CS181-S23

Assignment #1

Due: 11:59pm ET, February 9, 2023

Homework 1: Regression

Introduction

This homework is on different three different forms of regression: kernelized regression, nearest neighbors regression, and linear regression. We will discuss implementation and examine their tradeoffs by implementing them on the same dataset, which consists of temperature over the past 800,000 years taken from ice core samples.

The folder `data` contains the data you will use for this problem. There are two files:

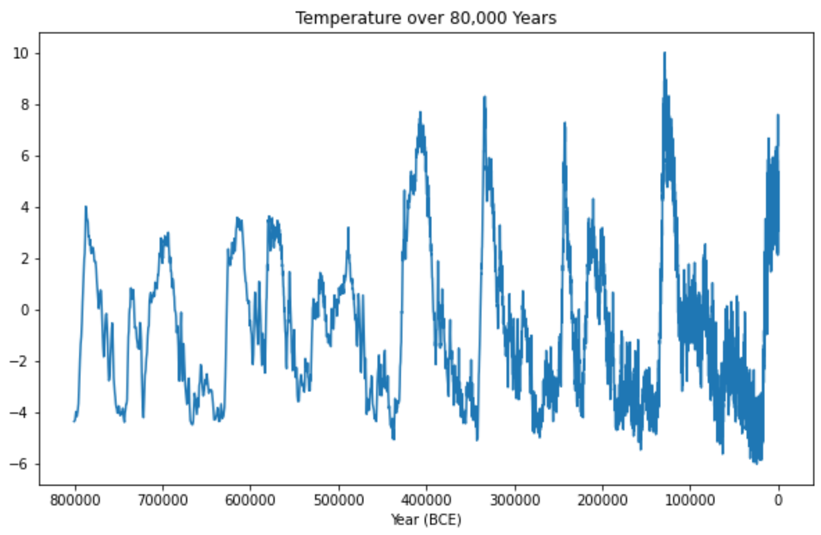
- `earth_temperature_sampled_train.csv`
- `earth_temperature_sampled_test.csv`

Each has two columns. The first column is the age of the ice core sample. For our purposes we can think of this column as the calendar year BC. The second column is the approximate difference in yearly temperature (K) from the mean over a 5000 year time window starting at the given age. The temperatures were retrieved from ice cores in Antarctica (Jouzel et al. 2007)[1].

The following is a snippet of the data file:

```
# Age, Temperature
3.999460000000000000e+05,5.090439218398755017e+00
4.099800000000000000e+05,6.150439218398755514e+00
```

Due to the large magnitude of the years, we will work in terms of thousands of years BCE in Problems 1-3. This is taken care of for you in the provided notebook.



[1] Retrieved from https://www.ncei.noaa.gov/pub/data/paleo/icecore/antarctica/epica_domec/edc3deuttemp2007.txt
Jouzel, J., Masson-Delmotte, V., Cattani, O., Dreyfus, G., Falourd, S., Hoffmann, G., … Wolff, E. W. (2007). Orbital and Millennial Antarctic Climate Variability over the Past 800,000 Years. *Science, 317*(5839), 793–796. doi:10.1126/science.1141038

If you find that you are having trouble with the first couple problems, we recommend going over the fundamentals of linear algebra and matrix calculus (see links on website). The relevant parts of the cs181-textbook notes are Sections 2.1 - 2.7. We strongly recommend reading the textbook before beginning the homework.

We also encourage you to first read the Bishop textbook, particularly: Section 2.3 (Properties of Gaussian Distributions), Section 3.1 (Linear Basis Regression), and Section 3.3 (Bayesian Linear Regression). (Note that our notation is slightly different but the underlying mathematics remains the same!).

Please type your solutions after the corresponding problems using this LaTeX template, and start each problem on a new page. You may find the following introductory resources on LaTeX useful: LaTeX Basics and LaTeX tutorial with exercises in Overleaf

Homeworks will be submitted through Gradescope. You will be added to the course Gradescope once you join the course Canvas page. If you haven't received an invitation, contact the course staff through Ed.

Please submit the writeup PDF to the Gradescope assignment 'HW1'. Remember to assign pages for each question.

Please submit your LaTeXfile and code files to the Gradescope assignment 'HW1 - Supplemental'. Your files should be named in the same way as we provide them in the repository, e.g. `hw1.pdf`, etc.

Problem 1 (Optimizing a Kernel)

Kernel-based regression techniques are similar to nearest-neighbor regressors: rather than fit a parametric model, they predict values for new data points by interpolating values from existing points in the training set. In this problem, we will consider a kernel-based regressor of the form:

$$f_\tau(x^*) = \frac{\sum_n K_\tau(x_n, x^*) y_n}{\sum_n K_\tau(x_n, x^*)}$$

where $\{(x_n, y_n)\}_{n=1}^N$ are the training data points, and $K_\tau(x, x')$ is a kernel function that defines the similarity between two inputs x and x'. A popular choice of kernel is a function that decays as the distance between the two points increases, such as

$$K_\tau(x, x') = \exp\left\{-\frac{(x - x')^2}{\tau}\right\}$$

where τ represents the square of the lengthscale (a scalar value that dictates how quickly the kernel decays). In this problem, we will consider optimizing what that (squared) lengthscale should be.
Make sure to include all required plots in your PDF.

1. Let's first take a look at the behavior of the fitted model for different values of τ. Plot your model for years in the range 800,000 BC to 400,000 BC at 1000 year intervals for the following three values of τ: $1, 50, 2500$. Since we're working in terms of thousands of years, this means you should plot $(x, f_\tau(x))$ for $x = 400, 401, \ldots, 800$. The plotting has been set up for you in the notebook already.

   Include your plot in your solution PDF.

   In no more than 5 sentences, describe what happens in each of the three cases. How well do the models interpolate? If you were to choose one of these models to use for predicting the temperature at some year in this range, which would you use?

2. Say we instead wanted to empirically evaluate which value of τ to choose. One option is to evaluate the mean squared error (MSE) for f_τ on the training set and simply choose the value of τ that gives the lowest loss. Why is this a bad idea?

   Hint: consider what value of τ would be optimal, for τ ranging in $(0, \infty)$. We can consider $f_\tau(x^*)$ as a weighted average of the training responses, where the weights are proportional to the distance to x^*, and the distance is computed via the kernel. What happens to $K_\tau(x, x')$ as τ becomes very small, when $x = x'$? What about when $x \neq x'$?

3. We will evaluate the models by computing their MSE on the test set.

   Let $\{(x'_m, y'_m)\}_{m=1}^M$ denote the test set. Write down the form of the MSE of f_τ over the test set as a function of the training set and test set. Your answer may include $\{(x'_m, y'_m)\}_{m=1}^M$, $\{(x_n, y_n)\}_{n=1}^N$, and K_τ, but not f_τ.

4. We now compute the MSE on the provided test set. Write Python code to compute the MSE with respect to the same lengthscales as in Part 1. Which model yields the lowest test set MSE? Is this consistent with what you observed in Part 1?

5. Say you would like to send your friend your kernelized regressor, so that they can reproduce the same exact predictions as you. You of course will tell them the value of τ you selected, but what other information would they need, assuming they don't currently have any of your data or code? If our training set has size N, how does this amount of information grow as a function of N—that is, what is the space complexity of storing our model?

   What is the time complexity of your implementation, when computing your model on a new datapoint?

Solution 1: Optimizing a Kernel

1. Below, we see our model for different values of τ. We can see that the model with $\tau = 1$ has a lot of

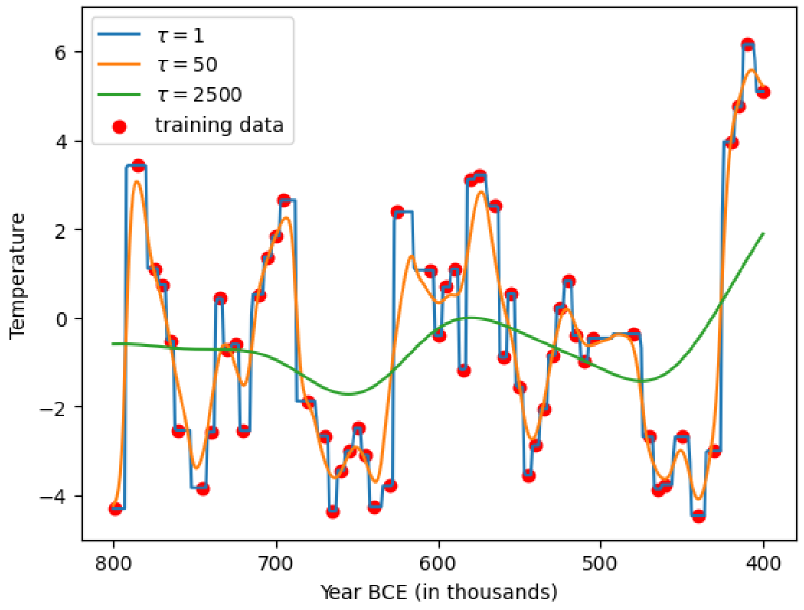


Figure 1: Surface

   variance, and it over-fits the data, exactly going through all of the points in the training data. The model with $\tau = 2500$ has much less variance, staying within a relatively narrow interval, and with gradual increases and decreases–it looks like it under-fits the data, since it doesn't really reflect the trend in the training data. The model with $\tau = 50$ has a lot of variance, but less than $\tau = 1$, and it seems to capture the trend better than the other models values of τ–this is the model that I would choose.

2. As τ approaches zero: when $x \neq x'$, we have that $K_\tau(x, x')$ approaches zero (since the exponent approaches $-\infty$, and $exp(-\infty) = 0$; when $x = x'$, $K_\tau(x, x')$ approaches 1. Thus, choosing the optimal value of τ will result in our model over-fitting our model, since the model would output the true value y_n when x' is x_n, and would be undefined for every other x'.

3. Our MSE over the test set is as follows:

$$MSE = \frac{1}{M}\sum_{m=1}^{M}(y'_m - \textbf{predictions}_m)^2 \tag{1}$$

$$= \frac{1}{M}\sum_{m=1}^{M}\left(y'_m - \frac{\sum_n K_\tau(x_n, x'_m)y_n}{\sum_n K_\tau(x_n, x'_m)}\right)^2 \tag{2}$$

4. Our results are as follows:

   - $\tau = 1$: loss = 1.9472621565209178
   - $\tau = 50$: loss = 1.8582899169613452
   - $\tau = 2500$: loss = 8.333886806980793

We can see that the model with $\tau = 50$ has the lowest MSE, followed closely by $\tau = 1$, and $\tau = 2500$ has a much higher MSE. This checks out with our answer in part 1.

5. They would need to have the function, and the same training data as me (and assuming they want to make the exact same predictions as me, they'd need the same test data, too.) Storing this information has space complexity of $O(N)$, since a training set of size N stores N pairs x, y, and $O(2N) = O(N)$. To compute the model on a new datapoint, you'd have to run it through f_τ, which iterates through every point in the training set, and so has a time complexity of $O(N)$.

Problem 2 (Kernels and kNN)

Now, let us compare the kernel-based approach to an approach based on nearest-neighbors. Recall that kNN uses a predictor of the form

$$f(x^*) = \frac{1}{k} \sum_n y_n \mathbb{I}(x_n \texttt{ is one of k-closest to } x^*)$$

where $\mathbb{I}$ is an indicator variable. For this problem, you will use the **same dataset as in Problem 1**.

Note that our set of test cases is not comprehensive: just because you pass does not mean your solution is correct! We strongly encourage you to write your own test cases and read more about ours in the comments of the Python script.

Make sure to include all required plots in your PDF.

1. Implement kNN for $k = \{1, 3, N-1\}$ where N is the size of the dataset, then plot the results for each k. To find the distance between points, use the kernel function from Problem 1 with lengthscale $\tau = 2500$.

   You will plot x^* on the year-axis and the prediction $f(x^*)$ on the temperature-axis. For the test inputs x^*, you should use an even grid spacing of 1 between $x^* = 800$ and $x^* = 400$. (Like in Problem 1, if a test point lies on top of a training input, use the formula without excluding that training input.) Again, this has been set up for you already.

   Please **write your own implementation of kNN** for full credit. Do not use external libraries to find nearest neighbors.

2. Describe what you see: what is the behavior of the functions in these three plots? How does it compare to the behavior of the functions in the three plots from Problem 1? In particular, which of the plots from Problem 1 look most similar to each in Problem 2? Are there situations in which kNN and kernel-based regression interpolate similarly?

3. Choose the kNN model you most prefer among the three. Which model did you choose and why? What is its mean squared error on the test set?

4. As before, say you wanted to send your friend your kNN, so that they can reproduce the same exact predictions as you. You will again tell them the value of the k you selected, but what other information would they need, assuming they do not currently have any of your data or code, and how does this information grow as a function of the size of the training set, N? Again worded more formally, what is the space complexity of storing your model?

   What is the time complexity of your implementation, when computing your model on a new datapoint? Give a brief overview of your implementation when you justify your answers.

Solution 2: Kernels and kNN

1. Below is the plot of the results for each k:

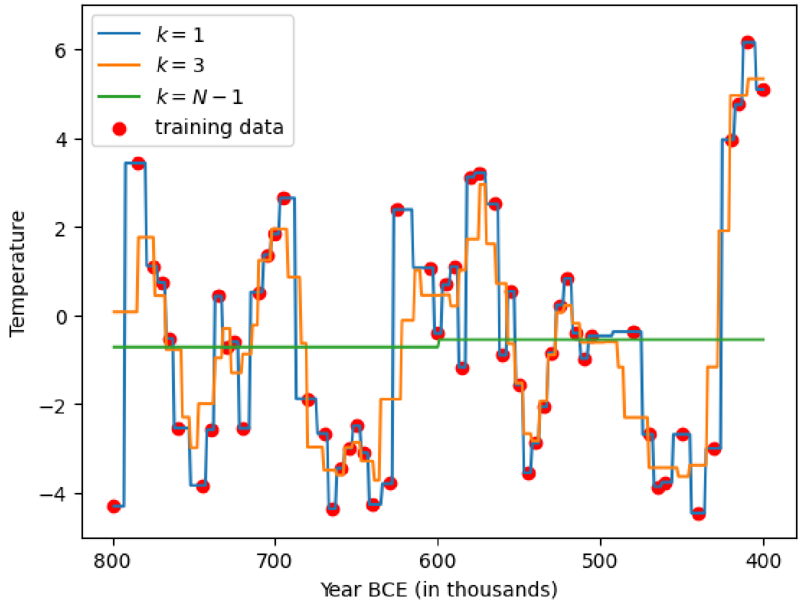


Figure 2: Plot of kNN models

2. We see that for $k = 1$, the plot over-fits, going through every single data point and highly fitting the noise. This is most similar to the plot of $\tau = 1$. For plot $k = 3$, it reflects the trend in the data without over-fitting or under-fitting. This is most similar to the plot of $\tau = 50$. For $k = N - 1$, the plot underfits the data–it is almost a straight line, and reflects no trend. This is most similar to the plot of $\tau = 2500$. It seems that they interpolate similarly when the choice of kernel function for a Kernel-based regression is one that decays as the distance between two points increases, since that more dramatically reduces the effects of points the further away they are from a given x', and highly prioritizing points that are closer, which has a similar effect to only counting the closest points.

3. I would choose the kNN model with $k = 3$, since the other 2 are clearly not appropriate models (either dramatically over-fitting or under-fitting), and out of the 3 of them, $k = 3$ best reflects the trend. Its mean squared error is 3.8907662222222212.

4. They would need to know the distance function, and they would need to have the same training data I used, which means the space complexity of the model is $O(N)$.

   For a given data point, the time complexity is $O(N)$ for calculating the distance between it and every training point (stored in an array), then $O(N \log N)$ for sorting the indices of the array by order of the values in the array (using numpy's argsort() function, which uses a quicksory algorithm) and then $O(k)$ for iterating through the resulting k closest neighbours and finding their average. Thus, the final time complexity for a given data point is $O(N + N \log N + k) = O(N \log N)$.

Problem 3 (Modeling Climate Change 800,000 Years Ago)

The objective of this problem is to learn about different forms of linear regression with basis functions.

Make sure to include all required plots in your PDF.

1. Recall that in *Ordinary Least Squares* (OLS) regression, we have data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N = \{\mathbf{X}, \mathbf{y}\}$ where $\mathbf{X} \in \mathbb{R}^{N \times D}$. The goal is to find the weights $\mathbf{w} \in \mathbb{R}^D$ for a model $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$ such that the MSE

$$\frac{1}{N}\|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

is minimized.

Without any novel bases, we have merely a single feature $D = 1$, the year, which is not enough to model our data. Hence, in this problem you will improve the expressivity of our regression model by implementing different bases functions $\phi = (\phi_1, \ldots, \phi_D)$. In order to avoid numerical instability, we must transform the data first. Let this transformation be f, which has been introduced in the code for you in the notebook.

(a) $\phi_j(x) = f(x)^j$ for $j = 1, \ldots, 9$. $f(x) = \frac{x}{1.81 \cdot 10^2}$.

(b) $\phi_j(x) = \exp\left\{-\frac{(f(x) - \mu_j)^2}{5}\right\}$ for $\mu_j = \frac{j+7}{8}$ with $j = 1, \ldots, 9$. $f(x) = \frac{x}{4.00 \cdot 10^2}$.

(c) $\phi_j(x) = \cos(f(x)/j)$ for $j = 1, \ldots, 9$. $f(x) = \frac{x}{1.81}$.

(d) $\phi_j(x) = \cos(f(x)/j)$ for $j = 1, \ldots, 49$. $f(x) = \frac{x}{1.81 \cdot 10^{-1}}$. [a]

* Note: Please make sure to add a bias term for all your basis functions above in your implementation of the `make_basis`.

Let

$$\phi(\mathbf{X}) = \begin{bmatrix} \phi(x_1) \\ \phi(x_2) \\ \vdots \\ \phi(x_N) \end{bmatrix} \in \mathbb{R}^{N \times D}.$$

You will complete the `make_basis` function which must return $\phi(\mathbf{X})$ for each part (a) - (d). You do NOT need to submit this code in your LaTeXwriteup.

For each basis create a plot of your code graphing the OLS regression line trained on your training data against a scatter plot of the training data. Boilerplate plotting code is provided in the notebook. **All you need to include in your writeup for 4.1 are these four plots.**

[a] For the trigonometric bases (c) and (d), the periodic nature of cosine requires us to transform the data such that the lengthscale is within the periods of each element of our basis.

Problem 3 (cont.)

2. We now have four different models to evaluate. Our models had no prior knowledge of any of the testing data, thus evaluating on the test set allows us to make stronger (but not definitive!) claims on the generalizability of our model.

   Observe that there is never an objectively "good" value of MSE or negative log likelihood - we can use them to compare models, but without context, they don't tell us whether or not our model performs well.

   For each basis function, complete three tasks and include the results in your writeup:

   - Compute the MSE on the train and test set.
   - Assume that the data is distributed as $y_i = \mathbf{w}^\top \mathbf{x}_i + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, we roll in the bias $\mathbf{x}_i = \begin{bmatrix} 1 \\ x_i \end{bmatrix}$, and each data point is drawn independently. Find σ_{MLE} and $\mathbf{w}_{\text{MLE}}$ (recall the formulas from class!) and use these to compute the negative log-likelihood of a model with parameters $\sigma_{\text{MLE}}, \mathbf{w}_{\text{MLE}}$ on your train and test sets. The following derives the likelihood.

   $$\begin{aligned} p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}, \sigma_{\text{MLE}}) &= \prod_{i=1}^{N} \mathcal{N}(\mathbf{y}_i \mid \mathbf{w}^\top \mathbf{x_i}, \sigma_{\text{MLE}}^2) \\ &= \prod_{i=1}^{N} \frac{1}{\sigma_{\text{MLE}}\sqrt{2\pi}} \exp\left(-\frac{(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}{2\sigma_{\text{MLE}}^2}\right) \end{aligned}$$

   - Make a claim regarding whether this basis overfits, underfits, or fits well. Write 1-2 sentences explaining your claim using the train and test negative log-likelihood and MSE.

3. For the third time, you wish to send your friend your model. Lets say you fitted some weight vector of dimension D. What information would you need to share with your friend for them to perform the same predictions as you? Do you need to share your entire training set with them this time? Again, what is the space complexity of storing your model?

   Given an arbitrary datapoint, what is the time complexity of computing the predicted value for this data point?

   How do these complexities compare to those of the kNN and kernelized regressor?

   Your response should be no longer than 5 sentences.

Note: Recall that we are using a different set of inputs $\mathbf{X}$ for each basis (a)-(d). Although it may seem as though this prevents us from being able to directly compare the MSE since we are using different data, each transformation can be considered as being a part of our model. Contrast this with transformations (such as standardization) that cause the variance of the target $\mathbf{y}$ to be different; in these cases the MSE can no longer be directly compared.

Solution 3: Modeling Climate Change 800,000 Years Ago

1. Below are the 4 plots:

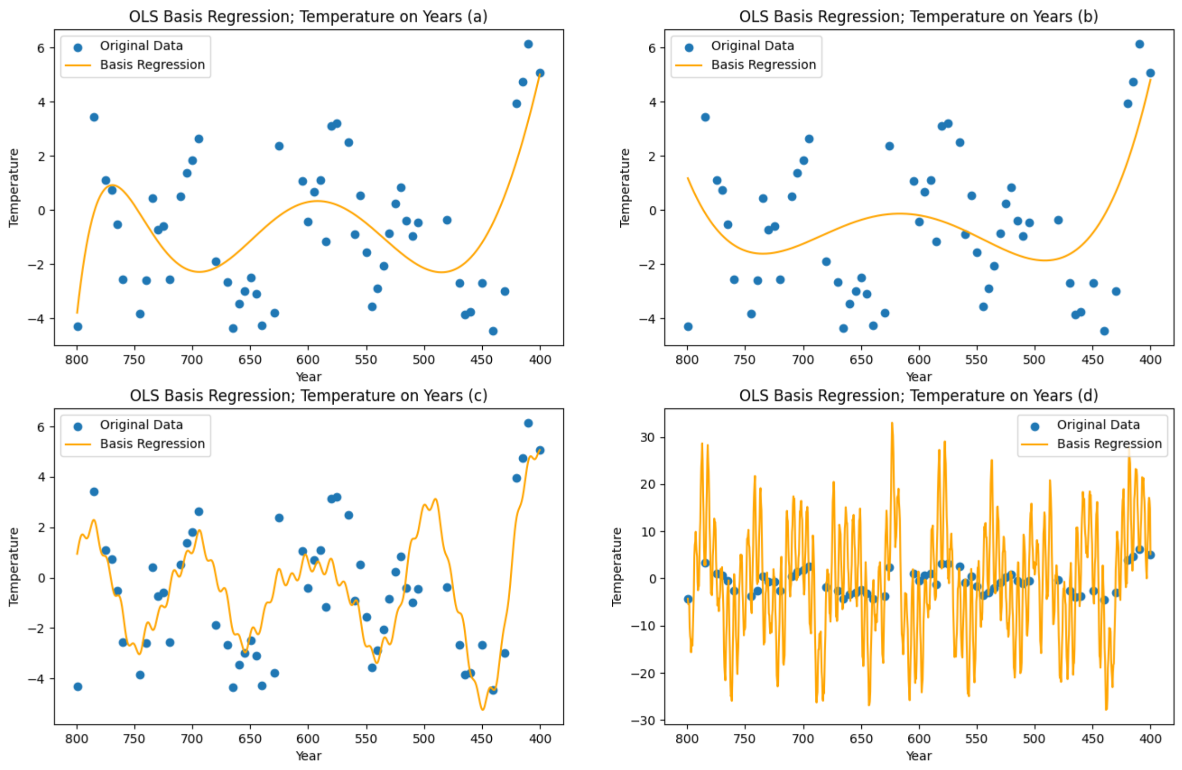


Figure 3: OLS regression plots for 4 different $\phi(\mathbf{X})$

2. The results are copied from the code and pasted below:

   (a) Train MSE: 4.83; Test MSE: 7.96

   Train Negative Log-Likelihood: 125.768; Test Negative Log-Likelihood: 63.256

   This fits the data well, as it follows a general trend without following the noise. The test and train MSE values are close in value, as are the train and test NLL values.

   (b) Train MSE: 5.53; Test MSE: 8.71

   Train Negative Log-Likelihood: 129.620; Test Negative Log-Likelihood: 64.035

   This fits the data well, as it follows a general trend without following the noise. The test and train MSE values are close in value, as are the train and test NLL values.

   (c) Train MSE: 2.88; Test MSE: 5.97

   Train Negative Log-Likelihood: 111.018; Test Negative Log-Likelihood: 62.098

   This fits the data well, as it follows a general trend without following the noise. The test and train MSE values are close in value, as are the train and test NLL values.

(d) Train MSE: 0.64; Test MSE: 58.90

Train Negative Log-Likelihood: 68.303; Test Negative Log-Likelihood: 1162.188

This overfits the data, as we can see that it goes through practically every data point in the plot. The test and train MSE values are also very different in value, as are the train and test NLL values, with both MSE and NLL dramatically higher for the test data than the train data.

3. This time round, we don't need to share the entire training set with them, since it's only needed to calculate the weights (which we've already fitted.) However, the weights were calculated using transformed data with a specific base, so that would have to be shared. This would have a space complexity of $O(N \cdot D)$, since the basis would be an array of size $N \times D$.

   Calculating the predicted value for an arbitrary datapoint, given the weights and the basis, involves a dot product between the basis array, of size $N \times D$, and $\mathbf{w}^T$ of size D, and so this has a time complexity of $O(N)$.

Problem 4 (Impact question: Building a descriptive (explanatory) linear regression model to understand the drivers of US energy consumption, to inform national policy decisions by the US President.)

Prompt: You are leading the machine learning team that is advising the US president. The US president is concerned about 3 things - climate change, the energy crisis in Europe and sustainable energy security in the US and asks you to help him understand what the driving factors of annual US energy consumption might be.

How would you build a regression model that can be used to explain the driving factors of the annual US energy consumption? Please answer the questions below by using concise language (350 - 700 words). Bullet points are appropriate.

This question is a conceptual question, and you are not required to implement the actual model. Yet, it is important that you think through your approach and its implications.

1. **Target variable:** What target variable would you choose and what would be its unit?
2. **Features:** List 5 possible features and explain your assumption why you think they might impact the target variable.
3. **Dataset size:** What should be the size of your dataset / covered time period? Why?
4. **Performance metric:** What metric would you use to assess the model's performance?
5. **Policy decision:** Explain one policy decision the US president could make based on your model.
6. **Trust:** What could be barriers for the US president to trust your model? List two possible barriers.
7. **Risk:** What happens if your model is wrong/inaccurate? List one real-world consequence.

Solution 4: Impact question

1. I would choose annual US energy consumption as the target variable, since that's what we are trying to explore. This is measured in KiloWatt Hours.

2. Considering the things that the US president is concerned about, 5 possible features we could use are:

   - US population. My assumption here is that there is a positive correlation between the population size and energy consumption.
   - Energy prices. This refers to how much individuals pay for electricity annually, and my assumption is that the more people have to pay, the more energy is being consumed.
   - Global levels of CO2. This is a big indicator of climate change, and although it is usually a feature that is affected by consumption of energy, it may also have an effect on the consumption of energy.
   - Electric vehicles owned. Climate change is commonly attributed to the usage of petrol/diesel powered vehicles. This is one of the aspects that the president was concerned about, and my assumption is that electric vehicles require less energy consumption.
   - Number of companies that adopted eco-friendly practices. My assumption is that eco-friendly practices reduce the consumption of electricity, and so there is a relationship between how much energy is used and how many companies are adopting these practices.

3. I would use data from 1990 to present, as the Third Industrial Revolution kicks off. This is I wouldn't want to include data that was before the main sources of energy consumption became consistent. In addition, I think data from 1990 is more likely to be reliable, with less noise. All in all, it's not dataset that's too large, but big enough to show a trend, if there is one.

4. MSE

5. The president could potentially decide on implementing policies that led to improving incentives for companies to implement eco-friendly practices.

6. He could be concerned that:

   - the data might not accurately reflect the effect of the features on the target variable–for example, the impact of a given company adopting eco-friendly on US energy consumption probably varies depending on how big the company is
   - the model's features are limited (with not enough focus on the international relationships and current affairs.)

7. It could lead to misinformed policy decisions that lead to unintended consequences–for example, missing features such as investments in renewable energy could lead to the US not keeping up with the rest of the world in global energy usage.

Name

Adam Mohamed

Collaborators and Resources

I worked mostly myself, and collaborated on some of the questions with Evan Jiang and Taj Gulati.

Calibration

This homework took about 15 hours to complete.