Homework 7

Due May 29th 2020 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 are 4 reading items and 2 exercises in this homework, including 1 exercise related to the data competition. The reading item is the preparation for the Week 9 class.

Reading Assignment

- Review Lecture 8.
- Review Computer Lab. 8 in *canvas.uw.edu/courses/1371621/pages/course-materials* .
- Read Section 7.7 in *An Introduction to Statistical Learning* .
- (Optional) Read Section 5.6 in *Mathematics of Machine Learning* .

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

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

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

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

Algorithm 1 Coordinate Descent Algorithm, general form

initialization $\beta = 0$.

repeat for $t = 0, 1, 2, \ldots$

- Pick a coordinate index j in $\{1, \ldots, d\}$
- Find β_j^{new} by minimizing $F(\beta)$ with respect to β_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.

The solution to the minimization problem

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

can be obtained as $\beta = (xy - \lambda)/x^2$ if $xy > +\lambda$, $\beta = (xy + \lambda)/x^2$ if $xy < -\lambda$, and $\beta = 0$ if $|xy| \leq \lambda$. Write a function that computes the formula of the solution of this minimization problem with respect to β for any x, y, and λ. In order to check your function, for various values of x, y, and λ, plot the function F with respect to β, and verify by visual inspection that the minimum coincides with the value given by the formula above.

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

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

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

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

Write a function that computes the formula of the solution of this partial minimization problem with respect to β_j for any $j = 1, \ldots, d$. You may want to build off the formula from the previous question. Check your function to make sure it works as you expect.

- Consider the Hitters dataset from *An Introduction to Statistical Learning* . 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 Data Competition Related Exercise

In this exercise, you are going to train support vector machines (SVMs) using the competition dataset with **all** classes. You will use a linear SVM classifier using scikit-learn's function `LinearSVC` for this exercise. You will consider here *all classes* in the dataset. You may work on this exercise on your own computer first. Whenever you are asked to run your code on AWS, *the screenshot of the output is supposed to show that you indeed ran your code on AWS*. You may want to read again the announcement "Data Competition" released earlier on Canvas.

- In a one-vs-one fashion, for each pair of classes, train a linear SVM classifier using scikit-learn's function `LinearSVC`, with the default value for the regularization parameter. Compute the *multi-class misclassification error* obtained using these classifiers trained in a one-vs-one fashion.

- In a one-vs-rest fashion, for each class, train a linear SVM classifier using scikit-learn's function `LinearSVC`, with the default value for the regularization parameter for each class. Compute the multi-class misclassification error obtained using these classifiers trained in a one-vs-rest fashion.

- Run your code on AWS. Read again the instructions given in the previous homework. Take a screenshot of (1) the output and (2) your instances `https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:sort=instanceId` . Submit your results to Kaggle.

- (Optional) Redo all questions above, now tuning both the regularization parameter and the imbalance parameter using scikit-learn's cross-validation. Run preliminary small-scale experiments before frantically submitting multiple submissions to Kaggle.