DAI Assignment-3

Aditya Neeraje, Balaji Karedla, Moulik Jindal

October 16, 2024

Contents

1 Finding optimal bandwidth

1.1 Part 1

1.1.1 Part (a)

We know that the estimator for distribution function $\hat{f}$ is given by

$$\hat{f}(x) = \sum_{j=1}^{m} \frac{\hat{p}_j}{h} \mathbb{1}\left[x \in B_j\right] \tag{1}$$

Also given, v_j is the number of points falling in the j^{th} bin and $\hat{p}_j = \frac{v_j}{n}$. Using this we get,

$$\begin{aligned}
\int \hat{f}(x)^2 dx &= \int \left(\sum_{j=1}^{m} \frac{\hat{p}_j}{h} \mathbb{1}\left[x \in B_j\right] \right)^2 dx \\
&= \sum_{i=1}^{m} \int_{B_i} \left(\sum_{j=1}^{m} \frac{\hat{p}_j}{h} \mathbb{1}\left[x \in B_j\right] \right)^2 dx && \text{(Segmenting the interval over the bins)} \\
&= \sum_{j=1}^{m} \int_{B_j} \left(\frac{\hat{p}_j}{h} \right)^2 dx && \text{(Only the jth interval sets the indicator function)} \\
&= \sum_{i=1}^{m} \left(\frac{\hat{p}_j}{h} \right)^2 \times h && \text{(Width of each bin is h)} \\
&= \frac{1}{n^2 h} \sum_{j=1}^{m} v_j^2
\end{aligned}$$

1.1.2 Part (b)

The histogram estimator after removing the i^{th} observation is given by

$$\hat{f}_{(-i)}(x) = \sum_{j=1}^{m} \frac{\hat{p}_{j,-i}}{h} \mathbb{1}\left[x \in B_j\right] \tag{2}$$

where $\hat{p}_{j,-i} = \frac{v_{j,-i}}{n-1}$ and $v_{j,-i}$ is the number of points falling in the j^{th} bin after removing the i^{th} observation. Now,

$$
\begin{aligned}
\sum_{i=1}^{n} \hat{f}_{(-i)}(X_i) &= \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\hat{p}_{j,-i}}{h} \mathbb{1}\left[X_i \in B_j\right] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{v_{j,-i}}{(n-1)h} \mathbb{1}\left[X_i \in B_j\right] \\
&= \sum_{j=1}^{m}\sum_{i=1}^{n} \frac{v_{j,-i}}{(n-1)h} \mathbb{1}\left[X_i \in B_j\right] \\
&= \sum_{j=1}^{m} \frac{v_j - 1}{(n-1)h} \left(\sum_{i=1}^{n} \mathbb{1}\left[X_i \in B_j\right]\right) \quad \text{(since whenever } \mathbb{1}[X_i \in B_j] = 1, v_{j,-i} = v_j - 1\text{)} \\
&= \sum_{j=1}^{m} \frac{v_j - 1}{(n-1)h} v_j \\
&= \frac{1}{(n-1)h} \sum_{j=1}^{m} (v_j^2 - v_j)
\end{aligned}
$$

1.2 Part 2

1.2.1 Part (a)

The histogram of the filtered data for number of bins = 10 is shown in Figure 1.

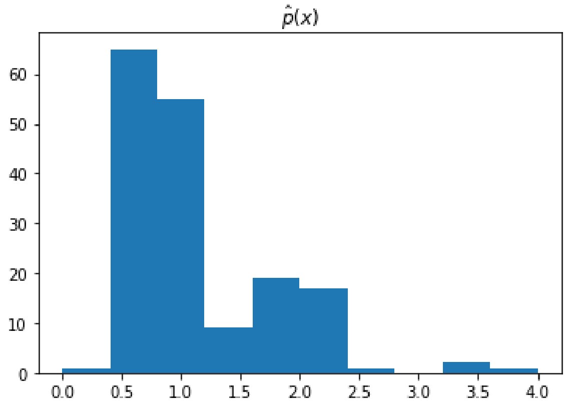

Figure 1: Histogram of filtered data with 10 bins

The values of $\hat{p}_j$ are as follows:

Bin	1	2	3	4	5	6	7	8	9	10
$\hat{p}_j$	0.602	1.428	0.138	0.12	0.396	0.172	0.017	0.0	0.034	0.017

1.2.2 Part (b)

The probablility distribution is undersmoothed as the number of bins is too low. The distribution is not smooth and the bins are too wide. This is evident from the histogram in Figure 1.

1.2.3 Part (c)

The graph of $\hat{J}(h)$ vs h for the number of bins ranging from 1 to 1000 is shown in Figure 2.

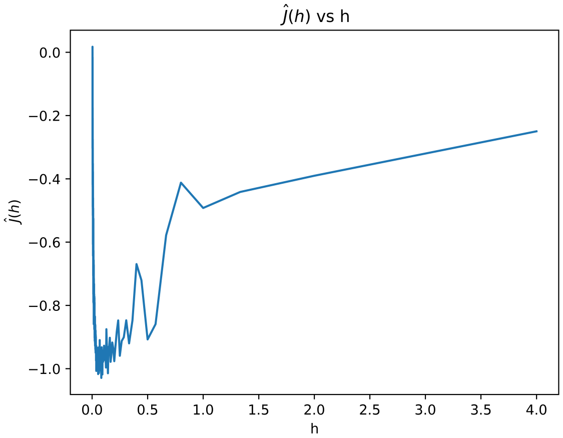

Figure 2: $\hat{J}(h)$ vs h

1.2.4 Part (d)

Based on the Cross Validation plot in Figure 2, the optimal number of bins is 50 with a value of $h^* = 0.06836$.

1.2.5 Part (e)

The histogram of the filtered data for number of bins = 50 is shown in Figure 3.

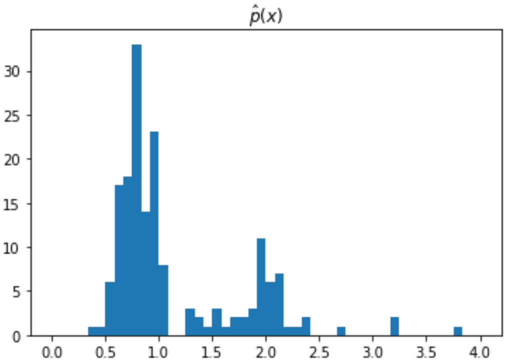

Figure 3: Histogram of filtered data with 50 bins

The graph is now just right and is neither oversmoothed nor undersmoothed. This is clearly evidnent when we compare the histograms in Figure 1 we can realize that the data will have a higher variance leading to an overall higher risk as compared to 3.

2 Detecting Anomalous Transactions using KDE

2.1 Designing a custom KDE Class

The KDE class with the Epachnikov kernel

$$K(x) = \frac{3}{\pi}(1 - ||x||_2^2) \quad \text{if} \quad ||x||_2 \leq 1 \tag{3}$$

The code was implemented in the file `2.py`.

2.2 Estimating Distribution of Transactions

After estimating the distribution of transactions using the Epachnikov kernel, the estimated probability density graph is shown in Figure 4.

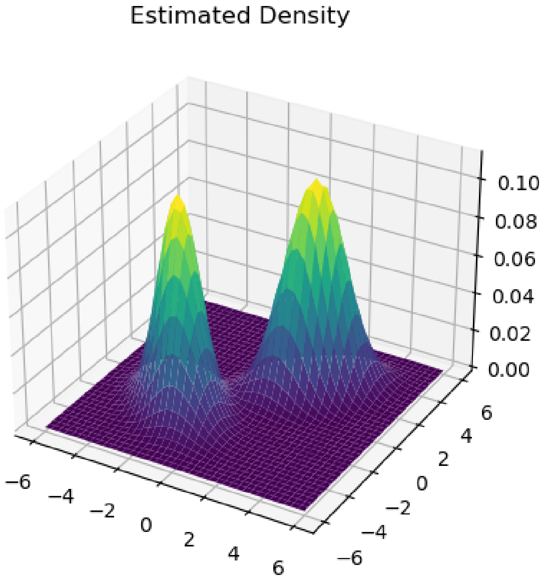

Figure 4: Estimated Probability Density of Transactions

3 Higher-Order Regression

3.1 Distribution of B:

Let us suppose we are trying to fit the dataset to a functional equation of the form:

$$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_m x^m + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. We have to find the value of B (the estimator of $(\beta_0, \beta_1, \cdots, \beta_m)$) that minimizes

$$\sum_{i=1}^{n} (Y_i - B_0 - B_1 x_i - B_2 x_i^2 - \cdots - B_m x_i^m)^2$$

Taking the derivative w.r.t each of the $B_i' s$, we get

$$\sum_{i=1}^{n} Y_i = nB_0 + B_1 \sum_{i=1}^{n} x_i + B_2 \sum_{i=1}^{n} x_i^2 + \cdots + B_m \sum_{i=1}^{n} x_i^m$$
$$\sum_{i=1}^{n} x_i \cdot Y_i = B_0 \sum_{i=1}^{n} x_i + B_1 \sum_{i=1}^{n} x_i^2 + B_2 \sum_{i=1}^{n} x_i^3 + \cdots + B_m \sum_{i=1}^{n} x_i^{m+1}$$
$$\cdots$$
$$\sum_{i=1}^{n} x_i^m \cdot Y_i = B_0 \sum_{i=1}^{n} x_i^m + B_1 \sum_{i=1}^{n} x_i^{m+1} + B_2 \sum_{i=1}^{n} x_i^{m+2} + \cdots + B_m \sum_{i=1}^{n} x_i^{2m}$$

Let us suppose that the matrix X is defined as $X_{ij} = x_i^j$. Note that $A = X^T \cdot X$ is a matrix that satisfies $A_{ij} = \sum_{k=1}^{n} x_k^{i+j}$ where A_{ij} is the element in the i^{th} row and j^{th} column of A. Let Y be the vector of Y_i's. Then, the above equations can be written as:

$$X^T \cdot Y = (X^T \cdot X) \cdot B$$

We claim that $X^T \cdot X$ is invertible as long as all values of x are distinct. To prove this, we first prove X is invertible. If that is the case, then X^T, which has the same determinant as X, is invertible, and their product also has a non-zero determinant and is invertible. For the proof of the claim that X is invertible, let us assume for the sake of contradiction that the columns of this matrix are not linearly independent.

That is, let us assume $c_0 v_0 + c_1 v_1 + \cdots + c_m v_m = 0$ where the c_i's are the columns of X and v_i's are real numbers, not all zero. Then we get on the k_{th} coordinate (for $k \in \{1, \cdots n\}$):

$$v_0 + v_1 x_k + \cdots + v_m x_k^m = 0$$

which means x_k is a root of the polynomial $v_0 + v_1 x + \cdots + v_m x^m$. Since this polynomial has at most m roots, we get that two of the x_k's must be equal, which contradicts our assumption that all x's are distinct.

Hence, $X^T \cdot X$ is invertible, and we get that B is given by:

$$B = (X^T \cdot X)^{-1} \cdot X^T \cdot Y$$

Thus, $B = C \cdot Y$ for some matrix C, with m rows and n columns.

$B_{i-1} = \sum_{j=1}^{n} C_{ij} Y_j$ is a linear combination of Gaussian random variables, and hence is Gaussian.

B is a (m+1)-tuple of Gaussian random variables.

3.2 Mean and Standard Deviation of B:

We know $Y = \beta X + \epsilon$, where $\beta = (\beta_0, \beta_1, \cdots, \beta_m)$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

$$\begin{aligned} E[B] &= E[(X^T \cdot X)^{-1} \cdot X^T \cdot Y] \\ &= (X^T \cdot X)^{-1} \cdot X^T \cdot E[Y] \\ &= (X^T \cdot X)^{-1} \cdot X^T \cdot \beta X \\ &= (X^T \cdot X)^{-1} \cdot X^T \cdot X \cdot \beta \\ &= \beta \end{aligned}$$

Above, notice that we have used the fact that the component-wise computation of the mean of B can be done simultaneously.

To find the variance, we again use matrix C as defined above. $B_{i-1} = \sum_{k=1}^{n} C_{ik} Y_k$ and $B_{j-1} = \sum_{k=1}^{n} C_{jk} Y_k$.
Hence, $\text{Cov}(B_{i-1}, B_{j-1}) = \text{Cov}(\sum_{k=1}^{n} C_{ik} Y_k, \sum_{k=1}^{n} C_{jk} Y_k) = \sum_{r=1}^{n} \sum_{l=1}^{n} C_{il} C_{jr} \text{Cov}(Y_l, Y_r)$.
Now, Y_l and Y_r are independent for $l \neq r$, and hence $\text{Cov}(Y_l, Y_r) = 0$ for $l \neq r$ and $\text{Var}(Y_l)$ if $l = r$.
Since $\text{Var}(Y_l)$=σ^2, we get that $\text{Cov}(B_{i-1}, B_{j-1}) = \sigma^2 \sum_{k=1}^{n} C_{ik} C_{jk}$.
The final term here is equal to the $(i, j)^{\text{th}}$ element of the matrix $C \cdot C^T$.
Thus, Cov(B)= $\sigma^2 C \cdot C^T$.
Now,

$$\begin{aligned} C^T = ((X^T \cdot X)^{-1} \cdot X^T)^T \\ &= X \cdot ((X^T \cdot X)^{-1})^T \\ &= X \cdot (X^T \cdot X)^{-1} \end{aligned}$$

Here, the last point follows from the symmetry of $X^T \cdot X$, which implies that its inverse is also a symmetric matrix.
The above statement can be proven as follows: suppose Y is a symmetric invertible matrix, then $Y \cdot Y^{-1} = I = (Y^{-1})^T \cdot Y^T = (Y^{-1})^T \cdot Y$.
Using the fact that $Y^{-1} \cdot Y = I = (Y^{-1})^T \cdot Y^T$, we get that Y^{-1} is symmetric, since inverses are unique.

Since $\text{Cov}(B_i, B_i)$=$\text{Var}(B_i)$, we can get the variance of B_i as the i^{th} diagonal element of $\sigma^2 X \cdot (X^T \cdot X)^{-1}$.

The quantity σ^2 can be estimated using the sum of squares of the residuals. That is, if we let

$$SS_R = \sum_{i=1}^{n} (Y_i - B_0 - B_1 x_i - B_2 x_i^2 - \cdots - B_m x_i^m)^2$$

then, it can be shown that $\sigma^2 = \frac{SS_R}{n-m-1}$. This is because $\frac{SS_R}{\sigma^2} \sim \chi^2_{n-(k+1)}$.

3.3 Proving B is unbiased:

In computing the mean of B_i above, we concluded that $E[B_i] = \beta_i$ for all $i \in \{0, \cdots, m\}$.
Thus, $E[B] = \beta$, where β is as defined before. This is the same as the true value of the coefficients, hence B is an unbiased estimator.

3.4 Part 1:

We have seen in class that the estimator in simple linear regression can be modelled as $\hat{y} = B\hat{x} + A$ where

$$B = \frac{\sum_{i=1}^{n} x_i \cdot y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}$$

$$A = \bar{y} - B\bar{x}$$

Using this estimate at $y = \bar{x}$, we get $\hat{y} = B\bar{x} + \bar{y} - B\bar{x} = \bar{y}$. Thus, $(\bar{x}, \bar{y})$ is always on the regression line.

3.5 Part 2:

We want to reduce the sum of squares error, which is given by:

$$SS_R = \sum_{i=1}^{n} (y_i - \beta_0^* - \beta_1^*(x_i - \bar{x}))^2$$

Taking the derivatives w.r.t β_0^* and β_1^*,

$$\sum_{i=1}^{n} (y_i - \beta_0^* - \beta_1^*(x_i - \bar{x})) = 0 \tag{4}$$

$$\sum_{i=1}^{n} (y_i - \beta_0^*) = 0 \tag{5}$$

$$\frac{\sum_{i=1}^{n} y_i}{n} = \beta_0^* \tag{6}$$

$$\beta_0^* = \bar{y} \tag{7}$$

The second last line stems from the fact that $\sum_{i=1}^{n} (x_i - \bar{x}) = 0$.

$$\sum_{i=1}^{n} (y_i - \beta_0^* - \beta_1^*(x_i - \bar{x}))(x_i - \bar{x}) = 0 \tag{8}$$

$$\sum_{i=1}^{n} (y_i)(x_i - \bar{x}) - \beta_1^* \sum_{i=1}^{n} (x_i - \bar{x})^2 = 0 \tag{9}$$

$$\beta_1^* = \frac{\sum_{i=1}^{n} (y_i \cdot x_i) - n\bar{y}\bar{x}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} \tag{10}$$

Thus, $\beta_0^* = \bar{y}$ and $\beta_1^* = B$, with $B = \frac{\sum_{i=1}^{n} (y_i \cdot x_i) - n\bar{y}\bar{x}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}$.
In comparison, the simple linear regression model gives $\beta_0 = \bar{y} - B\bar{x}$ and $\beta_1 = B$. These results are very intuitive and in fact both models give us the same result, with $A^* = \bar{y} - B\bar{z} = \bar{y}$ (since $\sum_{i=1}^{n} (x_i - \bar{x}) = 0$). As for B, B is the ratio of covariance of z,y to variance of z, and since variance and covariance are not affected by a linear shift, these are equivalent to the covariance of x, y and variance of x respectively.

3.6 Explaining the code:

The code contains a HigherOrderRegression class. You should pass the desired degree of the class while initializing it. Eg. regressor = HigherOrderRegression(3) will create a regressor that fits a cubic polynomial to the data.
regressor.fit(X, Y) function fits the data to the polynomial.
regressor.cross_validation(params) can be used to visualize the results of fitting the model to various degree polynomials. For each degree passed in params, a plot will be generated. The function also prints out the SS_R scores for every parameter, enabling identification of the optimal degree.
Note: To actually display these plots and save the figures, one would have to comment out a few lines of code in the cross_validation function, which I have commented out for now to avoid needless generation of graphs and saving of files.
regressor.sum_of_squares(Y_true, Y_pred) and regressor.r2_score(Y_true, Y_pred) return the SS_R and R^2 scores respectively.

3.7 underfitting, overfitting and Correct Fit:

I got a severe underfit at degree 2 and some amount of noticeable underfitting at degree 3 (especially at the left top corner).

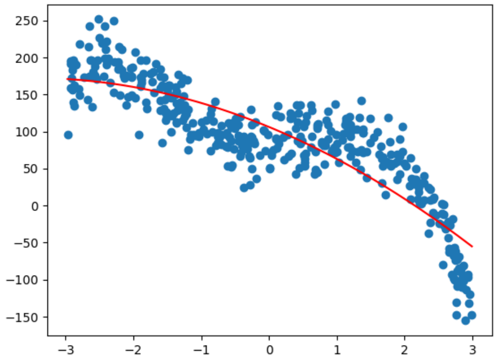

Figure 5: Degree 2 - underfitting

I got severe overfitting at degree 20 and visible overfitiing at degree 18. On a k-fold cross-validation test, degree 20 had an SS_R almost 1000 greater than that of degree 5 (24631 vs 23135), and at degree 25 the SSR was more than 300 times the optimal value (8681858 vs 23135).

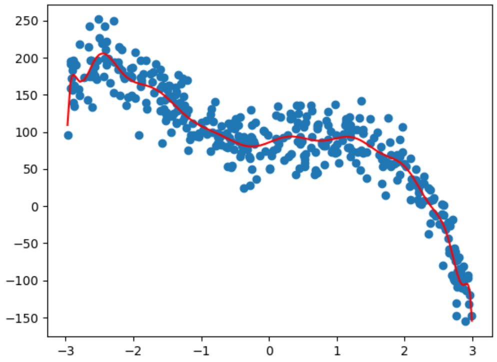

Figure 6: Degree 20 - overfitting

Optimal fitting that minimized the SS_R during k-fold cross-validation with k=10 occured at degree 5. The SS_R here was 23135.

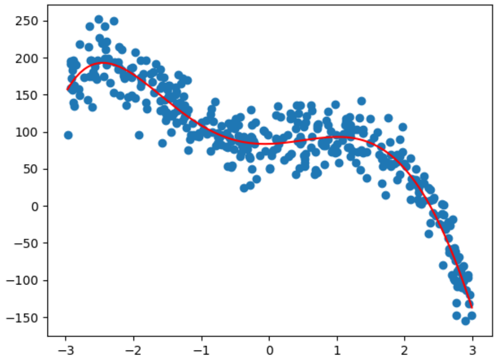

Figure 7: Degree 5 - optimal fitting

3.8 SS_R and R^2 scores:

The SS_R scores given below are calculated as the mean of the sum of squares of residuals for roughy 10 different splits of the shuffled data on a 90-10 train-test split multiplied by the size of the data during 10-fold cross-validation. The R^2 scores are also the average of the R^2 scores for the same splits, but are not multiplied by the number of elements in the entire data (because r2_score is not additive).
The SS_R scores are as follows (from degree 1 to degree 24): [678747.9382965628, 592366.7833805815, 408863.87658873753, 252157.416703885, 241342.14391359143, 242087.9009216432, 243876.70040396016, 246526.1638885481, 245291.59829327973, 247463.42707817038, 248606.92864321292, 253615.77339479985, 258441.89366066793, 259842.7638342046, 262545.24268273375, 260498.2392055063, 327502.366708921, 2159702.53448649, 2237333.434887825, 2548830.073823605, 2625229.0104435543, 2828216.5837240503, 3084863.5053297495, 3043823.152044983]
The scores corresponding to degree 2, degree 20 and degree 5 are 56427.175480097954, 24631.227983779838 and 23135.01264391381 respectively.

Similarly, the R^2 scores are as follows (from degree 1 to degree 24): [0.7073677200099657, 0.7405248627800795, 0.8110940468157531, 0.8855334495350349, 0.8890032715554146, 0.8890277101115786, 0.8883324870047348, 0.8875398759197806, 0.8879689135097628, 0.8869996684541892, 0.8860458568265249, 0.8838815125716601, 0.882803463178799, 0.8820051686946446, 0.8815504766729326, 0.8811998128010107, 0.8481736884907953, 0.015264747634069962, -0.01811629656120142, -0.1687725538754416, -0.20659071646514895, -0.29430921457705345, -0.4353047105419893, -0.39995843700122163]
The scores corresponding to degree 2, degree 20 and degree 5 are 0.7405248627800795, 0.8816272586091627 and 0.8890032715554141 respectively.

4 Non-parametric regression

4.1 Explaining the code:

The code contains a NadarayaWatsonRegressor class and three kernel types that can be used - Gaussian, Epachnikov and Square (Tophat). Let us suppose our model is called "regressor". We first need regressor to store our data, which is done by calling regressor.fit(X, Y). regressor.display(bandwidth) plots the estimated KDE for the particular bandwidth.

Noticing that the data we have been given has a range of x values which are greater than 0 and less than 4, I have taken these as my limits for plotting.

Within the range of min(X) to max(X), we plot values corresponding to the y_labels given (note again that I am using the fact that the x values in the graph are already very dense, in order to save on computation). Outside this range, I plot the estimated yvery 0.001 units.

regressor.plot_cross_validator(bandwidths) can be used to generate a plot of cross-validation scores. Assumptions made here include the assumption that bandwidths is an array of non-zero values and is in increasing order. While trying to find the optimal bandwidth, plot_cross_validator() was called with bandwidths being np.linspace(0.1, 2, 200)

Finally, regressor.display_4_plots can be used to generate the picture required by the assignment. The user needs to pass as arguments the bandwidths he wants to assign to the top left, top right and bottom left graphs respectively. For instance, I have called regressor3.display_4_plots(0.01, 1.75, 0.32) to represent that h=0.01 leads to overfitting and undersmoothening, h=1.75 leads to underfitting and oversmoothening and h=0.32 seems optimal based on cross validation scores.

4.2 Bandwidth corresponding to minimum estimated risk:

For a Gaussian kernel, the computed optimal bandwidth was 0.13.
For a Epachnikov kernel, the computed optimal bandwidth was 0.32.
For a Tophat/Square kernel, the computed optimal bandwidth was 0.31.

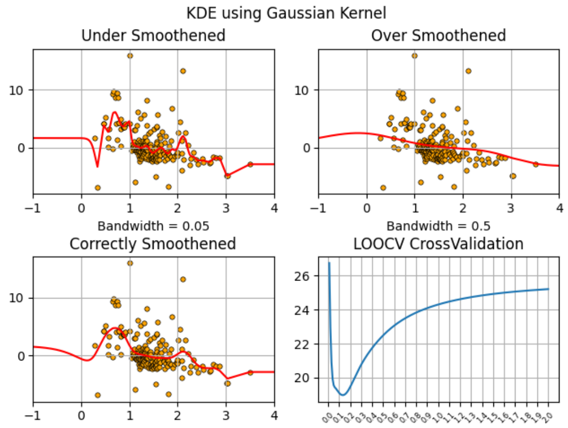

Figure 8: Gaussian Kernel Regression - undersmoothening, oversmoothening, optimal bandwidth and Cross Validation curve

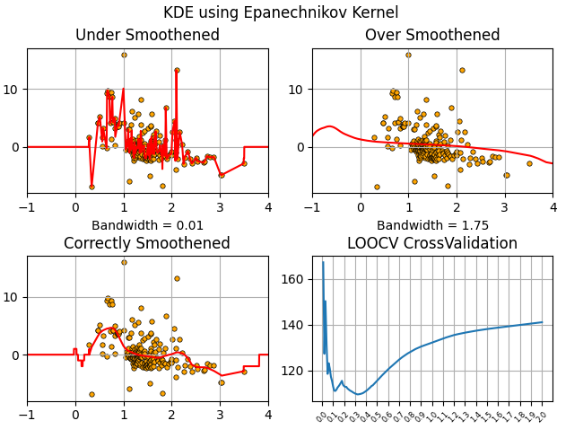

Figure 9: Epachnikov Kernel Regression - undersmoothening, oversmoothening, optimal bandwidth and Cross Validation curve

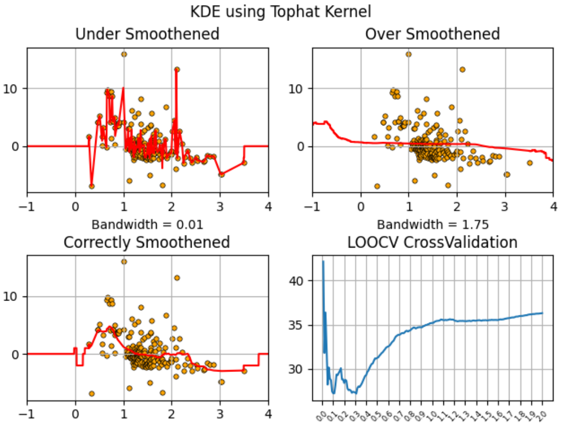

Figure 10: Tophat Kernel Regression - undersmoothening, oversmoothening, optimal bandwidth and Cross Validation curve

4.3 Comments on similarity and dissimilarity

The Gaussian estimator seems a lot smoother than the others, especially compared to Tophat, presumably because of the gaussian kernel itself being smooth and not jerky like the tophat kernel. The Gaussian estimator has a lower optimal bandwidth, presumably because it assigns a non-zero weight to all points, thus even points which are far away have some smoothening influence, whereas with Epachnikov or Tophat, both of which have finite support, we need a larger bandwidth in order to capture enough neighboring points for a smooth estimate.

The EPA and Tophat estimators do not have as smooth cross-validation curves as the Gaussian, presumably because they are abrupt jumps in the risk when a transition between bandwidths causes one object to lose all influence over the kernel at a point.