STAT 339: HOMEWORK 1B (LINEAR REGRESSION)

DUE VIA GITHUB BY CLASS TIME ON WEDNESDAY 2/26

Instructions. Create a directory called `hw1b` in your `stat339` GitHub repo. Your main writeup should be called `hw1b.pdf`, and any source code should either be in that directory, a subdirectory within it, or a "library" directory at the top level (in the case of files defining functions used in multiple assignments).

I suggest placing the definitions of any "helper" functions in a separate file which you load (in Python, `import`) from your main file.

I will access your work by cloning your repository; make sure that any file path information is written relative to your repo – don't use absolute paths on your machine, or the code won't run for me!

You may use any language you like to do this assignment — the tasks are stated in a language-neutral way — but Python is recommended.

You may also use any typesetting software to prepare your writeup, but the final document should be a PDF. LaTeXis highly encouraged.

All data files referred to in the problems below can be found at

`http://colindawson.net/data/<filename>.csv`.

1. **Implementing a Regression Solver From Scratch.**
   (a) (Adapted from FCML Ex. 1.2) Write a function that takes a dataset consisting of ordered pairs of the form (x_n, t_n), $n = 1, \dots, N$ (you can decide on an appropriate data structure representation for the input; one standard possibility would be an $N \times 2$ matrix/array) and returns the OLS coefficients (w_0, w_1) for a linear model of the form $t_n = w_0 + w_1 \cdot x_n$.
   (b) (Adapted from FCML Ex. 1.6) Verify that your function works using the data in `womens100.csv` containing winning race times from the Olympic women's 100m event in various years. Plot the data and the resulting line and compare the coefficients returned by your model to the coefficients returned by a pre-packaged regression solver (see also Sec. 1.2.1 of the textbook).

Date: Last Revised: February 25, 2020.

(c) The data ends with the 2008 olympics. Use your model to "predict" the winning times for the 2012 and 2016 olympics. Now look up the actual times; how does the model do? Report the squared prediction error for each year.

(d) Modify your solver function to allow multiple input variables (e.g., columns) (if it does not already allow this).

(e) Write a function that takes a single predictor column and a positive integer D, and creates a predictor matrix whose columns form a polynomial basis of order D (i.e., to append entries to the input consisting of $x_n^2, ..., x_n^D$)

(f) Write a function that takes a target vector, a single predictor column, and a positive integer, D, and returns the coefficients of the OLS polynomial function of order D.

(g) (Adapted from FCML Ex. 1.9) Apply your function to the data in `synthdata2016.csv` using a cubic polynomial ($D = 3$). The data file as it is consists simply of (x, t) pairs. Plot the data and the fitted cubic curve.

(h) Modify your solver to allow the user to specify a **regularization parameter**, λ, to do ridge regression (i.e., use the modified loss function defined as:

$$\mathcal{L}(\mathbf{w}) = (\mathbf{t} - \mathbf{X}\mathbf{w})^\mathsf{T}(\mathbf{t} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^\mathsf{T}\mathbf{w}$$

Compare results for various choices of λ using a cubic model on the `synthdata2016.csv` data.

(i) Create a version of your solver that, instead of returning the coefficients, returns a *function* which, when called on a new predictor matrix returns a vector of predictions (this will be useful for the purposes of recycling cross-validation code you wrote for the previous problem set).

2. **Cross-Validation With Regression.**

(a) Write a function that takes as input a target vector and a prediction vector, and returns the mean squared prediction error (MSE).

(b) Write a function that takes the following inputs:

(i) A target vector, $\mathbf{t}$

(ii) A predictor matrix, $\mathbf{X}$

(iii) A number of folds, K (default to 10)

(iv) A random seed (default to 1)

(v) A regularization parameter, λ (default to 0)

(vi) A boolean flag to govern whether to return training error (default to False).

and performs K-fold cross-validation. Reuse as much code as possible to have this function split $\mathbf{X}$ into K random folds, fit a (ridge) regression model to each set of $K-1$ folds as the training set, compute the mean squared prediction error (MSE) on the remaining fold (as the validation set), and return the mean and standard deviation across folds of the MSE.

Be careful when randomly splitting the data into folds that you keep input variables together with the corresponding target.

(c) Write a function to select the best polynomial order using cross-validation. It should take as inputs:

(i) A target vector

(ii) A (single column) predictor matrix

(iii) A positive integer D, representing the maximum order of polynomial to consider (default to N, the number of data points)

(iv) An integer K (at least 2), representing the number of folds to use

(v) A random seed

(vi) A boolean flag to govern whether to return training error (default to False).

Your function should perform cross-validation for each polynomial order ranging from 0 to D, and return the mean and standard deviation across folds of the MSE for each order. If the option to return training error is turned on, it should also return the mean and standard deviation across folds of the training error for each polynomial order. In addition, it should find and return the polynomial order with the lowest average cross-validation MSE.

(d) Apply the function you wrote in 2c to both the `synthdata2016.csv` and the `womens100.csv` datasets to identify the optimal polynomial order. Try both $K = 10$ and $K = N$ (i.e., leave-one-out) cross-validation. Plot the mean training and validation errors for each polynomial order, as well as error bars 1 standard deviation in either direction, and comment on what the plots show.

(e) For the `womens100.csv` dataset, we can use the 2012 and 2016 times as a true test set. Compute squared prediction errors for each of the polynomial degree models (fit on the full dataset) for 2012 and 2016. Did cross validation give the best model on this metric?

(f) Now use 10-fold cross-validation and a grid search to find a good combination of values of λ and D for a ridge regression model. How does generalization performance (on the true test set) for the chosen model and regularization parameter compare to that for the best OLS regression fit?

3. **Product Rule of Matrix Differentiation.** (No coding here)

(a) Show that for functions f and g that take vectors in $\mathbb{R}^N$ and return vectors in $\mathbb{R}^M$, N-dimensional vector $\mathbf{v}$, and scalar

$$a = f(\mathbf{v})^\mathsf{T} g(\mathbf{v})$$

the following "product rule" applies

$$\frac{da}{d\mathbf{v}} = f(\mathbf{v})^\mathsf{T} \frac{dg}{d\mathbf{v}} + g(\mathbf{v})^\mathsf{T} \frac{df}{d\mathbf{v}}$$

(b) Apply the product rule to the special case where $M = N$, f is the identity function, and $g(\mathbf{v}) = \mathbf{A}\mathbf{v}$ for some **symmetric** matrix $\mathbf{A}$ that does not depend on $\mathbf{v}$ to show that

$$\frac{d}{d\mathbf{v}} \mathbf{v}^\mathsf{T} \mathbf{A} \mathbf{v} = 2\mathbf{v}^\mathsf{T} \mathbf{A}$$

4. **Deriving a Weighted Least Squares Regression Fit.** (adapted from FCML Ex. 1.11) (No coding here) Consider a generalization of the OLS loss function that applies (known) weights to each observation, given by:

$$\mathcal{L}(\mathbf{w}; \mathbf{x}, \mathbf{t}) = \frac{1}{N} \sum_{n=1}^{N} \alpha_n (t_n - \mathbf{w}^\mathsf{T} \mathbf{x}_n)^2$$

(or, equivalently, by

$$\mathcal{L}(\mathbf{w}; \mathbf{x}, \mathbf{t}) = \frac{1}{N} (\mathbf{t} - \mathbf{X}\mathbf{w})^\mathsf{T} \mathbf{A} (\mathbf{t} - \mathbf{X}\mathbf{w})$$

where $\mathbf{X}$ is the matrix whose nth row consists of $\mathbf{x}_n$, and $\mathbf{A}_{nn} = \alpha_n$, $n = 1, \ldots, N$ and $\mathbf{A}_{mn} = 0, m \neq n$.)

Derive an expression for the estimated weight vector $\hat{\mathbf{w}}$ that minimizes this loss.