(수치해석 과제 #6)

6장, P227~p267

(In and Out Practic)

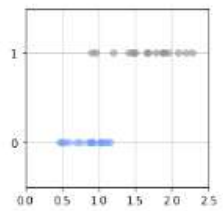
소프트웨어학과 20162820 김영민

#list 6-1-(1) ~ 6-1-(3)

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

np.random.seed(seed=0) # 난수를 고정
X_min = 0
X_max = 2.5
X_n = 30
X_col = ['cornflowerblue', 'gray']
X = np.zeros(X_n) # 입력 데이터
T = np.zeros(X_n, dtype=np.uint8) # 목표 데이터
Dist_s = [0.4, 0.8] # 분포의 시작 지점
Dist_w = [0.8, 1.6] # 분포의 폭
Pi = 0.5 # 클래스 0의 비율
for n in range(X_n):
    wk = np.random.rand()
    T[n] = 0 * (wk < Pi) + 1 * (wk >= Pi) # (A)
    X[n] = np.random.rand() * Dist_w[T[n]] + Dist_s[T[n]] # (B)

print('X=' + str(np.round(X, 2)))
print('T=' + str(T))
```

```
X=[1.94 1.67 0.92 1.11 1.41 1.65 2.25 0.47 1.07 2.19 2.05 1.02 0.91 1.16
 1.46 1.02 0.85 0.69 1.79 1.89 0.75 0.9  1.87 0.5  0.69 1.5  0.98 0.59
 1.21 0.8 ]
T=[1 1 0 0 1 1 1 0 0 1 1 0 0 0 1 0 0 0 1 1 0 1 1 0 0 1 1 0 1 0]
```

```
def show_data1(x, t):
    K = np.max(t) + 1
    for k in range(K): # (A)
        plt.plot(x[t == k], t[t == k], X_col[k], alpha=0.5,
                 linestyle='none', marker='o') # (B)
        plt.grid(True)
        plt.ylim(-.5, 1.5)
        plt.xlim(X_min, X_max)
        plt.yticks([0, 1])

fig = plt.figure(figsize=(8, 3))
show_data1(X, T)
plt.show()
```



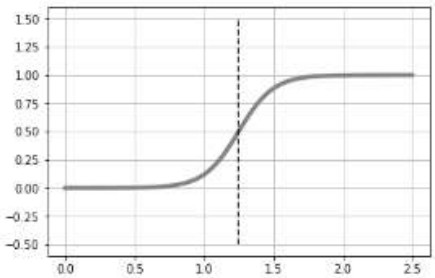

```
def logistic(x, w):
    y = 1 / (1 + np.exp(-(w[0] * x + w[1])))
    return y
```

#list 6-1-(4) ~ 6-1-(5)

```
def logistic(x, w):
    y = 1 / (1 + np.exp(-(w[0] * x + w[1])))
    return y
```

```
def show_logistic(w):
    xb = np.linspace(X_min, X_max, 100)
    y = logistic(xb, w)
    plt.plot(xb, y, color='gray', linewidth=4)
    i = np.min(np.where(y > 0.5)) # (A)
    B = (xb[i - 1] + xb[i]) / 2 # (B)
    plt.plot([B, B], [-.5, 1.5], color='k', linestyle='--')
    plt.grid(True)
    return B

W = [8, -10]
show_logistic(W)
```

```
1.25
```




```
# 리스트 6-1-(5)
# 평균 교차 엔트로피 오차 ----------------------
def cee_logistic(w, x, t):
    y = logistic(x, w)
    cee = 0
    for n in range(len(y)):
        cee = cee - (t[n] * np.log(y[n]) + (1 - t[n]) * np.log(1 - y[n]))
    cee = cee / X_n
    return cee


# test
W=[1,1]
cee_logistic(W, X, T)
```

```
1.0266191541651066
```

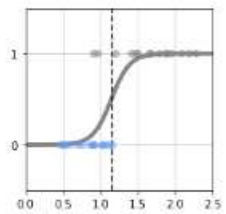
#list 6-1-(6)

```
def dcee_logistic(w, x, t):
    y = logistic(x, w)
    dcee = np.zeros(2)
    for n in range(len(y)):
        dcee[0] = dcee[0] + (y[n] - t[n]) * x[n]
        dcee[1] = dcee[1] + (y[n] - t[n])
    dcee = dcee / X_n
    return dcee

W=[1, 1]
dcee_logistic(W, X, T)
```

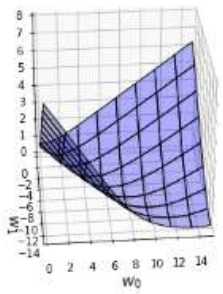
```
array([0.30657905, 0.38485474])
```

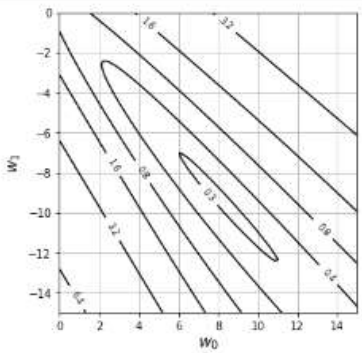
```
from scipy.optimize import minimize

def fit_logistic(w_init, x, t):
    res1 = minimize(cee_logistic, w_init, args=(x, t),
                    jac=dcee_logistic, method="CG") # (A)
    return res1.x

plt.figure(1, figsize=(8, 8))
W_init=[1,-1]
W = fit_logistic(W_init, X, T)
print("w0 = {0:.2f}, w1 = {1:.2f}".format(W[0], W[1]))
B=show_logistic(W)
show_data1(X, T)
plt.ylim(-.5, 1.5)
plt.xlim(X_min, X_max)
cee = cee_logistic(W, X, T)
print("CEE = {0:.2f}".format(cee))
print("Boundary = {0:.2f} g".format(B))
plt.show()
```

```
w0 = 8.16, w1 = -9.86
CEE = 0.25
Boundary = 1.15 g
```



In [8]:
```
from mpl_toolkits.mplot3d import Axes3D

xn = 80 # 등고선 표시 해상도
w_range = np.array([[0, 15], [-15, 0]])
x0 = np.linspace(w_range[0, 0], w_range[0, 1], xn)
x1 = np.linspace(w_range[1, 0], w_range[1, 1], xn)
xx0, xx1 = np.meshgrid(x0, x1)
C = np.zeros((len(x1), len(x0)))
w = np.zeros(2)
for i0 in range(xn):
    for i1 in range(xn):
        w[0] = x0[i0]
        w[1] = x1[i1]
        C[i1, i0] = cee_logistic(w, X, T)

plt.figure(figsize=(12, 5))
plt.subplots_adjust(wspace=0.5)
ax = plt.subplot(1, 2, 1, projection='3d')
ax.plot_surface(xx0, xx1, C, color='blue', edgecolor='black',
                rstride=10, cstride=10, alpha=0.8)
ax.set_xlabel('$w_0$', fontsize=14)
ax.set_ylabel('$w_1$', fontsize=14)
ax.set_xlim(0, 15)
ax.set_ylim(-15, 0)
ax.set_zlim(0, 8)
ax.view_init(30, -95)

plt.subplot(1, 2, 2)
cont = plt.contour(xx0, xx1, C, 20, colors='black',
                   levels=[0.28, 0.4, 0.8, 1.6, 3.2, 6.4])
cont.clabel(fmt='%1.1f', fontsize=8)
plt.xlabel('$w_0$', fontsize=14)
plt.ylabel('$w_1$', fontsize=14)
plt.grid(True)
plt.show()
```





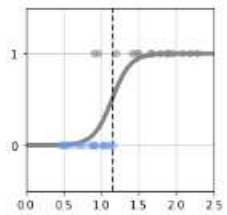
#list 6-1-(7) ~ (8)

```
def dcee_logistic(w, x, t):
    y = logistic(x, w)
    dcee = np.zeros(2)
    for n in range(len(y)):
        dcee[0] = dcee[0] + (y[n] - t[n]) * x[n]
        dcee[1] = dcee[1] + (y[n] - t[n])
    dcee = dcee / X_n
    return dcee

W=[1, 1]
dcee_logistic(W, X, T)
```

```
array([0.30657905, 0.89485474])
```

```
from scipy.optimize import minimize

def fit_logistic(w_init, x, t):
    res1 = minimize(cee_logistic, w_init, args=(x, t),
                    jac=dcee_logistic, method="CG") # (A)
    return res1.x

plt.figure(1, figsize=(8, 8))
W_init=[1,-1]
W = fit_logistic(W_init, X, T)
print("w0 = {0:.2f}, w1 = {1:.2f}".format(W[0], W[1]))
B=show_logistic(W)
show_data1(X, T)
plt.ylim(-.5, 1.5)
plt.xlim(X_min, X_max)
cee = cee_logistic(W, X, T)
print("CEE = {0:.2f}".format(cee))
print("Boundary = {0:.2f} g".format(B))
plt.show()
```

```
w0 = 8.16, w1 = -9.86
CEE = 0.25
Boundary = 1.15 g
```



#list 6-2-(1) ~ (4)

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

np.random.seed(seed=1)  # 난수를 고정
N = 100 # 데이터의 수
K = 3 # 분포 수
T3 = np.zeros((N, 3), dtype=np.uint8)
T2 = np.zeros((N, 2), dtype=np.uint8)
X = np.zeros((N, 2))
X_range0 = [-3, 3] # X0 범위 표시 용
X_range1 = [-3, 3] # X1의 범위 표시 용
Mu = np.array([[-.5, -.5], [.5, 1.0], [1, -.5]]) # 분포의 중심
Sig = np.array([[.7, .7], [.8, .8], [.8, .8]]) # 분포의 분산
Pi = np.array([0.4, 0.6, 1]) # (A) 각 분포에 대한 비율 0.4 0.6 1
for n in range(N):
    wk = np.random.rand()
    for k in range(K): # (B)
        if wk < Pi[k]:
            T3[n, k] = 1
            break
    for k in range(2):
        X[n, k] = (np.random.randn() * Sig[T3[n, :] == 1, k]
                   + Mu[T3[n, :] == 1, k])
T2[:, 0] = T3[:, 0]
T2[:, 1] = T3[:, 1] | T3[:, 2]
```

```
print(X[:5,:])
```

```
[[-0.14178627  0.66533886]
 [-0.86972028 -1.25107604]
 [-2.15442602  0.29474174]
 [ 0.75523126  0.92516669]
 [-1.10198462  0.74062584]]
```

```
print(T2[:5,:])
```

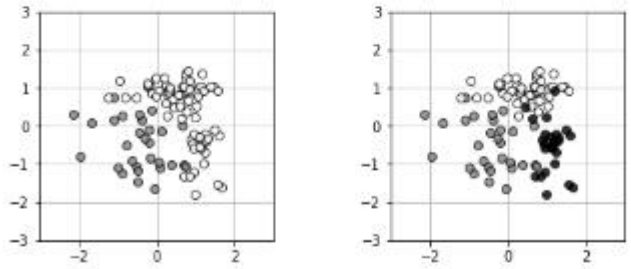
```
[[0 1]
 [1 0]
 [1 0]
 [0 1]
 [1 0]]
```

```
print(T3[:5,:])
```

```
[[0 1 0]
 [1 0 0]
 [1 0 0]
 [0 1 0]
 [1 0 0]]
```

#list 6-2-(5) ~ (6)

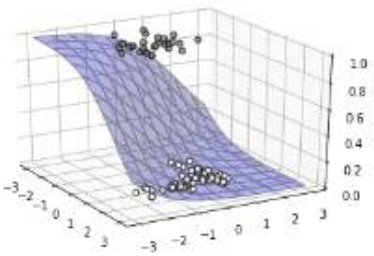
```
def show_data2(x, t):
    wk, K = t.shape
    c = [[.5, .5, .5], [1, 1, 1], [0, 0, 0]]
    for k in range(K):
        plt.plot(x[t[:, k] == 1, 0], x[t[:, k] == 1, 1],
                 linestyle='none', markeredgecolor='black',
                 marker='o', color=c[k], alpha=0.8)
        plt.grid(True)


# 메인 ----------------------------
plt.figure(figsize=(7.5, 3))
plt.subplots_adjust(wspace=0.5)
plt.subplot(1, 2, 1)
show_data2(X, T2)
plt.xlim(X_range0)
plt.ylim(X_range1)


plt.subplot(1, 2, 2)
show_data2(X, T3)
plt.xlim(X_range0)
plt.ylim(X_range1)
plt.show()
```



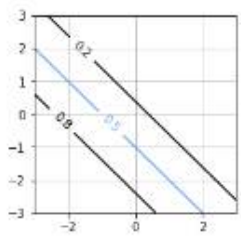
```
def logistic2(x0, x1, w):
    y = 1 / (1 + np.exp(-(w[0] * x0 + w[1] * x1 + w[2])))
    return y
```

#list 6-2-(7) ~ (8)

```
from mpl_toolkits.mplot3d import axes3d


def show3d_logistic2(ax, w):
    xn = 50
    x0 = np.linspace(X_range0[0], X_range0[1], xn)
    x1 = np.linspace(X_range1[0], X_range1[1], xn)
    xx0, xx1 = np.meshgrid(x0, x1)
    y = logistic2(xx0, xx1, w)
    ax.plot_surface(xx0, xx1, y, color='blue', edgecolor='gray',
                    rstride=5, cstride=5, alpha=0.8)


def show_data2_3d(ax, x, t):
    c = [[.5, .5, .5], [1, 1, 1]]
    for i in range(2):
        ax.plot(x[t[:, i] == 1, 0], x[t[:, i] == 1, 1], 1 - i,
                marker='o', color=c[i], markeredgecolor='black',
                linestyle='none', markersize=6, alpha=0.8)
    Ax.view_init(elev=25, azim=-30)


# test ---
Ax = plt.subplot(1, 1, 1, projection='3d')
W=[-1, -1, -1]
show3d_logistic2(Ax, W)
show_data2_3d(Ax,X,T2)
```



```
def show_contour_logistic2(w):
    xn = 30 # 파라미터의 분할 수
    x0 = np.linspace(X_range0[0], X_range0[1], xn)
    x1 = np.linspace(X_range1[0], X_range1[1], xn)
    xx0, xx1 = np.meshgrid(x0, x1)
    y = logistic2(xx0, xx1, w)
    cont = plt.contour(xx0, xx1, y, levels=(0.2, 0.5, 0.8),
                       colors=['k', 'cornflowerblue', 'k'])
    cont.clabel(fmt='%1.1f', fontsize=10)
    plt.grid(True)


# test ---
plt.figure(figsize=(3,3))
W=[-1, -1, -1]
show_contour_logistic2(W)
```



#list 6-2-(9) ~ (11)

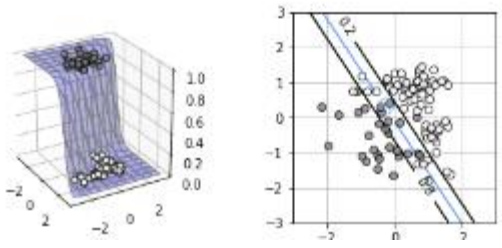
```
def cee_logistic2(w, x, t):
    X_n = x.shape[0]
    y = logistic2(x[:, 0], x[:, 1], w)
    cee = 0
    for n in range(len(y)):
        cee = cee - (t[n, 0] * np.log(y[n]) +
                     (1 - t[n, 0]) * np.log(1 - y[n]))
    cee = cee / X_n
    return cee
```

```
def dcee_logistic2(w, x, t):
    X_n=x.shape[0]
    y = logistic2(x[:, 0], x[:, 1], w)
    dcee = np.zeros(3)
    for n in range(len(y)):
        dcee[0] = dcee[0] + (y[n] - t[n, 0]) * x[n, 0]
        dcee[1] = dcee[1] + (y[n] - t[n, 0]) * x[n, 1]
        dcee[2] = dcee[2] + (y[n] - t[n, 0])
    dcee = dcee / X_n
    return dcee


# test ---
W=[-1, -1, -1]
dcee_logistic2(W, X, T2)
```

```
array([ 0.10272008,  0.04450983, -0.06307245])
```

```
from scipy.optimize import minimize


def fit_logistic2(w_init, x, t):
    res = minimize(cee_logistic2, w_init, args=(x, t),
                   jac=dcee_logistic2, method="CG")
    return res.x


# 메인 ----------------------------------
plt.figure(1, figsize=(7, 3))
plt.subplots_adjust(wspace=0.5)


Ax = plt.subplot(1, 2, 1, projection='3d')
W_init = [-1, 0, 0]
W = fit_logistic2(W_init, X, T2)
print("w0 = {0:.2f}, w1 = {1:.2f}, w2 = {2:.2f}".format(W[0], W[1], W[2]))
show3d_logistic2(Ax, W)


show_data2_3d(Ax, X, T2)
cee = cee_logistic2(W, X, T2)
print("CEE = {0:.2f}".format(cee))


Ax = plt.subplot(1, 2, 2)
show_data2(X, T2)
show_contour_logistic2(W)
plt.show()
```

```
w0 = -3.70, w1 = -2.64, w2 = -0.28
CEE = 0.22
```



#list 6-2-(12) ~ (15)

```
def logistic3(x0, x1, w):
    K = 3
    w = w.reshape((3, 3))
    n = len(x1)
    y = np.zeros((n, K))
    for k in range(K):
        y[:, k] = np.exp(w[k, 0] * x0 + w[k, 1] * x1 + w[k, 2])
    wk = np.sum(y, axis=1)
    wk = y.T / wk
    y = wk.T
    return y

W = np.array([1, 2, 3, 4 ,5, 6, 7, 8, 9])
y = logistic3(X[:3, 0], X[:3, 1], W)
print(np.round(y, 3))
```

```
[[0.    0.006 0.994]
 [0.965 0.033 0.001]
 [0.925 0.07  0.006]]
```

```
def cee_logistic3(w, x, t):
    X_n = x.shape[0]
    y = logistic3(x[:, 0], x[:, 1], w)
    cee = 0
    N, K = y.shape
    for n in range(N):
        for k in range(K):
            cee = cee - (t[n, k] * np.log(y[n, k]))
    cee = cee / X_n
    return cee
W = np.array([1, 2, 3, 4 ,5, 6, 7, 8, 9])
cee_logistic3(W, X, T3)
```

```
3.9824582404787288
```

```
def dcee_logistic3(w, x, t):
    X_n = x.shape[0]
    y = logistic3(x[:, 0], x[:, 1], w)
    dcee = np.zeros((3, 3)) # (클래스의 수 K) x (x의 차원 D+1)
    N, K = y.shape
    for n in range(N):
        for k in range(K):
            dcee[k, :] = dcee[k, :] - (t[n, k] - y[n, k])* np.r_[x[n, :], 1]
    dcee = dcee / X_n
    return dcee.reshape(-1)

W = np.array([1, 2, 3, 4 ,5, 6, 7, 8, 9])
dcee_logistic3(W, X, T3)
```

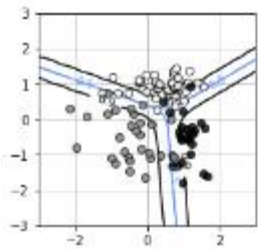
```
array([ 0.03778433,  0.03708109, -0.1841851 , -0.21235188, -0.44408101,
       -0.38340835,  0.17456764,  0.40699992,  0.66759346])
```

#list 6-2-(16) ~ (17)

```
def show_contour_logistic3(w):
    xn = 30
    x0 = np.linspace(X_range0[0], X_range0[1], xn)
    x1 = np.linspace(X_range1[0], X_range1[1], xn)

    xx0, xx1 = np.meshgrid(x0, x1)
    y = np.zeros((xn, xn, 3))
    for i in range(xn):
        wk = logistic3(xx0[:, i], xx1[:, i], w)
        for j in range(3):
            y[:, i, j] = wk[:, j]
    for j in range(3):
        cont = plt.contour(xx0, xx1, y[:, :, j],
                           levels=[0.5, 0.9],
                           colors=['cornflowerblue', 'k'])
        cont.clabel(fmt='%1.1f', fontsize=9)
    plt.grid(True)
```

```
M_init = np.zeros((3, 3))
W = fit_logistic3(M_init, X, T3)
print(np.round(W.reshape((3, 3)),2))
cee = cee_logistic3(W, X, T3)
print("CEE = {0:.2f}".format(cee))


plt.figure(figsize=(3, 3))
show_data2(X, T3)
show_contour_logistic3(W)
plt.show()
```

```
[[-8.2  -2.69  2.25]
 [-0.49  4.8  -0.69]
 [ 8.69 -2.11 -1.56]]
CEE = 0.23
```



< 소감 >

이번 6장에서는 '분류 문제'를 배웠다. 이전 회귀 문제에서는 목표 데이터가 연속된 수치였지만, 여기서는 목표 데이터는 '클래스'라고 보면 된다. 그리고 또 이장에서 중요한 개념인 '확률'개념에 대해서도 새롭게 배웟다. 그것을 이용하여 예측의 '불확실성'을 정량적으로 다루는 법도 배웠다. 인공데이터를 만들고 그를이용해 확률을 설정해 샘플링하는 방법을 배웠다. 여기서 결정 경계 즉 경계선을 그으면 새로운 질량에대해 값을 예상할 수 있는데 경계 결정은 클래스를 0과 1의 값으로 해석하여 데이터의 분포에 직선을 맞추어 주면 경계결정을 할 수 있다. 하지만 이방법은 잘 통하지 않는 경우가 있었는데 직선 데이터에 점이 겹쳐 져 있어서 오차가 발생한다. 확률로 나타내는 클래스를 분류할 때 암컷 수컷 개미를 구분할 때 질량이 x < 0.8g이라면 확실히 그 곤충은 암컷이 되고 1.2 < x 이라면 수컷이라고 할 수 있다. 0.8 ~ 1.2 사이일 때만 둘 다 있으므로 100%예측이 불가능하다는 것을 알았다 그때 쓰이는 jt은 모호성을 확률로 포함한 예측이 가능하다는 것을 알았다. 그리고 조건부확률을 이용하여 수컷일 확률은 1/3이다라는 것을 본다. P(t = 1|x)를 이용한다. 그리고 또한 최대가능도법을 배웠는데, 이 것은 P(t = 1|x) = w 로 설정하여 확률 w에서 t = 1을 생성하는 모델을 설정한다. 그리고 P(T = 0,0,0,1|x) = (1-w)3(제곱)w를 이용하여 그래프를 그리면 위로 솟은 산같은 형태로 나타나게 되고 이산의 최대치를 갖는 w가 가장 적절한 값이다 추정치가 된다. 이것이 바로 최대가능도법이다.
평균엔트로피오차를 계산하는 함수 cee_logistic(w,x,t)를 배웠는데 이는 엔트로피 오차가 어떤 모양인지 확인 할수 있고 보자기의 대각선 모서리를 잡고 들어올리는 모양을 하고 있는 것을 알았다. 그리고 최소치를 구할 수 있다는 것도 알았다. 그리고 경사

하강법에 의한 해를 배웠는데 scipy.optimize라이브러리에 포함된 minimize()함수로 경사하강법을 시도하고 이함수는 학습률을 내부에서 자동으로 설정해주기 때문에 매우 편리안 함수라는 것을 알았다. 그리고 데이터가 2차원인 경우에서도 생각하는 것을 배웠는데, print(X[:5,:])를 통해 2차원 입력데이터를 확인 할 수 있다. 그렇게 T1,T2,T3 의 클래스 데이터의 2차원 데이터를 출력하는 방법은 파이썬과 동일하다는 것을 알았다. pi = np.array([0.4,0.8,1])로 확률을 클래스에 설정하면 그 난 수를 생성하여 wk에 넣고 그것이 pi[0]보다 작으면 클래스 0 pi[1]보다 작으면 클래스 1 pi[2]보다 작으면 2로 하는 방법을 알앗다. 경사하강법에 의한 해 e(w)를 최소화하는 배열 w를 구하려면 e(w)의 각 wki에 관한 편미분이 필요하다는 것도 배웠다. 그리고 각 매개변수에 대한 미분값을 출력하는 함수 dcee_loigcstic3인것도 알았다.그리고 등고선에 결과를 표시하는 함수 show_contour_logistic3도 있는 것을 알았다. 이는 가중치 매개 변수 w를 전달하면ㄴ 표시할 입력공간을 30X30으로 분할하여 모든 입력에 대해 네트워크의 출력을 확인하고 등고선ㄴ으로 표시하는 역할을한다.