Kernels, Spike-Triggered-Averages, Convolution and Cross-Correlation

I’ve spent some time doing tests with Matlab, and I think I understand the issues.  Currently I’m not convinced a convolution is what we really want. I’ve attached a quick-and-dirty Matlab script and its PDF output to show what I’ve found about aligning time functions.

Each first row in the plot (testConv.pdf) shows a fake kernel, a fake spike-triggered average (a negative delta for simplicity) and their convolution.  The first row just shows that if you use Matlab’s conv(kernel, start) you get a function that is as long as the sum of the kernel and STA lengths (800 + 500 = 1300).  The STA length is effectively pre-pended to the convolution, so the kernel peak is delayed by the STA timing in the convolution (350 ms for this delta). (The small vertical steps on the convolution baseline happen because I offset the kernel slight from zero so we could see where the convolution is valid.).

The second row shows only that if you use the ’same’ shape argument for the convolution, (conv(kernel, sta, ’same’)), Matlab clips out a section of the middle that has the same length as the kernel.  In this case it removes 250 from each end (500/2), so the peak that was at 849 is now at 599). So far, so good.

But the third row shows that the convolution peak time depends critically on how we frame the STA (or, equivalently, the kernel) . If we offset our STA clip slightly in time, the convolution peak moves.  You might think we just need to be systematic and keep t=0 centered for the kernel and STA, but it gets worse.

The fourth row shows if you keep t=0 fixed and get a STA that peaks 50 ms later, the convolution peak occurs 50 ms later.  Superficially it seems OK, but in fact it’s backward.  An STA peak at 0 implies an instantaneous response (no delay).  An STA peak at -50 implies that spikes are determined by what happened 50 ms in the past (a 50 ms delay).  So moving the STA peak 50 ms back (left) in time should move the response 50 ms forward (right, delayed) in time.

(The fifth row is just a cautionary note that the order of the kernel and STA determines which function sets the convolution length when you use the ’same’ shape parameter.)

I think I know why the convolution shifts in the wrong direction.  A convolution effectively reverses one function in time before performing the integration (e.g., see <https://en.wikipedia.org/wiki/Convolution>).  This is necessary, because it gets the temporal interactions correct. But we don’t what that.  We want the STA to effectively be the impulse response function for the neurons (which is essentially what an STA is). Then we want to use that impulse response to predict the response to an arbitrary stimulus (the kernel, in this case).  I’m not certain, but I think what we want is a cross-correlation, not a convolution.  But I need to think about it some more.

We definitely want to use a cross-correlation, not a convolution.  I think I’ve got a handle on it now.

The PDF and Matlab script I’ve attached here extend what I sent yesterday. Nothing has changed on the first PDF page (convolution), except I made the kernel asymmetric.  That is irrelevant for the convolution, but is useful for the cross-correlation.

The second page of the PDF closely follows the format of the first page, except for cross-correlation. The first row is a cross-correlation between the baseline kernel and STA.  Unlike the convolution, the cross correlation has a length that is twice as long as the longer function (kernel, in this case: 800 \* 2 = 1600.  Points in the shorter function are correlated with every point in the longer function both forward and backward in time, so the cross-correlation extends from -kernel-length to +kernel length.  Points beyond -STA-length and +STA-length aren’t valid.  There is a ‘maxlag’ argument that can clip the invalid parts of the function.  Row 2 uses that argument to clip the result in row 1 as a demonstration, but I haven’t used it in later rows.

The cross-correlation peak in the first row is at 851 ms.  Zero time for the cross-correlation is at the middle of the function (400 ms), so this peak is at t = +50ms.  xcorr() considers the input functions (kernel and STA) to be aligned at their start (labelled -400 ms for both the kernel and STA).  The kernel peaks 400 ms later, the STA peaks 350 ms later. The 50 ms offset in their peaks shifts the cross-correlation 50 ms from t = 0 (bin 800), so the peak in the cross-correlation is at 851.

In row 3 I aligned the kernel and STA peaks in cross-correlation time by putting them both 400 ms from their respective starts. With the two peaks aligned, the peak in the cross-correlation is at t = 0 (bin 800).

Row 4 shows that if instead we move the STA peak to 300 ms after the start, then the cross-correlation peak is shifted later, to 900.  This is correct: the STA peak moving earlier means that spikes are influenced by things that happened earlier in time (i.e., spikes are delayed relative to the stimulus), so the cross-correlation is delayed to 100 ms.  Rows 1, 3 and 4 show that the timing of the cross-correlation works intuitively.  If we align the kernel and STA properly, we can think of t=0 (middle bin in the cross-correlation as the start of the visual stimulus).

Again, row 5 is cautionary.  The order of the two input functions can flip time in the cross-correlation.  We want the kernel as the first argument.

The summary is that  we should use cross-correlation, not convolution, and that it is critical that the kernel and STA clips be aligned in time.  For example, in Jackson’s latest example, the kernel begins at -400 relative to stimon, but at -350 relative to spike time. If we want to relate spike time to stimon, we need to align these two (perhaps adding 50 ms of zero padding onto the front of the STA.

