Byron Yu
Carnegie Mellon University
CoSMo 2012 August 15-16, 2012

Brain-machine interfaces: Matlab exercises

We will implement Kalman filter and apply it to multi-electrode data recorded in motor cortex. Please download the file `Stevenson_2011/Stevenson_2011_e1.mat` from the DREAM website `http://crcns.org/data-sets/movements/dream`.

1. Data preprocessing

   The first step is to prepare the arm state vectors and take binned spike counts. There are two subjects in the dataset. Let's start with `Subject(1)`.

   (a) We will use a 4-dimensional arm state $\mathbf{z}_t$, where

   $$\mathbf{z}_t = \begin{bmatrix} \text{horz position} \\ \text{vert position} \\ \text{horz velocity} \\ \text{vert velocity} \end{bmatrix}.$$

   For each trial, find the time series $\mathbf{z}_t$, where $t = 1, \ldots, T$ indexes timesteps of 20 ms. Because position and velocity are already sampled in 10 ms intervals, this amounts to simply taking every other value in `Subject(1).Trial(n).HandPos` and `HandVel`. The sampling times are listed in `Subject(1).Trial(n).Time`, whose values are in units of seconds. For each trial, the result should be a $4 \times T$ matrix, where T is the number of 20 ms timesteps on that trial.

   (b) For each trial, take spike counts in non-overlapping 20 ms bins. A list of spike times for unit `k` on trial `n` can be found in `Subject(1).Trial(n).Neuron(k).Spike`. The kinematic sampling times in (a) can be used as spike count bin boundaries here. Note that the spike times are recorded on the same clock as the times shown in `Subject(1).Trial(n).Time`. For each trial, the result should be a $196 \times T$ matrix, where T is the number of 20 ms time bins on that trial.

   (c) For each unit, find its overall mean firing rate collapsed over all reach targets. Remove any units with overall mean firing rate less than 1 spike/sec, since they can make the decoder unstable (due to many 0 spike counts) and probably don't contribute much to the decoded trajectory anyway.

   (d) Because motor cortex drives the arm movements (i.e., it's a physical system with conduction and processing delays), we expect there to be a lag between neural activity in motor cortex and arm movements. Previous studies suggest that this lag is approximately 150 ms, but this value may vary for different units. For simplicity here, let's assume a 7 timestep (i.e., 140 ms) lag for all units. To introduce this lag, simply remove 7 timesteps of arm kinematics at the beginning

of each trial and 7 timesteps of neural activity at the end of each trial. If you have time at the very end of the tutorial session, you can examine how decoding performance varies with the lag.

(e) Divide the 194 trials into two halves: one for training and the other for testing.

2. Training phase

   Fit the model parameters $\theta = \{A, Q, \boldsymbol{\pi}, V, C, R\}$ to the training data. Show the values in the 4×4 matrices A and Q. (Hint: The relationship between position and velocity defined by A should be consistent with the laws of physics.)

3. Test phase

   Using the model parameters found in Problem 2, apply the Kalman filter to decode an arm trajectory for each test trial. We will create two plots, one for each of the first two test trials. In each plot, show:

   - the mean position estimate $\boldsymbol{\mu}_t^t$ $(t = 1, \ldots, T)$ as red points connected by a red line (note: $\boldsymbol{\mu}_t^t$ is a 4-dimensional vector that includes both position and velocity estimates, but you can simply ignore the velocity estimate here),
   - the one-standard-deviation confidence ellipse corresponding to the uncertainty Σ_t^t of each position estimate in red, and
   - the actual arm trajectory as a black line.

4. We will plot the same data as in Problem 3, but now versus time. For *each* of the two trials, create a two-panel plot, where one panel shows horizontal position versus time, and the other panel shows vertical position versus time. In each panel, show:

   - the decoded mean trajectory as a red solid line,
   - the one-standard-deviation confidence intervals using red dotted lines (note: there should be one dotted line above and another below the decoded mean trajectory), and
   - the actual arm trajectory as a black line.

5. Performance evaluation

   At each timepoint, find the distance (in mm) between the decoded position and actual position. Average these distance errors across the timepoints of all test trials, then report this average error.