Lab 9: Kalman filter

Introduction

In this lab, you will be modifying your ridge regression algorithm to use the L_1 norm along with creating a Kalman filter to predict movement from neural data and physics. Use the same processed data from Lab 8 (contdata.mat).

Similar to lab 8, this lab provides little guidance on code structure, leaving you to determine the most appropriate form of implementation. Note that many of these algorithms can be implemented very efficiently and concisely using matrix notation provided in lecture. For the Kalman filter code should be written from the ground up.

Software

This lab must be completed using MATLAB.

Part 1) LASSO

LASSO is a modified version of ridge regression that uses the L_1 norm instead of the L_2 norm, resulting the following objective function:

$$\min_{\vec{w},b}\left[\frac{1}{n}\sum_{i=1}^{n}(\vec{w}^T\vec{x}_i + b - y_i)^2 + \lambda\left|\left|\vec{w}\right|\right|_1^2\right]$$

Despite the apparently small change in the objective function, the LASSO algorithm results in a significantly different, although very similar, output, when compared to ridge regression.

1. Using MATLAB's built-in `lasso()` function and the best value of λ found in Lab 8 Part 2, train a decoder using LASSO and evaluate its performance using test data. Note that the lasso function can be quite slow in Matlab (takes a minute or so to run).

2. Examine the values of the weight matrix B (or w, in the objective function above). What do you notice about the values, when compared to the values of B yielded by ridge regression? Why might this be useful?

Part 2) Kalman filter

Note that linear regression, ridge regression, and LASSO all predict motion using neural data only. Simple physics, however, could also contribute to prediction of motion. For example, we know that if an object is moving left at time point t, then it *should* be to the left of its original location at time point $t + 1$. Kalman filters use a weighted combination of neural data (based on linear regression) and simple physics to predict motion in this way.

Train a Kalman filter on the training data and evaluate its performance on the test data. The relevant equations can be found in the lecture, and the steps are as follows:

1. Using the training data alone, fit the four parameters A, C, W, Q using the regression equations given in lecture. You will need to create two state vectors (i.e. X) that offset from each other by one timestep (i.e. the same X matrix data indexed from 1:N-1 and 2:N). **Hint:** You can check if your regressions are giving you significant correlation values similar to what you did for linear regression. Also, the values of A should make sense from the perspective of implementing physics. (I.e. the x position from one time step should depend on the x position from the previous timestep in a way that makes sense.)

2. After you have your parameters, implement your Kalman loop, which will loop through all of the timesteps of your test dataset. For the first timestep, you can set your starting X value to the first real test value, assuming that you have a known starting point. You can set your starting P value to P=W

3. Now loop through all timesteps carrying out the "Predict", "Innovate" and "Update" steps as describes in the lecture notes. The variables are explained throughout the lecture, but everything you need is actually on the last slide. For debugging, try carrying out these operations at the command line one by one.

4. Once it's working, it may take a few minutes to run. Evaluate the performance using the same metrics you used in Lab 7. How does its performance compare to the previous algorithms (including the algorithms from Lab 8)? Explain why you think it performed better or worse.

Guidelines for Lab Report (on Labs 7, 8, and 9 together)

Introduction: The introduction should be one paragraph long summarizing the motivation for developing the tools used in this lab and what they can be used for, along with a brief summary of everything you will show in this lab report.

Methods: From Lab 9, there should be methods paragraphs (and diagrams where necessary) on:

1. Assumptions of the algorithms used
2. How the algorithms were implemented

Include the code as an Appendix to your report. Cite sources for any values used in your models.

Results: You should include the following in your Results:

1. The outcome of each algorithm (correlation and mean squared error).
2. The limitations of each algorithm.
3. How the algorithms compare to each other.

Include all figures produced by MATLAB that could help explain and illustrate your findings.

Discussion: Should be ~2 paragraphs long describing what you could use these tools for in the future.

This report will be combined with Labs 7 and 8 to create one cohesive report. The report (not including Appendix) should be no longer than 6 pages. Use 12 pt. font and 1.15-1.5 line spacing. If your text is over the 6-page limit with figures, you can move your figures to an appendix section that goes beyond the 6-page limit. Please upload your report to Canvas.