Wiener-Hopf equations. Convolution and correlation in continuous time

Sebastian Seung

9.29 Lecture 3: February 11, 2003

When analyzing neural data, the firing rate of a neuron is sometimes modeled as a linear filtering of the stimulus. Alternatively, the stimulus is modeled as a linear filtering of the spike train. To construct such models, the optimal filter must be determined from the data. This problem was studied by the famous mathematician Norbert Wiener in the 1940s. It requires the solution of the famous Wiener-Hopf equations.

1 The Wiener-Hopf equations

Suppose that we'd like to model the time series y_i as a filtered version of x_i, i.e. find the h that optimizes the approximation

$$y_i \approx \sum_j h_j x_{i-j}$$

We assume that both x and y have had their means subtracted out, so that no additive constant is needed in the model. Also, h_j is assumed to be zero for $j < M_1$ or $j > M_2$. This constrains how far forward or backward in time the kernel extends. For example, $M_1 = 0$ corresponds to the case of a causal filter.

The best approximation in the least squares sense is obtained by minimizing the squared error

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

relative to h_j for $j = M_1$ to M_2. This is analogous to the squared error function for linear regression, which we saw in the first lecture.

The minimum is given by the equations, $\partial E / \partial h_k = 0$, for $k = M_1$ to M_2. These are the famous Wiener-Hopf equations,

$$C_k^{xy} = \sum_{j=M_1}^{M_2} h_j C_{k-j}^{xx} \qquad k = M_1, \ldots, M_2 \tag{1}$$

where the shorthand notation

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

has been used for the cross-covariance and auto-covariance. You'll be asked to prove this in the homework. This is a set of $M_2 - M_1 + 1$ linear equations in $M_2 - M_1 + 1$ unknowns, so it typically has a unique solution. For our purposes, it will be sufficient to solve them using the backslash (\) and `toeplitz` commands in MATLAB. If you're worried about minimizing computation time, there are more efficient methods, like Levinson-Durbin recursion.

Recall that in simple linear regression, the slope of the optimal line times the variance of x is equal to the covariance of x and y. This is a special case of the Wiener-Hopf equations. In particular, linear regression corresponds to the case $M_1 = M_2 = 0$, for which

$$h_0 = C_0^{xy}/C_0^{xx}$$

2 White noise analysis

If the input x is Gaussian white noise, then the solution of the Wiener-Hopf equation is trivial, because $C_{k-j}^{xx} = C_0^{xx}\delta_{kj}$. Therefore

$$h_k = \frac{C_k^{xy}}{C_0^{xx}} \tag{2}$$

So a simple way to model a linear system is to stimulate it with white noise, and correlate the input with the output. This method is called reverse correlation or white noise analysis.

If the input x is not white noise, then you must actually do some work to solve the Wiener-Hopf equations. But if the input x is close to being white noise, you might get away with being lazy. Just choose the filter to be proportional to the xy cross-correlation, $h_k = C_k^{xy}/\gamma$, as in the formula (2). The optimal choice of the normalization factor γ is

$$\gamma = \frac{\sum_{jl} C_j^{xy} C_{j-l}^{xx} C_l^{xy}}{\sum_m C_m^{xy} C_m^{xy}}$$

where the summations run from M_1 to M_2. Note this reduces to $\gamma = C_0^{xx}$ in the case of white noise, as in Eq. (2).

3 Discrete versus continuous time

In the previous lecture, the convolution, correlation, and the Wiener-Hopf equations were defined for data sampled at discrete time points. In the remainder of this lecture, the parallel definitions will be given for continuous time.

Before the advent of the digital computer, the continuous time formulation was more important, because of its convenience for symbolic calculations. But for numerical analysis of experimental data, it is the discrete time formulation that is essential.

4 Convolution

Consider two functions g and h defined on the real line. Their convolution $g * h$ is defined as

$$(g * h)(t) = \int_{-\infty}^{\infty} dt' g(t - t')h(t')$$

The continuous variables t and t' have taken the place of the discrete indices i and j. Again, you should verify commutativity and associativity.

If g and h are only defined on finite intervals, they can be extended to the entire real line using the zero padding trick. For example, if h vanishes outside the interval $[0, T]$, then

$$(g * h)(t) = \int_{0}^{T} dt' g(t - t')h(t')$$

5 Firing rate

To define the continuous-time representation of a spike train, we need to make use of a mathematical construct called the Dirac delta function. The delta function is zero everywhere, except at the origin, where it is infinite. You can imagine it as a box of width Δt and height $1/\Delta t$ centered around the origin, with the limit $\Delta t \to 0$. The delta function is defined by the identity

$$h(t) = \int_{-\infty}^{\infty} dt' \delta(t - t')h(t')$$

In other words, when the delta function is convolved with a function, the result is the same function, or $h = \delta * h$. A special case of this formula is the normalization condition

$$1 = \int_{-\infty}^{\infty} dt' \delta(t - t')$$

Note that the delta function has dimensions of inverse time.

The delta function represents a single spike at the origin. A spike train with spikes at times t_a can be written as a sum of delta functions,

$$\rho(t) = \sum_a \delta(t - t_a)$$

A standard way of estimating firing rate from a spike train is to convolve it with a

response function w

$$\begin{aligned}
\nu(t) &= \int dt\, w(t-t')\rho(t') && (3)\\
&= \int dt\, w(t-t') \sum_a \delta(t'-t_a) && (4)\\
&= \sum_a \int dt\, w(t-t')\delta(t'-t_a) && (5)\\
&= \sum_a w(t-t_a) && (6)
\end{aligned}$$

So the convolution simply adds up copies of the response function centered around the spike times. Note that it's important to choose a kernel satisfying

$$\int dt\, w(t) = 1$$

so that

$$\int dt\, \nu(t) = \int dt\, \rho(t)$$

Since the Dirac delta function has dimensions of inverse time, smoothing $\rho(t)$ results in an estimate of firing rate. In contrast, the discrete spike train ρ_i is dimensionless, so smoothing it results in an estimate of probability of firing. You can think of $\rho(t)$ as the $\Delta t \to 0$ limit of $\rho_i/\Delta t$.

6 Low-pass filter

To see the convolution in action, consider the differential equation

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

This is an equation for a low-pass filter with time constant τ. Given a signal h, the output of the filter is a signal x that is smoothed over the time scale τ. The solution can be written as the convolution $x = g * h$, where the "impulse response function" g is defined as

$$g(t) = \frac{1}{\tau} e^{-t/\tau} \theta(t)$$

and we have defined the Heaviside step function $\theta(t)$, which is zero for all negative time and one for all positive time. The response function g is zero for all negative time, jumps to a nonzero value at time zero, and then decays exponentially for positive time.

To construct the function x, the convolution places a copy of the response function $g(t-t')$ at every time t'. Each copy gets weighted by $h(t')$, and they are all summed to obtain $x(t)$. The response function is sometimes called the kernel of the convolution.

To see another application of the delta function, note that the impulse response function for the low-pass filter satisfies the differential equation

$$\tau \frac{dg}{dt} + g = \delta(t)$$

In other words, g is the response to driving the low-pass filter with an "impulse" $\delta(t)$, which is why it's called the impulse response.

7 Correlation

The correlation is defined as

$$\mathrm{Corr}[g,h](t) = \int_{-\infty}^{\infty} dt' g(t')h(t+t')$$

This compares g and h at times separated by the lag t.[1] Note that $\mathrm{Corr}[g,h](t) = \mathrm{Corr}[h,g](-t)$, so that the correlation operation is not commutative.

As before, if g and h are only defined on the interval $[0,T]$, they can be extended by defining them to be zero outside the interval. Then the above definition is equivalent to

$$\mathrm{Corr}[g,h](t) = \int_{0}^{T} dt' g(t')h(t+t')$$

This is the unnormalized version of the correlation. In the Dayan and Abbott textbook, $Q_{gh}(t) = (1/T)\,\mathrm{Corr}[g,h](t)$, which is the normalized correlation.

8 The spike-triggered average

Dayan and Abbott define the spike-triggered average of the stimulus as the average value of the stimulus at time τ *before* a spike,

$$C(\tau) = \frac{1}{n}\sum_a s(t_a - \tau)$$

where n is the number of spikes. Then in Figure 1.9 they plot $C(\tau)$ with the positive τ axis pointing left. This sign convention may be standard, but it is certainly confusing.

Exactly the same graph would be produced by the alternative convention of taking $C(\tau)$ to be the average value of the stimulus at time τ *after* a spike, and plotting it with the positive τ axis pointing right. Note that in this convention, $C(\tau)$ would have the

[1] The expression above is the definition used in the Dayan and Abbott book, but take note that the opposite convention is used in other books like *Numerical Recipes*, which call the above integral $\mathrm{Corr}[h,g](t)$.

same shape as the cross-correlation of ρ and s,

$$\begin{aligned} \mathrm{Corr}[\rho, s](\tau) &= \int dt\, \rho(t) s(t+\tau) && (7) \\ &= \int dt \sum_a \delta(t - t_a) s(t+\tau) && (8) \\ &= \sum_a s(t_a + \tau) && (9) \end{aligned}$$

9 Visual images

So far we've discussed situations where the neural response encodes a single time-varying scalar variable. In the case of visual images, the stimulus is a function of space as well as time. This means that a more complex linear model is necessary for modeling the relationship between stimulus and response. Let the stimulus be denoted by s_i^{ab}, where the indices a and b specify pixel location in the two-dimensional image. Construct $x_i^{ab} = s_i^{ab} - \langle s^{ab} \rangle$ by subtracting out the pixel means. Similarly, let y_i denote the neural response with the mean subtracted out. Then consider the linear model

$$y_i \approx \sum_{jab} h_j^{ab} x_{i-j}^{ab}$$

We won't derive the Wiener-Hopf equations for this case, as the indices get messy.

But for white noise the optimal filter is given by the cross correlation

$$h_j^{ab} \propto \sum_i x_i^{ab} y_{i+j}$$

White noise is defined so that

$$\frac{1}{m} \sum_{i=1}^{m} x_i^{ab} x_{i+j}^{cd} = \sigma_x^2 \delta_j \delta_{ac} \delta_{bd}$$