CS181-S21

Assignment #1

Due: 7:59pm ET, February 4, 2021

Collaborators:

Homework 1: Regression

Introduction

This homework is on different forms of linear regression and focuses on loss functions, optimizers, and regularization. Linear regression will be one of the few models that we see that has an analytical solution. These problems focus on deriving these solutions and exploring their properties.

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. `T1_P1.py`, etc.

Problem 1 (Optimizing a Kernel, 15pts)

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:

$$f(x^*) = \frac{\sum_n K(x_n, x^*) y_n}{\sum_n K(x_n, x^*)}$$

where (x_n, y_n) are the training data points, and $K(x, x')$ is a kernel function that defines the similarity between two inputs x and x'. Assume that each x_i is represented as a column vector, i.e. a D by 1 vector where D is the number of features for each data point. A popular choice of kernel is a function that decays as the distance between the two points increases, such as

$$K(x, x') = \exp(-||x - x'||_2^2) = \exp(-(x - x')^T (x - x'))$$

However, the squared Euclidean distance $||x - x'||_2^2$ may not always be the right choice. In this problem, we will consider optimizing over squared Mahalanobis distances

$$K(x, x') = \exp(-(x - x')^T W (x - x'))$$

where W is a symmetric D by D matrix. Intuitively, introducing the weight matrix W allows for different dimensions to matter differently when defining similarity.

1. Let $\{(x_n, y_n)\}_{n=1}^N$ be our training data set. Suppose we are interested in minimizing the residual sum of squares. Write down this loss over the training data $\mathcal{L}(W)$ as a function of W.

   Important: When computing the prediction $f(x_i)$ for a point x_i in the training set, carefully consider for which points x' you should be including the term $K(x_i, x')$ in the sum.

2. In the following, let us assume that $D = 2$. That means that W has three parameters: W_{11}, W_{22}, and $W_{12} = W_{21}$. Expand the formula for the loss function to be a function of these three parameters.

3. Derive the gradients of the loss function with respect to each of the parameters of W for the $D = 2$ case. (This will look a bit messy!)

Problem 1 (cont.)

4. Consider the following data set:

```
x1 , x2 , y
  0 , 0 , 0
  0 , .5 , 0
  0 , 1 , 0
  .5 , 0 , .5
  .5 , .5 , .5
  .5 , 1 , .5
  1 , 0 , 1
  1 , .5 , 1
  1 , 1 , 1
```

   And the following kernels:

$$W_1 = \alpha \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad W_2 = \alpha \begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix} \qquad W_3 = \alpha \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

   with $\alpha = 10$. Write some Python code to compute the loss with respect to each kernel for the dataset provided above. Which kernel does best? Why? How does the choice of α affect the loss?

   For this problem, you can use our staff **script to compare your code to a set of staff-written test cases.** This requires, however, that you use the structure of the starter code provided in `T1_P1.py`. More specific instructions can be found at the top of the file `T1_P1_Testcases.py`. You may run the test cases in the command-line using `python T1_P1_TestCases.py`. **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.**

5. Bonus: Code up a gradient descent to optimize the kernel for the data set above. Start your gradient descent from W_1. Report on what you find.
   Gradient descent is discussed in Section 3.4 of the cs181-textbook notes and Section 5.2.4 of Bishop, and will be covered later in the course!

Problem 2 (Kernels and kNN, 10pts)

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 kernels as Problem 1, and dataset `data/p2.csv`.

For this problem, you can use our staff **script to compare your code to a set of staff-written test cases.** This requires, however, that you use the structure of the starter code provided in `T1_P2.py`. More specific instructions can be found at the top of the file `T1_P2_Testcases.py`. You may run the test cases in the command-line using `python T1_P2_TestCases.py`. **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. We will be making 6 plots comparing kernel-based and nearest neighbor-based predictors, all using the Mahalanobis distance corresponding to W_1 from Problem 1. In each plot, you will plot the predicted value of y, given x_1 (horizontal axis) and x_2 (vertical axis), as the color of each point (grayscale between 0 and 1). Include the x_1 and x_2 axes, with tick marks spaced every 0.1 units for $x_1 = 0$ to $x_1 = 1$ and $x_2 = 0$ to $x_2 = 1$.

   For the first three plots, use the kernel-based predictor varying $\alpha = \{0.1, 3, 10\}$. For the next three plots, use the kNN predictor with $\alpha = 1$, $k = \{1, 5, N-1\}$, where N is the size of the data set.

   Print the total least squares loss on the training set for each of the 6 plots.

   You may choose to use some starter Python code to create your plots provided in `T1_P2.py`. Please **write your own implementation of kNN** for full credit. Do not use external libraries to find nearest neighbors.

2. Do any of the kernel-based regression plots look like the 1NN? The $(N-1)$NN? Why or why not?

3. Suppose we are given some W for a Mahalanobis distance or kernel function. Then, in general, there exist values of k for which kernel-based regression and kNN disagree (i.e., make different predictions) on at least one input - for all choices of α. Explain why by means of an example (i.e., show that for some value k of your choosing, no value of α will produce two classifiers that are the same).

4. Why did we not vary α for the kNN approach?

Problem 3 (Deriving Linear Regression, 10pts)

In class, we noted that the solution for the least squares linear regressions "looked" 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 come from some distribution $p(x, y)$. We will consider the process of fitting these data 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.

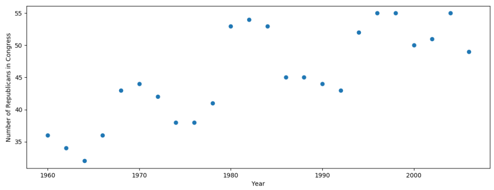
1. Derive an expression for the optimal w, that is, the w that minimizes the expected squared loss above. You should leave your answer in terms of moments of the data, e.g. terms like $E_x[x]$, $E_x[x^2]$, $E_y[y]$, $E_y[y^2]$, $E_{x,y}[xy]$ etc.
2. Provide unbiased and consistent formulas to estimate $E_{x,y}[yx]$ and $E_x[x^2]$ given observed data $\{(x_n, y_n)\}_{n=1}^N$.
3. In general, moment terms like $E_{x,y}[yx]$, $E_{x,y}[x^2]$, 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 derived in class/Section 2.6 of the cs181-textbook?
4. As discussed in lecture, 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?

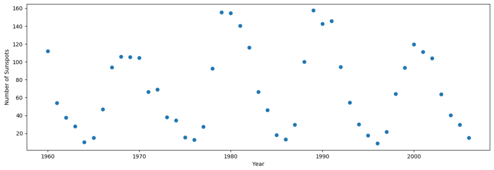
Problem 4 (Modeling Changes in Republicans and Sunspots, 15pts)

The objective of this problem is to learn about linear regression with basis functions by modeling the number of Republicans in the Senate. The file `data/year-sunspots-republicans.csv` contains the data you will use for this problem. It has three columns. The first one is an integer that indicates the year. The second is the number of Sunspots observed in that year. The third is the number of Republicans in the Senate for that year. The data file looks like this:

```
Year,Sunspot_Count,Republican_Count
1960,112.3,36
1962,37.6,34
1964,10.2,32
1966,47.0,36
```

You can see scatterplots of the data in the figures below. The horizontal axis is the Year, and the vertical axis is the Number of Republicans and the Number of Sunspots, respectively.





(Data Source: http://www.realclimate.org/data/senators_sunspots.txt)

Make sure to include all required plots in your PDF.

1. In this problem you will implement ordinary least squares regression using 4 different basis functions for **Year (x-axis)** v. **Number of Republicans in the Senate (y-axis)**. Some starter Python code that implements simple linear regression is provided in `T1_P4.py`.

   First, plot the data and regression lines for each of the following sets of basis functions, and include the generated plot as an image in your submission PDF. You will therefore make 4 total plots:

   (a) $\phi_j(x) = x^j$ for $j = 1, \ldots, 5$
   ie, use basis $y = a_1x^1 + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$ for some constants $\{a_1, ..., a_5\}$.

   (b) $\phi_j(x) = \exp \frac{-(x-\mu_j)^2}{25}$ for $\mu_j = 1960, 1965, 1970, 1975, \ldots 2010$

   (c) $\phi_j(x) = \cos(x/j)$ for $j = 1, \ldots, 5$

   (d) $\phi_j(x) = \cos(x/j)$ for $j = 1, \ldots, 25$

   * Note: Be sure to add a bias term for each of the basis functions above.

   Second, for each plot include the residual sum of squares error. Submit the generated plot and residual sum-of-squares error for each basis in your LaTeX write-up.

Problem 4 (cont.)

2. Repeat the same exact process as above but for **Number of Sunspots (x-axis)** v. **Number of Republicans in the Senate (y-axis)**. Now, however, only use data from before 1985, and only use basis functions (a), (c), and (d) – ignore basis (b). You will therefore make 3 total plots. For each plot make sure to also include the residual sum of squares error.

   Which of the three bases (a, c, d) provided the "best" fit? **Choose one**, and keep in mind the generalizability of the model.

   Given the quality of this fit, do you believe that the number of sunspots controls the number of Republicans in the senate (Yes or No)?