(PSL) Coding Assignment 3

Fall 2023

Contents

Part I: Optimal span for LOESS

Objective

Write your own function to employ LOO-CV and GCV in selecting the optimal span for LOESS. Definitions of LOO-CV and GCV can be found on page 33 of [lec_W5_NonlinearRegression.pdf]

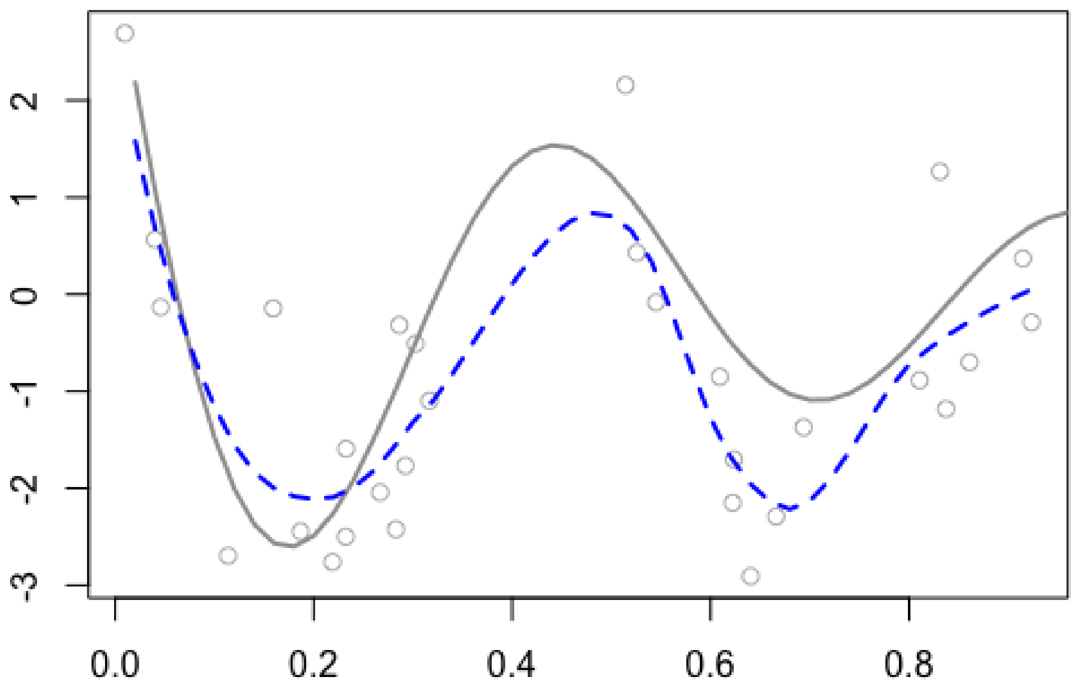
Task

1. Test your code using data set [Coding3_Data.csv]
2. Report your CV and GCV for the following 15 span values: 0.20, 0.25, . . . , 0.90.
3. Report the optimal span value(s) based on CV and GCV.
4. Display the original data points and overlay them with the true curve and the fitted curve(s) generated using the optimal span value(s).

The true curve is

$$f(x) = \frac{\sin(12(x + 0.2))}{x + 0.2}$$

Note: Your plot may differ from the one provided below. You're encouraged to use your own color schemes and styles.



Steps

- **Computing the Diagonal of the Smoother Matrix:** Create a function to retrieve the diagonal of the smoother matrix. We're only interested in the diagonal entries (which will be used in computing LOO-CV and GCV), so this function should return an n-by-1 vector.
  - **Inputs**: `x` (an n-by-1 feature vector) and `span` (a numerical value).
  - **Output**: n-by-1 vector representing the diagonal of the smoother matrix S.
  - **Tip**: Review the technique we used for the smoother matrix in smoothing spline models and adapt it for LOESS.
- **Span Value Iteration**:
  - Iterate over the specified span values.
  - For each span, calculate the CV and GCV values.
  - Post iteration, compile lists of CV and GCV values corresponding to each span.
  - Determine the best span based on the CV and GCV results. It's possible for both methods to recommend the same span.

Part II: Clustering time series

Objective

Cluster time series data based on their fluctuation patterns using natural cubic splines.

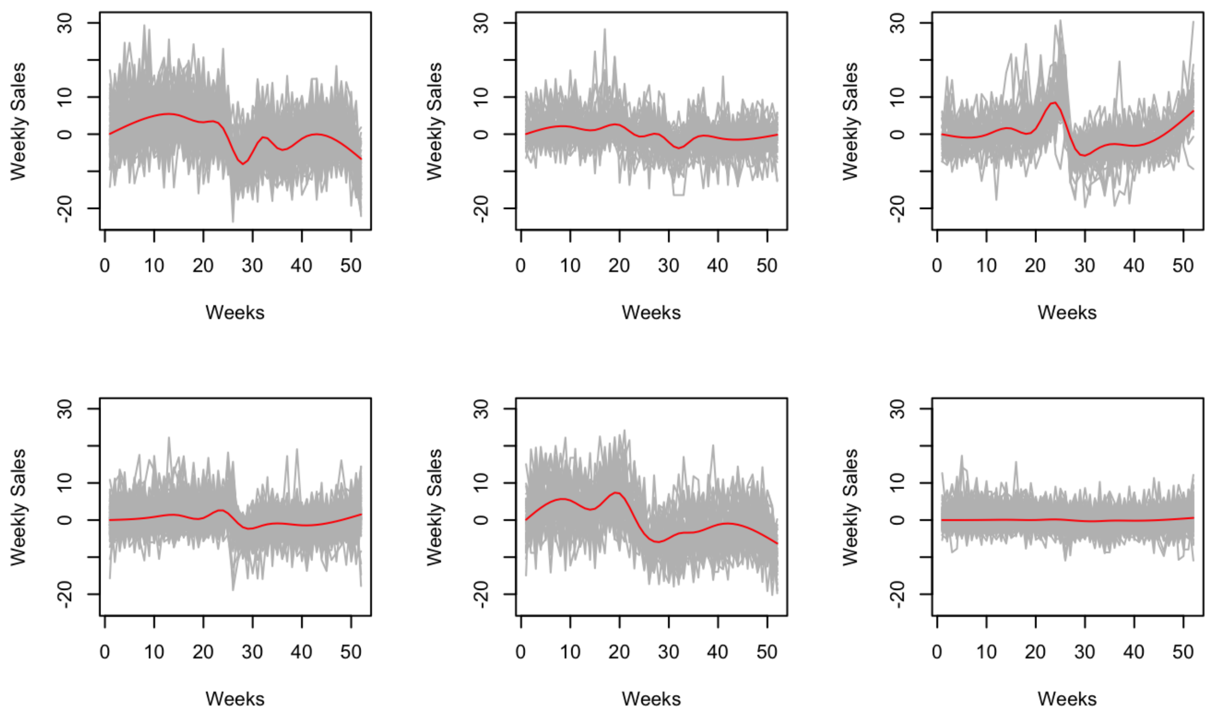
Data

Download the **'Sales_Transactions_Dataset_Weekly_datase'** from UCI Machine Learning Repository [Link]

- This dataset contains weekly purchased quantities of 811 products over 52 weeks, resulting in each product having a time series with 52 data points.
- Normalize each time series by **removing its mean**. Store the de-meaned data in an 811-by-52 matrix $\mathbf{X}$.

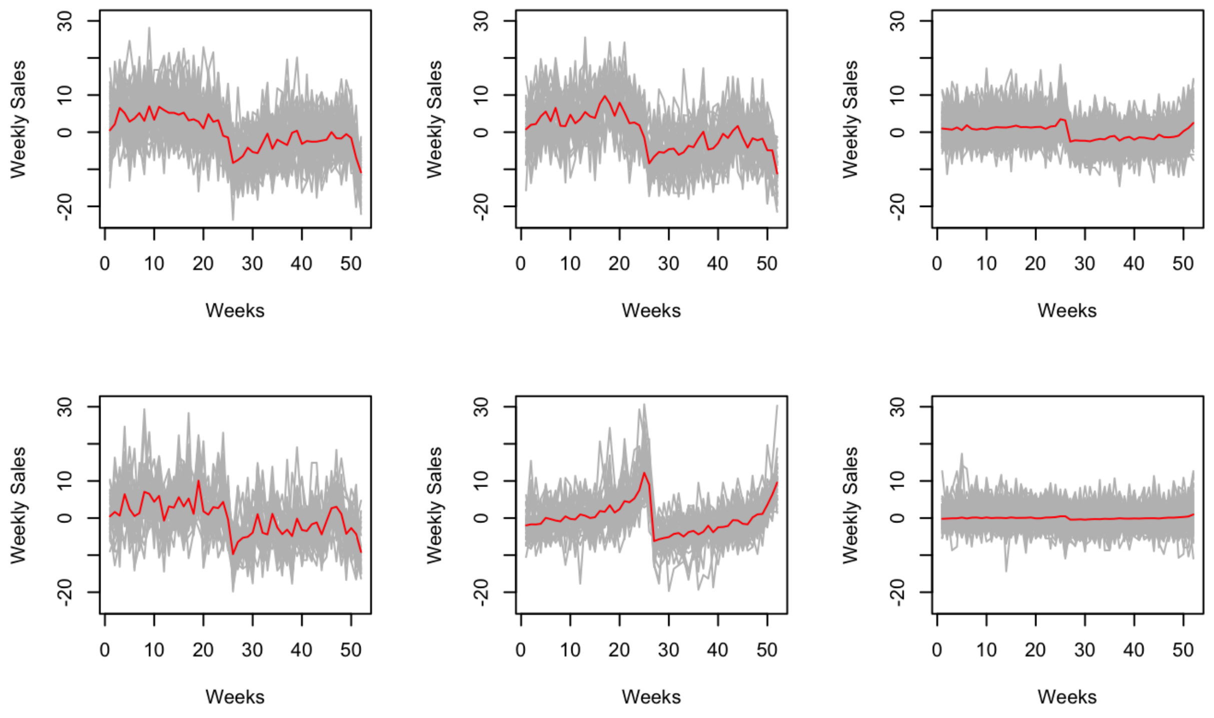
Task

1. **Fitting NCS**
   - Fit each time series with an NCS with `df = 10`. This corresponds to an NCS with 8 interior knots. Each row of $\mathbf{X}_{811\times52}$ represents the response, with the 1-dimensional feature being the index from 1 to 52. Row = purchase of one product
   - Store the NCS coefficients (excluding the intercept) in an 811-by-9 matrix $\mathbf{B}_{811\times9}$.
   - Matrix $\mathbf{B}$ can be derived as:
     - $\mathbf{F}$ is a 52-by-9 design matrix without the intercept. For instance, this can be obtained using the `ns` command in R. Remove the column mean from $\mathbf{F}$ to disregard the intercept.
     - The matrix equation is:
       $$\mathbf{B}^t = (\mathbf{F}^t\mathbf{F})^{-1}\mathbf{F}^t\mathbf{X}^t.$$
     - The formula above is given for the transpose of $\mathbf{B}$, since it is equivalent to fitting 811 linear regression models: the design matrix stays the same (i.e., $\mathbf{F}$) but the response vector — there are 811 response vectors corresponding to the 811 rows of $\mathbf{X}$ — would vary; each nine dimensional regression coefficient vector corresponds to a row in $\mathbf{B}$ (or equivalently, column in $\mathbf{B}^t$).
2. **Clustering using Matrix B**
   - Run the k-means algorithm on matrix $\mathbf{B}$ to cluster the 811 products into 6 clusters.
   - Visualize the **centered** time series (colored in grey) of products grouped by their clusters. Overlay the plots with the cluster centers (colored in red). Arrange the visualizations in a 2-by-3 format.
   - Note: When using matrix $\mathbf{B}$ for clustering, the centers are the average of the rows of $\mathbf{B}$ within each cluster. To get the corresponding time series from a cluster center $\mathbf{b}$, use the matrix product $\mathbf{Fb}$.



3. **Clustering using Matrix X**

- Run the k-means algorithm on matrix **X** to cluster the 811 products into 6 clusters.
- Similarly, visualize the **centered** time series of products grouped by their clusters, accompanied by their respective cluster centers. Arrange the visualizations in a 2-by-3 format.



Note: Your plots may differ from the examples provided above. You're encouraged to use your own color schemes and styles to visualize the time series.

Learning Outcome

This assignment provides insights into how local polynomials can be leveraged to derive meaningful descriptions of time series or curves.

Converting the raw data from matrix **X** to **B** acts as a dimension reduction method, particularly beneficial for functional data where each entry represents a curve or sequence observed at discrete points.

One key observation is that the cluster centers derived from Step 2 are smoother than those from Step 3.

Part III: Ridgeless and double descent

Objective

So far in our course, we've utilized the U-shaped bias-variance trade-off curve as a pivotal tool for model selection. This has aided us in methodologies such as ridge/lasso regression, tree pruning, and smoothing splines, among others.

A key observation is that when a model interpolates training data to the extent that the Residual Sum of Squares (RSS) equals zero, it's typically a red flag signaling overfitting. Such models are anticipated to perform inadequately when presented with new, unseen data.

> However, in modern practice, very rich models such as neural networks are trained to exactly fit (i.e., interpolate) the data. Classically, such models would be considered overfitted, and yet they often obtain high accuracy on test data. This apparent contradiction has raised questions about the mathematical foundations of machine learning and their relevance to practitioners. (Belkin et al. 2019)

In this assignment, we will use **Ridgeless** to illustrate the **double descent** phenomenon. Our setup is similar to, but not the same as, Section 8 in Hastie (2020).

Data

Remember the dataset used in Coding 2 Part I? It consisted of 506 rows (i.e., $n = 506$) and 14 columns: Y, X_1 through X_{13}.

Based on this dataset, we have formed Coding3_dataH.csv, which is structured as follows:

- It contains 506 rows, corresponding to $n = 506$.
- There are 241 columns in total. The first column represents Y. The subsequent 240 columns relate to the NCS basis functions for each of the 13 X variables. The number of knots are individually determined for each feature.

Task 1: Ridgeless Function

Ridgeless least squares can be equated with principal component regression (PCR) when all principal components are employed. For our simulation study, we'll employ the PCR version with the `scale = FALSE` option, implying that we'll center each column of the design matrix from the training data without scaling.

Your task is to write a function that accepts training and test datasets and returns the training and test errors of the ridgeless estimator. For both datasets, the initial column represents the response vector Y.

- You can use R/Python packages or built-in functions for PCA/SVD, but you are not allowed to use packages or functions tailored for linear regression, PCR, or ridge regression.
- Post PCA/SVD, you'll notice that the updated design matrix comprises orthogonal columns. This allows for the calculation of least squares coefficients through simple matrix multiplication, eliminating the need for matrix inversion.
- For computation stability, you need to exclude directions with extremely small eigenvalues (in PCA) or singular values (in SVD). As a reference, consider setting `eps = 1e-10` as the threshold for singular values.
- Although training errors aren't a requisite for our simulation, I recommend including them in the `ridgeless` output. This serves as a useful debugging tool. Ideally, your training error should align with the RSS derived from a standard linear regression model.

Task 2: Simulation study

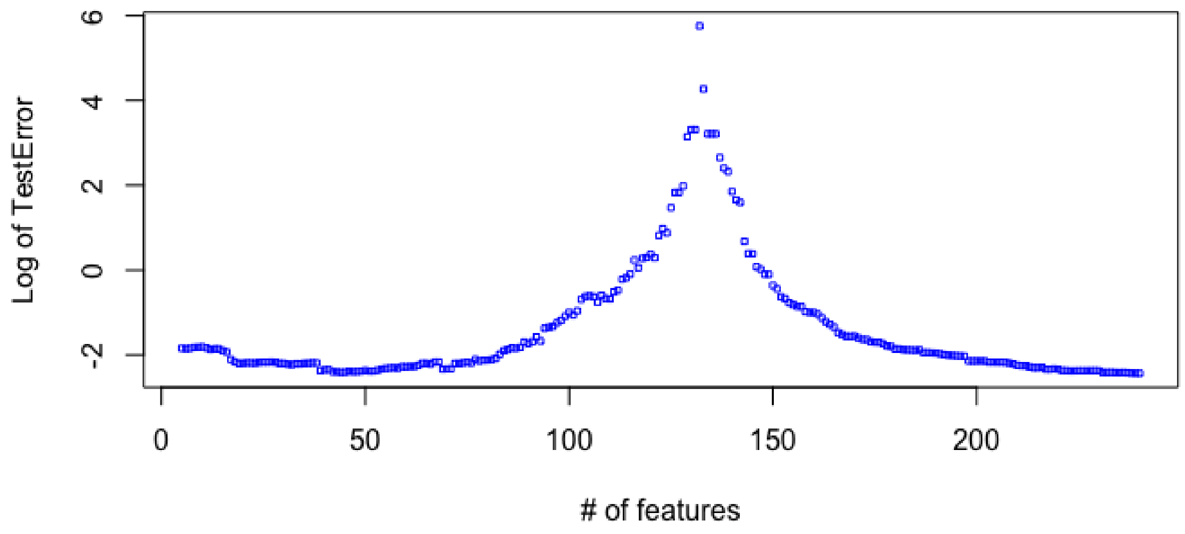
Execute the procedure below for $T = 30$ times.

In each iteration,

- Randomly partition the data into training (25%) and test (75%).
- Calculate and log the test error from the `ridgeless` method using the first d columns of `myData`, where d ranges from 6 to 241. Keep in mind that the number of regression parameters spans from 5 to 240 because the first column represents Y.

This will result in recording 236 test errors per iteration. These errors are the averaged mean squared errors based on the test data. One practical way to manage this data would be to maintain a matrix of dimensions 30-by-236 to house the test errors derived from this simulation study.

Graphical display: Plot the median of the test errors (collated over the 30 iterations) **in log scale** against the count of regression parameters, which spans from 5 to 240.



Submission Requirements

1. Submit a Markdown (or Notebook) document converted to HTML format. This should encompass all essential code along with the corresponding outputs/results.
2. For both Part II and Part III, initialize the seed value using the last four digits of your UIN. This ensures reproducibility, allowing us to obtain identical simulation results when re-running your code.
    - Note: Setting a seed for Part I isn't necessary since it doesn't involve any random processes.