EE 486E: Homework 3

Spring 2015

39 points

Due: WED 18 FEB

Directions for submission

Turn in a paper copy of your writeup at the beginning of class on the due date. Your writeup should consist of typed, numbered answers to each question followed by an addendum containing only the **PLOTS** requested in all-caps bold red font, followed by your code. Include two plots per page, each with a title indicating which problem it answers and axes labels that have proper units. In addition, upload your *Matlab code only* to Google Drive (`EE486E_CLASS\EE486E_DROPBOX\<YOURNAME>_EE486E_DROPBOX`). Your code may be split into more than one m-file but it must have a single main file, `RunHw<X>_<YOURNAME>.m`, that calls all of the others.

Motor Prediction

One of the oldest paradigms in BMI research involves predicting the movement of a limb using recordings from an ensemble of cells involved in motor control (usually in primary motor cortex, often called area M1). The data set in this assignment comes form a macaque monkey holding a pen on a digitizing tablet so that the arm's 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 `motorData15.mat` contains three variables: `spikes`, the number of spikes that each of the 40 neurons (columns) fires in each 70-ms time bin (rows), `X`, the horizontal position of the monkey's hand, and `Y`, the vertical position of the monkey's hand.

Optimal Linear Decoder

In this homework, you will use the *optimal linear decoder* method described by Warland et al.[1] We will recapitulate the method in this section[2] but you may wish to consult the paper for more details. The algorithm involves using the spike rates of ν neurons (for us, $\nu = 40$) during N time bins prior to and including a given time bin (for us, $N = 20$) to predict the vertical and horizontal position of the hand for that time bin. Time bin indices run from 1 to M (for us, $M = 1991$) but notice that positional data are only available for the *last* 1972 time bins. The first 19 elements of

[1] Warland DK, Reinagel P, and Meister M. 1997. Decoding visual information from a population of retinal ganglion cells. *Journal of Neurophysiology*. 78:2336-2350.

[2] Note, however, that we have changed notation in several places: Our spike matrix is $\mathbf{S}$ rather than $\mathbf{R}$. Our subscripts start at 1, theirs at 0. We have distinguished between the index j of each neuron and the total number of neurons ν. Our last time bin has index M, theirs N + M - 2.

X and Y are `NaN` since the 20th time bin is the first for which a valid prediction can be made given the available spike data.

You will use the responses (i.e., the number of spikes) of *all* the cells over the N time bins preceding and including a given time bin to make an X or Y positional prediction for that time bin. The approach that Warland et al. (and we will) take is to arrange these responses in a matrix with the following form:

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

$$\mathbf{S} = \begin{bmatrix} 1 & s_1^1 & s_2^1 & \dots & s_N^1 & s_1^2 & s_2^2 & \dots & s_N^2 & \dots & \dots & s_1^\nu & s_2^\nu & \dots & s_N^\nu \\ 1 & s_2^1 & s_3^1 & \dots & s_{N+1}^1 & s_2^2 & s_3^2 & \dots & s_{N+1}^2 & \dots & \dots & s_2^\nu & s_3^\nu & \dots & s_{N+1}^\nu \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & s_{M-N+1}^1 & s_{M-N+2}^1 & \dots & s_M^1 & s_{M-N+1}^2 & s_{M-N+2}^2 & \dots & s_M^2 & \dots & \dots & s_{M-N+1}^\nu & s_{M-N+2}^\nu & \dots & s_M^\nu \end{bmatrix}$$

Make sure you understand what this matrix means before moving on. Note that for our 40 neurons, each row of $\mathbf{S}$ looks like

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

We denote the true X (or Y) position M-vector for the hand as $\mathbf{p}$ and the predicted position M-vector as $\hat{\mathbf{p}}$. The residual sum of squares (RSS) between the true and predicted positions is given by $(\mathbf{p} - \mathbf{S}\hat{\beta})^T(\mathbf{p} - \mathbf{S}\hat{\beta})$. Minimizing the RSS with respect to the unknown weights vector, $\hat{\beta}$, leads to:

$$\hat{\beta} = (\mathbf{S}^T\mathbf{S})^{-1}\mathbf{S}^T\mathbf{p} \tag{1}$$

Although Warland et al. don't refer to it as such, this is *exactly* the same as linear regression, one of the most commonly used algorithms in practical machine learning. Not only is this algorithm remarkably powerful, but it has a beautiful analytic form for estimating the model parameters (here, the $\hat{\beta}$ vector), a rarity in a field where almost all optimizations involve some sort of iterative approach. After learning $\hat{\beta}$ we can calculate the predictions as:

$$\hat{\mathbf{p}} = \mathbf{S}\hat{\beta} \tag{2}$$

1 Motor Prediction

In this section, you will explore some of the basics of motor prediction using the optimal linear decoder method described above.

1. For our set of 40 neurons and 1972 X and Y position values, what will the size of the $\mathbf{S}$ matrix be if we use N = 20 time bins of spiking history for each neuron? (1 pt)

2. Compute the spike matrix $\mathbf{S}$ and verify that the quantity `mean(mean(S))` is 1.5096. (This is mostly to ensure you've calculated $\mathbf{S}$ correctly, without which the rest of the homework will not work). (4 pts)

3. Calculate the weights vector $\hat{\beta}$ as defined by Equation 1 for the X position and then for the Y position. (Note: remember that $(\mathbf{S}^T\mathbf{S})^{-1}$ represents a matrix inverse, not an element-wise reciprocal.)

   (a) Reshape the weights for X and Y, excluding the first "intercept" weight (sometimes called the bias term), into two 40×20 matrices. Using `imagesc`, **PLOT** these matrices in a 2×1 `subplot` (top plot X, bottom plot Y), putting time bins on the x-axis and the different neurons on the y-axis. Make sure to include a `colorbar` with each image. (4 pts)

   (b) Describe the *temporal* patterns (if any) in the two weights images. Are they similar or different? Explain in a few sentences. (3 pts)

   (c) What conclusions can you draw about the cells involved in predicting the X position versus the Y position? (2 pts)

4. Use Equation 2 to calculate the predicted X and Y positions over the 1972 time bins for which position data are available.

   (a) In a 2×1 `subplot`, **PLOT** the actual and predicted positions overlayed for the X (top plot) and Y (bottom plot) coordinates. Incude an appropriate legend in each plot. (3 pts)

   (b) For both the X and Y positions, calculate the R^2 statistic:

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=N}^{M}(p_i - \hat{p}_i)^2}{\sum_{i=N}^{M}(p_i - \overline{p})^2}$$

   with p_i and $\hat{p}_i$ the true and estimated X or Y position for time bin i and $\overline{p}$ the average X or Y position over all time bins. (2 pts)

5. Create a movie showing the evolution of the actual and predicted positions by plotting them as 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 `getframe` to store a snapshot of each plot as a movie frame. Then use `movie` to play back the result, using a frame rate of 28 frames per second so that the movie plays at approximately twice real-time speed. In each frame (i.e., plot), limit your X and Y ranges to $[8000, 12000]$ and $[12000, 16000]$, respectively. Include a **PLOT** of the last frame of the movie in your written report and give your observations about this movie in a few sentences. (4 pts)

2 A More Realistic Scenario

In this section, you will explore a few modifications that will make the decoding paradigm a bit more realistic with respect to real-world applications.

1. Hopefully you noticed that in Section 1, we committed a cardinal sin in machine learning: using training data to evaluate performance. For reasons you already know, this approach, while often useful for a rough first-pass assessment of performance, is not at all appropriate for a final performance assessment.

   Set aside the first 400 rows of the **S** matrix as a testing set, and use the remainder as a training set to re-learn your X and Y $\hat{\beta}$ vectors. What are your two new R^2 values (i.e., one for X, one for Y) for the training data? (4 pts)

2. What are your **test set** MSEs for X and Y (computed separately)? (2 pts)

3. Suppose we wanted to develop an implantable device that used wireless telemetry to transmit data out of the brain to an external device (e.g., to a computer in order to control a cursor or perhaps a robotic arm). Sending large amounts of data can carry significant costs both in terms of power and control delay, so it's often a design goal to minimize the amount of data transmitted while still maintaining a desirable level of performance.

   For this question, you will eliminate as many columns[3] as possible from the training data matrix while maintaining MSE values of no more than 60,000 square units for each of X and Y on the training data. Note that since in practice you're going to be transmitting just one set of spike rates (ultimately corresponding to a row of the **S** matrix), you cannot optimize separately for X and Y.

   Since each subject may be different, your data reduction technique must be generalizable across subjects. This means you should not hard-code specific columns to remove. Describe the rationale for your approach in a short paragraph, then summarize the specific details of your algorithm in an ordered list (e.g., Step 1, Step 2, Step 2.a, Step 2.b, etc.). (4 pts)

4. How many columns were you able to remove? (2 pts; +4 pts bonus for whoever has the largest number here)

5. What are your *training set* MSEs for X and Y (computed separately)? (2 pts)

6. What are your *test set* MSEs for X and Y (computed separately)? (2 pts)

7. In a 2×1 `subplot`, **PLOT** the actual and predicted positions overlayed for the X (top plot) and Y (bottom plot) coordinates. Incude an appropriate legend in each plot. (2 pts)

[3] In practice we might transmit data at every time step to ensure real-time operation, and in that case we would only be concerned at the transmitter about which *neurons* to use, but think of more generally reducing the columns of **S** as an optimization for the external decoding algorithm: fewer data points means faster algorithm execution.