Homework 2

Due Friday April 16, 2021, by 11:59pm

Instructions: This homework consists of a reading assignment, two mathematical exercises, and two coding exercises. Please submit your solutions via Gradescope. Solutions should consist of three files: a PDF containing your solutions to the *non-coding questions*; a Jupyter notebook (`.ipynb` file) and an html print-out (`.html` file) with your solution to the *coding exercise*. **All coding exercises must be completed in Python.** Please be sure to comment the code appropriately. Students are encouraged to discuss homework problems, particularly on Canvas and in the TA hours, but must submit their own solutions.

Reading Assignments

- Review Lab 2 (available from the Course Materials page on Canvas).
- Study Sec. 5.1 to 5.5 in *Mathematics of Machine Learning*.

Problems

Please submit your solutions as a single PDF file under the `Homework 2 - Problems` assignment in Gradescope.

1 Exercise 1

Recall from Lab 2 that we defined

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}},$$

with

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \bar{y})^2 \text{ and } \text{TSS} = \sum_{i=1}^{n} (y_i - \bar{y})^2.$$

We claimed that $0 \le R^2 \le 1$. Prove this by showing that

$$\sum_{i=1}^{n} (y_i - \bar{y})^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2.$$

2 Exercise 2

In Section 2.2. of Lab 2, we showed that the MSE decomposed as

$$\text{MSE} = \sigma^2 + \text{Bias}(\hat{\beta})^2 + \text{Var}(\hat{\beta}^T x).$$

The proof assumed the following identity:

$$\mathbb{E}_y \left[(y - \bar{\beta}^\top x)^2 \right] = \mathbb{E}_y \left[(y - f(x))^2 \right] + \left(f(x) - \bar{\beta}^\top x \right)^2 .$$

Prove this result.

Coding

Please submit your solutions the coding exercise below as a Jupyter notebook (`.ipynb` file) and an html print-out (`.html` file) under the `Homework 2 - Coding` assignment in Gradescope. **Please run all cells in your notebook prior to submission, so we can view their output.**

3 Exercise 3

In this exercise, you will implement a first version of *your own gradient descent algorithm* to solve the ridge regression problem. Throughout the homeworks, you will keep improving and extending your gradient descent optimization algorithm. In this homework, you will implement a basic version of the algorithm.

Recall from Week 1 and Week 2 Lectures that the ridge regression problem writes as

$$\min_{\beta \in \mathbb{R}^d} \frac{1}{2n} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 \; + \frac{\lambda}{n} \|\beta\|_2^2 \,, \tag{1}$$

that is, if you expand,

$$\min_{\beta \in \mathbb{R}^d} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{d} \beta_j x_{ij} \right)^2 + \frac{\lambda}{n} \sum_{j=1}^{d} \beta_j^2 \,. \tag{2}$$

3.1 Remarks

Several remarks are in order.

Normalization Note that there is a $1/2n$ normalization factor in the empirical risk term in the equations. Note also that there is a λ/n multiplicative factor in the regularization penalty term in the equations.

You can actually normalize the terms any way you want *as long as you are consistent all the way through* in your mathematical derivations, your codes, and your experiments, especially when you search over parameters.

Here is some general advice:

- Do normalize the empirical risk term so that it is an average, not a sum; this normalization will be important for large scale problems where the sum can become very large. Indeed, with the normalization, the average remains of the same order of magnitude regardless of the number of terms in the sum.

- Check what optimization problem exactly is solved when you use a library, so you can compare your solution to the optimization problem to the solution found by the library and compare the optimal value of the regularization found by your grid search to the one found the library's grid search.

Intercept It is common in traditional statistics and machine learning books and libraries to include an intercept β_0 in the statistical model. Having a separate intercept coefficient is actually not that important, and provably so, especially if the data was properly centered and standardized beforehand.

There is actually a simple way to bypass the issue of having a separate intercept coefficient by adding a constant variable 1 in the variables. See Sec. 2.3.1 of *The Elements of Statistical Learning*. So the d variables in the equations correspond to the $(d-1)$ original variables plus 1 dummy variable equal to 1. See also Lab 2.

3.2 Gradient descent

The gradient descent algorithm is an iterative algorithm that is able to solve differentiable optimization problems such as (1). Define

$$F(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 \ + \frac{\lambda}{n}\|\beta\|_2^2 \ . \tag{3}$$

Gradient descent generates a sequence of iterates[1] (β_t) that converges to the optimal solution $\beta^\star$ of (1). The optimal solution of (1) is defined as

$$F(\beta^\star) = \min_{\beta\in\mathbb{R}^d} F(\beta) \ . \tag{4}$$

Gradient descent is outlined in Algorithm 1. The algorithm requires a sub-routine that computes the gradient for any β. The algorithm also takes as input the value of the constant step-size η.

- Assume that $d = 1$ and $n = 1$. The sample is then of size 1 and boils down to just (x, y). The function F writes simply as

$$F(\beta) = \frac{1}{2}(y - x\,\beta)^2 \ + \lambda\beta^2 \ . \tag{5}$$

  Compute and write down the gradient ∇F of F.

- Assume now that $d > 1$ and $n > 1$. Using the previous result and the linearity of differentiation, compute and write down the gradient $\nabla F(\beta)$ of F.

[1]The subscript t refers to the iteration counter here, not to the coordinates of the vector β.

- Consider the Penguins dataset, which you should load and divide into training and test sets using the code below.[2]

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import preprocessing

# Load the data
file = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/penguins.csv'
penguins = pd.read_csv(file, sep=',', header=0)
penguins = penguins.dropna()

# Create our X matrix with the predictors and y vector with the response
X = penguins.drop('body_mass_g', axis=1)
X = pd.get_dummies(X, drop_first=True)
y = penguins['body_mass_g']

# Divide the data into training and test sets. By default, 25% goes into the test set.
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

  Standardize the data. Note that you can convert a data frame into an array by using `np.array()`.

- Write a function *computegrad* that computes and returns $\nabla F(\beta)$ for any β. Avoid using `for` loops by vectorizing the computation.

- Write a function *graddescent* that implements the gradient descent algorithm described in Algorithm 1. The function *graddescent* calls the function *computegrad* as a subroutine. The function takes as input the initial point, the constant step-size value, and the maximum number of iterations. The stopping criterion is the maximum number of iterations.

- Set the constant step-size to $\eta = 0.5$ and the maximum number of iterations to 1000. Run *graddescent* on the training set of the Penguins dataset for $\lambda = -5.00$. Plot the curve of the objective value $F(\beta_t)$ versus the iteration counter t. Again, avoid using `for` loops when computing the objective values. What do you observe?

- Set the constant step-size to $\eta = 0.5$ and the maximum number of iterations to 1000. Run *graddescent* on the training set of the Penguins dataset for $\lambda = +0.05$. Plot the curve of the objective value $F(\beta_t)$ versus the iteration counter t. Again, avoid using `for` loops when computing the objective values. What do you observe?

- Denote β_T the final iterate of your gradient descent algorithm. Compare β_T to the $\beta^\star$ found by *sklearn.linear_model.Ridge*. Compare the objective value for β_T to the one for $\beta^\star$. What do you observe?

- Run your gradient algorithm for many values of η on a logarithmic scale. Find the final iterate, across all runs for all the values of η, that achieves the smallest value of

[2] You may encounter problems with the quotes when copying and pasting it. If so, delete the quotes that are there and retype the quotes.

Algorithm 1 Gradient Descent algorithm with fixed constant step-size

input step-size η
initialization $\beta_0 = 0$
repeat for $t = 0, 1, 2, \ldots$
$\beta_{t+1} = \beta_t - \eta \nabla F(\beta_t)$
until the stopping criterion is satisfied.

the objective. Compare β_T to the $\beta^\star$ found by *sklearn.linear_model.Ridge*. Compare the objective value for β_T to the $\beta^\star$. What conclusion to you draw?

- Change the stopping criterion from being a maximum number of iterations to an ε-stationarity condition $\|\nabla F(\beta)\| \leq \varepsilon$. Redo the last three questions now with this stopping criterion with $\varepsilon = 0.005$. Report your observations.

4 Exercise 4

Exercise 3.8 in Chapter 3 of *An Introduction to Statistical Learning* (in Python):
This question involves the use of simple linear regression on the `Auto` data set.

(a) Read in the dataset. The data can be downloaded from this url: `http://www-bcf.usc.edu/~gareth/ISL/Auto.csv` When reading in the data use the option `na_values='?'`. Then drop all NaN values using `dropna()`.

(b) Use the `OLS` function from the `statsmodels` package to perform a simple linear regression with `mpg` as the response and `weight` as the predictor. Be sure to include an intercept. Use the `summary()` attribute to print the results. Comment on the output. For example:

   (i) Is there a relationship between the predictor and the response?

   (ii) How strong is the relationship between the predictor and the response?

   (iii) Is the relationship between the predictor and the response positive or negative?

   Hint: See this URL for help with the statsmodels functions: `http://www.statsmodels.org/dev/regression.html#examples`

(c) Plot the response and the predictor using the `plot_fit` function (`http://www.statsmodels.org/dev/generated/statsmodels.graphics.regressionplots.plot_fit.html`)

(d) Plot the residuals vs. fitted values. Comment on any problems you see with the fit.