CS181-S24

Assignment #1
Due: 11:59pm ET, February 10, 2024

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:

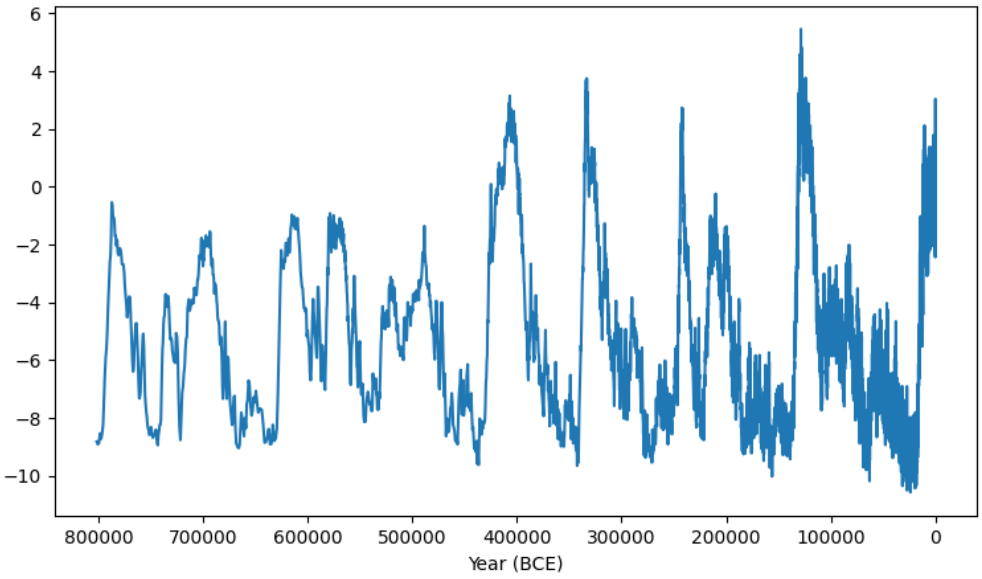
- `earth_temperature_sampled_train.csv`
- `earth_temperature_sampled_test.csv`

Each has two columns. The first column is the age of the ice core sample. 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 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

Resources and Submission Instructions

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.

Name: Kelsey Chen
Collaborators: Salma Douieb
Late days used: 0

Problem 1 (Setting up the Regression, 10pts)

As noted in the introduction, your goal is to predict temperature variation given the year. Before we start deriving and coding up our regressions, we will interrogate the set-up of our problem.

1. These data were derived from ice core samples in Antarctica. Take a brief look at the original data file. Briefly discuss how the data were processed: what kinds of decisions and corrections had to be made? We know that different places on earth have different temperatures: what does the temperature in the temperature column correspond to?

2. Even before doing any formal regressions, we see that there is some periodicity in the data: there are years that are warmer, and years that are cooler. Suppose you are a government official advising on how much to worry about climate change. Would it be reasonable to use this pattern as evidence that the earth will cool down again? Why or why not, or to what extent?

3. In the problems below, we will focus on interpolating temperatures for years not provided in the training set. What kind of application would such a regression be useful for?

Solution

1. From the abstract, we know that ice cores were drilled and extracted from Dome C in Antarctica. The deuterium content in the ice core was measured and used as a proxy for past temperatures. In terms of data corrections, the abstract states that the temperature needed to be corrected for sea-water isotopic composition and ice sheet elevation on the EDC3 age scale. The temperature in the temperature column corresponds to the temperature estimate for the temperature difference of the samples from the average of the last 1000 years.

2. It would not be reasonable to use this pattern as evidence that the earth will cool down again, since the samples are only drawn from a specific region (Dome C in Antarctica). Unless there are other studies that show correlation between temperatures from this region to the earth's temperature as a whole, the data from this dataset cannot be generalized to the earth.

3. Even though the data is only from one region of the earth, interpolating temperatures for years not provided in the training set can still be useful for climate change research. For example, the data could be used in conjunction with other data sets to reconstruct historical climate conditions, which in turn could contribute to our understanding of climate change patterns and trends. Furthermore, if we are missing temperature data from years that are covered by this data set, you can use this data to interpolate those values.

Problem 2 (Optimizing a Kernel, 30pts)

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 $\mathcal{D}_{\texttt{train}} = \{(x_n, y_n)\}_{n=1}^N$ 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). 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. 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 $x = 400, 401, \ldots, 800$. **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 how the fits change with τ.

2. Now, we will evaluate the quality of each model *quantitatively* by computing the error on some test set $\mathcal{D}_{\texttt{test}} = \{(x'_m, y'_m)\}_{m=1}^M$. Write down the expression for 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.

3. Suppose we used the training set as our test set, that is, we evaluated the expression above with $\mathcal{D}_{\texttt{test}} = \mathcal{D}_{\texttt{train}}$, and chose the value of τ which gave the smallest loss. What value of τ would be picked? Why is setting $\mathcal{D}_{\texttt{test}} = \mathcal{D}_{\texttt{train}}$ 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 $x \neq x'$?

4. Let us compute the MSE on the provided test set (that is, not the 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?

5. Describe the time and space complexity of kernelized regression 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?

Solution

1. The curve gets smoother as τ increases. For $\tau = 50$, the curve seems to be still fairly accurate while being smooth, but for $\tau = 2500$, the curve is incredibly smooth with larger MSE. This makes sense because with tau very large, the model gives too much weight to points that are far from the prediction x-value, hence leading to inaccurate predictions. For $\tau = 1$, the curve goes through all the data points in the training dataset. While this fits the training data incredibly well, it may lead to higher loss due to the model fitting too closely to the data points and failing to generalize to new data points.

2. $$\text{MSE} = \frac{1}{M}\sum_{M}\left[y_m - f(x_m; w)\right]^2 \text{ from lecture}$$
$$= \frac{1}{M}\sum_{M}\left[y_m - \frac{\sum_n K_\tau(x_n, x_m)y_n}{\sum_n K_\tau(x_n, x_m)}\right]^2 \text{ by replacing } f(x_m; w)$$

3. The value of τ that would minimize the loss would be very small, since it would give essentially 0 weight to all the other points other than the training data point, i.e. τ would be set to 0. However, this would make the kernel function undefined, since if $x = x'$, then the kernel function becomes exp(0/0). If $\mathcal{D}_{\texttt{test}} = \mathcal{D}_{\texttt{train}}$, there will be at least one point for which $x = x'$ in the sum of the weights, and hence $f(x^*)$ is poorly defined for every test value. With $x \neq x'$ and τ small, the expression inside exp() is very negative, so $K_t(x, x^*)$ is close to 0 but not undefined. This makes sense because we are giving less weight to points that are further from the prediction x-value. Hence, it's bad to set $\mathcal{D}_{\texttt{test}} = \mathcal{D}_{\texttt{train}}$ since the model will overfit to exactly the training data and will only be able to replicate the training data without providing generalized information on new data.

4. Results:

   (a) $\tau = 1$: loss = 1.9472621565209178

   (b) $\tau = 50$: loss = 1.8582899169613447

   (c) $\tau = 2500$: loss = 8.333886806980791

   $\tau = 50$ yields the lowest MSE of the three options ($\tau = 1$, $\tau = 50$, $\tau = 2500$).

5. Our time complexity is $O(N)$ since for each of the n x-values x in the training set, we need to compute the similarity between x and the new point x' using our kernel function. Hence computing these similarities is $O(N)$ in total. Then we calculate $f_\tau(x^*)$, for each of the n x-values in the training set, which is again $O(N)$. Hence in total our time for making a single prediction is $O(N)$, so the number of computations required to make a prediction is linear with the size of the training set.

   For space complexity, we will need to store N pairs of training data, as well as N weights for each prediction. Hence the space complexity is $O(N)$.

Problem 3 (Kernels and kNN, 20pts)

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. 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 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, are there situations in which kNN and kernel-based regression interpolate similarly?

2. Compute the MSE on the test set for each value of k. Which solution has the lowest MSE?

3. As in Problem 1, describe the space and time complexity of kNN. How does what is stored to compute predictions change with the size of the training set N? How does the computation needed to compute the prediction for a new input depend on the size of the training set N? (For the latter, justify based off of your implementation.)

Solution

1. The behavior of the plots for KNN (varying k) are quite similar to those for KR (varying tau). In particular, k=1 looks similar to $\tau = 1$, fitting very closely to the training data (but perhaps not well to the test data because of overfitting, as we saw in the previous question). Similarly, k=3 looks similar to $\tau = 50$, but fitting less closely to the training data. This makes sense since KNN only considers a fixed number of nearest neighbors without weighing the distances differently. Finally, k=$N-1$ is practically a horizontal line, since it's simply the average of all the data points except for the furthest one. It makes sense that the behavior varying k is similar to the behavior varying τ because both KNN and KR are locally adaptive, so we can control which points the model uses for its predictions using k and τ respectively.

2. Results:

   (a) k = 1: MSE loss = 1.7406000000000004
   (b) k = 3: MSE loss = 3.890766222222223
   (c) k = 55: MSE loss = 9.66397324033058
   (d) k = 56: MSE loss = 9.528571442602042

   The value of k with the lowest MSE is $k = 1$.

3. For a new data point, we get the distances from each N training point, which is $O(N)$. We then sort the neighbors in ascending order to get the k closest points, which is $O(N \log N)$. Summing each row for a single point is $O(N)$, so we have that the overall time complexity is $O(N \log N)$.

Problem 4 (Modeling Climate Change 800,000 Years Ago, 30pts)

Our last regression will be linear regression. We currently only have a one dimensional input, 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. (The first basis has been implemented for you in the notebook as an example.) 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.

   For each basis create a plot of your code graphing the least squares 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 this part are these four plots.**

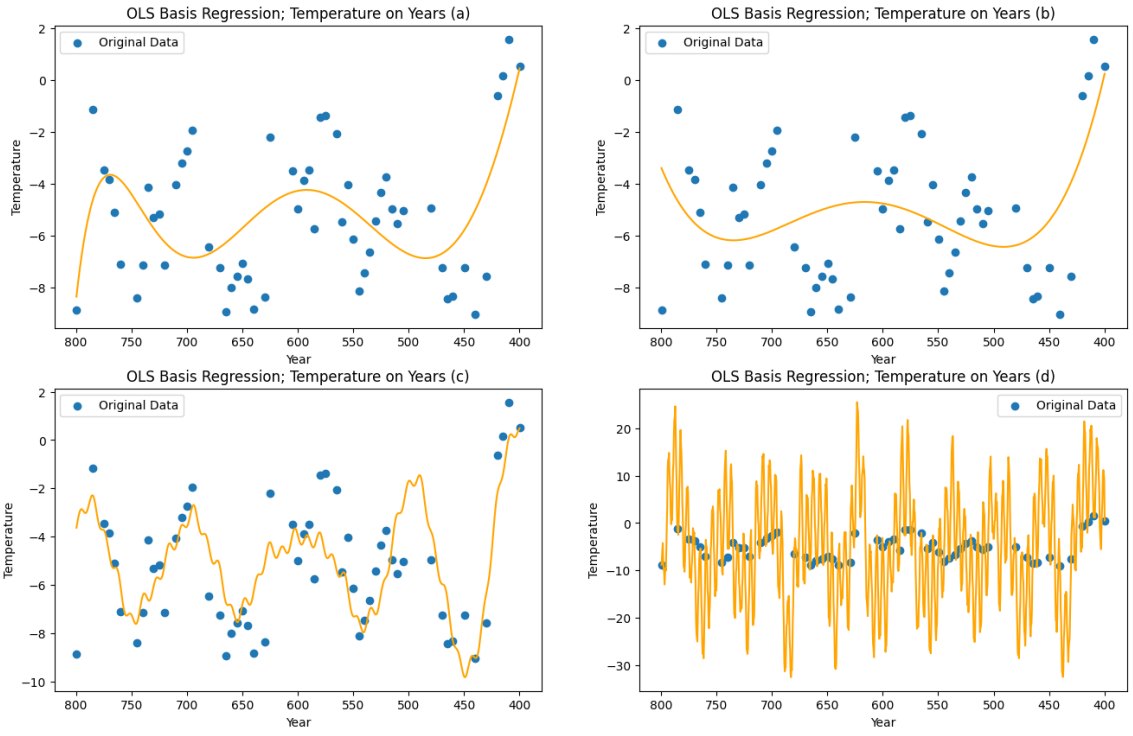
[a] For the trigonometric bases (c) and (d), the periodic nature of cosine requires us to transform the data such that the lengthscale is within the periods of each element of our basis.

Problem 4 (cont.)

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 Problems 1 and 2, 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? 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: 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



1.

2. Results:

(a) Train MSE: 4.82; Test MSE: 7.96

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

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

(d) Train MSE: 0.65; Test MSE: 60.32

We can see from the training and testing MSEs that basis (d) seems to overfit, since the training MSE is much lower while the test MSE is much higher than the others. Relative to (c), basis (a) and (b) underfit.

3. A basis function allows us to extend linear regression to fit higher order functions. For instance, temperature data will vary by time, and will likely follow a sin or cos curve since it follows a yearly cycle. Hence transforming our data with sin or cos allows us to still use linear regression on our data.

   On the other hand, the function f can be used to scale our inputs so that they are easier to work with. For instance, if our input range is very large, we can divide the entire input vector by $1.81 * 10^2$ for numerical stability.

4. What is stored to compute predictions is just $O(d)$ space where d is the dimension of our data, since we only need to store the weight for each dimension. To compute the prediction for a new input, we can simply perform a calculation with the stored weights of the dimensions, which is $O(d)$ time. The computation needed to compute the prediction thus doesn't depend on the size of the training set N, merely the dimension of the data.

5. KNN and KR are similar in that they are both locality adaptive (can adjust how global/local the points are) and geometry adaptive (can adjust the way we compare points). KR is like a more generalized version of KNN, using a kernel function instead of distance to adjust for locality and geometry. As we saw in lecture, we can compare the performance of KNN and KR by looking at a sparsely sampled highly changing function (like the sin or cos wave). Both perform poorly as we increase the locality, but KR outperforms KNN as the kernel gives more weight to nearest neighbors. For linear regression, some basis functions will overfit or underfit, as we saw in the example in the previous parts. Furthermore, another limitation of linear regression is that we can only represent linear data without basis functions, so if we don't have some prior intuition of what basis function to use, it may be harder to make a prediction with linear regression.

Problem 5 (Deriving Linear Regression, 10pts)

In the previous problems, you focused on implementing regressions and exploring their fits on data. Now we will turn to some more analytic work. Specifically, the solution for the least squares linear regressions "looks" kind of like a ratio of covariance and variance terms. In this problem, we will make that connection more explicit.

Let us assume that our data are tuples of scalars (x, y) that are described by some joint distribution $p(x, y)$. For clarification, the joint distribution $p(x, y)$ is just another way of saying the "joint PDF" $f(x, y)$, which may be more familiar to those who have taken Stat 110, or equivalent.

We will consider the process of fitting these data from this distribution with the best linear model possible, that is a linear model of the form $\hat{y} = wx$ that minimizes the expected squared loss $E_{x,y}[(y - \hat{y})^2]$.

Notes: The notation $E_{x,y}$ indicates an expectation taken over the joint distribution $p(x, y)$. Since x and y are scalars, w is also a scalar.

1. Derive an expression for the optimal w, that is, the w that minimizes the expected squared loss above. You should leave your answer in terms of moments of the distribution, e.g. terms like $E_x[x]$, $E_x[x^2]$, $E_y[y]$, $E_y[y^2]$, $E_{x,y}[xy]$ etc.
2. Provide unbiased and consistent formulas to estimate $E_{x,y}[yx]$ and $E_x[x^2]$ given observed data $\{(x_n, y_n)\}_{n=1}^N$.
3. In general, moment terms like $E_{x,y}[yx]$, $E_{x,y}[x^2]$, $E_{x,y}[yx^3]$, $E_{x,y}[\frac{x}{y}]$, etc. can easily be estimated from the data (like you did above). If you substitute in these empirical moments, how does your expression for the optimal w^* in this problem compare with the optimal w^* that we see in Section 2.6 of the cs181-textbook?
4. Many common probabilistic linear regression models assume that variables x and y are jointly Gaussian. Did any of your above derivations rely on the assumption that x and y are jointly Gaussian? Why or why not?

Solution

1. We want to minimize $E_{x,y}[(y - \hat{y})^2] = \arg\min_w E_{x,y}[(y - wx)^2]$. We have that:

$$\begin{aligned} E_{x,y}[(y - wx)^2] = E_{x,y}[y^2 - 2ywx + (wx)^2] &= E_{x,y}[y^2] - 2E_{x,y}[ywx] + E_{x,y}[(wx)^2] \text{ by linearity} \\ &= E_{x,y}[y^2] - 2wE_{x,y}[yx] + w^2E_{x,y}[x^2] \text{ since w is constant w.r.t. } E_{x,y} \end{aligned}$$

To find the optimal w we set the derivative of the expression above to be equal to 0:

$$\begin{aligned} 0 = \frac{d}{dw}E_{x,y}[y^2] - \frac{d}{dw}2wE_{x,y}[yx] + \frac{d}{dw}w^2E_{x,y}[x^2] \\ = 0 - 2E_{x,y}[yx] + 2wE_{x,y}[x^2] \\ \Rightarrow w^* = \frac{E_{x,y}[yx]}{E_{x,y}[x^2]} \end{aligned}$$

2. We want our estimator $\hat{w}^*$ for w^* to be consistent ($\hat{w}^* \to w^*$ in probability) and unbiased ($E[\hat{w}^*] - w^* = 0$). By law of large numbers, the sample mean converges to the theoretical mean with probability 1,

so we can use the sample means as consistent estimators: $\hat{E}_{x,y}[yx] = \frac{1}{N}\sum_{i=1}^{N} y_i x_i$ and $\hat{E}_{x,y}[x^2] = \frac{1}{N}\sum_{i=1}^{N} x_i^2$. Now to show that the sample mean is unbiased, we can use linearity of expectation and the assumption that the data points are independent:

$$E\left[\frac{1}{N}\sum_{i=1}^{N} y_i x_i\right] = \frac{1}{N}$$
$$E\left[\sum_{i=1}^{N} y_i x_i\right] = \frac{1}{N} N E\left[yx\right]$$

Hence we have that the sample mean is unbiased for the term in the numerator, and by similar logic we can show the same for the term in the denominator, so our unbiased and consistent estimator is

$$\hat{w}^* = \frac{\sum_{i=1}^{N} y_i x_i}{\sum_{i=1}^{N} x_i^2}$$

3. From the textbook, we have that the optimal $w^* = (X^T X)^{-1} X^T y$. This is analogous to the estimator from part 2, except we use matrices instead of scalar values. In particular, $(X^T X)^{-1}$ corresponds to the denominator, $\sum_{i=1}^{N} x_i^2$, and the term $X^T y$ corresponds to the numerator, $\sum_{i=1}^{N} y_i x_i$.

4. We did not rely on the data being jointly gaussian, since our calculation of the expectation holds for all random variables, and we do not use any properties of the Normal. Hence our derivations should hold for all random variables.

Name

Collaborators and Resources

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