Homework 6

Due May 24, 2019 by 11:59pm

Instructions: Upload your answers to the questions below to Canvas. Submit the answers to the questions in a PDF file and your code in a (single) separate file. Be sure to comment your code to indicate which lines of your code correspond to which question part. There is 1 reading assignment and 4 exercises in this homework.

Reading Assignment

Read Ch. 6. Sec. 6.1.1 to Sec. 6.3.1, and Ch. 10 Sec. 10.2.1 to 10.3.1 in [1]. You may find the pdf of the book at the url below.

`http://www-bcf.usc.edu/~gareth/ISL/`

1 Exercise 1

In this exercise, you will implement in **Python** a first version of *your own coordinate descent algorithm* to solve the LASSO problem, that is the ℓ_1-regularized least-squares regression problem.

Recall from the lectures that the LASSO problem writes as

$$\min_{\beta \in \mathbb{R}^d} F(\beta) := \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 \; + \lambda \|\beta\|_1 \; . \tag{1}$$

Coordinate Descent

The coordinate descent algorithm is outlined in Algorithm 1. The algorithm requires a subroutine that performs the partial minimization of the objective for any coordinate β_j.

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

  $$F(\beta) = (y - x\,\beta)^2 \; + \lambda|\beta| \; . \tag{2}$$

  The solution to the minimization problem

  $$\min_{\beta} F(\beta) \tag{3}$$

  is given in Slide 71 of the Week 5 Lecture. Write a function that computes the formula of the solution of this minimization problem.

```
Algorithm 1 Coordinate Descent Algorithm, general form
initialization   $\beta = 0$.
repeat for $t = 0, 1, 2, \dots$
    – Pick a coordinate index $j$ in $\{1, \dots, d\}$
    – Find $\beta_j^{\text{new}}$ by minimizing $F(\beta)$ with respect to $\beta_j$ only.
    – Perform update $\beta_j^{(t+1)} = \beta_j^{\text{new}}$.
    – Perform update $\beta_\ell^{(t+1)} = \beta_\ell^{(t)}$ for all $\ell \neq j$.
until the stopping criterion is satisfied.
```

- Assume now that $d > 1$ and $n > 1$. The full minimization problem now writes as

$$F(\beta) := \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 \ + \lambda\|\beta\|_1 \ . \tag{4}$$

  Coordinate descent proceeds by sequential partial minimization with respect to each coordinate β_j, that by solving partial minimization problems of form

$$\begin{aligned}\min_{\beta_j} \quad & \frac{1}{n}\sum_{i=1}^{n}\left\{y_i - (\beta_1 x_{i,1} + \cdots + \beta_j x_{i,j} + \cdots + \beta_d x_{i,d})\right\}^2 \\ & + \lambda\left\{|\beta_1| + \cdots + |\beta_j| + \cdots + |\beta_d|\right\} \ .\end{aligned}$$

  Write a function that computes the formula of the solution of this partial minimization problem with respect to β_j for any $j = 1, \dots, d$.

- Consider the Hitters dataset from [1]. You may download it via

```
import pandas as pd
hitters = pd.read_csv('https://raw.githubusercontent.com/selva86/
  ↪ datasets/master/Hitters.csv', sep=',', header=0)
```

  (Note that the arrow is only there to indicate the line has been wrapped.) Standardize the data, i.e., center the features and divide them by their standard deviation, and center the outputs. For any categorical variables you should first convert them to indicator variables and then perform the standardization.

- Write a function *computeobj* that computes and returns $F(\beta)$ for any β.

- Write a function *cycliccoorddescent* that implements the *cyclic coordinate descent* algorithm. The cyclic coordinate descent algorithm proceeds sequentially. At each iteration, the algorithm increments the index j_t of the coordinate to minimize over. Then the algorithm performs partial minimization with respect to the coordinate β_{j_t} corresponding to that index. After updating the coordinate β_{j_t}, the algorithm proceeds to the next iteration. The function takes as input the initial point and the maximum number of iterations. The stopping criterion is the maximum number of iterations.

- Write a function *pickcoord* that samples uniformly from the set $\{1, \ldots, d\}$.
- Write a function *randcoorddescent* that implements the *randomized coordinate descent* algorithm. The randomized coordinate descent algorithm proceeds as follows. At each iteration, the algorithm samples the index j_t of the coordinate to minimize over. Then the algorithm performs partial minimization with respect to the coordinate β_{j_t} corresponding to that index. After updating the coordinate β_{j_t}, the algorithm proceeds to the next iteration. The function takes as input the initial point and the maximum number of iterations. The stopping criterion is the maximum number of iterations.
- Set the maximum number of iterations to 1000. In the remainder, the iteration counter iter refers here to t/d, that is the effective number of passes over all coordinates. Run cross-validation on the training set of the Hitters dataset using *scikit-learn* to find the optimal value of λ. Run *cycliccoorddescent* and *randcoorddescent* on the training set of the Hitters dataset for that value of λ found by cross-validation. Plot the curves of the objective values $F(\beta_t)$ for both algorithms versus the iteration counter iter (use different colors). What do you observe?
- Denote by β_T the final iterates of your coordinate descent algorithms for that value of λ. Compute $\beta^\star$ found by *scikit-learn* for that value of λ. Plot the curves of the fraction of correct non-zero coefficients (with respect to $\beta^\star$) for both algorithms versus the iteration counter iter (use different colors). What do you observe? Plot the curves of the fraction of correct zero coefficients for both algorithms (with respect to $\beta^\star$) versus the iteration counter iter (use different colors). What do you observe?

2 Exercise 2

In this problem you will generate simulated data and then perform PCA on the data. You will use *your own normalized Oja algorithm* for PCA. Note that "first two principal component score vectors" refers to the results from projecting the original data to a two-dimensional space with PCA.

(a) Generate a simulated data set with 30 observations in each of three classes (i.e. 90 observations total), and 60 features. Hint: There are a number of functions in numpy that you can use to generate data. One example is the `numpy.random.normal()` function; `numpy.random.uniform()` is another option. Be sure to add a mean shift to the observations in each class so that there are three distinct classes.

(b) Run *your own normalized Oja algorithm* on the 90 observations. This algorithm was discussed in the week 8 lecture. Plot the first two principal component score vectors. Compare your results to the ones obtained with scikit-learn's PCA algorithm. Use a different color to indicate the observations in each of the three classes. If the three classes appear separated in this plot, then you're done. If not, then return to part (a) and modify the simulation so that there is greater separation between the three classes.

3 Exercise 3

In this exercise, we will generate simulated data, and will then use this data to fit a lasso model. You should use Scikit-learn in this exercise for fitting the lasso model and performing cross-validation.

(a) Use the `numpy.random.normal()` function to generate a predictor X of length $n = 100$, as well as a noise vector ϵ of length $n = 100$.

(b) Generate a response vector Y of length $n = 100$ according to the model

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon,$$

where β_0, β_1, β_2, and β_3 are constants of your choice. Here superscripts denote powers of X.

(c) Now fit a lasso model to the simulated data, using $X, X^2, \ldots, X^{10}$ as predictors. Use cross-validation to select the optimal value of λ. Create plots of the cross-validation error as a function of λ. Report the resulting coefficient estimates, and discuss the results obtained.

(d) Now generate a response vector Y according to the model

$$Y = \beta_0 + \beta_7 X^7 + \epsilon,$$

and fit a lasso model. Discuss the results obtained.

4 Data Competition Project

Read the announcement "Data Competition 2" on Canvas. Convert each image into a feature vector using the provided extract_features.py script. We strongly recommend you perform this task on AWS.

- Pick two classes of your choice from the dataset. Train an ℓ_2^2-regularized logistic regression classifier on the training set using your own fast gradient algorithm with $\lambda = 1$. Plot, with different colors, the *misclassification error* on the training set and on the validation set vs iterations.

- Find the value of the regularization parameter λ using cross-validation; you may use scikit-learn's built-in functions for this purpose. Train an ℓ_2^2-regularized logistic regression classifier on the training set using your own fast gradient algorithm with that value of λ found by cross-validation. Plot, with different colors, the *misclassification error* on the training set and on the validation set vs iterations.

References

[1] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R.* Springer Publishing Company, Incorporated, 2014.