

Hint: Use the method of Lagrange multipliers from multivariate calculus (read the textbook for more about this).

5. Wiener-Hopf equations. In class we discussed the problem of optimizing the approximation

$$y_i \approx \sum_{j=M_1}^{M_2} h_j x_{i-j} \quad (1)$$

with respect to the filter h_i . It was stated without proof that h_j is the solution of the Wiener-Hopf equations

$$C_k^{xy} = \sum_{j=M_1}^{M_2} h_j C_{k-j}^{xx}, \quad k = M_1, \dots, M_2 \quad (2)$$

where the correlations are defined by

$$C_k^{xy} = \sum_i x_i y_{i+k} \quad C_l^{xx} = \sum_i x_i x_{i+l}$$

If not otherwise noted, all summations are from $-\infty$ to ∞ , and assumed to be finite.

Derive the Wiener-Hopf equations by minimizing the cost function

$$E = \sum_i \frac{1}{2} \left(y_i - \sum_{j=M_1}^{M_2} h_j x_{i-j} \right)^2 \quad (3)$$

with respect to h_j , for $j = M_1$ to M_2 . You will need to compute the partial derivatives $\partial E / \partial h_k$, set them to zero, and play around with summations.

6. Stimulus reconstruction from spike trains. In class, we discussed using the Wiener-Hopf equations to model neural response as a filtered version of the stimulus. In this exercise, we'll work in the opposite direction: the stimulus will be modeled as a filtered version of the spike train. This method was invented by Bill Bialek, Rob de Ruyter van Steveninck, and co-workers, and is described in Section 3.4 of Dayan and Abbott. We'll apply it to the same *Eigenmannia* data that was used in the first problem set. Define $y_i = s_i - \langle s \rangle$ and $x_i = \rho_i - \langle \rho \rangle$, where s_i is the stimulus and ρ_i is the spike train. Use the model of Eq. (1) with $M_1 = -100$ and $M_2 = 300$.

If the spike train were white noise, the optimal filter would be of the form $h_k \propto C_k^{xy}$. This turns out to be a good approximation, even though the spike train is not really white noise. Compute the cross-covariance of the spike train and the stimulus, and normalize by the number of spikes (this is similar to the spike-triggered average of the stimulus). Plot the filter h . Make sure to only plot the elements corresponding to h_{M_1}, \dots, h_{M_2} .

7. Compute $h * x$. Again, the challenge here is to discard the proper elements of the convolution so that $h * x$ lines up with the stimulus y . Plot the first 1000 elements of $h * x$ and y on the same graph. If you've done everything right, you should see very good agreement. Calculate the squared error of the approximation, as defined in Eq. (3).
8. While the preceding filter is good, the optimal filter is found by solving the Wiener-Hopf equations (2). Compute the appropriate elements of the auto-covariance C_i^{xx} , and transform them into a matrix of the form C_{j-k}^{xx} using the `toeplitz` command. Also compute the appropriate elements of the cross-covariance C_i^{xy} . Then solve the Wiener-Hopf equations for h using the backslash (`\`) command. Plot your result for h .
9. Now for the Wiener filter, plot the first 1000 elements of $h * x$ and y on the same graph. If you've done everything right, the agreement should be even better than before. Calculate the squared error of the approximation, as defined in Eq. (3). This number should be lower than before.