Homework 3

Due April 26, 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, including for the data competition exercise. Be sure to comment your code to indicate which lines of your code correspond to which question part. There is 1 reading assignment and 5 exercises in this homework. Exercise 4 is the first milestone of the data competition. Exercise 5 is optional, not to be turned in.

Reading Assignment

Read Sec. 4.4 to 4.5 in *The Elements of Statistical Learning*.

1 Exercise 1

In this exercise, you will implement in **Python** a first version of *your own fast gradient algorithm* to solve the ℓ_2^2-regularized logistic regression problem.

Recall from the lectures that the logistic regression problem writes as

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

We use here the machine learning convention for the labels that is $y_i \in \{-1, +1\}$.

1.1 Fast Gradient

The fast gradient algorithm is outlined in Algorithm 1. The algorithm requires a subroutine that computes the gradient for any β.

- 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

$$F(\beta) = \log(1 + \exp(-yx\,\beta) \; + \lambda \beta^2 \, . \tag{2}$$

  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.

Algorithm 1 Fast Gradient Algorithm

input step-size η_0, target accuracy ε
initialization $\beta_0 = 0$, $\theta_0 = 0$
repeat for $t = 0, 1, 2, \ldots$
Find η_t with backtracking rule
$\beta_{t+1} = \theta_t - \eta_t \nabla F(\theta_t)$
$\theta_{t+1} = \beta_{t+1} + \frac{t}{t+3}(\beta_{t+1} - \beta_t)$
until the stopping criterion $\|\nabla F\| \leq \varepsilon$.

- Consider the Spam dataset from *The Elements of Statistical Learning* (You can get it here: `https://web.stanford.edu/~hastie/ElemStatLearn/`). Standardize the data (i.e., center the features and divide them by their standard deviation, and also change the output labels to +/- 1).
- Write a function *computegrad* that computes and returns $\nabla F(\beta)$ for any β.
- Write a function *backtracking* that implements the backtracking rule.
- Write a function *graddescent* that implements the gradient descent algorithm with the backtracking rule to tune the step-size. The function *graddescent* calls *computegrad* and *backtracking* as subroutines. The function takes as input the initial point, the initial step-size value, and the target accuracy ε. The stopping criterion is $\|\nabla F\| \leq \varepsilon$.
- Write a function *fastgradalgo* that implements the fast gradient algorithm described in Algorithm 1. The function *fastgradalgo* calls *computegrad* and *backtracking* as subroutines. The function takes as input the initial step-size value for the backtracking rule and the target accuracy ε. The stopping criterion is $\|\nabla F\| \leq \varepsilon$.
- Use the estimate described in the course to initialize the step-size. Set the target accuracy to $\varepsilon = 10^{-4}$. Run *graddescent* and *fastgradalgo* on the training set of the Spam dataset for $\lambda = 0.1$. Plot the curve of the objective values $F(\beta_t)$ for both algorithms versus the iteration counter t (use different colors). What do you observe?
- Denote by β_T the final iterate of your fast gradient algorithm. Compare β_T to the $\beta^\star$ found by *scikit-learn*. Compare the objective value for β_T to the one for $\beta^\star$. What do you observe?
- Run cross-validation on the training set of the Spam dataset using *scikit-learn* to find the optimal value of λ. Run *graddescent* and *fastgradalgo* to optimize the objective with that value of λ. Plot the curve of the objective values $F(\beta_t)$ for both algorithms versus the iteration counter t. Plot the misclassification error on the training set for both algorithms versus the iteration counter t. Plot the misclassification error on the test set for both algorithms versus the iteration counter t. What do you observe?

2 Exercise 2

Suppose we estimate the regression coefficients in a logistic regression model by minimizing

$$F(\beta) := \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + \exp(-y_i\, x_i^T \beta)\right) \; + \lambda \|\beta\|_2^2$$

for a particular value of λ. For parts (a) through (e), indicate which of (i) through (v) is correct. Justify your answer.

(a) As we increase λ from 0, the misclassification error on the test set will:

(i) Increase initially, and then eventually start decreasing in an inverted U shape.

(ii) Decrease initially, and then eventually start increasing in a U shape.

(iii) Steadily increase.

(iv) Steadily decrease.

(v) Remain constant.

(b) Repeat (a) for the misclassification error on the training set.

3 Exercise 3

In this exercise, you will use Amazon Web Services (AWS) to run a nearest neighbors algorithm on data from the Sloan Digital Sky Survey. The goals of this exercise are:

1. To teach you how to use AWS's Elastic Compute Cloud (EC2) and Simple Storage Service (S3)

2. To show you that computing with GPUs can be much faster than computing with CPUs

3. To demonstrate a fast algorithm for computing nearest neighbors (from [1])

Background: The Sloan Digital Sky Survey (SDSS) has gathered data on many objects in the sky. The data set you will use in this assignment consists of (1) a "training" set of photometric data on known astronomical objects; and (2) a "test" set of additional photometric data on more objects. Astronomers often need to determine which objects in a new data set are worth further examination because their telescope time is very limited. By finding the nearest neighbors in the training set to each object in the test set, they can tell whether the object is interesting or not. For example, if an object in the test set is closest to sun-like stars, then they might find that one very boring and not follow up on it.

Instructions: Following the AWS tutorial for help, do the following:

1. Create a p2.xlarge spot instance with the "Deep Learning AMI (Ubuntu) Version 22.0" AMI. If this instance type doesn't work, try another one of the gpu instances. You need to use a GPU instance for this assignment.

2. Install Swig and bufferkdtree on your instance.

3. On your instance, run the astronomy example from here `https://github.com/gieseke/bufferkdtree/tree/master/examples` and save all of the output to the file "output.txt". **Hint:** You will need to transfer both astronomy.py and generate.py on EC2. In addition, in the file astronomy.py you will need to change the line

   ```
   plat_dev_ids = {0:[0,1,2,3]}
   ```

   to

   ```
   plat_dev_ids = {0:[0]}
   ```

   The data will automatically download the first time you run it **after user input.** Therefore, do not try to save the output to a file via `>` the first time you run it or else it will hang indefinitely. Either use the `tee` command (`https://en.wikipedia.org/wiki/Tee_(command)`), run the script once before using `>` or copy and paste the output to a file. If you are still having issues with it hanging when saving directly to a file, check the output file. It's possible that it finished writing out the output.

4. Transfer the file output.txt to an S3 bucket.

5. Go to the S3 interface `https://console.aws.amazon.com/s3/home?region=us-east-1` and make that file public.

6. In your homework submission please include the following:

   (a) The fitting and testing times on both the CPU and GPU versions. You can find these in the output.

   (b) The url from the previous step (Check to make sure you successfully made it public!)

   (c) A statement of any problems you encountered during this exercise and how you overcame them (or if you didn't).

   (d) How long it took you to complete this exercise (for our reference–we're not grading you on how long it took).

Remember to terminate your instance when you are done with it!

4 Exercise 4

Read the announcement "Data Competition, Part 1" released on Canvas. We strongly recommend you perform this task on AWS. You will use *your own ℓ_2^2-regularized logistic regression* for this exercise. After completing this exercise, submit your predictions to the data competition Kaggle website.

- Download the data for the Kaggle competition. Run the script `extract_features.py` to extract features from the images. This script was written in Python 3 and depends on the library PyTorch.

- 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 $\lambda = 1$. Be sure to use the features you generated above rather than the raw image features. 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.

5 Optional Exercise

It is well-known that ridge regression tends to give similar coefficient values to correlated variables, whereas the lasso may give quite different coefficient values to correlated variables. We will now explore this property in a very simple setting.

Suppose that $n = 2, p = 2, x_{11} = x_{12}, x_{21} = x_{22}$. Furthermore, suppose that $y_1 + y_2 = 0$ and $x_{11} + x_{21} = 0$ and $x_{12} + x_{22} = 0$, so that the estimate for the intercept in a least squares, ridge regression, or lasso model is zero: $\hat{\beta}_0 = 0$.

(a) Write out the ridge regression optimization problem in this setting.

(b) Argue that in this setting, the ridge coefficient estimates satisfy $\beta_1 = \beta_2$.

References

[1] F. Gieseke, J. Heinermann, C. E. Oancea, and C. Igel. Buffer kd trees: Processing massive nearest neighbor queries on gpus. In *ICML*, pages 172–180, 2014.