Homework #1

Due: February 13, 2026 at 11:59 PM

Regression

Introduction

This homework is on different three different forms of regression: nearest neighbors regression, kernelized 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:

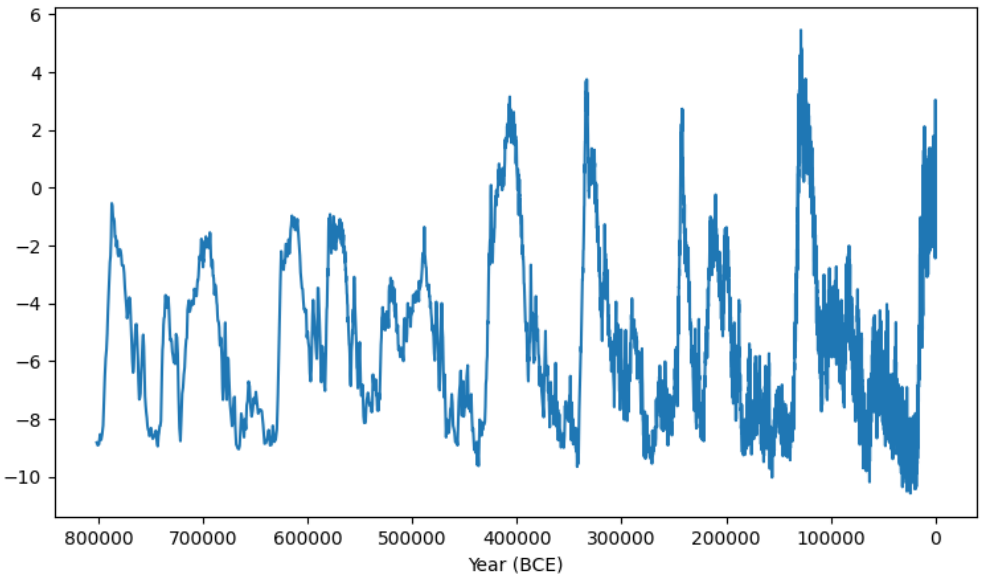
- `earth_temperature_sampled_train.csv`
- `earth_temperature_sampled_test.csv`

Each has two columns. The first column is the age of the ice core sample. The second column is the approximate difference in a year's temperature (K) from the average temperature of the 1,000 years preceding it. 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
399946,0.51
409980,1.57
```

And this is a visualization of the full dataset:



Due to the large magnitude of the years, we will work in terms of thousands of years BCE in these problems. 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

Resources and Submission Instructions

Please type your solutions after the corresponding problems using this LaTeX template, and start each main problem on a new page.

Please submit the writeup PDF to the Gradescope assignment 'HW1'. Remember to assign pages for each question. **You must include any plots in your writeup PDF.** . Please submit your LaTeXfile and code files to the Gradescope assignment 'HW1 - Supplemental.' The supplemental files will only be checked in special cases, e.g. honor code issues, etc. Your files should be named in the same way as we provide them in the repository, e.g. `hw1.pdf`, etc.

If you find that you are having trouble with the first couple problems, it may be helpful to refer to Section 0 notes and review some linear algebra and matrix calculus.

Problem 1 (kNN and Kernels, 35pts)

You will now implement two non-parametric regressions to model temperatures over time.

Make sure to include all required plots in your PDF. Passing all test cases does not guarantee that your solution is correct, and we encourage you to write your own.

1. 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.

   (a) The kNN implementation **has been provided for you** in the notebook. Run the cells to plot the results for $k = \{1, 3, N-1\}$, where N is the size of the dataset. Describe how the fits change with k. Please include your plot in your solution PDF.

   (b) Now, we will evaluate the quality of each model *quantitatively* by computing the error on the provided test set. Write Python code to compute the test MSE for each value of k. Report the values here. Which solution has the lowest MSE?

2. *Kernel-based regression* techniques are another form of non-parametric regression. 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 $\mathcal{D}_{\texttt{train}} = \{(x_n, y_n)\}_{n=1}^N$ are the training data points, and x^* is the point for which you want to make the prediction. The kernel $K_\tau(x, x')$ is a 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).

   (a) First, implement the `kernel_regressor` function in the notebook, and 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 $x = 400, 401, \ldots, 800$. **In no more than 10 lines**, describe how the fits change with τ. Please include your plot in your solution PDF.

   (b) Denote the test set as $\mathcal{D}_{\texttt{test}} = \{(x'_m, y'_m)\}_{m=1}^M$. Write down the expression for 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.

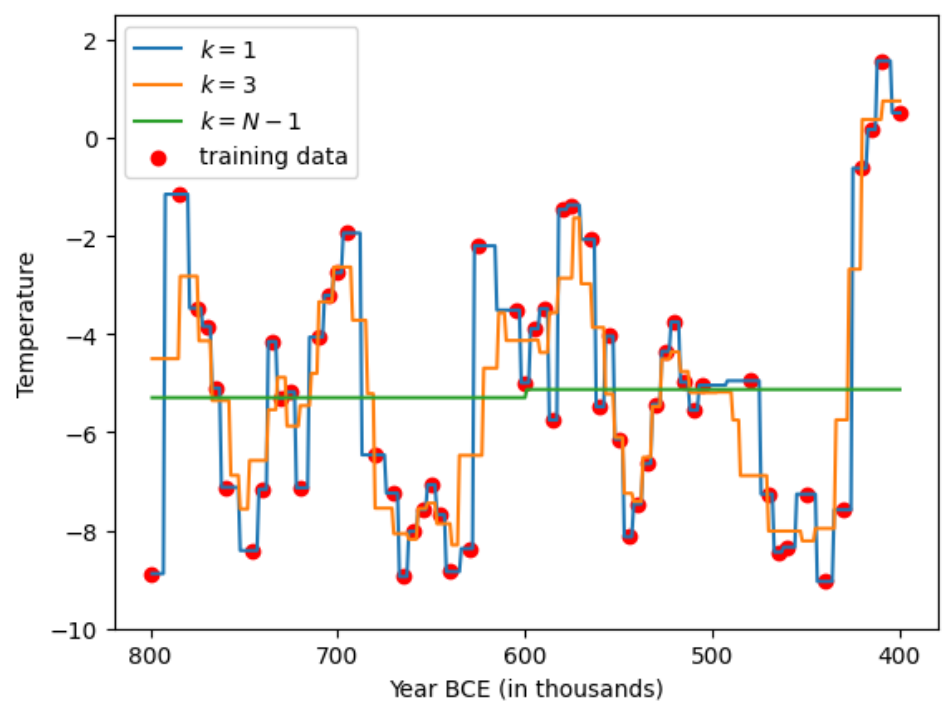
   (c) Compute the MSE on the provided test set for the three values of τ. Report the values here. Which model yields the lowest MSE? Conceptually, why is this the case? Why would choosing τ based on $\mathcal{D}_{\texttt{train}}$ rather than $\mathcal{D}_{\texttt{test}}$ be a bad idea?

(d) Describe the time and space complexity of both kernelized regression and kNN with respect to the size of the training set N. How, if at all, does the size of the model—everything that needs to be stored to make predictions—change with the size of the training set N? How, if at all, do the number of computations required to make a prediction for some input x^* change with the size of the training set N?.

(e) What is the exact form of $\lim_{\tau \to 0} f_\tau(x^*)$?

Solution: **Problem 1**:

1. (a) The fits become smoother as k increases. For $k = 1$, the fit is very jagged and closely follows the training data points, which leads to overfitting. For $k = 3$, the fit is smoother and captures the general trend of the data while still being responsive to local variations (though still somewhat overfitting). For $k = N - 1$, the fit is very smooth and essentially averages all the training data points, which can lead to underfitting as it does not capture local patterns in the data at all.

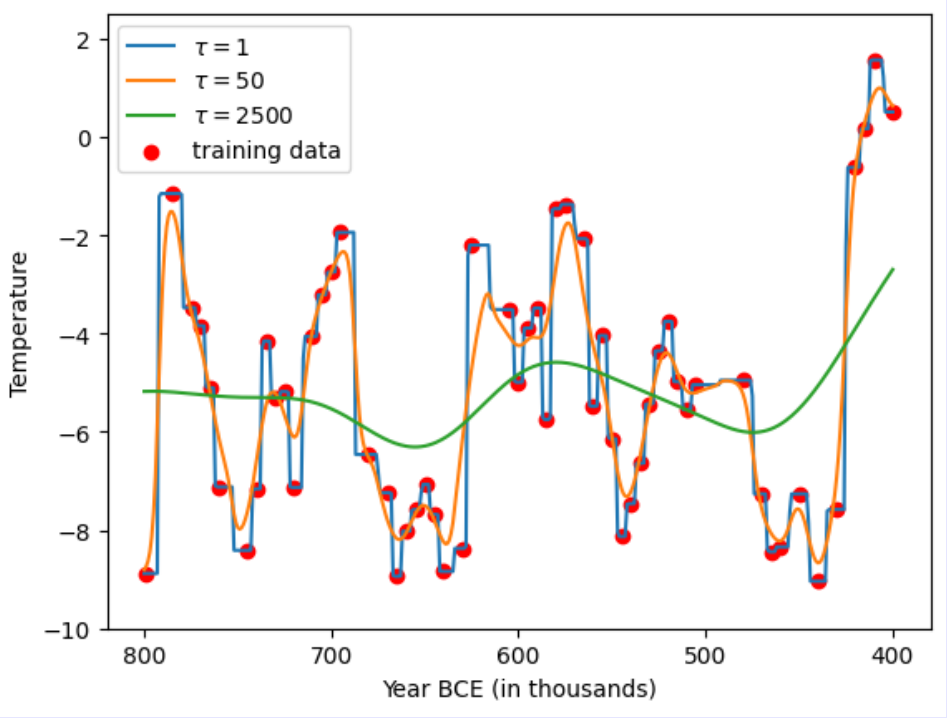


(b) The MSE values for each k are as follows:

- For $k = 1$: MSE $= 1.74$
- For $k = 3$: MSE $= 3.89$
- For $k = N - 1$: MSE $= 9.53$

The model with $k = 1$ yields the lowest MSE, basically because it's so biased to the data and is essentially overfitting everywhere.

2. (a) See the plot below:



The presence of τ controls the "smoothness" of the regrression. Looking at the three cases, we have a really small, medium, and large value for τ:

- In the small case, the small τ makes the kernel decay very quickly, resulting in the consideration of points super close to $x*$, leading to a spiky plot with overfitting and passing through every point.

- In the medium case, the medium τ is easing from the rapid decay a bit, showing a little bit less variance on the individual points and capturing less of the noise. It seems very responsive to the fluctuations in the graph right now, so maybe increasing it a bit more would make a better generalization of the data.

- In the high case, this is considerably underfitting with the τ value because of the need to average many points all together, leading to the line being much more flat (and seeming like a nice curve, but actually heavily biased).

(b) The MSE over a test set $\mathcal{D}_{\text{test}} = \{(x'_m, y'_m)\}_{m=1}^{M}$ is defined as:

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

Given the definition of the kernel-based regressor $f_\tau(x^*)$ using $\mathcal{D}_{\text{train}} = \{(x_n, y_n)\}_{n=1}^{N}$:

$$f_\tau(x^*) = \frac{\sum_{n=1}^{N} K_\tau(x_n, x^*) y_n}{\sum_{n=1}^{N} K_\tau(x_n, x^*)} \tag{2}$$

By substituting the regressor expression $f_\tau(x'_m)$ into the MSE formula, we obtain the expression

as a function of the training and test sets:

$$MSE = \frac{1}{M}\sum_{m=1}^{M}\left(y'_m - \frac{\sum_{n=1}^{N} K_\tau(x_n, x'_m)y_n}{\sum_{n=1}^{N} K_\tau(x_n, x'_m)}\right)^2 \tag{3}$$

(c) The computed MSE are as follows:

- For $\tau = 1$: 1.95
- For $\tau = 50$: 1.86
- For $\tau = 2500$: 8.33

Between them, the medium case (50) works best with the lowest MSE because it optimizes the bias-variance tradeoff. Choosing to go with the smallest possible τ will always minimize MSE, but it will fail to capture the trend of the graph because we're fully overfitting and heavily limiting the predictive capability of the model. Any new data will be off from predictions because we are heavily varied.

Using D_{train} would minimize the MSE, but this would perfectly interpolate the training data which causes $\tau \to 0$ to give an MSE of 0, causing the same issues noted above with overfitting.

(d) Their complexities are as follows:

- **Space Complexity:** For both models, the space complexity is $O(N)$. Since there is no "training" phase that compresses the data into a fixed number of parameters (like in linear regression), the entire training set $\mathcal{D}_{\text{train}}$ must be stored in memory to make future predictions. Thus, the model size grows linearly with the size of the training set N.
- **Time Complexity:** To make a single prediction for an input x^*, both algorithms require $O(N)$ computations.
  - In **kNN**, one must calculate the distance between x^* and all N training points to identify the k nearest neighbors.
  - In **Kernel Regression**, one must evaluate the kernel $K_\tau(x_n, x^*)$ for all $n = 1, \ldots, N$ to compute the weighted average.

  Since there is no compounding step when doing these calculations, the number of computations required to make a prediction increases linearly as the training set N grows.

(e) As $\tau \to 0$, the kernel $K_\tau(x_n, x^*)$ becomes infinitely "peaked." If x^* is exactly equal to a training point x_i, $K_\tau(x_i, x^*)$ approaches 1 while all other $K_\tau(x_{n\neq i}, x^*)$ approach 0. If x^* is not a training point, the kernel that decays the slowest will dominate the sum.

The exact form of the limit is:

$$\lim_{\tau \to 0} f_\tau(x^*) = y_i \quad \text{where } i = \arg\min_n (x_n - x^*)^2 \tag{4}$$

As the lengthscale goes to zero, the kernel regressor recovers the **1-Nearest Neighbor (1-NN)** estimator. The prediction at x^* simply becomes the label y_n of the training point x_n that is spatially closest to it. It behaves something like a smaller and smaller window of points to consider, and so the smallest possible window will take into account the only closest point, hence the 1-NN.

Problem 2 (Geometric Least Squares, 20pts)

Linear regression can be understood using geometric intuition in $\mathbb{R}^n$. The design matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ spans a subspace $C(\mathbf{X})$, the column space of $\mathbf{X}$ (referred to in lecture as column span) which is a subspace of $\mathbb{R}^N$. If you wish to review the concept of a column space, consider visiting Section 0 notes.

Fitting by linear regression, sometimes called *ordinary least-squares* (OLS), is just projecting the observation vector $\mathbf{y} \in \mathbb{R}^N$ orthogonally onto that subspace. Lecture 2 slides provide a good graphic to visualize this, see the slide titled "Geometric Interpretation." From lecture, we also learned that $\hat{\mathbf{y}}$ lives in $C(\mathbf{X})$ and the residual $\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}}$ lives in the orthogonal complement.

1. Let $\mathbf{X} \in \mathbb{R}^{N \times D}$ have full column rank d and $\mathbf{y} \in \mathbb{R}^N$. Let the OLS estimator be $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} X^\top \mathbf{y}$ and the fitted vector $\mathbf{y} = \mathbf{X}\mathbf{w}^*$. Prove that $\hat{\mathbf{y}}$ is the *orthogonal projection* of y onto $C(\mathbf{X})$. In other words, show that $\hat{\mathbf{y}} \in C(\mathbf{X})$ and $\mathbf{y} - \hat{\mathbf{y}}$ is orthogonal to $C(\mathbf{X})$. *Hint: To show orthogonality, look at the gradient of $\mathcal{L}$, the loss, with respect to* $\mathbf{w}$.

2. Prove that among all vectors in $C(\mathbf{X})$, the fitted vector $\hat{\mathbf{y}}$ minimizes the Euclidean distance to $\mathbf{y}$. In other words, that for every vector $\mathbf{v} \in C(\mathbf{X})$:

   $$\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 \leq \|\mathbf{y} - \mathbf{v}\|_2^2$$

   Looking back at lecture, this is the formal proof of the phenomenon discussed in the image. *Hint: For two vectors, $\mathbf{v}, \mathbf{w}$, if $\mathbf{v}$ is orthogonal to $\mathbf{w}$, denoted as $\mathbf{v} \perp \mathbf{w}$, then $\|\mathbf{v} - \mathbf{w}\|_2^2 = \|\mathbf{v}\|_2^2 + \|\mathbf{w}\|_2^2$ (Pythagorean theorem).*

3. In lecture, we defined the projection matrix, $\mathbf{P} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$, which projects onto the subspace $C(\mathbf{X})$. The matrix $\mathbf{P}$ is often called the *hat matrix* because it maps $\mathbf{y}$ to its fitted values $\hat{\mathbf{y}} = \mathbf{P}\mathbf{y}$, i.e., it "puts a hat" on $\mathbf{y}$. Prove the following properties of $\mathbf{P}$:

   - Symmetry: $\mathbf{P}^\top = \mathbf{P}$
   - Idempotence: $\mathbf{P}^2 = \mathbf{P}$
   - Rank and Trace: $\text{rank}(\mathbf{P}) = d$ and $\text{trace}(\mathbf{P}) = d$.

   Hint: You may use the fact that any idempotent matrix has equal rank and trace. You do not need to prove this, but it may be helpful to think about why this is true.

4. Suppose you fit your model as in Problem 5.1. You observe that the **residual plot** exhibits a clear parabolic (U-shaped) pattern rather than random scatter around zero (as seen in lecture). Give a geometric interpretation of this phenomenon in terms of projection onto the column space of the design matrix. Also, explain how adding a quadratic basis function affects the geometry of the regression problem and the residuals.

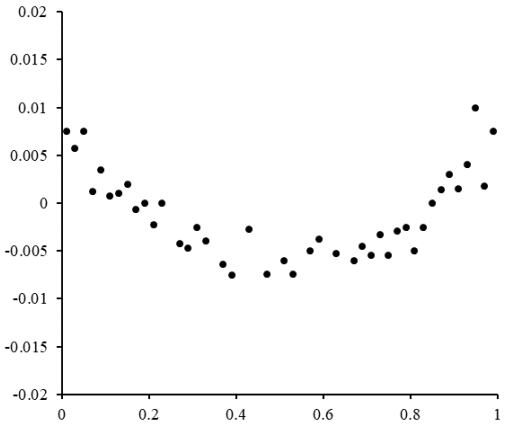


Figure 1: An example residual plot with the input variable x on the horizontal axis and residuals $y - \hat{y}$ on the vertical axis.

Solution: **Problem 2:**

1. **Proof: $\hat{\mathbf{y}}$ is the orthogonal projection of $\mathbf{y}$ onto $C(\mathbf{X})$**

   First, we show $\hat{\mathbf{y}} \in C(\mathbf{X})$. By definition, $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}^*$. Since $\hat{\mathbf{y}}$ is expressed as a linear combination of the columns of $\mathbf{X}$, it must lie in the column space $C(\mathbf{X})$.

   Next, we show that the residual $\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}}$ is orthogonal to $C(\mathbf{X})$. We consider the gradient of the loss function $\mathcal{L}(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$:

   $$\nabla_{\mathbf{w}} \mathcal{L} = -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) \tag{5}$$

   At the optimal solution $\mathbf{w}^*$, the gradient must be zero:

   $$-2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}^*) = 0 \implies \mathbf{X}^\top(\mathbf{y} - \hat{\mathbf{y}}) = 0 \tag{6}$$

   The equation $\mathbf{X}^\top(\mathbf{y} - \hat{\mathbf{y}}) = 0$ implies that the vector $(\mathbf{y} - \hat{\mathbf{y}})$ is orthogonal to every column of $\mathbf{X}$. Since the columns of $\mathbf{X}$ span $C(\mathbf{X})$, the residual is orthogonal to the entire subspace $C(\mathbf{X})$.

2. **Proof: $\hat{\mathbf{y}}$ minimizes Euclidean distance to $\mathbf{y}$**

   Let $\mathbf{v}$ be any vector in $C(\mathbf{X})$. We can write the distance from $\mathbf{y}$ to $\mathbf{v}$ as:

   $$\|\mathbf{y} - \mathbf{v}\|_2^2 = \|(\mathbf{y} - \hat{\mathbf{y}}) + (\hat{\mathbf{y}} - \mathbf{v})\|_2^2 \tag{7}$$

   Since $\hat{\mathbf{y}} \in C(\mathbf{X})$ and $\mathbf{v} \in C(\mathbf{X})$, their difference $(\hat{\mathbf{y}} - \mathbf{v})$ is also in $C(\mathbf{X})$. From Part 1, we know $(\mathbf{y} - \hat{\mathbf{y}}) \perp C(\mathbf{X})$, so $(\mathbf{y} - \hat{\mathbf{y}}) \perp (\hat{\mathbf{y}} - \mathbf{v})$. By the Pythagorean Theorem:

   $$\|\mathbf{y} - \mathbf{v}\|_2^2 = \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 + \|\hat{\mathbf{y}} - \mathbf{v}\|_2^2 \tag{8}$$

   Because $\|\hat{\mathbf{y}} - \mathbf{v}\|_2^2 \geq 0$ for all $\mathbf{v}$, it follows that $\|\mathbf{y} - \mathbf{v}\|_2^2 \geq \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$. Thus, $\hat{\mathbf{y}}$ is the vector in $C(\mathbf{X})$ closest to $\mathbf{y}$.

3. **Properties of the Projection Matrix $\mathbf{P} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$**

   - **Symmetry:** We show that $\mathbf{P}^\top = \mathbf{P}$. Using the property that $(\mathbf{AB})^\top = \mathbf{B}^\top\mathbf{A}^\top$:

   $$\begin{aligned}\mathbf{P}^\top &= \left(\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\right)^\top \\ &= (\mathbf{X}^\top)^\top \left((\mathbf{X}^\top\mathbf{X})^{-1}\right)^\top \mathbf{X}^\top \\ &= \mathbf{X}\left((\mathbf{X}^\top\mathbf{X})^\top\right)^{-1}\mathbf{X}^\top\end{aligned}$$

   Since $\mathbf{X}^\top\mathbf{X}$ is symmetric ($(\mathbf{X}^\top\mathbf{X})^\top = \mathbf{X}^\top\mathbf{X}$), its inverse is also symmetric. Thus:

   $$\mathbf{P}^\top = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top = \mathbf{P}$$

- **Idempotence:** We show that $\mathbf{P}^2 = \mathbf{P}$. Multiplying the matrix by itself:

$$\begin{aligned}\mathbf{P}^2 &= \left(\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\right)\left(\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\right) \\ &= \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\underbrace{(\mathbf{X}^\top\mathbf{X})(\mathbf{X}^\top\mathbf{X})^{-1}}_{\mathbf{I}}\mathbf{X}^\top \\ &= \mathbf{X}\mathbf{I}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top \\ &= \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top = \mathbf{P}\end{aligned}$$

- **Rank/Trace:** Since $\mathbf{P}$ is idempotent, its rank is equal to its trace ($\text{rank}(\mathbf{P}) = \text{tr}(\mathbf{P})$) as the problem hinted. Alternatively, to find the trace, we use the cyclic property (and I found this property online, then derived off of it) $\text{tr}(\mathbf{ABC}) = \text{tr}(\mathbf{BCA})$:

$$\begin{aligned}\text{tr}(\mathbf{P}) &= \text{tr}\left(\mathbf{X}\left[(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\right]\right) \\ &= \text{tr}\left(\left[(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\right]\mathbf{X}\right) \\ &= \text{tr}\left((\mathbf{X}^\top\mathbf{X})^{-1}(\mathbf{X}^\top\mathbf{X})\right) \\ &= \text{tr}(\mathbf{I}_d) = d\end{aligned}$$

  Because $\mathbf{P}$ acts like the identity matrix here, $\text{rank}(\mathbf{P}) = \text{tr}(\mathbf{P}) = d$.

4. **Residual Plot Interpretation**

   A parabolic (U-shaped) pattern in a residual plot indicates a **non-linear relationship** that the linear model has failed to capture. This suggests the presence of a quadratic trend in the data. To address this, one should perform a **basis expansion** to allow the model to fit the underlying curvature. This expands the column space of the design matrix to include quadratic terms, enabling the model to capture the non-linear relationship and thus reducing the systematic pattern in the residuals.

Problem 3 (Basis Regression, 30pts)

We now implement some linear regression models for the temperature. If we just directly use the data as given to us, we would only have a one dimensional input to our model, the year. To create a more expressive linear model, we will introduce basis functions.

Make sure to include all required plots in your PDF.

1. We will first implement the four basis regressions below. Note that we introduce an addition transform f (already into the provided notebook) to address concerns about numerical instabilities.

   (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.

   Then, create a plot of the fitted regression line for each basis against a scatter plot of the training data. Boilerplate plotting code is provided in the notebook—you will only need to finish up a part of it. **All you need to include in your writeup for this part are these four plots.**

2. Now we have trained each of our basis regressions. For each basis regression, compute the MSE on the test set. Discuss: do any of the bases seem to overfit? Underfit? Why?

3. Briefly describe what purpose the transforms ϕ serve: why are they helpful?

4. As in Problem 1, describe the space and time complexity of linear regression. How does what is stored to compute predictions change with the size of the training set N and the number of features D? How does the computation needed to compute the prediction for a new input depend on the size of the training set N? How do these complexities compare to those of the kNN and kernelized regressor?

5. Briefly compare and constrast the different regressors: kNN, kernelized regression, and linear regression (with bases). Are some regressions clearly worse than others? Is there one best regression? How would you use the fact that you have these multiple regression functions?

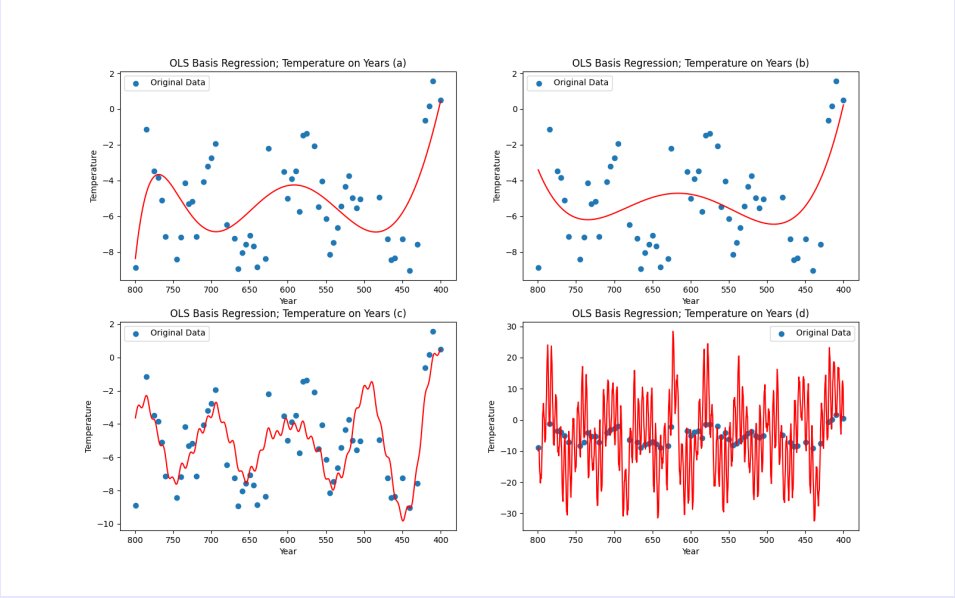
Note: You may be concerned that we are using a different set of inputs $\mathbf{X}$ for each basis (a)-(d), since it could seem as though this prevents us from being able to directly compare the MSE of the models since we are using different data as input. But this is not an issue, since each transformation is considered as

being a part of our model. This contrasts with transformations that cause the variance of the target $\mathbf{y}$ to be different (such as standardization); in these cases the MSE can no longer be directly compared.

[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.

Solution: **Problem 3:**

1. Below are the four plots for the fitted regression line for each basis against a scatter plot of the training data:



2. The MSE values for each basis regression are as follows:

   - MSE for Basis (a): 7.9559. This seems to underfit the data as the degree is still quite low to fully capture the fluctuations in the data towards the middle values.

   - MSE for Basis (b): 8.7081. This is even more udnerfitting than (a) because it appears to be too generallized and is not responsive enough to the fluctuations in the data.

   - MSE for Basis (c): 5.9670. This is the best fit of the three so far, as it seems to be capturing following the data quite well, but still general enough to not be overfitting.

   - MSE for Basis (d): 58.9367. This is a clear case of overfitting, as the model is so flexible that it is capturing the noise in the data, leading to a very high MSE on the test set. It makes sense in a Fourier way to cover every data point, but not interpretable at all for generalization.

3. The basis transformations ϕ serve to map the original input data into a higher-dimensional feature space, which increases the expressivity of the model. This is helpful because it allows a linear regression framework to capture complex, non-linear relationships while remaining linear in the parameters w. By doing so, we can still utilize the computationally efficient closed-form solution $(\Phi^\top \Phi)^{-1}\Phi^\top \mathbf{y}$ to find the optimal model, even though the resulting fit is non-linear relative to the original input x. More features means more flexibility to be expressive, but obviously not too much, otherwise we run into overfitting issues.

4. The time and space complexity is as follows:

   - **Space Complexity:** Linear basis regression is a parametric method. Once the model is trained, we only need to store the weight vector $\mathbf{w} \in \mathbb{R}^D$, where D is the number of basis

functions. Therefore, the space complexity is $O(D)$. Unlike kNN or Kernel Regression, the size of the stored model **does not change** with the size of the training set N.

- **Time Complexity (Prediction):** To make a prediction for a new input x^*, we must compute the D basis transformations and perform a dot product $\phi(x^*)^\top \mathbf{w}$. This requires $O(D)$ operations. Notably, the number of computations **does not depend on** N, making this model significantly more efficient for inference on large datasets than memory-based methods.

Overall, the theme here is that the parametric nature of linear regression with basis allows for a fixed-size model that is independent of the training set size, leading to efficient storage and fast predictions, while kNN and Kernel Regression require storing the entire training set and have prediction times that grow with N.

5. The fundamental difference between Linear Regression and kNN/Kernel Regression is the distinction between parametric and non-parametric modeling.

   - **Computational Efficiency:** Linear regression is significantly more efficient for large-scale inference. Since it only stores a weight vector $\mathbf{w} \in \mathbb{R}^D$, its prediction time is $O(D)$ and does not grow with the training set size N. Conversely, kNN and Kernel regression require $O(ND)$ time (we had $D = 1$ for part 1) for prediction because they must consult every training point to compute a result.
   - **Memory Requirements:** Linear basis regression has a constant space complexity of $O(D)$ regardless of N. Non-parametric methods like kNN have a space complexity of $O(ND)$, as the entire dataset must be stored to perform predictions.
   - **Model Flexibility:** While kNN and Kernel regression are highly flexible and can adapt to complex patterns without explicit feature engineering, Linear basis regression is restricted by the expressivity of the chosen basis functions ϕ.

   Overall, what's better or worse depends on the goal of what we're trying to do. If we want runtime efficiency and a compact model, linear regression with an appropriate basis is often preferable, but we need to choose a good set of basis functions, otherwise the model may not fit well. If we want maximum flexibility and are less concerned about computational costs, non-parametric methods may be more suitable. In practice, one might use the multiple regression functions to compare performance and select the best model based on validation metrics or to ensemble them for improved predictions.

Problem 4 (Probablistic View of Regression and Regularization, 30pts)

Finally, we will explore an alternative view of linear regression to what was introduced in lecture. This view will be probabilistic. We will also introduce Bayesian regression under this probabilistic view. We will explore its connection to regularization for linear models, and then fit a regularized model to the temperature data. The probabilistic interpretation of linear regression is explored in more detail in the course notes under section 2.6.2, but we have also tried to make this question self-contained with all necessary content.

Recall that linear regression involves having N labeled data points, say, $(\mathbf{x}_n, y_n)$ for $n \in \{1, \ldots, N\}$. A probabilistic view of the linear regression problem supposes that the data actually came from a probabilistic model:

$$y_n = \mathbf{w}^\top \mathbf{x}_n + \epsilon_n, \quad \epsilon_n \sim \mathcal{N}(0, \sigma^2).$$

That is, we assume that there exists a set of coefficients $\mathbf{w}$ such that given data $\mathbf{x}_n$, the corresponding y_n results from taking the $\mathbf{w}^\top \mathbf{x}_n$ and adding some random noise ϵ_n. Here, we assume the noise is normally distributed with known mean and variance. The introduction of noise into the model accounts for the possibility of scatter, i.e., when the data does not literally follow a perfect line. It is shown in the aforementioned section of the course notes that under this probabilistic model, the data likelihood $p(\mathbf{y}|\mathbf{w}, \mathbf{X})$ is maximized by $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$, which, as we already saw in class, also minimizes the squared error. So, amazingly, the probabilistic view of regression leads to the view we saw in lecture, where we are trying to minimize a prediction error.

Now, Bayesian regression takes this probablistic view a step further. You may recall that Bayesian statistics involves choosing a prior distribution for the parameters, here $\mathbf{w}$, based on our prior beliefs. So, in Bayesian regression, we additionally assume the weights are distributed $p(\mathbf{w})$ and fit the weights $\mathbf{w}$ by maximizing the posterior likelihood

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w}, \mathbf{X}) p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}.$$

Note that since we maximize with respect to $\mathbf{w}$, it suffices to just maximize the numerator, while the denominator term does not need to be computed.

1. Suppose $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \frac{\sigma^2}{\lambda}\mathbf{I})$. Show that maximizing the posterior likelihood is equivalent to minimizing the loss function

   $$\mathcal{L}_{ridge}(\mathbf{w}) = \frac{1}{2}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2.$$

   For those who are familiar, note that minimizing $\mathcal{L}_{ridge}(\mathbf{w})$ is exactly what regression with ridge regularization does.

   Hint: You don't need to explicitly solve for the form of the maximizer/minimizer to show that the optimization problems are equivalent.

2. Solve for the value of $\mathbf{w}$ that minimizes $\mathcal{L}_{ridge}(\mathbf{w})$.

3. The Laplace distribution has the PDF

   $$L(a, b) = \frac{1}{2b} \exp\left(-\frac{|x - a|}{b}\right)$$

Show that if all $w_d \sim L\left(0, \frac{2\sigma^2}{\lambda}\right)$, maximizing the posterior likelihood is equivalent to minimizing the loss function

$$\mathcal{L}_{lasso}(\mathbf{w}) = \frac{1}{2}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \frac{\lambda}{2}||\mathbf{w}||_1.$$

For those who are familiar, note that minimizing $\mathcal{L}_{lasso}(\mathbf{w})$ is exactly what regression with LASSO regularization does.

4. The LASSO estimator is the value of $\mathbf{w}$ that minimizes $\mathcal{L}_{lasso}(\mathbf{w})$? It is very useful in certain real-world scenarios. Why is there no general closed form for the LASSO estimator?

5. Since there is no general closed form for the LASSO estimator $\mathbf{w}$, we use numerical methods for estimating $\mathbf{w}$. One approach is to use *coordinate descent*, which works as follows:

   (a) Initialize $\mathbf{w} = \mathbf{w}_0$.

   (b) For each $d = 1, \ldots, D$ do the following 2 steps consecutively:

      i. Compute $\rho_d = \tilde{\mathbf{x}}_d^\top(\mathbf{y} - (\mathbf{X}\mathbf{w} - w_d\tilde{\mathbf{x}}_d))$. We define $\tilde{\mathbf{x}}_d$ as the d-th column of $\mathbf{X}$.

      ii. If $d = 1$, set $w_1 = \frac{\rho_1}{||\tilde{\mathbf{x}}_1||_2^2}$. Otherwise if $d \neq 1$, compute $w_d = \frac{\text{sign}(\rho_d)\max\left\{|\rho_d| - \frac{\lambda}{2}, 0\right\}}{||\tilde{\mathbf{x}}_d||_2^2}$.

   (c) Repeat step (b) until convergence or the maximum number of iterations is reached.

   Implement the `find_lasso_weights` function according to the above algorithm, letting $\mathbf{w}_0$ be a vector of ones and the max number of iterations be 5000. Then, fit models with $\lambda = 1, 10$ to basis (d) from Problem 3 and plot the predictions on the train set. Finally, compute the test MSE's. You will need to do some preprocessing, but a completed helper function for this is already provided. How do the graphs and errors compare to those for the unregularized (i.e., vanilla) basis (d) model?

Solution: **Problem 4**:

1. **Proof: Maximizing the posterior likelihood is equivalent to minimizing $\mathcal{L}_{ridge}(\mathbf{w})$**

   We start with the posterior distribution of the weights given the data:

   $$p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})} \tag{9}$$

   Since the denominator $p(\mathbf{y}|\mathbf{X})$ is independent of $\mathbf{w}$, maximizing the posterior is equivalent to maximizing the numerator $p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})$.

   Using the likelihood $y_n|\mathbf{x}_n,\mathbf{w} \sim \mathcal{N}(\mathbf{w}^\top\mathbf{x}_n,\sigma^2)$ and the prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0},\frac{\sigma^2}{\lambda}\mathbf{I})$ (where each data point is i.i.d.), we have:

   $$p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w}) = \left(\prod_{n=1}^{N}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_n-\mathbf{w}^\top\mathbf{x}_n)^2}{2\sigma^2}\right)\right)\cdot\frac{1}{\sqrt{(2\pi)^D|\frac{\sigma^2}{\lambda}\mathbf{I}|}}\exp\left(-\frac{\lambda}{2\sigma^2}||\mathbf{w}||_2^2\right) \tag{10}$$

   Taking the log-posterior and dropping terms that do not depend on $\mathbf{w}$ because we only care about the numerator:

   $$\begin{aligned}\log p(\mathbf{w}|\mathbf{X},\mathbf{y}) &\propto \log p(\mathbf{y}|\mathbf{X},\mathbf{w}) + \log p(\mathbf{w})\\ &= -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n-\mathbf{w}^\top\mathbf{x}_n)^2 - \frac{\lambda}{2\sigma^2}||\mathbf{w}||_2^2 + \text{const}\\ &= -\frac{1}{2\sigma^2}\left(||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_2^2\right) + \text{const}\end{aligned}$$

   Maximizing this log-likelihood is equivalent to minimizing the negative of the expression. By scaling by the constant σ^2 (which does not change the argmin), we get:

   $$\arg\max_{\mathbf{w}} p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \arg\min_{\mathbf{w}}\left(\frac{1}{2}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2\right) \tag{11}$$

   The right-hand side is exactly $\mathcal{L}_{ridge}(\mathbf{w})$. Thus, the maximizing the posterior likelihood is equivalent to minimizing the Ridge loss function.

2. To find the weight vector $\mathbf{w}^*$ that minimizes the Ridge loss function, we take the gradient of the loss with respect to $\mathbf{w}$ and set it to zero.

   Starting with the loss function:

   $$\mathcal{L}_{ridge}(\mathbf{w}) = \frac{1}{2}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2 \tag{12}$$

   Expanding the squared terms:

   $$\mathcal{L}_{ridge}(\mathbf{w}) = \frac{1}{2}(\mathbf{y}-\mathbf{X}\mathbf{w})^\top(\mathbf{y}-\mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} \tag{13}$$

Taking the gradient $\nabla_{\mathbf{w}}$:

$$\begin{aligned}\nabla_{\mathbf{w}}\mathcal{L}_{ridge}(\mathbf{w}) &= \nabla_{\mathbf{w}}\left(\frac{1}{2}\mathbf{y}^\top\mathbf{y} - \mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \frac{1}{2}\mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w} + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w}\right) \\ &= -\mathbf{X}^\top\mathbf{y} + \mathbf{X}^\top\mathbf{X}\mathbf{w} + \lambda\mathbf{w}\end{aligned}$$

Setting the gradient to the zero vector $\mathbf{0}$, we can end up comparing the two set of terms:

$$\begin{aligned}\mathbf{X}^\top\mathbf{X}\mathbf{w} + \lambda\mathbf{w} &= \mathbf{X}^\top\mathbf{y} \\ (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})\mathbf{w} &= \mathbf{X}^\top\mathbf{y}\end{aligned}$$

Solving for $\mathbf{w}$, we obtain the closed-form solution:

$$\mathbf{w}^* = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y} \tag{14}$$

This allows us to obtain the closed form for the maximized $\mathbf{w}^*$, with the presence of the regularization term λ to minimize overfitting in $\mathcal{L}_{ridge}$.

3. **Proof: Maximizing the posterior likelihood with a Laplace prior is equivalent to minimizing $\mathcal{L}_{lasso}(\mathbf{w})$ (similar to 4.1):**

   We start with the posterior distribution of the weights given the data:

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})} \tag{15}$$

   We maximize the numerator $p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})$ since the denominator is independent of $\mathbf{w}$.

   Using the Gaussian likelihood for i.i.d. observations and the Laplace prior given, $w_d \sim L\left(0, \frac{2\sigma^2}{\lambda}\right)$, where $p(w_d) = \frac{1}{2b}\exp\left(-\frac{|w_d|}{b}\right)$ with $b = \frac{2\sigma^2}{\lambda}$, we have:

$$p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w}) = \left(\prod_{n=1}^{N}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_n - \mathbf{w}^\top\mathbf{x}_n)^2}{2\sigma^2}\right)\right)\cdot\left(\prod_{d=1}^{D}\frac{1}{2b}\exp\left(-\frac{\lambda|w_d|}{2\sigma^2}\right)\right) \tag{16}$$

   Taking the natural logarithm and dropping constants that do not depend on $\mathbf{w}$:

$$\begin{aligned}\log p(\mathbf{w}|\mathbf{X},\mathbf{y}) &\propto \log p(\mathbf{y}|\mathbf{X},\mathbf{w}) + \log p(\mathbf{w}) \\ &= -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \mathbf{w}^\top\mathbf{x}_n)^2 - \sum_{d=1}^{D}\frac{\lambda|w_d|}{2\sigma^2} + \text{const} \\ &= -\frac{1}{2\sigma^2}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 - \frac{\lambda}{2\sigma^2}||\mathbf{w}||_1 + \text{const}\end{aligned}$$

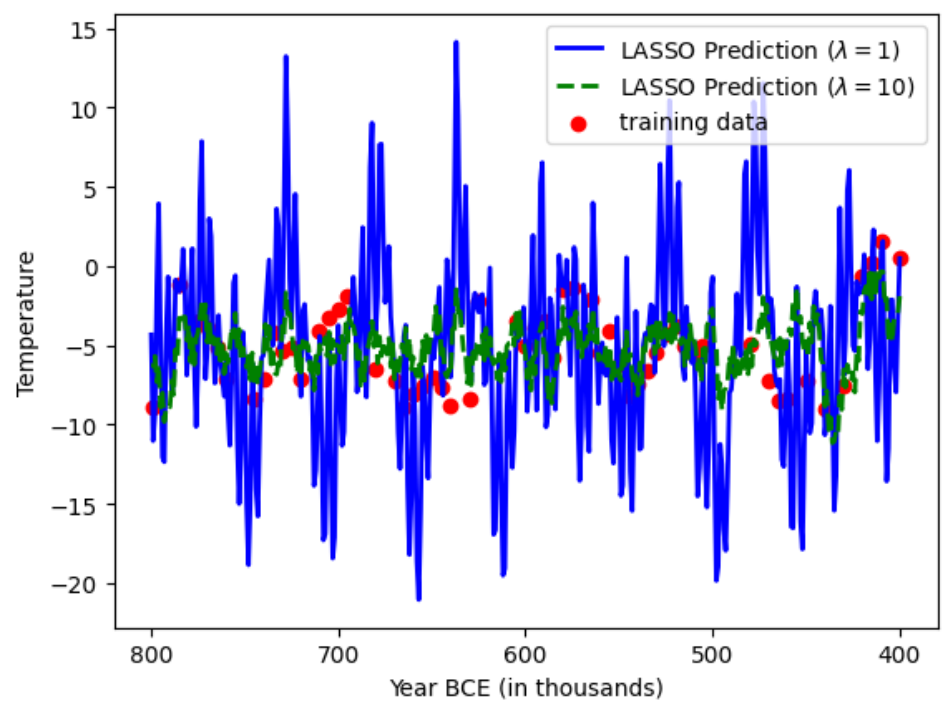
   Maximizing the log-posterior is equivalent to minimizing its negative. By scaling the entire expression by the constant σ^2 (which does not change the argmin), we obtain:

$$\arg\max_{\mathbf{w}}\log p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \arg\min_{\mathbf{w}}\left(\frac{1}{2}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \frac{\lambda}{2}||\mathbf{w}||_1\right) \tag{17}$$

The right-hand side is exactly the loss function $\mathcal{L}_{lasso}(\mathbf{w})$ as defined in the problem. Same way as going about this as 4.1, maximizing the posterior likelihood with this Laplace prior is equivalent to minimizing the Lasso loss function.

4. The Lasso loss function $\mathcal{L}_{lasso}(\mathbf{w})$ includes an L_1 regularization term, which is not differentiable at zero. This non-differentiability arises because the L_1 norm involves the absolute value function, which ruins our approach to find a closed form solution by setting the gradient to zero. As a result, we have to find ways to numerically optimize like in number 5.

5. The graph for this is below:



For MSE's, we have the following values:

- Test MSE for lambda=1: 30.06

- Test MSE for lambda=10: 15.62

Errors are much lower than the unregularized basis (d) model, which had a test MSE of 58.94. The higher lambda has the lowest test MSE. The regularization term in Lasso helps to prevent overfitting by encouraging sparsity in the weight vector $\mathbf{w}$, which leads to better generalization on the test set. The graph also shows that the predictions with regularization are smoother and less sensitive to noise in the training data compared to the unregularized model, which is consistent with the observed reduction in test MSE. The model with $\lambda = 10$ also seems to generalize and fit well with the overall trend of the data.

Name: Khang Nguyen

Collaborators and Resources: Looked up a few things regarding derivation online for Rank/Trace in Problem 2. I did LaTeX in VSCode, which had in-line completions, but only after I laid out the core ideas and details already. It's moreso to format the LaTeX than helping with the problem.