BE 521: Homework 7

Motor Prediction

Spring 2015

54 points

Due: Thursday, 4/02/2015 11:59 PM

Objective: Predict position and angle of limb using EEG recordings of neurons.

Homework Policy

1. Piazza should be used for peer discussion for all questions related to course material. Please also use Piazza to contact teaching staff for all questions. TA's will be available to help during office hours and occasionally on Piazza.
2. Submit LaTeX write-up (pdf) and Matlab code to Canvas as pennkey_hwx.pdf, .m before listed deadline.
3. Assignments will be returned electronically on Canvas.
4. Collaboration is encouraged but individual write-ups are required. Please list any collaborators. Honor code will be strictly enforced. Note: submitted code is routinely passed through a plagiarism checker.
5. Late Policy: 5% per day. **No homework is accepted after the 5th late day.** (e.g. If originally due Tuesday, 11:59PM, last day to turn in is Sunday, 11:59 PM).

Motor Planning

One of the oldest paradigms of BCI research is motor planning: predicting the movement of a limb using recordings from an ensemble of cells involved in motor control (usually in primary motor cortex, often called M1). The data from this experiment comes from a macaque monkey holding a pen on a digitizing tablet (so that its X and Y location can be recorded). As the monkey moves its arm, simultaneous recordings from 40 neurons capture the spikes of each neuron over time.

The data in `I521_A0007_D001` contains the number of spikes fired in time bins of 70-ms for each of the 40 neurons. The data in `I521_A0007_D002` contains `X`, the horizontal position of the monkey's hand, and `Y`, the vertical position of the monkey's hand.

Optimal Linear Decoder

You will use the *optimal linear decoder* method in this homework as described in Warland et al., 1997. We will recapitulate the method in this section[1], but consult the paper for more details. This

[1]We have changed minor notation in a few places for clarity. Our subscripts start at 1, theirs at 0. We have also distinguished between the index j of the neurons and the total number of neurons ν.

paradigm involves using the firing rates of the ν cells (for us, $\nu = 40$ neurons) from the N time bins before (for us, $N = 20$, including the current time bin) to predict the vertical and horizontal position of the hand at a given time bin. The position data is captured for M (for us, $M = 1972$) total time bins.

You will use the responses (i.e. the number of spikes) of all the cells over a given number of time bins before the current one to make the prediction. The approach that Warland et al., 1997 (and we will) take is to construct a row vector containing spike counts for all the neurons over the relevant N time bins (thus, there will be a good amount of redundancy between row vectors of adjacent time bins, but that's ok).

Let r_i^j be the the number of spikes fired by neuron j in time bin i. Let the response matrix $\mathbf{R}$ be defined as

$$\mathbf{R} = \begin{bmatrix} 1 & r_1^1 & r_2^1 & \dots & r_N^1 & r_1^2 & r_2^2 & \dots & r_N^2 & \dots & \dots & r_1^\nu & r_2^\nu & \dots & r_N^\nu \\ 1 & r_2^1 & r_3^1 & \dots & r_{N+1}^1 & r_2^2 & r_3^2 & \dots & r_{N+1}^2 & \dots & \dots & r_2^\nu & r_3^\nu & \dots & r_{N+1}^\nu \\ 1 & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & r_M^1 & r_{M+1}^1 & \dots & r_{N+M-1}^1 & r_M^2 & r_{M+1}^2 & \dots & r_{N+M-1}^2 & \dots & \dots & r_M^\nu & r_{M+1}^\nu & \dots & r_{N+M-1}^\nu \end{bmatrix}$$

This is also referred to as the design or feature matrix, with each column being a predictor, or feature. The column of 1's accounts for the intercept term in linear regression/decoding. Make sure you understand what this matrix means before moving on. Note that for our 40 neurons, each row of R looks like

$$[1 \quad \text{neuron 1 rates} \quad \text{neuron 2 rates} \quad \dots \quad \text{neuron 40 rates}]$$

We denote the target[2] vector[3] (e.g. the X or Y position of the hand) as $\mathbf{s}$ and the reconstruction (e.g. the predicted X or Y position) as $\mathbf{u}$. Solving for the minimum least-squares difference between the stimulus and reconstruction, $(\mathbf{s} - \mathbf{u})^T(\mathbf{s} - \mathbf{u})$, we get the analytic form for the optimal filter,

$$\mathbf{f} = (\mathbf{R}^T\mathbf{R})^{-1}(\mathbf{R}^T\mathbf{s}) \tag{1}$$

This equation should take a familiar form. Warland et al., 1997 don't refer to it as such, but this is *exactly* the same as linear regression, one of the most commonly used algorithms in practical machine learning. Not only is this algorithm remarkable powerful, but it has a beautiful analytic form for learning the "weights" (here, the $\mathbf{f}$ vector), a rarity in a field where almost all optimizations involve some sort of iterative algorithm.

After learning the filter weights $\mathbf{f}$, we can calculate the optimal predictions as

$$\mathbf{u} = \mathbf{R}\mathbf{f} \tag{2}$$

1 Motor Prediction (20 pts)

In this section, you will explore some of the basics of motor planning prediction using an optimal linear decoder.

[2] In Warland et al., 1997, this quantity is referred to as the stimulus vector since they are talking about decoding the stimulus from neural data after it. We, on the other hand, are trying to decode the stimulus using data before it, but we can conveniently use the same method.

[3] It is generally assumed here and in most other places that vectors are column-vectors unless otherwise specified.

1. For our set of 40 neurons and set of 1972 X and Y position values, what would the size of the $\mathbf{R}$ matrix be if we use all 20 time bins? (1 pt)

2. Compute the response matrix $\mathbf{R}$ and verify that the quantity `mean(mean(R))` is 1.5096. (Note that you should be getting values $[1 : (N + M - 1)]$ for each neuron, although the displayed vector size is a few points fewer). (4 pts)

3. Calculate the linear filter (i.e. the weights vector) $\mathbf{f}$ as defined by Equation 1 for the X position and then for the Y position. (N.B., instead of using the matrix inverse of $\mathbf{R}^T\mathbf{R}$, use Matlab `mldivide` function, which is generally more stable.)

   (a) Reshape the weights, excluding the first (bias) weight, into a 40×20 matrix and show it with `imagesc` with time bins on the x-axis and different cells on the y-axis. Make sure to include a `colorbar` with each image. (4 pts)

   (b) In four or five sentences, comment on what conclusions you can draw from the filter weights. (4 pts)

4. Use Equation 2 to calculate the predicted X and Y positions over the 1972 time points.

   (a) Use `subplot` to plot the actual and predicted positions on top of each other for the X position (top plot) and Y position (bottom plot). (3 pts)

   (b) Calculate the correlation coefficient ρ for the X predictions and the Y predictions separately. (Hint: see Matlab's `corr` function, but only give a scalar value for X and a scalar value for Y.) (2 pts)

5. Create a "movie" of the actual position and predicted position by plotting two points (e.g. a blue dot and a red circle) in 2D space over time. Use a `for` loop to iterate over the time bins and use `pause(0.035)` to pause for 35 ms between each plot (so the "video" will be at 2x speed). In each frame limit your X and Y ranges to $[8000, 12000]$ and $[12000, 16000]$, respectively.

   You can't include this "movie" in your report, but be sure to include the code (which can be commented out, if that's easier for publishing) you use to generate it. (2 pts)

2 A More Realistic Setting (14 pts)

In this section, you will explore a few modifications that will make the decoding paradigm a bit more realistic for a real-world application.

1. Set aside the first 400 rows of the $\mathbf{R}$ matrix as a testing set, and use the rest to re-estimate your X and Y filters. What are your new correlations for X and Y? Comment on differences compared to correlations from section 1 (3 pts)

2. Suppose we wanted to develop an implantable device that uses wireless telemetry to transmit data to some external output (e.g. a cursor on a screen or perhaps even a robotic arm). Transmitting lots of data costs in both power and transmission speed, so we logically might want to reduce the amount of data we transfer "out" to as little as possible, while still maintaining a desirable level of performance. In addition to this benefit, remember that as more

features are added to a model, complexity increases and overfitting may occur. Thus, feature reduction can also improve generalizability.

In this question, you will eliminate columns, or features, from the $\mathbf{R}$ matrix. There are many methods that can be used for this purpose. Some of these include principal component analysis (PCA), forward/backward/stepwise selection, and penalized least squares. We will focus on one useful penalized least squares method: the least absolute shrinkage and selection operator (LASSO) [4].

Recall from section 1 that optimization of linear regression minimizes $(\mathbf{s}-\mathbf{u})^\mathbf{2} = (\mathbf{s}-\mathbf{Rf})^\mathbf{2}$. In LASSO, the **objective function** minimizes $(\mathbf{s}-\mathbf{Rf})^\mathbf{2} + \lambda|\mathbf{f}|_\mathbf{1}$. The $|\mathbf{f}|_\mathbf{1}$ term is the L_1 norm, which expands to $\sum_i^p |f_i|$. Thus, LASSO minimization favors weights (`f`) close to 0 because it minimizes the objective function and as a result tends to reduce the number of non-zero values in the $\mathbf{f}$ vector (from a Bayesian perspective, it places a Laplace prior on the weights).

(a) Using the same training and test sets as in question 1 of this section, train a model to predict the X position using Matlab's `lasso` command with the default parameters. It should return 100 different models. Plot the number of non-zero feature weights in your models on the X-axis versus the corresponding test correlations for your test set on the Y-axis. (7 pts)

(b) What determines the number of zeroed feature weights? Explain. (4 pts)

3 Putting your skills to the test (20 pts)

For this section, your task is to predict the angle of a limb based on recordings from the dorsal root ganglion (DRG). Recall that in the spinal cord, afferent sensory pathways from our limbs traverse through the dorsal root ganglion (part of the PNS), where it synapses and travels up the dorsal part of the spinal cord to the brain (CNS). Some of this data is responsible for proprioception, which tells us where our limb is in space. The data from this section contains recordings from a Utah Array implanted in the dorsal root ganglion of a cat as the hind limb is moved by an experimenter. The goal is to predict limb movement based on recordings from the DRG.

The task of clustering the spikes into different cells, as you did in HW6, has been done for you. 146 different cells and their spiking activity have been extracted and presented. Your job is to build an algorithm that uses the spiking activity to predict limb position - specifically the x-, y-, and z-angle of the limb in three dimensional space.

Dataset (`HW7Data.mat`)

- `time` : time (in seconds)
- `angles` : corresponding angles of limb (one dimension in each column) recorded over time. Column 1 = x, 2 = y, 3 = z.

[4] Regression Shrinkage and Selection via the Lasso, Tibshirani 1996

- `count` : cell average spiking activity/time bin for each cell corresponding to the given angles.
- `test_count` : spiking activity for the test set.

Rules

Build your algorithm using the training data provided, that is, start with the cell spiking activity and build a model(s) that predicts the limb x-, y-, and z- angles. Then, run your algorithm on the spiking activity provided in `test_count`. Submit a .mat file consisting of the variable `predicted_angles` which is a 594 x 3 matrix, where predict x, y, and z positions are in columns 1,2, and 3, respectively. Performance will be measured by the correlation between your `predicted_angles` and the true angles, calculated with the `corr` command in Matlab. For full credit you must meet performance correlations of 0.89 for each angle.

- There are no restrictions on how you create your predicted angles, besides that it must involve training model(s) on the training data. You must justify your steps and algorithm(s).
- Be sure to submit your .m, .pdf, and .mat to Canvas before the deadline.
- You may work in teams of up to 3 people. List your collaborators.
- 12 points for the algorithm and its description.
- 8 points for meeting performance criteria
- an additional 7, 5, or 3 points (extra credit) for the first, second, and third highest total correlations.