EE412 Homework 2

Quang Minh Nguyen
Student ID: 20200854

November 26, 2022

Contents

1 Link Analysis

(a)

For this part, I use 10^{-8} as the distance between two consecutive values of v at which the computation is considered converged.

Exercise 5.1.2

The values in the list are of pages A, B, and C, in that order.

```
Exercise 5.1.2
PageRank:
 [0.2592592599681075, 0.308641974292488, 0.43209876573940437]
```

Exercise 5.3.1

The values in the lists are of pages A, B, C, and D, in that order.

```
Exercise 5.3.1, (a)
Topic-sensitive PageRank for A:
 [0.42857142660801495, 0.19047619113066164, 0.19047619113066164, 0.19047619113066164]
Exercise 5.3.1, (b)
Topic-sensitive PageRank for A and C:
 [0.38571428326001866, 0.17142857224666042, 0.2714285722466604, 0.17142857224666042]
```

Source code

Since `numpy` is not allowed, I first defined matrix and vector operations. In the function `get_pagerank`, I followed exactly the formula $v = \beta M v + (1 - \beta)e/n$ in section 5.1.5 of MMDS book. In the function `get_topic_sensitive_pagerank`, I followed exactly the formula $v = \beta M v + (1 - \beta)e_S/|S|$ in section 5.3.2 of MMDS book.

```
"""
Filename: hw3_1a.py
Author: Quang Minh Nguyen

Python source code for homework 3, task 1a
References: MMDS book
"""

# linear algebra operations


def multiply_mv(M, v):
    """Multiply a matrix M with a column vector v

    Parameters:
        M: list[list[float]], size n x n
        v: list[float], size n
    """
    return [sum([r*c for r, c in zip(row, v)]) for row in M]


def add_vv(u, v):
    """Add two vectors u and v

    Parameters:
        u: list[float], size n
        v: list[float], size n
    """
    return [x + y for x, y in zip(u, v)]


def multiply_sv(s, v):
    """Multiply a scalar s with a vector v

    Parameters:
        s: float
        v: list[float], size n
    """
    return [s * x for x in v]


def distance(u, v):
    """Calculate the Euclidean distance between two vectors u and v

    Parameters:
        u: list[float], size n
        v: list[float], size n
```

```
    """
    return sum([(x - y)**2 for x, y in zip(u, v)])**0.5


# PageRank


def get_pagerank(M, beta, tol):
    """
    Get the pagerank of a matrix M with
    1 - beta being the probability
    of teleporting to a random page and
    tol being the tolerance of convergence.

    Parameters:
        M: list[list[float]], size n x n
        beta: float
        tol: float
    """
    n = len(M)
    v_cur = [1. / n] * n
    while True:
        v = add_vv(multiply_sv(beta, multiply_mv(M, v_cur)) , \
            multiply_sv(1 - beta, [1. / n] * n))
        if distance(v, v_cur) < tol:
            break
        v_cur = v
    return v

def get_topic_sensitive_pagerank(M, beta, topic, tol):
    """
    Get the topic-sensitive pagerank of a matrix M with
    1 - beta being the probability
    of teleporting to a random page,
    topic being the indicator vector for the teleport
    set, and tol being the tolerance of convergence.

    Parameters:
        M: list[list[float]], size n x n
        beta: float
        topic: list[int], size n
        tol: float
    """
    n = len(M)
    S = sum(topic)
    v_cur = [1. / n] * n
    while True:
```

```
        v = add_vv(multiply_sv(beta, multiply_mv(M, v_cur)) , \
            multiply_sv((1 - beta)/S, topic))
        if distance(v, v_cur) < tol:
            break
        v_cur = v
    return v


# Exercise 5.1.2

M = [
    [1/3, 1/2, 0],
    [1/3, 0, 1/2],
    [1/3, 1/2, 1/2]]

EPSILON = 1e-8
BETA = .8
print("Exercise 5.1.2\nPageRank:\n", get_pagerank(M=M, beta=BETA, tol=EPSILON))


# Exercise 5.3.1

M = [
    [0, 1/2, 1, 0],
    [1/3, 0, 0, 1/2],
    [1/3, 0, 0, 1/2],
    [1/3, 1/2, 0, 0]]

EPSILON = 1e-8
BETA = .8
TOPIC = [1, 0, 0, 0] # A
print("Exercise 5.3.1, (a)\nTopic-sensitive PageRank for A:\n",
    get_topic_sensitive_pagerank(M=M, beta=BETA, topic=TOPIC, tol=EPSILON))
TOPIC = [1, 0, 1, 0] # A and C
print("Exercise 5.3.1, (b)\nTopic-sensitive PageRank for A and C:\n",
    get_topic_sensitive_pagerank(M=M, beta=BETA, topic=TOPIC, tol=EPSILON))
```

(b) Implement the PageRank algorithm using Spark

The result is

```
263     0.00202
537     0.00194
965     0.00193
243     0.00185
285     0.00183
255     0.00180
126     0.00180
```

```
502     0.00180
16      0.00179
747     0.00178
```

2 Mining Social-Network Graphs

(a)

Exercise 10.3.2

We use the following principle, which is derived using the frequent itemset argument in section 10.3.4:

$$\text{If } n\binom{d}{t}\binom{n}{t}^{-1} \geq s \text{ then the existence of } K_{s,t} \text{ is guaranteed.} \tag{1}$$

To find all maximal pairs, we vary t from 1 to d and find $s = \lfloor n\binom{d}{t}\binom{n}{t}^{-1} \rfloor$. Then we filter this with the criterion $t \leq s$.

(a) $n = 20, d = 5$. The only pair is $(1, 5)$.

(b) $n = 200, d = 150$. The pairs are $\{(1, 150), (2, 112), (3, 83), (4, 62), (5, 46), (6, 34), (7, 25), (8, 19), (9, 14), (10, 10)\}$.

Exercise 10.5.2

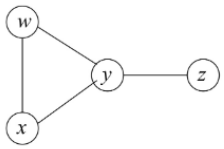

Figure 10.20: A social graph

Figure 1: Social graph for Exercise 10.5.2

The Likelihood can be computed by

$$p_{wx}p_{wy}p_{xy}p_{yz}(1 - p_{wz})(1 - p_{xz}) \tag{2}$$

for both parts (a) and (b).

(a) $C = \{w, x\}, D = \{y, z\}$.

p_{wx}	$1 - (1 - p_C) = p_C$
p_{wy}	ϵ
p_{xy}	ϵ
p_{yz}	$1 - (1 - p_D) = p_D$
$1 - p_{wz}$	$1 - \epsilon$
$1 - p_{xz}$	$1 - \epsilon$
Likelihod	$p_C p_D \epsilon^2 (1 - \epsilon)^2$
MLE	The maximum likelihood is $\epsilon^2(1 - \epsilon)^2$, at $p_C = p_D = 1$

Table 1: Computation table for exercise 10.5.2(a)

p_{wx}	$1-(1-p_C)=p_C$
p_{wy}	$1-(1-p_C)=p_C$
p_{xy}	$1-(1-p_C)(1-p_D)=p_C+p_D-p_Cp_D$
p_{yz}	$1-(1-p_C)(1-p_D)=p_C+p_D-p_Cp_D$
$1-p_{wz}$	$1-p_C$
$1-p_{xz}$	$(1-p_C)(1-p_D)$
Likelihod	$L=p_C^2(p_C+p_D-p_Cp_D)^2(1-p_C)^2(1-p_D)$
MLE	The maximum likelihood is $\frac{16}{729}$, at $p_C=2/3, p_D=0$

Table 2: Computation table for exercise 10.5.2(b)

(b) $C=\{w,x,y,z\}, D=\{x,y,z\}$.

Solving for MLE:

- Writing p_C, p_D as x, y for notational simplicity,

$$\begin{aligned}\nabla L = (-2(x-1)x(y-1)(xy-x-y)(3x^2(y-1)+x(2-4y)+y),\\ -(x-1)^2x^2(3x(y-1)-3y+2)(xy-x-y))\end{aligned} \tag{3}$$

- There exists values of x, y for which $L>0$, namely $x=y=0.5$, so the maximum of L is not achieved at $x=0$, $x=1$, or $y=1$, or $x+y-xy=0$, i.e., conditions that make $L=0$. So to find the maximiser through $\nabla L=0$, we can ignore factors x, $x-1$, $y-1$, $xy-x-y$.
- $\nabla L=0$ therefore leads to

$$\begin{aligned}3x^2(y-1)+x(2-4y)+y=0\,,\\ 3x(y-1)-3y+2=0\,,\end{aligned} \tag{4}$$

  solving which yields solutions $x_0=2/3, y_0=0$.

- $$\begin{aligned}&f_{xx}(x_0,y_0)f_{yy}(x_0,y_0)-f_{xy}(x_0,y_0)^2\\ &=[-2(-15x_0^4+20x_0^3-6x_0^2)][-2(-1+x_0)^3x_0^2(1+-3x_0)]-4(-1+x_0)^2x_0^2(9x_0^3-11x_0^2+3x_0)^2\\ &=\frac{64}{6561}>0\end{aligned} \tag{5}$$

  and

$$f_{xx}=-2(-15x_0^4+20x_0^3-6x_0^2)=\frac{-16}{27}<0\,. \tag{6}$$

  So (x_0,y_0) is a local maximum point.

- Therefore we know that $(x_0,y_0)=(2/3,0)$ is the global maximum point.

(b) Implement the triangle finding algorithm in MMDS Chapter 10.7.2

The total number of triangles is 3501542.

3 Large-Scale Machine Learning

(a) Exercise 12.5.3 (modified)

Since there are only two classes, we can write GINI impurity as

$$G(x)=1-x^2-(1-x)^2=2x-2x^2 \tag{7}$$

and Entropy measure of impurity as

$$H(x) = -x \log x - (1-x) \log(1-x)\,. \tag{8}$$

Claim. If $f''(x) < 0$ for all x then f is concave.
Proof. Due to Taylor's theorem, for any x and x_0, we can find an x between x and x_0 such that

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(x_0)(x - x_0)^2\,. \tag{9}$$

Since $f''(x) < 0$ for all x, the last term is negative, implying

$$f(x) < f(x_0) + f'(x_0)(x - x_0)\,. \tag{10}$$

Letting $x_0 \leftarrow z$ while $x \leftarrow x$ and $x \leftarrow y$, we get, respectively,

$$\begin{aligned} f(x) &< f(z) + f'(z)(x - z) \\ f(y) &< f(z) + f'(z)(y - z)\,. \end{aligned} \tag{11}$$

As such

$$\begin{aligned} \frac{y-z}{y-x} f(x) &< \frac{y-z}{y-x} f(z) + \frac{y-z}{y-x} f'(z)(x - z) \\ \frac{z-x}{y-x} f(y) &< \frac{z-x}{y-x} f(z) + \frac{z-x}{y-x} f'(z)(y - z)\,. \end{aligned} \tag{12}$$

Summing both sides of these two inequalities, we obtain

$$\frac{y-z}{y-x} f(x) + \frac{z-x}{y-x} f(y) < f(z)\,, \tag{13}$$

showing that f is concave. □

Now we check the second derivatives of $G(x)$ and $H(x)$.

- $G(x) = 2x - 2x^2 \to G'(x) = 2 - 4x \to G''(x) = -4 < 0, \forall x.$
- $H(x) = -x \log x - (1-x)\log(1-x) \to H'(x) = -\log x + \log(1-x) \to H''(x) = -\frac{1}{x} - \frac{1}{1-x} = -\frac{1}{x(1-x)} < 0, \forall x \in (0,1).$

Therefore $G(x)$ and $H(x)$ are concave, due to the claim proved above.

(b) Implementation of gradient descent SVM

I performed a grid search over $C \in$ `np.arange(0.1, 1, 0.2)`, $\eta \in$ `np.arange(0.0001, 0.0011, 0.0001)` (see Figure 2). The best result was (with average accuracy, C, and η on each line)

```
0.8453333333333333
0.5000000000000001
0.0004
```

The current code fixes C and η for fast execution. If a grid search is to be conducted, please uncomment this following last part of the code:

```
# best_c, best_eta, best_acc = grid_search(X, Y)
# print(best_acc)
# print(best_c)
# print(best_eta)
```

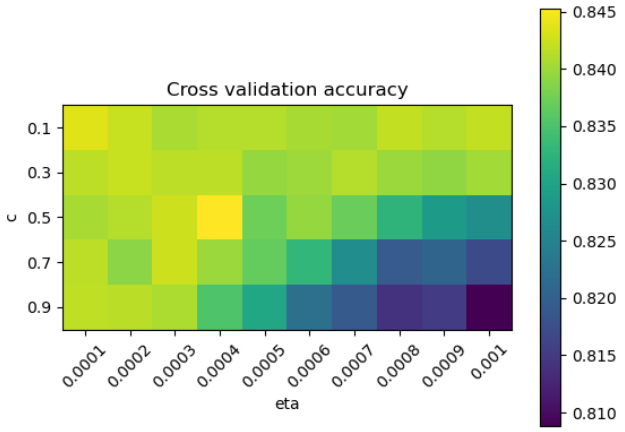

Figure 2: Grid search over $C \in$ `np.arange(0.1, 1, 0.2)`, $\eta \in$ `np.arange(0.0001, 0.0011, 0.0001)` in Problem 3(b)