CS 215
Assignment 3

Due Date: 15^{th} October, 2024

Instructions

1. You should type out a report containing all the answers to the written problems in Word (with the equation editor) or using LaTeX, or write it neatly on paper and scan it. In either case, prepare a single pdf file.

2. The report should contain names and roll numbers of all group members on the first page as a header.

3. Once the pdf and code are ready, `zip` them into one folder with the name `N` = `A3-FirstStudentRollNumber-SecondStudentRollNumber-ThirdStudentRollNumber.zip` Please name the zip file `N.zip` You may use the command `zip -r N.zip N` from the directory containing your submission directory `N` to do this. If the assignment was done in a group of size not equal to 3, the name of the folder and zip file will be `A3` followed by the roll numbers of each member; each item separated by hyphens. Any letters in the roll number must be capitalized. The submissions for this assignment **MUST** follow the given directory structure and assignments not adhering to this risk not being graded.

```
N
├── report.pdf
├── code
│   ├── 1.py
│   ├── 2.py
│   ├── 3.py/3.ipynb
│   ├── 3_predictions.csv
│   ├── 3_weights.pkl
│   ├── 4.py/4.ipynb
│   └── 5.ipynb
└── images
    ├── 1
    │   ├── 10binhistogram.png
    │   ├── crossvalidation.png
    │   └── optimalhistogram.png
    ├── 2
    │   └── transaction_distribution.png
    ├── 3
    │   ├── 3_overfit.png
    │   ├── 3_underfit.png
    │   └── 3_correctfit.png
    └── 4
        ├── ⟨kernel_function1_name⟩_kernel_regression.png
        └── ⟨kernel_function2_name⟩_kernel_regression.png
```

4. Put the pdf file and the code for the programming parts all in one zip file. The pdf should contain the names and ID numbers of all students in the group within the header. The pdf file should also contain instructions for running your code. Name the zip file as follows: A3-RollNumberOfFirstStudent-RollNumberOfSecondStudent-RollNumberofThirdStudent.zip. (If you are doing the assignment alone, the name of the zip file is A3-RollNumber.zip, if there are two students it should be A3-RollNumberOfFirstStudent-RollNumberOfSecondStudent.zip).

5. Upload the file on moodle BEFORE 11:59 pm on the due date. No assignments will be accepted thereafter.

6. Note that only one student per group should upload their work on moodle, though all group members will receive grades.

7. Please preserve a copy of all your work until the end of the semester.

1 Finding optimal bandwidth

The formula for the optimal binwidth h^*, which you saw in the class, is of theoretical interest but is not useful in practice since it depends on the unknown probability distribution function $f(x)$. We define loss of an estimator, $\hat{f}(x)$ as

$$L = \int (\hat{f}(x) - f(x))^2 dx \tag{1}$$

Let us consider the case of the histogram estimator. The estimator is a function of h, the bin width/ bandwidth of the histogram. The loss of the estimator simplifies to

$$J(h) = \int \hat{f}(x)^2 dx - 2 \int \hat{f}(x) f(x) dx \tag{2}$$

Given the dataset $\{X_1, X_2, ...X_n\}$, we want to minimize this quantity to get the optimal h, bin width of the histogram estimator. Notice that the $f(x)^2$ term has been omitted because it is constant and does not affect the argmin.

To approximate the second term above, we define the cross-validation estimator as,

$$\hat{J}(h) = \int \hat{f}(x)^2 dx - \frac{2}{n} \sum_{1}^{n} \hat{f}_{(-i)}(X_i) \tag{3}$$

Where, $\hat{f}_{(-i)}$ is the histogram estimator after removing the i^{th} observation.

1. We will prove that for histogram estimator, with m bins, the cross validation estimator can be written as

$$\hat{J}(h) = \frac{2}{(n-1)h} - \frac{n+1}{(n-1)h} \sum_{j=1}^{m} \hat{p}_j^2 \tag{4}$$

Where $\hat{p}_j$ is the estimated probability that a points falls in the j^{th} bin i.e. $\hat{p}_j = \frac{v_j}{n}$, v_j is the number of points that fall in the j^{th} bin and n is the total number of points.

(a) For the first part prove that:

$$\int \hat{f}(x)^2 dx = \frac{1}{n^2 h} \sum_{j=1}^{m} v_j^2 \tag{5}$$

Hint: Notice that $\hat{f}(x) = \sum_{j=1}^{m} \frac{\hat{p}_j}{h} \mathbb{1}[x \in B_j]$. Express $\hat{p}_j$ in terms of sum of indicator random variables and then simplify.

[3 marks]

(b) For evaluating the second term, prove that:

$$\sum_{1}^{n} \hat{f}_{(-i)}(X_i) = \frac{1}{(n-1)h} \sum_{j=1}^{m} (v_j^2 - v_j) \tag{6}$$

Note: Even if you are unable to prove the above, you can directly use the result for the next part.

[3 marks]

2. Now, we will use the cross validation estimator, derived above to find the optimal bin width in a real-world setting. We will use data from NASA/IPAC Extragalactic Database (NED). Download the data from here.

   A histogram of metric distance (in units of Megaparsecs, Mpc) helps astronomers visualize the distribution of galaxies and other extragalactic objects in space. By analyzing such histograms, scientists can identify clusters, voids, and large-scale structures in the universe. Additionally, this data can be used to study the relationship between distance and redshift, helping in the determination of the universe's expansion rate and understanding cosmic evolution.

   Here, we will use the distance of the extragalactic object from the earth (column `D (Mpc)` in the dataset). As a first filtering step, filter out all the other rows except the first 1500 datapoints. We want to analyse the histogram only for those objects which are less than 4 Mpc distance away from Earth, so filter accordingly. (Note: Do not reverse the order of above two steps)

   (a) Plot a histogram of the filtered data for number of bins = 10. Calculate and print the estimated probabilities $\hat{p}_j$ for all bins. Mention the values in the report. Save the plot as `10binhistogram.png`

   [2 marks]

   (b) Comment whether the probability distribution is an underfit/overfit/just-right fit.

   [1 mark]

   (c) Use the above formula to calculate the cross-validation score for bin width h, corresponding to 1 to 1000 bins. Save the plot as `crossvalidation.png`

   [2 marks]

   (d) Based on the cross-validation plot, what is the optimal value of bin width, h? Any answer correct within a range would receive full credit.

   [2 marks]

   (e) Plot the histogram with the optimal h^* and compare it with the histogram in part (a). Comment on how it is different. Save the plot as `optimalhistogram.png`

   [2 marks]

   (f) Save the code as `1.py` as given in the directory structure.

2 Detecting Anomalous Transactions using KDE

In financial systems, understanding the distribution of transaction patterns is crucial to detect anomalous activities. Kernel Density Estimation (KDE) provides a powerful non-parametric method to estimate the probability distribution of transaction data. Using KDE, we can identify transactions that fall in regions of low probability, enabling the detection of potentially fraudulent or suspicious activities in real time.

You have been provided with a 2D latent representation of transactions, derived through dimension reduction of a high-dimensional latent transaction representation.

As a first step for estimating the distribution of transactions, you will implement your custom KDE class. Download the data and boiler plate code from here.

2.1 Designing a custom KDE Class

A commonly used kernel is the Epanechnikov Kernel, which is defined as:

$$K(x) = \begin{cases} \frac{3}{4}(1 - ||x||_2^2) \ for \ ||x||_2 \leq 1 \\ 0 \ otherwise \end{cases} \tag{7}$$

You are provided with the structure of the `EpanechnikovKDE` class. Each function serves a distinct purpose in implementing the kernel density estimation algorithm. Your task is to complete the following sub-functions:

1. `__init__(self, bandwidth=1.0)`: Initialize the class with a given bandwidth parameter bandwidth. Set up a placeholder for the data (initially `None`).
2. `fit(self, data)`: Implement this function to store the provided data points as a NumPy array, which was initialized to `None`. Do not return anything.
3. `epanechnikov_kernel(self, x, xi)`: Implement the Epanechnikov kernel function. The function returns $K(\frac{x-x_i}{h})$.
4. `evaluate(self, x)`: Compute the KDE density estimate for a set of points x. Return

$$f(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x - x_i}{h}\right) \tag{8}$$

Note: The above part will be auto-graded, so please be careful with function names etc.

[1 + 1 + 3 + 3 marks]

2.2 Estimating Distribution of Transactions

In this task we will estimate the probability distribution of transactions carried out by a financial institution. In the given boiler plate code, the data is already loaded.

1. Initialize the object of the `EpanechnikovKDE` class with an appropriate width.
2. Fit the data (in `data` in the code) using the `fit` function defined above.
3. Estimate and plot the probability density of transactions (within an appropriate range) so that the resulting plot is reasonably smooth. How many modes does the resulting distribution contain? Include the plot and the answer in the report.
4. Save the plot as `transaction_distribution.png` and all code as `2.py` as given in the directory structure.

[2 marks]

3 Higher-Order Regression

1. Show that in a simple linear regression model the point $(\bar{x}, \bar{y})$ lies exactly on the least squares regression line. [1 mark]

2. Consider the simple linear regression model

$$Y = \beta_0 + \beta_1 x + \epsilon$$

Suppose that you as an analyst decide to use $z = x - \bar{x}$ as the regressor variable. So the new model becomes

$$Y = \beta_0^* + \beta_1^* z + \epsilon$$

Find least squares estimates of β_0^* and β_1^*. How do they relate to least square estimates of β_0 and β_1 ? How are these two models different? [3 marks]

3. As covered in class, to model $f(Y|x)$ we assume a polynomial function of form

$$f(Y|x) \sim \mathcal{N}(\mu_x, \sigma^2) \quad \text{where} \quad \mu_x = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \alpha \tag{9}$$

where x has k features, μ_x is degree 1 polynomial and A, B are estimators of α , β respectively which are also normally distributed random variables. Now, consider the case where

$$\mu_x = \beta_1 x_1 + \beta_2 x_1^2 + \ldots + \beta_m x_1^m + \beta_0 \tag{10}$$

i.e. k=1 and degree of polynomial is m. Let us express μ_x in vector notation as Xβ where $X_i = (1, x_1, x_1^2 \ldots x_1^m)$ and $\beta = (\beta_0, \beta_1, \ldots \beta_m)^T$

   (a) For this part you are provided training data with 400 instances. Data contains 1 independent variable in input and 1 continuous dependent variable.

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2) \ldots, (x_{400}, y_{400})\}$$

Here we will go with the bare hands approach so, your task is to implement OLS from scratch using standard python libraries. Try to implement the algorithm such that it generalises to arbitrary number of features. Explore regression with higher degree polynomials. Use this model to obtain predictions on test set.

Report/code to submit:

- code file(s)
- parameters learnt for best fit model, save $\beta = (\beta_0, \beta_1, \ldots \beta_m)$ parameters which should be a numpy array of shape (m+1,) as `3_weights.pkl`
- test set predictions, refer sample.csv for format

Test set predictions will be used to check performance against gold data of test set (kept private) and evaluated using SS_R. .

Training Data
Test Data

Note: Since this part is auto-graded, ensure count of data points in test set predictions match provided test data.

Hint: Consider matrix notation for generalised implementation [3 marks]

(b) As you might have observed, degree 1 polynomial would result in underfitting and high degree (say 50) would result in overfitting. Report for what degree of polynomial you get underfit (more than degree 1 polynomial), correct fit and overfit distribution.

Provide plots (one for each fit) to support your observations. You can use scatter plot (for gold data) and line plot (for predictions) to visualise model performance. There might be multiple answers to each fit, try to provide best approximation.
For example, suppose original data is sampled from a distribution best estimated with degree 8 polynomial, all models with degree less than or equal to 7 are underfitting and degree more than or equal to 10 are overfitting then report degree 7 for underfitting, 8 for correct fit and 10 for overfitting. Split training data in appropriate ratio (say 90:10) to obtain dev-set (which is never used in training), that can be used to evaluate whether model has over/under/correct fit.
Hint: Consider sorting (x_i, y_i) along x before plotting to generate correct plots.

[2 marks]

(c) Implement sum of squares of the residuals (SS_R) and coefficient of determination (R^2). Recall that

$$SS_R = \sum_{i=1}^{n} (Y_i - X_i\beta)^2 \tag{11}$$

and

$$R^2 = 1 - \frac{SS_R}{SS_Y} \tag{12}$$

Report these metrics for models corresponding to solutions provided in part (b) along with relevant code files. You can submit code for different part of this section in a single file.

Note: Please use meaningful variable names and comments in code. Use of pandas, numpy, matplotlib allowed for part a, b, c. [1 mark]

4 Non-parametric regression

4.1 Introduction

Implement a Nadaraya-Watson kernel regression model from scratch to estimate Y given x.

4.2 Data

Provided here is data on fragments of glass collected in forensic work.
Forensic Glass Data

Here, let Y be refractive index and let x be aluminum content (the fourth variable).

4.3 Setup

Recall definition of Nadaraya-Watson kernel

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right) y_i}{\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right)} \tag{13}$$

where,

1. $\hat{m}(x)$ is the estimated regression function at point x,
2. $K(\cdot)$ is the kernel function (e.g., Gaussian kernel),

3. h is the bandwidth parameter that controls the smoothness of the estimator,
4. x_i and y_i are the input and response values of the i-th observation,
5. n is the number of data points and
6. $w_i(x) = \frac{\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right)}$ is the weight corresponding to y_i

Perform nonparametric regression to fit the model $Y = f(x) + \epsilon$. You have to perform regression using atleast 2 kernel functions covered in class (both functions to be implemented from scratch). It can be shown theoretically and empirically that the choice of K is not as crucial. However, the choice of bandwidth h is very important. So you will use cross-validation to estimate the bandwidth.

4.4 Report

1. Code with correct implementation from scratch [3 marks]
2. For each kernel function,
   (a) Provide 4 plots corresponding to
      i. oversmoothed
      ii. undersmoothed
      iii. just right
      iv. cross-validation curve - plot estimated risk v/s bandwidth

      Plots corresponding to a kernel function should be grouped in grid using plt.subplot(2, 2) and naming convention to follow is $\langle kernel\_function\_name \rangle$`_kernel_regression.png`. Refer 1 for sample plot.

      For example: if chosen kernel functions are Epanechnikov and Gaussian then plots should be saved as `epanechnikov_kernel_regression.png` and `gaussian_kernel_regression.png`.

      [2+2 marks]

   (b) Report bandwidth corresponding to minimum estimated risk. [1 mark]
3. Comment on similarities and differences due to choice of different kernel functions using plots to support your points. You can also add other plots and metrics if they are relevant to your findings. [2 marks]

For this task you can choose between

- **Lp OCV** (**LOOCV** is a special case of Lp OCV where $p = 1$)
- **k-fold cross-validation**

Important things to remember:

- shuffle the data before splitting (to avoid any possible bias in data that may break i.i.d assumption)
- split data in appropriate ratio and evaluate risk using dev-set only

Note: Implementation of Nadaraya-Watson kernel estimator, kernel functions, cross-validation, risk estimation are to be done from scratch using pandas, numpy. Plots can be generated using matplotlib.

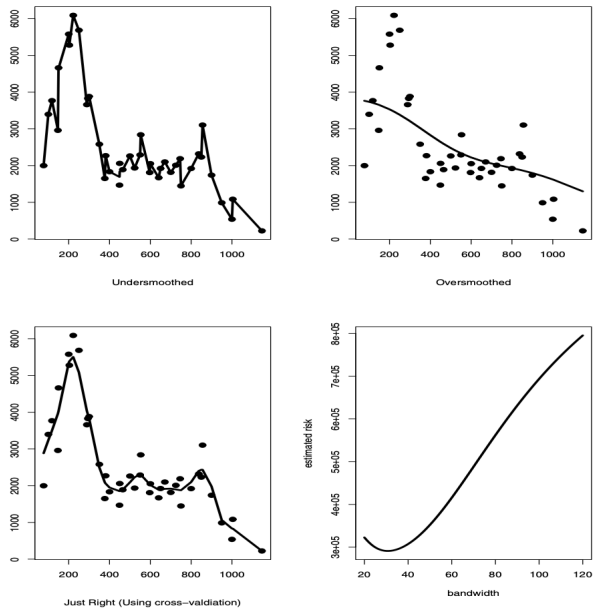

FIGURE 20.8. Regression analysis of the CMB data. The first fit is undersmoothed, the second is oversmoothed, and the third is based on cross-validation. The last panel shows the estimated risk versus the bandwidth of the smoother. The data are from BOOMERaNG, Maxima, and DASI.

Figure 1: Sample plots to be reported

5 Multivariate Insights Unlocked!

5.1 Introduction

Welcome to the Wild Blueberry Yield Prediction Challenge! In this Kaggle competition, you will work with the Wild Blueberry Yield Prediction Dataset to explore the relationships between multiple variables and predict yield using multiple linear regression. Your task is to implement regression from scratch. Explore multi-variate OLS and kernel regression techniques. Use code implemented in previous questions. You can extend kernel regression to n-dimensions based on material covered in class. Uncover hidden patterns among the covariates, and, if necessary, choose the most relevant subset of features for final modeling.

Kaggle Competition Link.
Team member(s) joining the competition might need to share their mail-id. Only 1 person from every team needs to mandatorily join the competition. Rest are optional.

Here, in addition to multi-variate regression, focus will be on data analysis such as testing linear regression assumptions (linearity, normality of errors, multicollinearity of features etc.)

5.2 Data

You will find 3 files in data section:

1. train.csv - contains both X_i and Y_i, used to estimate model parameters

2. test.csv - contains only X_i, which will be used to process submissions after competition is over

3. sample_submission.csv - sample format to follow when submitting predictions for test.csv

Data used here is AI generated synthetic version of "Wild blueberry Yield Prediction Dataset". You can find more information regarding covariates/features in data section on competition webpage.

5.3 Regression Implementation and Analysis

In previous questions, you would have implemented OLS from scratch for higher dimensional polynomials or kernel regression for 1-dimensional data. Here, you have to use same implementations. You can perform analysis in jupyter notebook and submit it as specified in instructions. Highlight analysis along with proper conclusions and findings in notebook and no need to add same information in report.
Refer Mastering Jupyter Notebooks: Best Practices for Data Science and notebooks shared in lecture slides for good coding practices. Try to derive insights and present your findings with supporting plots and metrics.

(Hint: In matrix notation of k-dimensional linear regression, as covered in class, k-features corresponding to a data point form a row vector in X. For higher dimensional regression you can consider $x_1, {x_1}^2, {x_1}^3, \ldots {x_1}^m$ as features introduced manually in-place of k-features and apply regression as you would in case of m-dimensional regression. Refer class notes for extension of single to multi-variate kernel regression.)

Note: For this section you can use any python library for analysis and visualisation but implementation of OLS, Kernel Regression and MAE should be from scratch.

5.4 Evaluation

Submissions will be evaluated using MAE which is to be, again, implemented from scratch. You will receive full credit(scored out of 10) for score above a certain threshold and correct implementation of regression models and metrics. Bonus marks will be awarded to top 5 teams based on kaggle leaderboard capped by total marks. 5 marks will be awarded for analysis and code quality.
You can refer Multi-Linear Regression Notebook for analysis in multi-variable setup.

Deadline: Same as assignment deadline.

[10+5 marks]