EE412 Foundation of Big Data Analytics, Fall 2018
HW3

Name: 김경만

Student ID: 20150073

Discussion Group (People with whom you discussed ideas used in your answers):

On-line or hardcopy documents used as part of your answers:

Answer to Problem 1

Exercise 5.1.2 (뒤에 코드 첨부)

$M = \begin{pmatrix} 1/3 & 1/2 & 0 \\ 1/3 & 0 & 1/2 \\ 1/3 & 1/2 & 1/2 \end{pmatrix}$ 이므로 $v' = 0.8 \cdot \begin{pmatrix} 1/3 & 1/2 & 0 \\ 1/3 & 0 & 1/2 \\ 1/3 & 1/2 & 1/2 \end{pmatrix} v + 0.2 \cdot \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix}$ 이다. 첫 $v = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix}$ 이며

수렴할때까지 python 으로 반복하였다. 수렴결과, $v = \begin{pmatrix} 0.2593 \\ 0.3086 \\ 0.4321 \end{pmatrix}$ 이다.

Exercise 5.3.1 (뒤에 코드 첨부)

$M = \begin{pmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix}$ 이며 $\beta = 0.8$ 이므로 각 계산식은

(a) $S = \{A\}$ 일때, $v' = \beta M v + (1-\beta) \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ 이며 첫 $v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ 이다.

계산 결과, $v = \begin{pmatrix} 0.4285 \\ 0.1905 \\ 0.1905 \\ 0.1905 \end{pmatrix}$ 로 수렴했다

(b) $S = \{A, C\}$ 일때, $v' = \beta M v + (1-\beta) \cdot \begin{pmatrix} 1/2 \\ 0 \\ 1/2 \\ 0 \end{pmatrix}$ 이며 첫 $v = \begin{pmatrix} 1/2 \\ 0 \\ 1/2 \\ 0 \end{pmatrix}$ 이다.

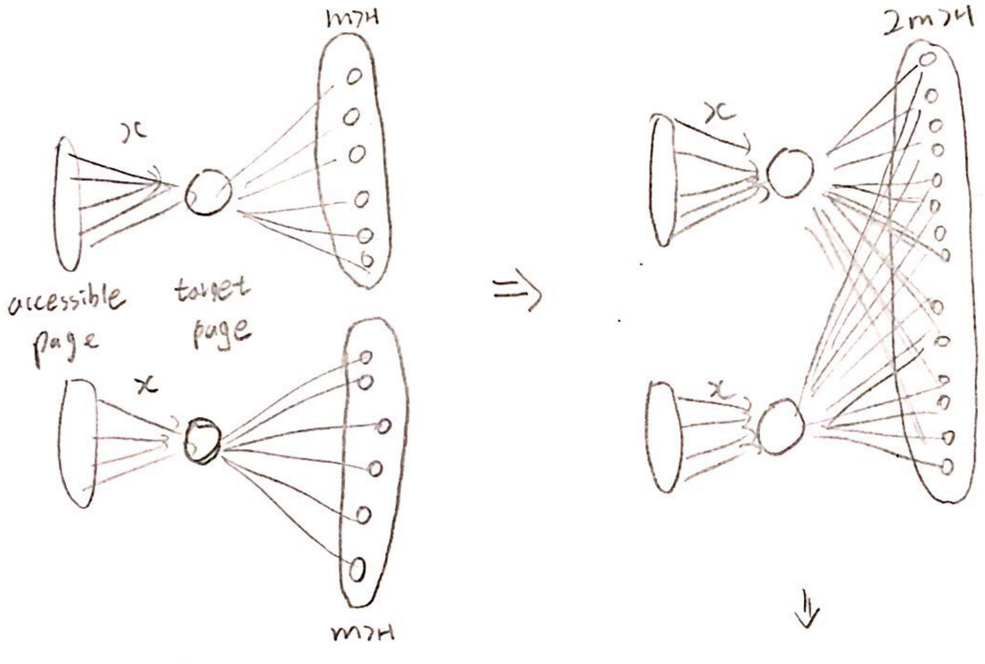
계산결과, $v = \begin{pmatrix} 0.3858 \\ 0.1714 \\ 0.2714 \\ 0.1714 \end{pmatrix}$ 로 수렴하였다.

Exercise 5.4.3

기존 target page의 Page rank가 y, m개의 supporting pages 가 있을 때 $c = \frac{\beta}{1+\beta}$

$y =$ 외부의 $x + \beta m \left(\frac{\beta y}{m} + \frac{1-\beta}{n}\right)$ 로 $y = \frac{x}{1-\beta^2} + c\frac{m}{n}$ 이다. 여기서

2개의 같은 spam farm을 합쳐 각 farm의 supporting pages 들이 2개의 target page에 연결된다고 할 때, 각 supporting pages들은 $\beta \cdot \frac{y}{2m} \cdot \underbrace{2}_{\text{2개로부터 받기때문}} + \frac{1-\beta}{n}$ 의 pagerank를 가지게 되므로 $y =$ 외부의 $x + \beta \cdot \underbrace{2m}_{\text{2m개로부터}} \cdot \underbrace{\frac{1}{2}}_{\text{각 supporting page는 2개의 target 한테 보내므로}} \cdot \left(\frac{\beta y}{m} + \frac{1-\beta}{n}\right)$ 로 같은 page rank를 갖게된다.



각 $y = \frac{x}{1-\beta^2} + c \cdot \frac{m}{n}$ (같은 spam farm) $\longleftrightarrow$ advantage가 없다. 각 $y = \frac{x}{1-\beta^2} + c \cdot \frac{m}{n}$

그러므로 2개의 target page가 supporting pages 들을 공유함으로써 생기는 advantage는 없다. 그래서 supporting pages 들은 그대로두고, target page를 연결해 보았다.

2개의 target page를 서로 연결하면 각 supporting pages 들은

$\beta \frac{y}{m+1} + \frac{1-\beta}{n}$ 의 page rank를 가지게 되므로 $y =$ 외부의 $x + \beta \cdot m \cdot \left(\beta \frac{y}{m+1} + \frac{1-\beta}{n}\right) + \underbrace{\beta \frac{y}{m+1}}_{\text{target page로부터}}$

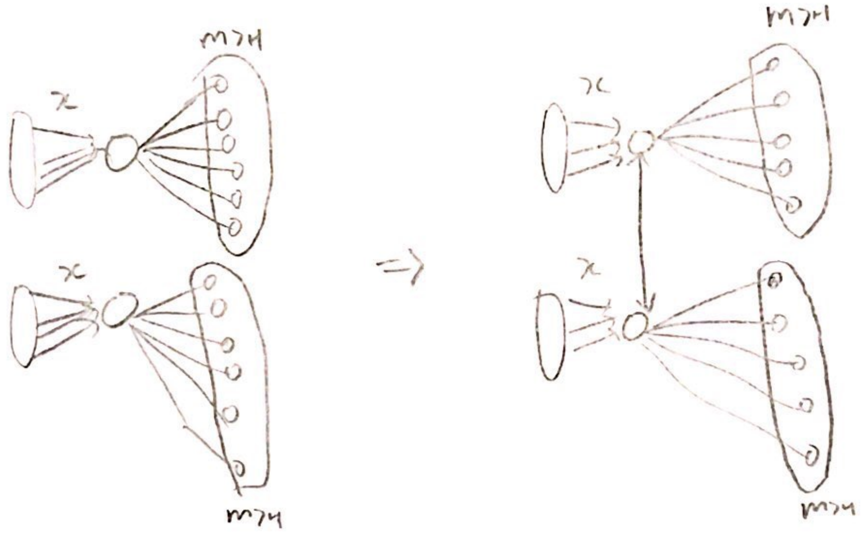
가 되므로 $y = x + \left(\frac{m\beta^2+\beta}{m+1}\right) y + \frac{m}{n} \cdot \beta(1-\beta)$ 가 된다. 그러므로 기존 식에서는

$y = x + \beta^2 y + \frac{m}{n}\beta(1-\beta)$ 의 형태이므로 기존의 y를 y_o, 새로운 구조의 y를 y_n 라 할때

$(1-\beta^2) y_o = x + \frac{m}{n}\beta(1-\beta) = \left(1 - \frac{m\beta^2+\beta}{m+1}\right) y_n$ 이다. 여기서 β는 traction으로

$0 < \beta < 1$ 이다. 그러므로, $\beta > \beta^2$ 이며, 따라서 $1-\beta^2 > 1 - \frac{m\beta^2+\beta}{m+1}$ 이다.

그러므로 $y_o < y_n$, 즉 새로운 구조의 page rank가 더 크다.



$(1-\beta^2) y = x + \frac{m}{n}\beta \cdot (1-\beta) \Longleftrightarrow \left(1 - \frac{m\beta^2+\beta}{m+1}\right) y = x + \frac{m}{n}\beta(1-\beta)$

advantage 있다.

supporting pages 들은 그대로 두고 2개의 target page를 연결하자

각 target page의 page rank에 advantage가 생겼다.

Answer to Problem 2

Exercise 10.1.1

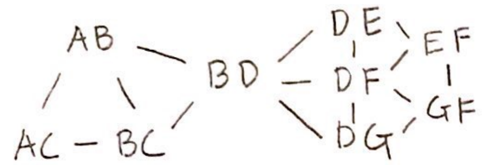
(a) 기존 그래프 G의 node가 사람, edge가 친구관계임을 설명해준다면 G'의 edge는 두 친구관계(node) 사이에 공통으로 사람(edge)을 의미하게된다

끼어있는

Y가 공통된 사람이다

X − Y − Z ⇒ XY —— YZ

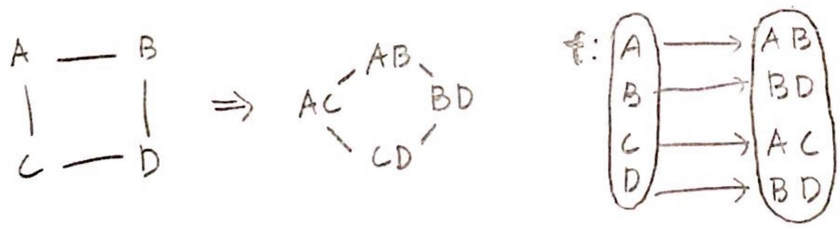
(b)



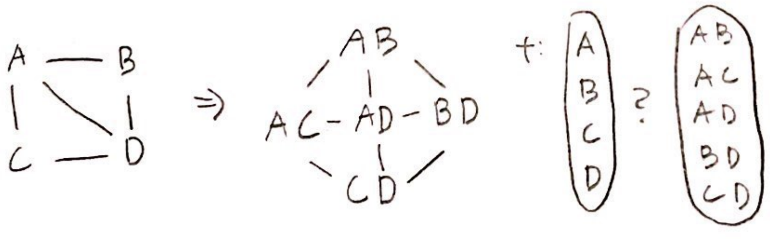
(c) degree(XY) = degree(X) + degree(Y) − 2 이다. XY의 이웃 중 X가 공통된 이웃 degree(X)−1 개, Y가 공통된 이웃 degree(Y)−1 개 이기 때문이다.

↳ $\overline{XY}$ ↳ $\overline{YX}$

(d)



G와 G'은 isomorphic 하다.



G와 G'은 one-to-one & onto function f를 가질수없으므로 not isomorphic 하다.

Exercise 10.3.2

(a) $n\left(\frac{d}{n}\right)^t \geq s$ 가 만족되어야 $K_{s,t}$를 갖는게 보장된다. $20 \cdot \left(\frac{1}{4}\right)^t \geq s$, $t \leq s$ 를 만족하는 maximal pair (t,s) 는 $\{(1,5)\}$ 이다.

(b) 위와 같이, $200 \cdot \left(\frac{3}{4}\right)^t \geq s$, $t \leq s$ 를 만족하는 set of maximal pair (t,s) 는 $\{(1,150), (2,112), (3,84), (4,63), (5,47), (6,35), (7,26), (8,20), (9,15), (10,11)\}$ 이다

Exercise 10.5.2

(a) $C = \{w, x\}$, $D = \{y, z\}$ 라고 할 때, $P_{wx} = 1-(1-P_C) = P_C$, $P_{yz} = 1-(1-P_D) = P_D$ 이며

나머지 $P_{wy} = P_{wz} = P_{xy} = P_{xz} = \epsilon$ 이다. 그러므로 이 그래프의 MLE는

$P_{wx}P_{wy}P_{xy}P_{yz}(1-P_{wz})(1-P_{xz}) = P_C \cdot P_D \cdot \epsilon^2 (1-\epsilon)^2$ 이다 그러므로 maximize를

위해 $P_C = 1$, $P_D = 1$ 이다. 즉, C 와 D의 멤버가 확실히 (100%) 그들끼리의

edge를 가지고 있을때 Figure 10.20 와 Maximum Likelihood 가 된다

(b) $C = \{w, x, y, z\}$, $D = \{x, y, z\}$ 라고 할때, $P_{wx} = P_{wy} = P_{wz} = 1-(1-P_C) = P_C$ 이며

$P_{xy} = P_{zz} = P_{yz} = 1-(1-P_C)(1-P_D) = P_C + P_D - P_C P_D$ 이다 그러므로 이 그래프의 MLE는

$P_{wx}P_{wy}P_{xy}P_{yz}(1-P_{wz})(1-P_{xz}) = P_C^2 \cdot (P_C+P_D-P_CP_D)^2 (1-P_C)(1-P_C-P_D+P_CP_D)$

$= P_C^2 \cdot (P_C+P_D-P_CP_D)^2 (1-P_C)^2 (1-P_D)$ 이다 이를 maximize 하는 P_C, P_D 값은

$P_C = \frac{2}{3}$ $P_D = 0$ 이다. 즉, C의 멤버끼리는 $\frac{2}{3}$의 확률로 edge가 존재하고,

D 멤버끼리는 0의 확률로 edge를 가지고 있을때 Figure 10.20 와

Maximum Likelihood 가 된다.

(그들끼리 edge를 가지지 않을때)

Answer to Problem 3

Exercise 12.3.2 (python 코드 참고)

먼저 positive example은 합이 10이 넘고, negative sample은 합이 10이 안되므로 초기 w와 b는

$w = [x, y, z] = [1, 1, 1]$, $b = -10$ 으로 정한다. 그럼 모든 sample에 대해 $y(wx+b) \geq 1$ 을 만족한다.

이제 각 weight, bias 에 대해 $w_j := w_j - \eta \frac{\partial f}{\partial w_j} = w_j - \eta \left(w_j + C \sum_{i=1}^{6} \overbrace{\left(-y_i x_{ij} \right)}^{\text{misclassify 된 점만 세기}} \right)$,

$b := b - \eta \left(b + C \sum_{i=1}^{6} \overbrace{(-y_i)}^{\text{misclassify만 세기}} \right)$ 를 이용하여 python으로 update 했다.

$\eta = 0.001$, $C = 5$ (misclassify 된 점을 없게하기 위해) 를 이용한 결과,

$w = [x, y, z] = [0.024, 0.534, 0.516]$, $b = -3.789$ 가 나왔다.

이를 다시 점에 대입하여 $w \cdot x + b = \pm 1$ 과 가장 가까운 점을 따졌을 때,

([3,4,5],+1), ([2,7,2],+1), ([3,3,2],-1) 이 세점이 support vector라고 할 수 있다.

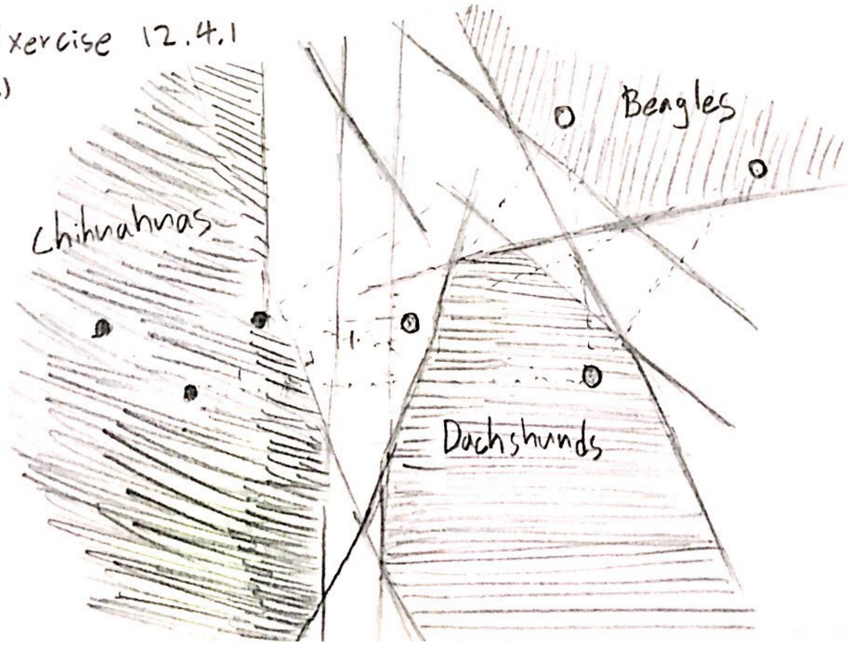
(a) ([3,4,5],+1), ([2,7,2],+1), ([3,3,2],-1)

(b) $w = [1,1,1]$, $b = -10$

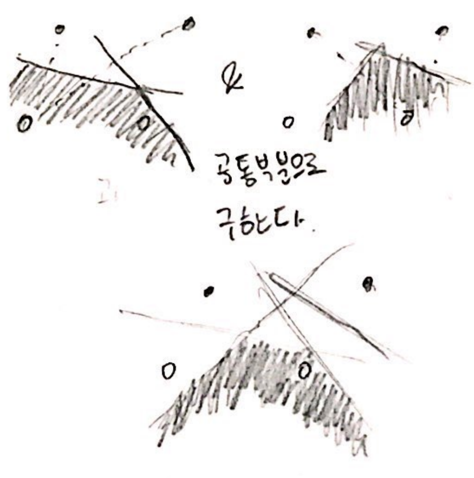
(c) $w = [0.024, 0.534, 0.516]$, $b = -3.789$

Exercise 12.4.1

(a)



빗금 없는 곳은 unclassified 이다.



(b) 그렇다. 2-NN에서의 class A의 region은 A의 점이 n개 있을때, 우선 각 점에서

다른 class의 점보다 더 가까운 범위를 그린다.

그리고 모든 nC_2 의 순서쌍의 공통된 범위를 모두 합친다. 이것이 2-NN에서의

region이다. 위에선 순서쌍이 1개이므로 (점이 2개니까) 공통된 범위도 한 개다.

즉, 각 점에서의 다른 class 점보다 더 가까운 범위는 직선으로 이루어지며 이러한

범위들의 공통된 부분의 합으로 2-NN region이 결정되므로 boundary는

무조건 직선 segments로 이루어져있다.

Exercise 12.5.3

(a) 2개의 class로 P_1, P_2 중 $P_1 = x$, $P_2 = 1-x$ 이므로 $f(x) = 1 - x^2 - (1-x)^2 = -2x^2 + 2x$ 이다.

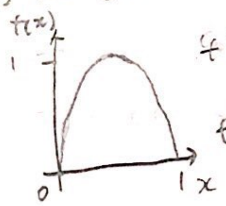
$x < z < y$ 일때, 주어진 식은 $-2z^2 + 2z > \frac{z-x}{y-x} \cdot (-2y^2 + 2y) + \frac{y-z}{y-x} \cdot (-2x^2 + 2x)$ 이며 정리하면

$zy(y-z) + zx(z-x) > xy(y-x) = xy(y-z) + xy(z-x)$ 이므로 이항하면

$y(z-x)(y-z) > x(y-z)(z-x)$ 가 되어 $y > x$ 로 항상 성립한다. 그러므로,

GINI impurity 는 오목함수 (concave) 하다.

(b) 위와같이 $f(x) = -x\log x - (1-x)\log(1-x)$ 이며 $0 \le x \le 1$ 이며 그래프 개형은 다음과같다.

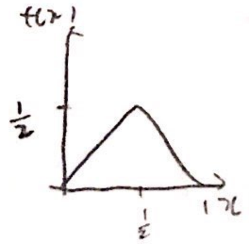


$f'(x) = -\log x - \frac{1}{\ln 2} + \log(1-x) + \frac{1}{\ln 2} = \log(1-x) - \log x$

$f''(x) = \frac{-1}{(1-x)\ln 2} - \frac{1}{x \cdot \ln 2} = -\frac{1}{\ln 2} \cdot \left(\frac{1}{x} + \frac{1}{(1-x)}\right)$, $0 < x < 1$ 에서

$\frac{1}{x} + \frac{1}{1-x}$ 은 항상 양수이므로 $f''(x) < 0$ 이다. 그러므로 $f(x)$는 오목함수(concave)이다

(c) $f(x) = 1 - \max(x, 1-x)$ 로 그래프 개형은 다음과같다.



$0 \le x < z < y \le 1$ 에서 $x = 0$, $y = \frac{1}{2}$ 일때 z는 $\overline{xy}$ 직선과 겹쳐 다니므로

항상 $f(z) = \frac{z-x}{y-x} f(y) + \frac{y-z}{y-x} f(x) = 2z \cdot \frac{1}{2} + 0 = z = 1 - (1-z)$

$= 1 - \max(z, 1-z)$ 이다. ($0 < z < \frac{1}{2}$ 이므로) 그러므로 $x=0$, $y=\frac{1}{2}$ 일 때

concave 하지 않다.

EE412 HW3 20150073 김경만

1. Link Analysis

Exercise 5.1.2

```
import numpy as np

M = np.array([[1/3.0, 1/2.0,   0  ],
              [1/3.0,   0,   1/2.0],
              [1/3.0, 1/2.0, 1/2.0]])
v = np.array([[1/3.0],
              [1/3.0],
              [1/3.0]])
e_n = v
for _ in range(50):
    v = 0.8 * np.dot(M, v) + 0.2 * e_n

print(v)
```

50번 반복한 결과, 더 이상 반복했을 때의 결과와 차이가 없이 수렴했음을 확인했다. 수렴 결과 각 a, b, c의 page rank는 0.2593, 0.3086, 0.4321 이었다.

Exercise 5.3.1-(a)

```
import numpy as np

M = np.array([[  0,    1/2.0,   1,    0  ],
              [1/3.0,   0,      0,  1/2.0],
              [1/3.0,   0,      0,  1/2.0],
              [1/3.0, 1/2.0,    0,    0  ]])
v = np.array([[1],
              [0],
              [0],
              [0]])
e_n = v
for _ in range(50):
    v = 0.8 * np.dot(M, v) + 0.2 * e_n

print(v)
```

50번 반복한 결과, 더 이상 반복했을 때의 결과와 차이가 없이 수렴했음을 확인했다. 수렴 결과 각 A, B, C, D의 page rank는 0.4285, 0.1905, 0.1905, 0.1905이었다.

Exercise 5.3.1-(b)

```python
import numpy as np

M = np.array([[  0,   1/2.0,   1,    0  ],
              [1/3.0,   0,     0,  1/2.0],
              [1/3.0,   0,     0,  1/2.0],
              [1/3.0, 1/2.0,   0,    0  ]])
v = np.array([[1/2.0],
              [  0  ],
              [1/2.0],
              [  0  ]])
e_n = v
for _ in range(50):
    v = 0.8 * np.dot(M, v) + 0.2 * e_n

print(v)
```

50번 반복한 결과, 더 이상 반복했을 때의 결과와 차이가 없이 수렴했음을 확인했다. 수렴 결과 각 A, B, C, D의 page rank는 0.3858, 0.1714, 0.2714, 0.1714이었다.

3. Large-Scale Machine Learning

Exercise 12.3.2

```
import numpy as np

x = np.array([[3,4,5],
              [2,7,2],
              [5,5,5],
              [1,2,3],
              [3,3,2],
              [2,4,1]])
y = [1, 1, 1, -1, -1, -1]

def misclassify(w, b, x, y):
    index = []
    for i in range(6):
        if y[i] * (np.dot(w, x[i].T) + b) < 1:
            index.append(i)
    return index

w = np.array([1.0, 1.0, 1.0])
b = -10.0
C = 5
learning_rate = 0.001
for _ in range(1000):
    index = misclassify(w, b, x, y)
    sum_0 = 0
    sum_1 = 0
    sum_2 = 0
    sum_b = 0
    for i in index:
        sum_0 += -1 * y[i] * x[i][0]
        sum_1 += -1 * y[i] * x[i][1]
        sum_2 += -1 * y[i] * x[i][2]
        sum_b += -1 * y[i]
    w[0] -= learning_rate * (w[0] + C * sum_0)
    w[1] -= learning_rate * (w[1] + C * sum_1)
    w[2] -= learning_rate * (w[2] + C * sum_2)
    b -= learning_rate * (b + C * sum_b)

print(misclassify(w,b,x,y))
print(w, b)
```

Misclassify 함수는 misclassified 된 점들의 index를 반환하는 함수다. 이를 이용하여 gradient descent를 진행하였고 결과적으로 모든 점이 제대로 분류되었으며 w = [0.024, 0.534, 0.516], b = -3.789였다.