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:

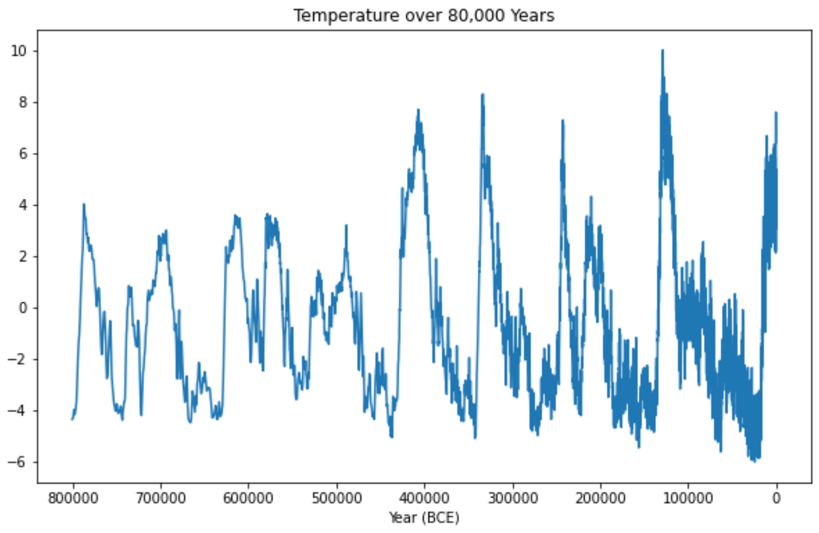
- `earth_temperature_sampled_train.csv`
- `earth_temperature_sampled_test.csv`

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.

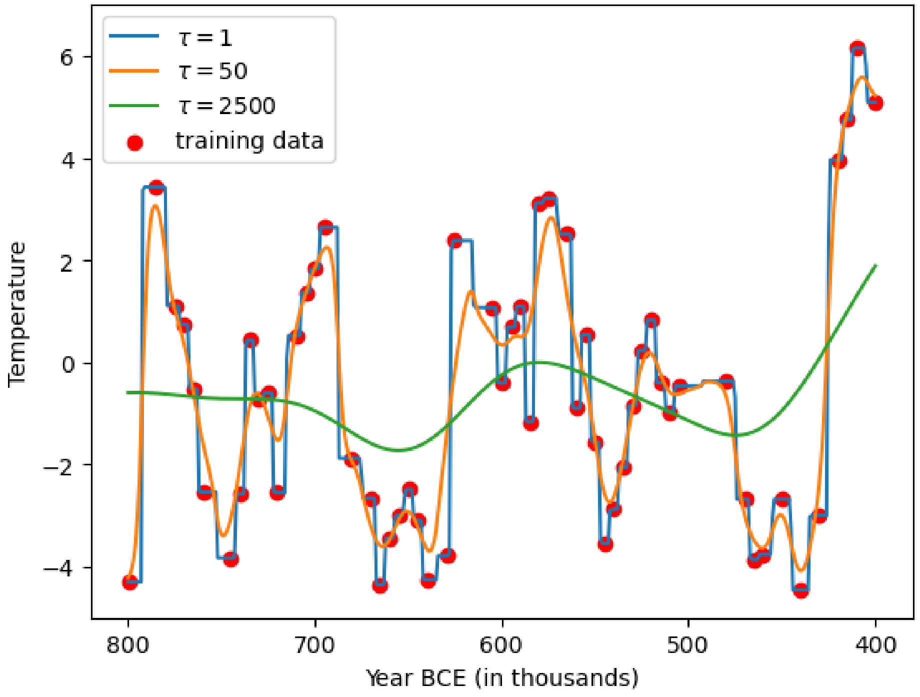
4. We now compute the MSE on the provided training set. Write Python code to compute the MSE with respect to the same lengthscales as in Part 1. Which model yields the lowest test set MSE? Is this consistent with what you observed in Part 1?

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?

1: Optimizing a Kernel

(1.1) We see that as τ increases, our model is penalized less for not perfectly fitting the available datapoints: when $\tau = 1$, we see that the model perfectly intersects each datapoint but is jagged and discontinuous, at $\tau = 50$, the interpolation between points is smoother and not all training points are intersected, and when $\tau = 2500$, the model is skewed less by the variations in temperature overall. The model with $\tau = 50$ is the model I would choose for predicting temperatures within our date range, because it smoothly interpolates between the datapoints while still capturing the changes in the temperature trends happening on the 1000-year scale.



(1.2) It's a bad idea to set the value of τ such that the MSE for the training set is minimized; this is a sure-fire way to overfit to the training dataset and end up with a model that hasn't actually learned anything useful more generally! The model would just learn to set *tau* to a very small number, which whould result in a train MSE approaching 0.

(1.3) Here we write the form of the MSE of f_τ over the test set as a function of the training set and test set:

$$\begin{aligned} MSE_{test} &= \frac{1}{M}\sum_{i=1}^{M}\left(y_i' - f_\tau(x_i')\right)^2 \\ &= \frac{1}{M}\sum_{i=1}^{M}\left(y_i' - \frac{\sum_{j=1}^{N} K_\tau(x_j, x_i')y_j}{\sum_{j=1}^{N} K_\tau(x_j, x_i')}\right)^2 \end{aligned}$$

(1.4) When we consider the test dataset, we see that choosing $\tau = 50$ yields the smallest MSE (see the table below) and thus we will choose this model. This is the same model that I chose in part 1.1.

(1.5) For someone to be able to make predictions with the kernelized regressor, they will need all of the training datapoints in addition to the value of τ, namely $\{(x_n, y_n)\}_{n=1}^{N}$. The storage space requirements will

τ	1	50	2500
test MSE	1.947	1.858	8.334

scale as $O(N)$, aka linearly. The time complexity of making a single prediction will scale as $O(N)$ because we need to sum across all the N training datapoints in both the numerator and denominator.

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.

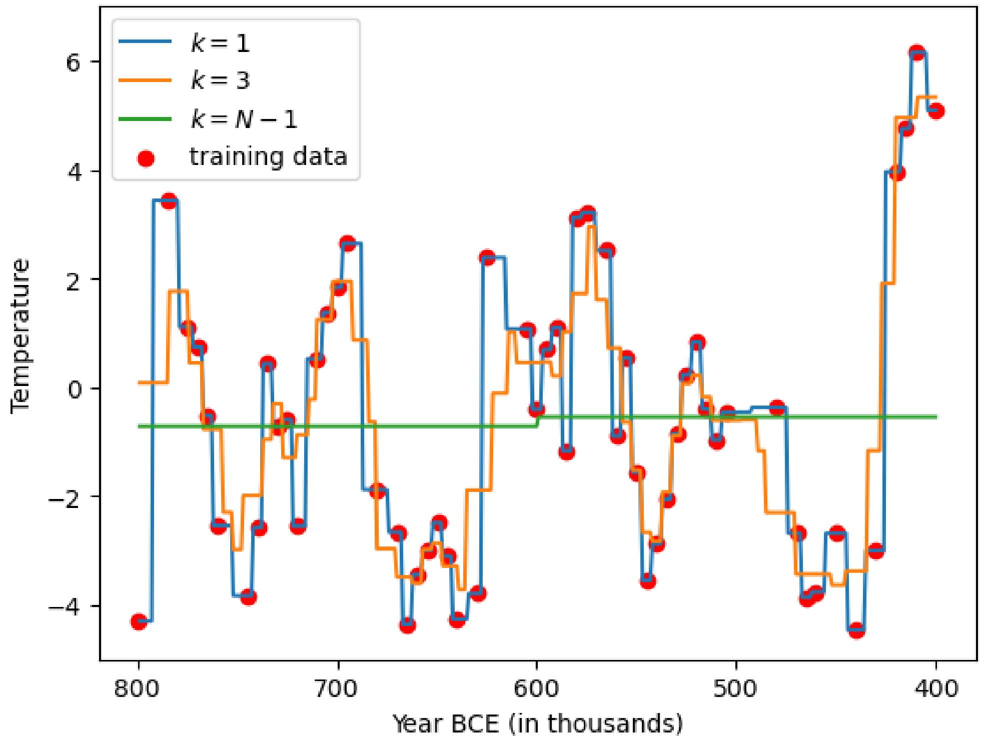
   Please **write your own implementation of kNN** for full credit. Do not use external libraries to find nearest neighbors.

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.



(2.1)

(2.2) In these plots, we see that the more neighbors used for generating the prediction, the less the prediction is pulled by "outlying" points. We see that for small k, the model predictions hew more closely to the local training data points whereas as k approaches the size of the training dataset, we start predicting close to the training temperature average across the whole domain.

We see that generally speaking, increased size of τ in the kernelized regression produces results similar to those seen when increasing the number of neighbors k in the kNN regression. Also, $\tau = 1$ looks remarkably similar to $k = 1$ and shows nearly identical (discontinuous) interpolation.

(2.3) Of the k-values assessed here, my favorite is $k = 3$ because it visually doesn't overfit as much as $k = 1$ while also capturing some of the variation across the domain (as opposed to $k = N - 1$). The test MSE is approximately 3.89.

(2.4) For my friend to be able to reproduce the same predictions as me, I will tell them the k I selected and the similarity kernel I was using (or I could just give them the function). In addition, in the special case when I know which years they which to make the prediction for, then I just need to provide the k nearest points which scales as $O(1)$ as a function of the training set size. However, if I don't know which years they want to predict over, then I will still need to provide the whole training dataset which would **scale as** $O(N)$. This is the space complexity of storing our model.

In terms of time complexity, the way I chose to implement isn't the most efficient because for each test point I chose to sort the array of kernel similarities and slice to find the k largest values (so this step was $O(NlogN)$ instead of $O(N)$ which we could get if we stored the k largest values as we looped through). In total, **the time complexity of computing a single prediction be** $O(NlogN)$ **as I implemented it.**

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 $\phi = (\phi_1, \dots, \phi_D)$. 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 $j = 1, \dots, 9$. $f(x) = \frac{x}{1.81 \cdot 10^2}$.

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

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

(d) $\phi_j(x) = \cos(f(x)/j)$ for $j = 1, \dots, 49$. $f(x) = \frac{x}{1.81 \cdot 10^{-1}}$. [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 3.1 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} (recall the formula from class!) and compute the negative log-likelihood for your train and test sets. The following derives the likelihood.

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

   - 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?

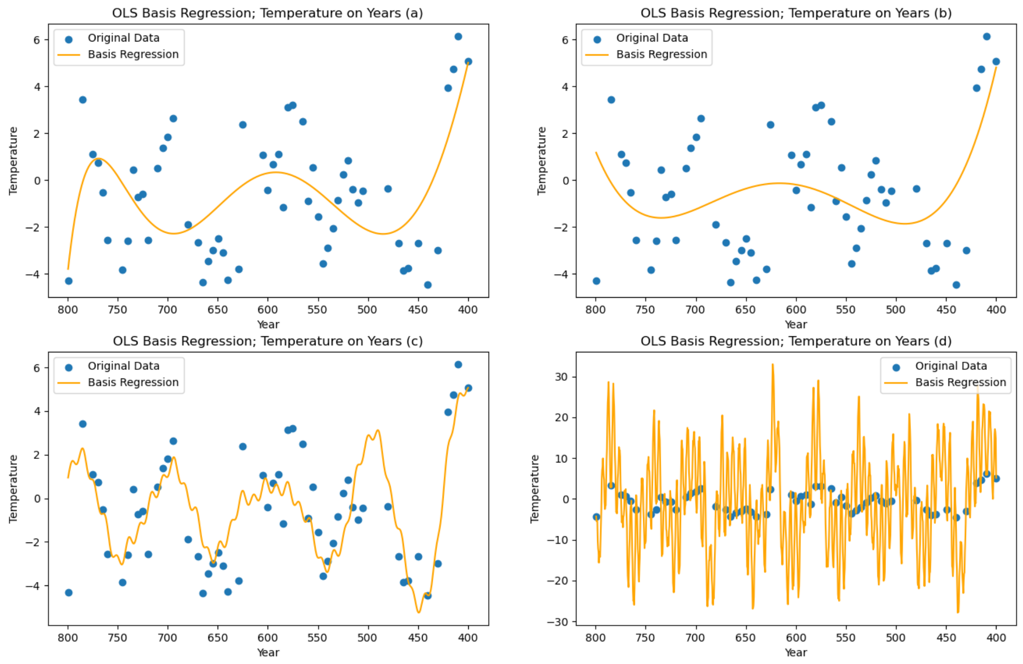
   How do these complexities compare to those of the kNN and kernelized regressor?

   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.

3: Modeling Climate Change 800,000 Years Ago

(3.1)



(3.2) We want to minimze the MSE and the negative log-likelihood w.r.t. the test dataset across our models (keeping common sense in mind, too). Both of these metrics are smallest when choosing basis (c): here, the training metrics are also relatively similar to the training metrics. This is especially evident when using the negative log-likelihood as our evaluation metric. NOTE: My second choice would be model (a) because it visuallt appears to capture most of the temperature trends on the scale of 1000s of years, but both its test and train MSE are significantly larger than that of basic (c).

Basis (d) shows evidence of overfitting because the metrics for the training dataset are much smaller than those for the test dataset. Bases (a) and (b) aren't too bad, but show some evidence of underfitting: the evaluation metrics are relatively large for both the training and testing datasets (the relatively large values for the training dataset compared to basis (c) are the classic warning sign of underfitting).

part	a	b	c	d
train MSE	4.83	5.53	2.88	0.64
test MSE	7.96	8.71	5.97	58.91
train NLL	127.522	130.364	119.590	110.502
test NLL	61.508	62.842	57.956	152.336

(3.3) The beauty of basis functions is how well it condenses the model while maintaining good performance: for my friend to use my basis model, all she needs is my D-dimensional weight vector and my basis function and scaling function so she can transform her data into the appropriate basis. She doesn't need access to the training data at all.

The space complexity of story this model is constant, $O(1)$: no matter how much data we use to train our model, it remains the same size.

Given an arbitrary datapoint, the time complexity to compute the predicted value using our basis regression is $O(1)$: all we need to do is generate the basis for the point (linear operation), multiply this with our known weights (also linear in time), and sum together.

Our basis regressor uses less time and space than either of our previous methods (the kNN regressor or the kernelized regressor) – you can't do better than constant-time and constant-space algorithms!

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.

4: Impact Question

1. **Target variable:** What target variable would you choose and what would be its unit?

   We will try to predict the net energy consumption of the USA in kilowatt-hours (kWh).

2. **Features:** List 5 possible features and explain your assumption why you think they might impact the target variable.

   Features to include: **month (or quarter)** (hot and cold months might see higher heating/coolding needs), population (more people likely to increase net consumption), **median US income** (increased income might be associated with higher consumption of items that require manufacturing), **cars per capita** (more cars per person means likely more fuel required), **average car MPG** (lower MPG results in more energy required per unit distance), **spending on public transport per capita** (public transport generally more efficient at moving people than car-centric transport).

3. **Dataset size:** What should be the size of your dataset / covered time period? Why?

   We will cover at least the past 30 years because we want to help the president understand the trends of the data, and because we also want to have enough data to average over random year-to-year fluctuations (and month-to-month fluctuations, since energy consumption likely systemically changes with the weather and consumer demands). Ideally, we will have monthly datapoints for all of our features but it is possible that we would only have access to quarterly reports.

4. **Performance metric:** What metric would you use to assess the model's performance?

   To assess the performance, I will use MSE on a held-out test set (e.g. either a random 80:20 train:test split or a temporal 80:20 train:test split). I will also generate bootstrap training datasets to assess the uncertainty in the weights of our linear model, which will help the president with interpretation.

5. **Policy decision:** Explain one policy decision the US president could make based on your model.

   From the coefficients (and their associated uncertainties), the US president will be able to identify which areas of the economy/country are consuming the most (and thus areas where reducing consumption by 5

6. **Trust:** What could be barriers for the US president to trust your model? List two possible barriers.

   The model may not achieve high test set performance, and thus may not be very useful. If the uncertainty in our coefficients is high, then the president may not trust that the model's implications are stable. Also, the model is relatively simple and may be missing features or interactive effects that could significantly change the 'best' policy decisions: this could make the president hesitant to use the model at all.

7. **Risk:** What happens if your model is wrong/inaccurate? List one real-world consequence.

   If our model is wrong or inaccurate, then the US could potentially waste enormous resources trying ineffectually to reduce energy consumption via policies doomed to fail. This wastes time, wastes money, and could also reduce the trust that the public has in the policymakers and potentially even the scientists who proposed the model (aka me).

Name

Gwen Miller

Collaborators and Resources

Whom did you work with, and did you use any resources beyond cs181-textbook and your notes?

I didn't work with anyone or use outside resources, except in debugging my kNN code where I looked up how to find the indexes of the k-smallest/largest values in a list (https://stackoverflow.com/questions/34226400/find-the-index-of-the-k-smallest-values-of-a-numpy-array).

Calibration

Approximately how long did this homework take you to complete (in hours)?

8 hr. Mostly for checking my work and expanding on my written answers.