Aidan Tai
aidantai@college.harvard.edu
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:

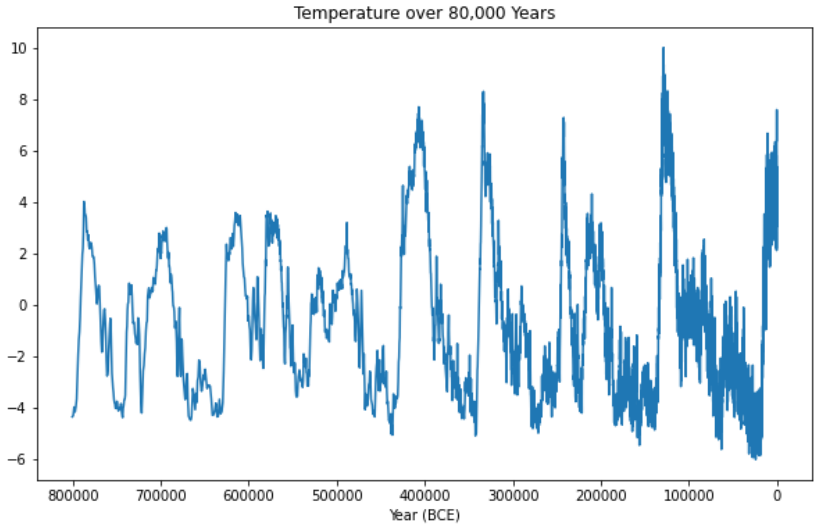
- `earth_temperature_sampled_train.csv`
- `earth_temperature_sampled_test.csv`

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
3.99946000000000000e+05,5.09043921839875501 7e+00
4.09980000000000000e+05,6.15043921839875514e+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.

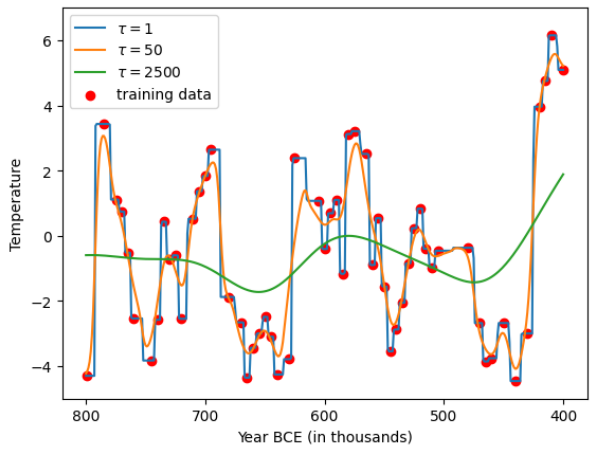
   Let $\{(x'_m, y'_m)\}_{m=1}^M$ denote the test set. Write down the form of the MSE of f_τ over the test set as a function of the training set and test set. Your answer may include $\{(x'_m, y'_m)\}_{m=1}^M$, $\{(x_n, y_n)\}_{n=1}^N$, and K_τ, but not f_τ.

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?

Solution 1 (Optimizing a Kernel)



1. The first and third graphs are overfit and underfit respectively. $\tau = 1$ is too biased (technically, bias is too low) toward the training data, so test predictions may be unnecessarily inaccurate due to the model overfitting to a unique quality of the training data that differs from future test data/the true data. Meanwhile, $\tau = 2500$ has been significantly underfit such that the "spiky" pattern in the training data is hardly represented by the model. In this case, the variance is over optimized to the point where excessive bias plagues the model. Based on intuition, it seems that the second graph is a better middle ground in terms of balancing bias and variance, so I would probably use tau=50 for predictions.

2. The MSE is minimized when tau is minimized, but this leads to overfitting. As tau approaches 0, the kernel function approaches $e^{-\infty}$ (approaches zero), meaning that we can effectively force the MSE to zero by minimizing tau, which severely overfits the MSE.

3. $$\begin{aligned} MSE &= \frac{1}{M}\Sigma_{m=1}^{M}(f_\tau(x'_m) - y'_m)^2 \\ &= \frac{1}{M}\Sigma_{m=1}^{M}(\frac{\Sigma_n K_\tau(x_n, x'_m)y_n}{\Sigma_n K_\tau(x_n, x'_m)} - y'_m)^2 \end{aligned}$$

4. (see code)
   $\tau = 50$ yields the lowest test set MSE (1.858), which is consistent with my observations in Part 1 (it has a slight edge over $\tau = 1$ because it's less overfit)

5. I would have to send the dataset because the function iterates over every data point, so the space complexity is $O(n)$. Calculating a single point has time complexity $O(n)$ because you have to iterate through each point to calculate the sum (bottom term) and also multiply y_n to each in the process (top term), but you can do both of these calculations together, so 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.

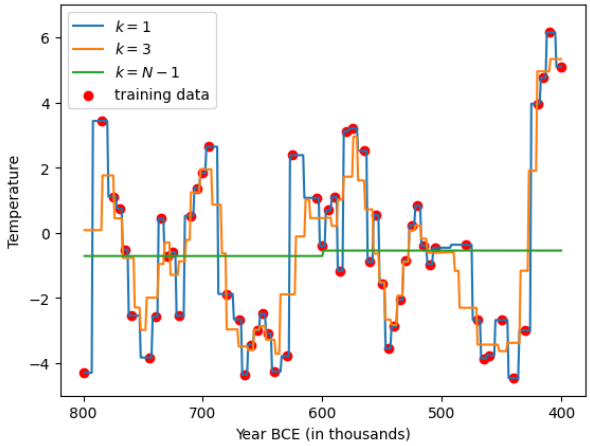
2. Describe what you see: what is the behavior of the functions in these three plots? How does it compare to the behavior of the functions in the three plots from Problem 1? In particular, which of the plots from Problem 1 look most similar to each in Problem 2? Are there situations in which kNN and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

2. $k = 1$ and $k = 3$ both appear reasonable, while $k = N - 1$ fails to capture the trends of the data. The kernelized regression graphs look smoother than the knn graphs, but overall $(\tau = 1, k = 1)$ and $(\tau = 50, k = 3)$ are very similar. Meanwhile, $k = N - 1$ looks very different from $\tau = 2500$, though they both "flatten" the graph too much.

3. Similar to problem 1, the middle graph appears to be the best because it offers a balance between the two other graphs (though k=N-1 can hardly be considered relevant). The bias-variance tradeoff is preferable in $k = 3$ because $k = 1$ is a very low parameter that likely overfits the training data. The MSE for $k = 3$ is 3.89, which is worse than the MSE for $k = 1$ (1.74). However, the MSE is not the only metric available for evaluating a model. $k = 3$ sacrifices some MSE-accuracy for better generalizability.

4. Once again, we are iterating through all of the points in the data set to calculate the nearest neighbors. Therefore, we need to store all of the data points, which has space complexity $O(n)$. However, in our implementation, we had to sort the data points in order to calculate the nearest neighbors. The sorting algorithm used has a time complexity of $O(n \log(n))$, and this is the slowest part of the model, so the overall time complexity is $O(n \log(n))$ for a single point.

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.

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

   - 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?

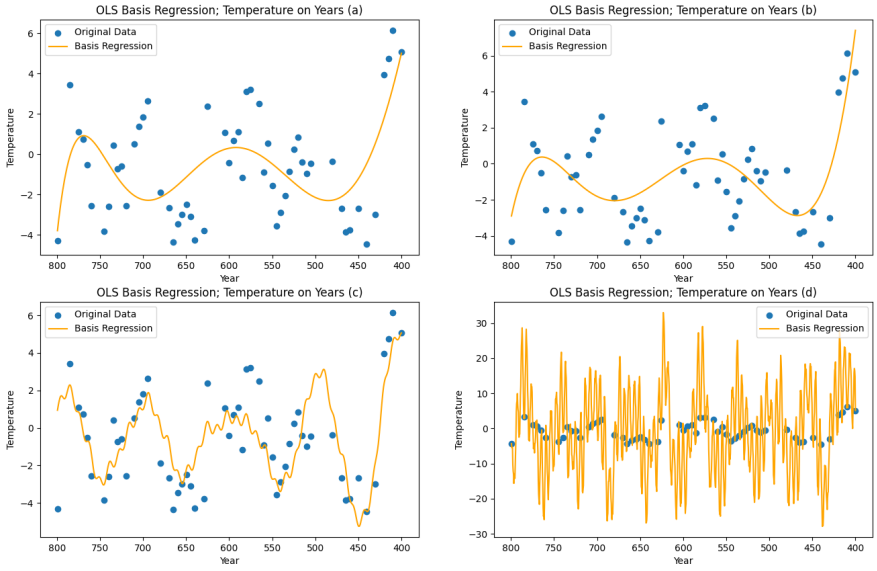
   Given an arbitrary datapoint, what is the time complexity of computing the predicted value for this data point?

   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.

Solution 3



1. (see graph above)
2. (see notes), $w^{MLE} = (X^T X)^{-1} X^T y$
   (a) Train MSE: 4.83; Test MSE: 7.96
   Train NLL: 762.399; Test NLL: 523.304
   This basis hardly captures the trend in the data and has poor metrics. We can probably do better.
   (b) Train MSE: 4.22; Test MSE: 7.76
   Train NLL: 601.027; Test NLL: 450.117
   Though this basis has better metrics, its only slightly better. The model could probably be improved (more complexity) to achieve better accuracy at the cost of good variance.
   (c) Train MSE: 2.88; Test MSE: 5.97
   Train NLL: 318.767; Test NLL: 250.941
   This basis looks good! The train and test metrics are pretty similar, so we know that, the model is not overfitting the training data. Additionally, the MSE values are decent, so the mode is probably complex enough to fit well. Visually, graph appears to fit the trend of the data fairly well.
   (d) Train MSE: 0.64; Test MSE: 58.91
   Train NLL: 51.594; Test NLL: 491.076
   This basis severely overfits. The Train metrics are extremely small, but the Test metrics are ridiculously high. Looking at the graph, we can see that the model has crazy waves but they line up on the training data very well.
3. I would just need to share the weight vector with dimension D, so the space complexity is just $O(D)$. The time complexity is also $O(k)$ because we are just multiplying once for each weight vector. These complexities are significantly better than kNN or kernelized regression. (unless the number of features is greater than $O(n)$ or $O(n \log(n))$

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)

1. My target variable would be total US energy consumption per capita per year (measured in gigajoules, so gj/capita measured across years)
2. There are a lot of options for choosing features, depending on what data is available. I would analyze how each sector in the US economy contributes to energy consumption (Industrial, Commercial, Residential, Transportation sectors). Therefore, I would look at driving factors behind these sectors, such as overall growth/cash flow of each sector (perhaps the Transportation sector dipped during 2020-2021 and thus had reduced contribution to energy consumption), the profits of each sector (a sector might be smaller but earn more profit, thus growing at a higher rate – new rising companies that consume a significant amount of energy would be relevant), or the subsidies/taxes/regulations on sectors/companies in each sector (this may be hard to measure, but if data is available to numerically represent overall "regulation" on a sector, then we could determine how government regulation can influence a sector's energy consumption). This would make 12 features (3 for each sector).
3. The dataset would focus on recent decades because going further back only brings data that is not representative of current circumstances.
4. I would use linear regression to predict energy usage (a coefficient for each feature) and assess its performance on a loss function that minimizes the MSE. The coefficients would reveal what impact each feature has on the total US energy consumption (thanks to the high interpretability of this linear regression)
5. If the model shows that applying taxes/regulation specifically upon the Industrial sector has a significant impact on US energy consumption (relative to other features), then the US president could consider increasing regulation on this sector.
6. The US president might be a skeptic of machine learning or might not trust it because he doesn't understand it. If he does have a good understanding, he might not trust the model if it has very high uncertainty and bad metrics.
7. If the model is wrong, the US president might make a poorly-informed policy decision. For example, he might greatly restrict a sector by imposing new regulation when it turns out that this regulation hardly improves US energy consumption.

HW1

Problem 1: Optimizing Kernel

$$f_\tau(x^*) = \frac{\sum_n K_\tau(x_n, x^*) y_n}{\sum_n K_\tau(x_n, x^*)} \qquad k_\tau(x, x') = \exp\left\{-\frac{(x-x')^2}{\tau}\right\}$$

1. tau = 50 is the best

2. MSE is minimized when tau is minimized

as $\lim_{\tau \to 0}$, if $x = x'$ then you near 0, but with $x \neq x'$ it runs $e^{-\infty}$

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

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

4. $\tau = 1$ yields the lowest test set MSE, which is consistent

5. The space complexity is $O(n)$, and the time complexity is $O(n)$ per point

HW1

Problem 2:

$$f(x^*) = \frac{1}{K} \sum y_n \mathbb{1}(x_n \text{ is one of } k\text{-closest to } x^*)$$

2. $K=1$, $K=3$ both have reasonable model fitting, while $K = N-1$ is basically a useless flat line. The kernel graphs look more boxy than the kernel. Also, the $(K=1, T=1)$ and $(K=3, T=50)$ pairs look very similar.

3. $K=3$ appears the best because it has a balance of bias and variance

4. Since they are using the same k, they only need the data points that were used in each neighbor calculation. If there are points that aren't used for any x^*, then you can cut them out. But it's still probably $O(n)$ space and $O(n + n\log(n)) = O(n\log(n))$. For each point have to find K nearest neighbors which means you to sort all n which is $O(n\log(n))$

HW1

Problem 3

$y = w_0 + w_1 x_i$ $\quad y = f_w(x) = w^T x$

$y = \begin{bmatrix} w_0 + w_1 x_1 \\ w + w x_2 \\ \vdots \end{bmatrix}$ $\quad \hat{y} = (w^*)^T x =$

$$\begin{bmatrix} 1 & x_1 & \cdots & x_n \\ 1 & & & \\ \vdots & & & \\ 1 & & & \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} w_0 + w_1 x_{1,n} \cdots \\ \vdots \\ w_0 + w_1 x_{n,1} \cdots \end{bmatrix}$$

2)a) $y_i = w^T x_i + \varepsilon$, roll in the bias $x_i = \begin{bmatrix} 1 \\ x_i \end{bmatrix}$ $\quad \varepsilon \sim N(0, \sigma^2)$

$L(w) = \frac{1}{N}(y - X_w)^T (y - X_w)$

$\nabla L = \frac{2}{N}(X^T X_w - X^T y) = 0$

$0 = X^T X_w - X^T y$

$X^T X_w = X^T y$

$w^{MLE} = (X^T X)^{-1} X^T y$

$$0 = \frac{d}{d\sigma} l(w^{MLE}, \sigma) = \frac{d}{d\sigma}\left(\sum_{n=1}^{N} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\right] - \sum_{n=1}^{N} \frac{(y_n - f_w(x_n))^2}{2\sigma^2} \right)$$

$$0 = \frac{d}{d\sigma} \frac{-1}{2\sigma^2} \left(\sum_{n=1}^{N} (y_n - f_w(x_n))^2 \right)$$

$$0 = \frac{d}{d\sigma} \frac{-N}{2\sigma^2} (MSE(w^{MLE}))$$

$$0 = \frac{N}{\sigma^3}(MSE) \qquad \sigma^2 = MSE$$

Name

Aidan Tai

Collaborators and Resources

None

Calibration

9 hours