EE546 HW1

Assigned. Sept. 13 Due Sept. 25

Kangyan Xu

1. Non-convex optimization

Assume: x^* is global minima of f. For all $x \in R^n$ obeying $\|x-x^*\|_{\ell_2} \le R$.

$\langle \nabla f(x), x-x^* \rangle \geq \frac{1}{\alpha}\|x-x^*\|_{\ell_2}^2 + \frac{1}{\beta}\|\nabla f(x)\|_{\ell_2}^2$ for some $\alpha > 0$.

$\|x_0 - x^*\|_{\ell_2} \le R$. $0 < \mu < \frac{2}{\beta}$. $x_{\tau+1} = x_\tau - \mu \nabla f(x_\tau)$

Prove: For all τ we have $\|x_\tau - x^*\|_{\ell_2} \le (1-\frac{2\mu}{\alpha})^\tau \|x_0 - x^*\|_{\ell_2}^2$

P: $\|x_\tau - x^*\|_{\ell_2}^2 = \|x_{\tau-1} - \mu\nabla f(x_{\tau-1}) - x^*\|_{\ell_2}^2 = \|x_{\tau-1} - x^* - \mu\nabla f(x_{\tau-1})\|_{\ell_2}^2$

$= \|x_{\tau-1} - x^*\|_{\ell_2}^2 - 2\mu\langle \nabla f(x_{\tau-1}), x_{\tau-1} - x^*\rangle + \mu^2\|\nabla f(x_{\tau-1})\|_{\ell_2}^2$

$\le \|x_{\tau-1} - x^*\|_{\ell_2}^2 - \frac{2\mu}{\alpha}\|x_{\tau-1} - x^*\|_{\ell_2}^2 - \frac{2\mu}{\beta}\|\nabla f(x_{\tau-1})\|_{\ell_2}^2 + \mu^2\|\nabla f(x_{\tau-1})\|_{\ell_2}^2$

$= (1-\frac{2\mu}{\alpha})\|x_{\tau-1} - x^*\|_{\ell_2}^2 + \mu(\mu - \frac{2}{\beta})\|\nabla f(x_{\tau-1})\|_{\ell_2}^2$

$\le (1-\frac{2\mu}{\alpha})\|x_{\tau-1} - x^*\|_{\ell_2}^2$

So: $\|x_\tau - x^*\|_{\ell_2}^2 \le (1-\frac{2\mu}{\alpha})\|x_{\tau-1} - x^*\|_{\ell_2}^2$

$\Rightarrow \|x_\tau - x^*\|_{\ell_2}^2 \le (1-\frac{2\mu}{\alpha})^\tau \|x_0 - x^*\|_{\ell_2}^2$

---

2. Convergence to stationary points and the PL-inequality

Assume: Let $f: R^n \to R$ be an L-smooth function. $x_{t+1} = x_t - \mu\nabla f(x_t)$.

(a) if $\mu \le \frac{1}{L}$ and function is bounded below ($f(x^*) \le f(x_t)$), then there exists: $\lim_{t\to+\infty}\|\nabla f(x_t)\|_{\ell_2} = 0$.

P: $f(x_{t+1}) \le f(x_t) - \mu(1-\frac{\mu L}{2})\|\nabla f(x_t)\|_{\ell_2}^2$

$\Rightarrow f(x_{t+1}) \le f(x_0) - \mu(1-\frac{\mu L}{2})\sum_{s=0}^{t}\|\nabla f(x_s)\|_{\ell_2}^2$

$\Rightarrow \sum_{s=0}^{t}\|\nabla f(x_s)\|_{\ell_2}^2 \cdot \mu(1-\frac{\mu L}{2}) \le f(x_0) - f(x_{t+1}) \le f(x_0) - f(x^*)$

$\Rightarrow \sum_{s=0}^{t}\|\nabla f(x_s)\|_{\ell_2}^2 \le \frac{1}{\mu(1-\frac{\mu L}{2})}(f(x_0) - f(x^*)) = \text{constant}$

So: $\lim_{t\to\infty}\sum_{s=0}^{t}\|\nabla f(x_s)\|_{\ell_2}^2 \le \text{constant} \Rightarrow \lim_{t\to\infty}\|\nabla f(x_t)\|_{\ell_2} = 0$

(b) Does x_t converges to a fixed point?

No.

Eg. function whose optim can be realized on interval $[x_i, x_j]$.

(c) Assume: function also obeys PL-inequality: $\|\nabla f(x)\|_{\ell_2}^2 \geq \gamma(f(x) - f(x^*))$, $\gamma > 0$. $\mu \leq 1/L$.

Prove: $(f(x_{t+1}) - f(x^*)) \leq (1 - \frac{\mu\gamma}{2})(f(x_t) - f(x^*))$.

P: Know: $f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\|\nabla f(x_t)\|_{\ell_2}^2$

$$\Rightarrow f(x_t) - f(x_{t+1}) \geq \frac{1}{2L}\|\nabla f(x_t)\|_{\ell_2}^2 \geq \frac{\gamma}{2L}(f(x_t) - f(x^*)) \geq \frac{\mu\gamma}{2}(f(x_t) - f(x^*))$$

$$\Rightarrow f(x_t) - \frac{\mu\gamma}{2} f(x_t) + \frac{\mu\gamma}{2} f(x^*) - f(x^*) \geq f(x_{t+1}) - f(x^*)$$

$$\Rightarrow (1 - \frac{\mu\gamma}{2})(f(x_t) - f(x^*)) \geq f(x_{t+1}) - f(x^*)$$

---

3. Logistic regression with momentum

$$(\hat{w}, \hat{b}) = \underset{(w,b)}{\operatorname{argmin}}\ f(w,b) = \sum_{i=1}^{N} \left[-y_i(w^T x_i + b) + \log(1 + \exp(w^T x_i + b))\right] + \frac{\lambda}{2}\|w\|_{\ell_2}^2$$

(ℓ_2-regularization)

$$b_{t+1} = b_t - \mu \cdot \sum_{i=1}^{N} \left(\frac{1}{1 + \exp -(w^T x_i + b)} - y_i \right)$$

$$W_{t+1} = W_t - \mu \left[\sum_{i=1}^{N} \left(\left(\frac{1}{1 + \exp -(w^T x_i + b)} - y_i \right) x_i \right) + \lambda W_t \right]$$

$b \in R$

$W \in R^{30}$

$X \in R^{500 \times 30}$

$Y \in R^{500}$

Combine these two parts:

A new $W \in R^{31}$, $W = \begin{pmatrix} w_1 \\ \vdots \\ w_{30} \\ b \end{pmatrix}_{31 \times 1}$

A nex $X \in R^{500 \times 31}$ $X = \begin{pmatrix} x_{11} & \cdots & x_{1,30} & 1 \\ \vdots & & \vdots & \\ x_{500,1} & \cdots & x_{500,30} & 1 \end{pmatrix}_{500 \times 31}$

so: $XW = \begin{pmatrix} w^T x_1 + b \\ \vdots \\ w^T x_{500} + b \end{pmatrix}_{500 \times 1}$

And gradient becomes $\left[\underset{31\times500}{X^T} \cdot (\text{sigmoid}(XW) - Y)_{500\times1} + \lambda \cdot \underset{31\times1}{W'} \right]_{31\times1}$, $\left(W' = \begin{pmatrix} w_1 \\ \vdots \\ w_{30} \\ 0 \end{pmatrix}\right)$

End of hand write part.

Coding follows.

3. Logistic regression with momentum (**Code Parts**)

Platform: Jupyter Notebook & Python3

a & b) Read in the data and normalize

```
In [1]: # Homework_1 / Sept. 2020  / Kangyan Xu

        import pandas as pd
        import numpy as np
        from sklearn.preprocessing import normalize
        from sklearn.model_selection import train_test_split
```

```
In [2]: data = pd.read_csv('wdbc.data', header = None)
        # number = data.shape[0] # number of patients
        # print(number)
        output = data.iloc[:,1] # 0-benign or 1-malignant
        feature_raw = data.iloc[:,2:32] # Each has 30 features
```

```
In [3]: # L2 Normalize
        mean = np.mean(feature_raw) # mean vector
        feature_substracted = feature_raw-mean

        feature = normalize(feature_substracted, axis = 1, norm = 'l2')
```

```
In [4]: def sigmoid(x):
            return 1.0 / (1 + np.exp(-x))
```

c) Report the average error over the 100 trials

step size used here: 0.01

```
    # prediction on test data
    X_test = np.column_stack((X_test, np.ones([69, 1])))
    y_test = y_test.to_frame().values

    prediction = np.round(sigmoid(np.dot(X_test, weight)))

    error[i] = int(69-sum(y_test == prediction))

print(error)
average_err = np.sum(error)/100
print("The average error over the 100 trials is %.2f" %average_err)
```

```
[ 3.  4.  6.  5.  8.  4.  1.  7.  5.  7.  6.  1.  8. 10.  6.  7.  8.  2.
  6.  3.  3.  6.  4.  2.  4.  6.  4.  5.  7.  8.  5.  5.  7.  7.  4.  2.
  5.  6.  2.  3.  4.  4.  2.  4.  4.  4.  4.  8.  3.  6.  3.  6.  1.  5.
  1.  4.  3.  6.  4.  6.  1.  5.  5.  4.  4.  2.  7.  5.  5.  5.  3.  2.
  8.  6.  2.  1.  4.  6.  8.  3.  3.  7.  4.  4.  4.  3.  7.  6.  6.  9.
  6.  3.  3.  4.  4.  7.  5.  4.  5.  3.]
The average error over the 100 trials is 4.67
```

d) Report the number of iterations it takes to get to an accuracy of 10^{-6}

```
print(iterations)
print(accuracy)
average_iter = np.sum(iterations)/100
print("The average iterations over the 100 trials is %d" %average_iter)
```

```
[11473. 11266. 11520. 11381. 12234. 11318. 11810. 11257. 11367. 10992.
 11619. 10871. 12052. 10784. 11919. 12872. 10792. 11139. 11212. 11616.
 11816. 12342. 11694. 11032. 11276. 11922. 12539. 11193. 12152. 12016.
 11836. 11873. 12361. 11567. 11355. 11085. 12118. 11387. 11188. 11124.
 11425. 11444. 11383. 11226. 11447. 11377. 11065. 11363. 13403. 12728.
 11449. 11382. 11205. 10875. 12683. 11037. 11550. 11139. 12220. 11142.
 11759. 11216. 11546. 11072. 11380. 11675. 11258. 11758. 11561. 11801.
 11811. 10765. 12705. 11937. 11387. 11303. 11347. 11155. 10905. 11087.
 11071. 11385. 11154. 10915. 11086. 12417. 11785. 12349. 12167. 11336.
 10893. 11202. 11602. 11169. 10884. 11728. 11524. 11576. 10811. 11360.]
[[9.99985152e-07]]
The average iterations over the 100 trials is 11527
```

e) Perform the experiment of part (d) but now add a momentum term (1) using the heavy ball method and (2) using Nesterov's accelerated scheme

eta used here: 0.96

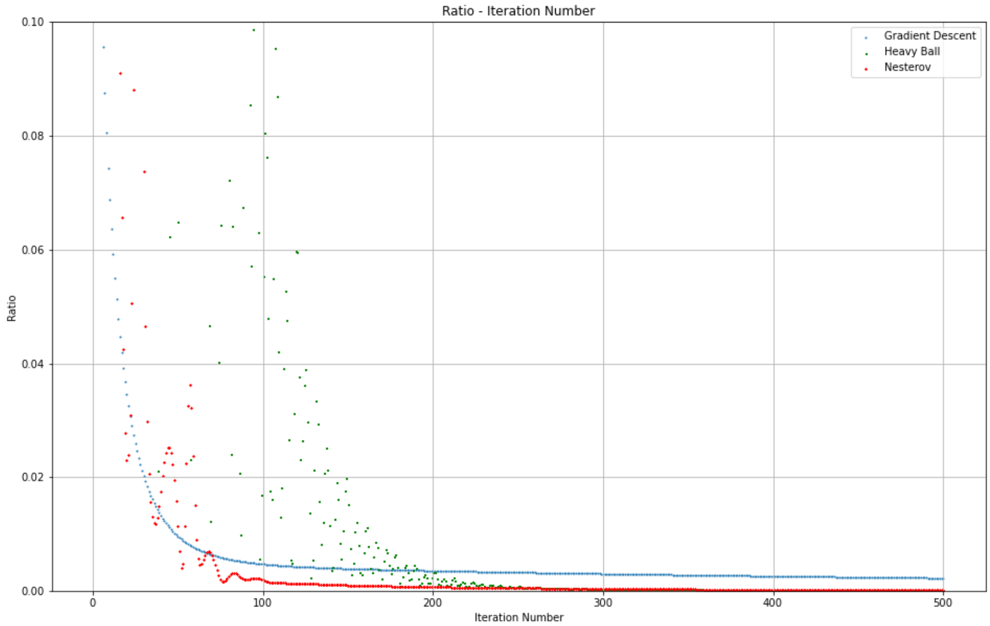
```python
print(iterations)
print(accuracy)
average_iter = np.sum(iterations)/100
print("Heavy ball: The average iterations over the 100 trials is %d" %average_iter)
```

```
[1401. 1402. 1405. 1409. 1412. 1403. 1397. 1403. 1400. 1402. 1405. 1405.
 1404. 1408. 1401. 1405. 1405. 1406. 1402. 1402. 1403. 1406. 1402. 1404.
 1403. 1407. 1406. 1401. 1406. 1408. 1405. 1403. 1405. 1408. 1403. 1400.
 1405. 1404. 1404. 1402. 1403. 1404. 1402. 1402. 1406. 1407. 1402. 1403.
 1403. 1401. 1406. 1405. 1402. 1404. 1405. 1401. 1404. 1404. 1401. 1398.
 1402. 1405. 1403. 1400. 1404. 1398. 1403. 1405. 1403. 1399. 1402. 1401.
 1405. 1406. 1403. 1404. 1402. 1402. 1406. 1402. 1403. 1405. 1403. 1400.
 1402. 1403. 1404. 1406. 1406. 1401. 1407. 1401. 1404. 1400. 1402. 1406.
 1400. 1402. 1403. 1404.]
[[9.96141206e-07]]
Heavy ball: The average iterations over the 100 trials is 1403
```

```python
print(iterations)
print(accuracy)
average_iter = np.sum(iterations)/100
print("Nesterov: The average iterations over the 100 trials is %d" %average_iter)
```

```
[1404. 1404. 1408. 1411. 1414. 1405. 1400. 1406. 1402. 1405. 1407. 1407.
 1406. 1411. 1404. 1408. 1408. 1408. 1405. 1404. 1405. 1408. 1405. 1407.
 1406. 1410. 1408. 1404. 1408. 1410. 1408. 1406. 1408. 1410. 1405. 1403.
 1407. 1407. 1406. 1405. 1406. 1407. 1405. 1405. 1409. 1410. 1404. 1406.
 1405. 1403. 1408. 1408. 1405. 1407. 1408. 1403. 1407. 1406. 1404. 1401.
 1405. 1407. 1406. 1403. 1406. 1401. 1406. 1408. 1405. 1402. 1404. 1404.
 1407. 1408. 1406. 1407. 1405. 1405. 1408. 1405. 1406. 1408. 1406. 1403.
 1405. 1406. 1407. 1408. 1409. 1404. 1409. 1404. 1407. 1403. 1405. 1408.
 1403. 1404. 1406. 1406.]
[[9.99822097e-07]]
Nesterov: The average iterations over the 100 trials is 1406
```

Draw the convergence of the three algorithms: gradient descent, heavy ball, Nesterov's accelerated scheme for one trial



Here Nesterov method is better. It converges quicker than other two methods.

end.