Ans-7 - (a) Iterative LS definition

$$x^{(k+1)} = x^{(k)} - \frac{1}{\|A\|^2} A^T (A x^{(k)} - b)$$

for $k = 0, 1, 2, \ldots$

$$\Rightarrow x^{(k+1)} = x^{(k)} - \frac{1}{\|A\|^2} A^T A x^{(k)} + \frac{A^T b}{\|A\|^2}$$

Let $P = I - \frac{1}{\|A\|^2} A^T A$, $Q = \frac{A^T b}{\|A\|^2}$

$$\boxed{x^{(k+1)} = P x^{(k)} + Q}$$

Spectral radius of $P < 1$ thus it will converge (it $= \{x^{(k)}\}$)

$\therefore$ expanding it more

$$x^{(k+1)} = P^{k+1} x^{(0)} + (P^k + P^{k-1} + \cdots + P^2 + P + I) Q$$

$$= \underbrace{(I + P + P^2 + \cdots + P^k)}_{S_k} Q$$

$$S = \lim_{k \to \infty} S_k = I + P + P^2 + \cdots + P^k$$

$$(I - P) S = I + \cancel{P} + P^2 + \cdots = I$$
$$\quad - \cancel{P} - P^2 + \cdots$$

$$\boxed{S = (I - P)^{-1}} \qquad \Rightarrow \lim_{k \to \infty} x^{(k+1)} = SQ$$

$$= (I - P)^{-1} Q$$

$$= \left(I - \left(I - \frac{1}{\|A\|^2} A^T A\right)\right)^{-1} \cdot \frac{1}{\|A\|^2} A^T b$$

Thus as $k \to \infty$ $\{x^{(k)}\}$ sequence converges to $\hat{x}$.

$$= \left(\frac{A^T A}{\|A\|^2}\right)^{-1} \cdot \frac{1}{\|A\|^2} A^T b = (A^T A)^{-1} \cdot A^T b.$$

(b) Computations

$A \in \mathbb{R}^{m \times n}$

$\|A\|_F^2 \longrightarrow$ computed in mn time (assuming we consider frobenius norm)

$A^T \longrightarrow$ computed in mn time (access each element)

order notations

Now for $Ax^{(k)} \longrightarrow mn$ steps (i.e. $O(mn)$ or $\approx mn$ steps)

$Ax^{(k)} - b \longrightarrow m$ steps

$A^T(Ax^{(k)} - b) \longrightarrow nm$ steps

$\frac{1}{\|A\|^2} A^T(Ax^{(k)} - b) \longrightarrow n$ steps

$x^{(k)} - \frac{1}{\|A\|^2} A^T(Ax^{(k)} - b) \longrightarrow n$ steps

sum ↓

$x^{(k+1)} \longrightarrow 2mn + m + 2n$ steps

To compute $\{x^{(k)}\}$

$\longrightarrow k(2mn + m + 2n)$ steps

$\longrightarrow O(kmn + km + kn)$ steps (Ans).

18CS10069_Q7

October 23, 2021

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import sklearn
import math
np.random.seed(0)
```

[2]:
```
# define A, b
A = np.random.standard_normal(size=(30,10))
b = np.random.standard_normal(size=(30,))
rankA = np.linalg.matrix_rank(A)                                                    ␣
 ↪# find the rank of matrix A
print(f'Rank of A: {rankA} \nFull column rank: {rankA == 10}')  # check rank of␣
 ↪matrix and if matrix A is full column rank
```

```
Rank of A: 10
Full column rank: True
```

[3]:
```
def ILS(A,b,steps):
    # iterative LS
    x = np.zeros(shape=(A.shape[1],))   # start with x = 0
    norm = np.linalg.norm(A)                    # find the norm for denominator    ␣
 ↪
    for i in range(steps):
        new_x  = x - (1/(norm**2)) * np.dot(np.transpose(A), np.dot(A,x) - b) #␣
 ↪new_x = x - 1/norm(A) * A.T *(A.Tx - b)
        x = new_x
    return x
```

[4]:
```
x_hat = np.linalg.pinv(A).dot(b)
x_ils = ILS(A, b, 100)
```

[5]:
```
print(f"Using pseudo inverse solution: \n {x_hat}")
```

```
Using pseudo inverse solution:
 [-0.01380771 -0.04248459  0.18512844  0.08649818 -0.08800931  0.20276114
  0.24758306 -0.01550312 -0.23988796  0.22112417]
```

[6]:
```
print(f'Iterative least squares solution: \n {x_ils}')
```

```
Iterative least squares solution:
 [-0.01435872 -0.04921549  0.18493234  0.08599041 -0.0831881   0.20676466
  0.24783107 -0.01755494 -0.23591761  0.21442819]
```

[7]:
```python
difference = np.linalg.norm(x_hat - x_ils)/math.sqrt(x_hat.shape[0])
print(f"Error (RM) between x_hat and x_ils: {rmse:.3f}")
```

```
Root mean squared error between x_hat and x_ils: 0.004
```

(c) Since the difference between Iterative LS (x_{ils}) and Pseudo Inverse ($\hat{x}$) is very small, hence algorithm converges to $\hat{x}$.

(d) Finding $\hat{x}$ involves computing the pseudo inverse, which is computationally expensive. But using Iterative LS, it is relatively easier to compute the norm and other operations. Thus, Iterative LS can be computationally beneficial.