Obligatory Assignment 3

AI801, Optimization
Due date: 31 May 2026, 11:59pm

Instructions

- Each *individual* student must submit their own solutions to be considered for a grade and the additional 2.5 ECTS. It is okay to discuss with other students, but the solutions and the code have to be *individually* written.
- Submit your solutions as a single zip file named *asg3.zip* to itsLearning, with one PDF containing all the explanations and the plots and 2 different .py or .ipynb files (one for question 1 and one for question 2 and 3) that has the code. Additionally, 'deklarationsformular-nat-gai.pdf' Generative AI declaration form from the Faculty must be duly filled and submitted as well. The document in Word format must be submitted compiled in PDF form. This form can be found on the course webpage.
- This assignment will be graded with a Pass/Fail grade and only those that obtain a passing grade will be awarded 2.5 ECTS.
- Attempt as many questions as possible to maximize your chances for passing.
- Show all relevant calculations and arguments.
- Clearly state any assumptions you make.
- The assignment has a total of **100 points**.
- The deadline is strict and no extension will be provided.

Question 1 **Total: 35 points**

We now study projected online gradient descent (OGD) for logistic regression. At each time $t = 1, \ldots, T$, the learner receives a sample

$$(a_t, y_t) \in \mathbb{R}^d \times \{-1, +1\},$$

suffers the logistic loss

$$h_t(w) = \log\left(1 + \exp(-y_t a_t^\top w)\right),$$

and updates its parameter using projected OGD over the constraint set

$$\mathcal{W} = \{w \in \mathbb{R}^d : \|w\|_2 \leq 2\}.$$

The update is

$$w_{t+1} = \Pi_{\mathcal{W}}\left(w_t - \eta \nabla h_t(w_t)\right),$$

where

$$\Pi_{\mathcal{W}}(u) = \begin{cases} u, & \text{if } \|u\|_2 \leq 2, \\ 2\dfrac{u}{\|u\|_2}, & \text{otherwise.} \end{cases}$$

Let

$$w_{\text{true}} = \frac{1}{\sqrt{d}}(1, \ldots, 1) \in \mathbb{R}^d.$$

Use $d = 20$ and $T = 1000$. The data stream is generated as follows. For each t, draw

$$a_t \sim \mathcal{N}(0, I_d),$$

where I_d is the d-dimensional identity matrix. For the first half of the stream, generate labels using

$$\mathbb{P}(y_t = +1 \mid a_t) = \sigma(w_{\text{true}}^\top a_t), \qquad t \le T/2.$$

For the second half of the stream, generate labels using the opposite classifier:

$$\mathbb{P}(y_t = +1 \mid a_t) = \sigma(-w_{\text{true}}^\top a_t), \qquad t > T/2.$$

Here

$$\sigma(z) = \frac{1}{1 + \exp(-z)}.$$

Use the following code to generate the data.

```
import numpy as np

def sigmoid(z):
    return 1.0 / (1.0 + np.exp(-z))

def generate_online_logistic_data(T=1000, d=20, seed=0):
    rng = np.random.default_rng(seed)

    w_true = np.ones(d) / np.sqrt(d)
    A = rng.normal(loc=0.0, scale=1.0, size=(T, d))

    logits = A @ w_true
    logits[T // 2:] *= -1.0

    probs = sigmoid(logits)
    y = np.where(rng.uniform(size=T) < probs, 1, -1)

    return A, y, w_true
```

(a) For projected OGD over a convex set $\mathcal{W}$, recall the standard regret bound for a constant stepsize $\eta > 0$:

$$\text{Regret}(T) \le \frac{D^2}{2\eta} + \frac{\eta}{2} G^2 T,$$

Using the regret bound above, choose the constant stepsize η that minimizes the upper bound. What regret rate does this choice imply? Once this is chosen, substitute $D = 4$, $G = 10$ and $T = 1000$ and report the value of η. **[10 points]**

(b) Implement projected OGD using the constant stepsize chosen in part (a) and Initialize $w_1 = 0$. Compute the regret against the best fixed comparator in hindsight:

$$\text{Regret}(T) = \sum_{t=1}^{T} h_t(w_t) - \min_{w \in \mathcal{W}} \sum_{t=1}^{T} h_t(w).$$

Approximate the minimizer using a standard numerical optimizer with the constraint

$$\|w\|_2 \le 2.$$

Consider using scipy.optimize.minimize with method="SLSQP" to solve the constrained optimization problem; see the official SciPy documentation.[1]. Plot the average regret

$$\frac{\text{Regret}(t)}{t}$$

for $t \in \{1, 2, \ldots, T\}$. **[15 points]**

(c) Compare the non-stationary stream above with an IID stream generated using

$$\mathbb{P}(y_t = +1 \mid a_t) = \sigma(w_{\text{true}}^\top a_t) \qquad \text{for all } t.$$

For comparison, use the following code to generate an IID online stream.

```
def generate_iid_online_logistic_data(T=1000, d=20, seed=1):
    rng = np.random.default_rng(seed)

    w_true = np.ones(d) / np.sqrt(d)
    A = rng.normal(loc=0.0, scale=1.0, size=(T, d))

    logits = A @ w_true
    probs = sigmoid(logits)
    y = np.where(rng.uniform(size=T) < probs, 1, -1)

    return A, y, w_true
```

Plot the average regret for both streams in the same figure. Should the average regret tend towards 0 in both these cases? Why or why not? Briefly explain any other differences in the plots for these two cases. (**Hint**: To save some computation, reuse the Regret values from the previous part and only compute fresh values for the IID case). **[10 points]**

Question 2 **Total: 40 points**

We study Synchronous SGD for regularized logistic regression. Let $d = 20$, and let $w_{\text{true}} \in \mathbb{R}^d$ be the data-generating parameter. Throughout the experiment, use

$$w_{\text{true}} = \frac{1}{\sqrt{d}}(1, \ldots, 1) \in \mathbb{R}^d.$$

For each sample $i = 1, \ldots, N$, draw

$$a_i \sim \mathcal{N}(0, I_d),$$

where I_d is the d-dimensional identity matrix and then draw $y_i \in \{-1, +1\}$ according to

$$\mathbb{P}(y_i = +1 \mid a_i) = \sigma(w_{\text{true}}^\top a_i), \qquad \sigma(z) = \frac{1}{1 + \exp(-z)}.$$

We fit the regularized logistic-regression objective

$$h(w) = \frac{1}{N} \sum_{i=1}^{N} \log\left(1 + \exp(-y_i a_i^\top w)\right) + \frac{\lambda}{2}\|w\|^2. \tag{1}$$

[1] `https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html`

Remark. The vector w_{true} is the parameter used to generate the data, but it need not minimize the finite-sample regularized objective h, which is minimized by w_{opt}, i.e., the minimizer of (1).

Use $N = 10000$, $d = 20$, and $\lambda = 10^{-2}$.

Use the following code to generate the data.

```
import numpy as np

def sigmoid(z):
    return 1.0 / (1.0 + np.exp(-z))

def generate_logistic_data(N=10000, d=20, seed=0):
    rng = np.random.default_rng(seed)

    # Data-generating parameter
    w_true = np.ones(d) / np.sqrt(d)

    # Features
    X = rng.normal(loc=0.0, scale=1.0, size=(N, d))

    # Label probabilities
    probs = sigmoid(X @ w_true)

    # Labels in {-1, +1}
    y = np.where(rng.uniform(size=N) < probs, 1, -1)

    return X, y, w_true
```

Generate $N = 10000$ samples with the above code and randomly partition the data into $M = 10$, IID (independent and identically distributed) shards

$$S_1, \ldots, S_{10}.$$

Implement Synchronous SGD, where each worker computes a mini-batch gradient with a batch-size $b = 10$ on its own shard and then the following update is performed at each iteration:

$$\text{For each worker } m = 1, \ldots, M, \text{ sample a mini-batch } B_t^m \subseteq S_m \text{ of size } b,$$
$$g_t^m = \frac{1}{b} \sum_{i \in B_t^m} \nabla h(w_t; (a_i, y_i)).$$

Then perform the following updates for $T_{\max} = 10000$ steps, with $w_0 = 0$.

$$w_{t+1} = w_t - \eta_t \frac{1}{M} \sum_{m=1}^{M} g_t^m.$$

Use the learning-rate schedule

$$\eta_t = \frac{1}{25.01 + 10^{-2} t}.$$

(a) Plot $\|w_T - w_{\text{true}}\|_2$ and $h(w_T) - h(w_{\text{opt}})$ as functions of T in separate plots, where w_{opt} is computed using either a long full-batch gradient descent run or a standard numerical optimizer (e.g., using scipy.optimize.minimize with method = "BFGS" or "L-BFGS"). Is the optimization problem in (1) convex, strongly convex, or neither? Justify your answer. Based on this classification, what convergence rate with respect to the number of iterations T should one expect for Synchronous SGD, assuming we have an appropriate learning-rate schedule.

[20 points]

(b) Repeat the experiment for

$$M \in \{1, 10, 100\}.$$

Keep all other parameters fixed, same as before (in part (a)). Plot $h(w_T) - h(w_{\text{opt}})$ as a function of T for the different values of M in the **same** plot. Briefly explain why increasing M can reduce the variance of the averaged stochastic gradient. Justify mathematically or experimentally. **[10 points]**

(c) Fix $M = 10$ and everything else as in part (a). Compare the following learning-rate schedules:

$$\eta_t = \frac{1}{L_h}, \qquad \eta_t = \frac{1}{L_h + \lambda t}, \qquad \eta_t = \frac{1}{L_h(t+1)^2}.$$

Use

$$\lambda = 10^{-2}, \qquad L_h = 25.01.$$

Plot the objective gap $h(w_T) - h(w_{\text{opt}})$ for each schedule in the **same** plot and briefly explain the observed behavior. **[10 points]**

Question 3 **Total: 25 points**

Use the same logistic-regression setup as in the previous question with the same global dataset (given by the python code that generates the data. Store and reuse it) and set:

$$w_0 = 0, \qquad \eta_t = \frac{1}{25.01 + 10^{-2}t}.$$

(a) Implement local-SGD with $M = 10$ workers and IID data shards (split them as in the previous question).

Let $T_{\max}$ be the total number of local SGD steps per worker and let H be the number of local steps between two communication rounds. Then how many rounds of communication ($R_{\max}$) are there in total? (**Hint**: Note that $T_{\max}$ is divisible by H)

For each $H \in \{10, 100\}$, run local-SGD and record the objective gap after each communication round:

$$h(w_R) - h(w_{\text{opt}}), \qquad R = 1, 2, \ldots, R_{\max}.$$

Plot these curves against the communication round R in the same figure. Explain the tradeoff as H increases. **[15 points]**

(b) Create a non-IID partition as follows. Sort the samples first by label, with -1 preceding $+1$. Within each label class, keep the original ordering of the samples, so that smaller original indices come before larger ones. Then split the sorted dataset into $M = 10$ consecutive shards.

Explain why this partition may violate the IID assumption across workers. Run local-SGD on this non-IID partition and compare its behavior with the IID partition from part (a) by plotting

$$h(w_R) - h(w_{\text{opt}}), \qquad R = 1, 2, \ldots, R_{\max}.$$

in the same figure for the IID and the non-IID sharding. For this fix $H = 10$. **[10 points]**

Remark. In this assignment for Questions 2 and 3, you should *simulate* the workers sequentially in a single Python program. That is, you do not need to use multiprocessing, GPUs, networking, or any distributed-computing library. The terms "worker" and "communication round" refer only to the mathematical algorithm: each worker corresponds to one data shard, and a communication round corresponds to averaging the simulated workers' model vectors in your code. To save time and compute, you can reuse the value of the solution to optimization problems that is already computed in Question 2 for Question 3 where applicable.