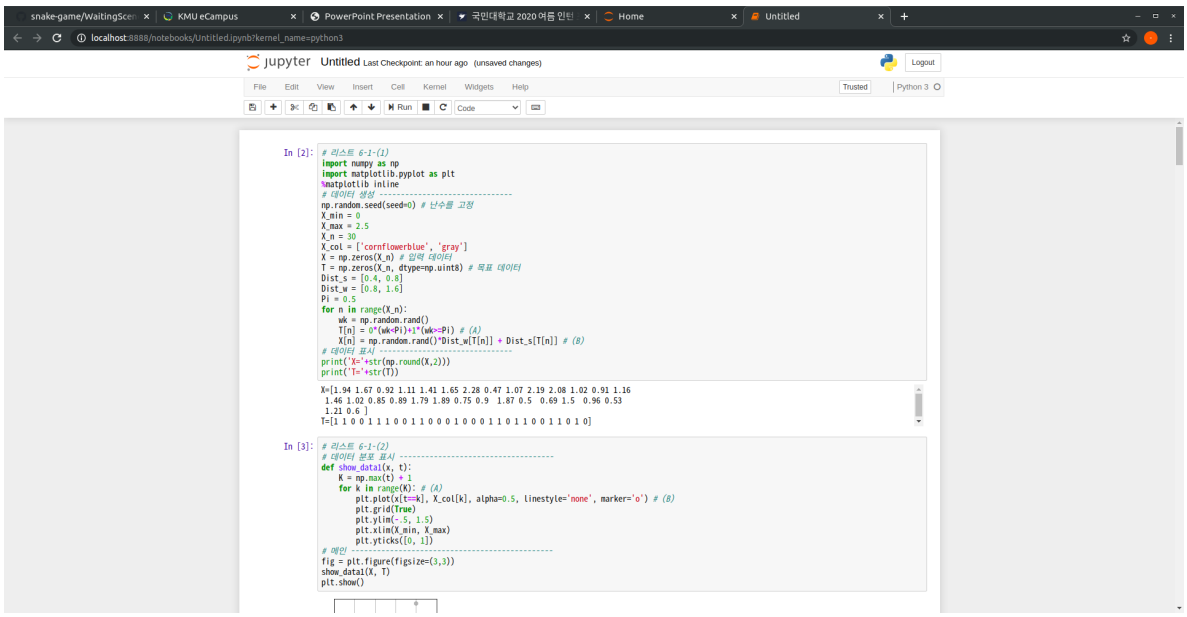
Homework #7

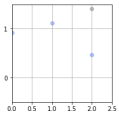
20171621 민대인

1. 목차

내용
1. 목차
2. 스크린샷
3. 간단한 소감



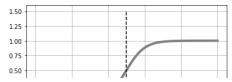
```
fig = plt.figure(figsize=(3,3))
show_data1(X, T)
plt.show()
```

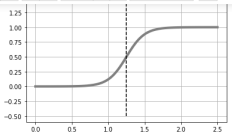


In [4]:
```
# 리스트 6-1-(3)
def logistic(x, w):
    y = 1/(1+np.exp(-(w[0]*x+w[1])))
    return y
```

In [5]:
```
# 리스트 6-1-(4)
def show_logistic(w):
    xb = np.linspace(X_min, X_max, 100)
    y = logistic(xb,w)
    plt.plot(xb, y, color='gray', linewidth=4)
    # 閾値i0 radr0
    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

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

Out[5]: 1.25

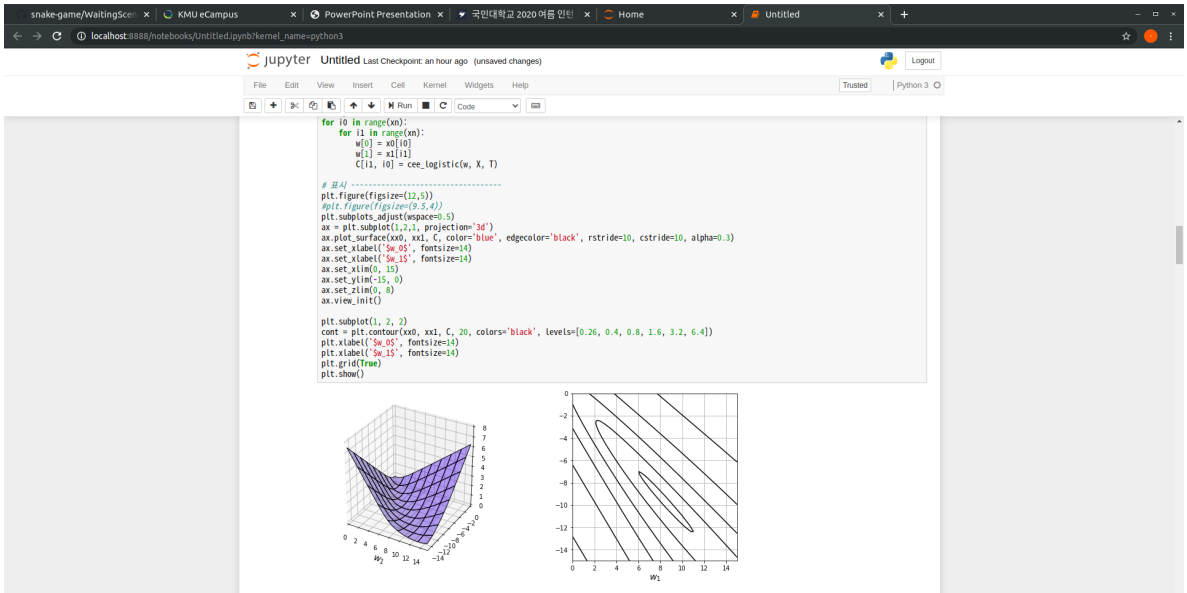




In [6]:
```
# 리스트 6-1-(5)
# 평균 교차 엔트로피 오차 -------------------
def cee_logistic(w, x, t):
    y = logistic(x, w)
    cee = 0
    for n in range(len(y)):
        cee = cee - (t[n] * np.log(y[n]) + (1-t[n]) * np.log(1-y[n]))
    cee = cee / X_n
    return cee

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

Out[6]: 1.0288191541851066

In [8]:
```
# 리스트 6-1-(6)
from mpl_toolkits.mplot3d import Axes3D

# 계산 ---------------------------------
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)
```



```
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)

# 표시 ---------------------------------
plt.figure(figsize=(12,5))
#plt.figure(figsize=(9.5,4))
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=0.3)
ax.set_xlabel('$w_0$', fontsize=14)
ax.set_xlabel('$w_1$', fontsize=14)
ax.set_xlim(0, 15)
ax.set_ylim(-15, 0)
ax.set_zlim(0, 8)
ax.view_init()

plt.subplot(1, 2, 2)
cont = plt.contour(xx0, xx1, C, 20, colors='black', levels=[0.26, 0.4, 0.8, 1.6, 3.2, 6.4])
plt.xlabel('$w_0$', fontsize=14)
plt.xlabel('$w_1$', fontsize=14)
plt.grid(True)
plt.show()
```


In [10]:
```
# 리스트 6-1-(7)
```

snake-game/WaitingScen | KMU eCampus | PowerPoint Presentation | 국민대학교 2020 여름 인턴 | Home | Untitled

localhost:8888/notebooks/Untitled.ipynb?kernel_name=python3

jupyter **Untitled** Last Checkpoint: an hour ago (unsaved changes) — Logout

File Edit View Insert Cell Kernel Widgets Help — Trusted — Python 3

Run Code

In [10]:
```
# 리스트 6-1-(7)
# 평균 교차 엔트로피 오차의 미분 ----------------------
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

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

Out[10]:
```
array([0.30857905, 0.39485474])
```

In [11]:
```
# 리스트 6-1-(8)
from scipy.optimize import minimize

# 매개 변수 검색
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

# 메인 ----------------------------
plt.figure(1, figsize=(3, 3))
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} g".format(B))
plt.show()
```

```
w0 = 8.18, w1 = -9.38
CEE = 1.15 g
```

snake-game/WaitingScen | KMU eCampus | PowerPoint Presentation | 국민대학교 2020 여름 인턴 | Home | Untitled

localhost:8888/notebooks/Untitled.ipynb?kernel_name=python3

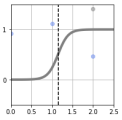
jupyter **Untitled** Last Checkpoint: an hour ago (unsaved changes) — Logout

File Edit View Insert Cell Kernel Widgets Help — Trusted — Python 3

Run Code

```
plt.show()
```

```
w0 = 8.18, w1 = -9.38
CEE = 1.15 g
```



In [12]:
```
# 리스트 6-2-(1)
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline


# 데이터 생성 ----------------------
np.random.seed(seed=1)
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 표시용
X_range1 = [-3, 3] # X1 범위 표시용
Mu = np.array([[-.5, -.5], [.5, 1.0], [1, -.5]]) # 분포의 중심
Sig = np.array([[.7, .7], [.8, .3], [.3, .8]]) # 분포의 분산
Pi = np.array([0.4, 0.8, 1]) # (A) 각 분포에 대한 비율 0.4 0.8 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]
```

In [13]:
```
# 리스트 6-2-(2)
```

snake-game/WaitingScen | KMU eCampus | PowerPoint Presentation | 국민대학교 2020 여름 인턴 | Home | Untitled

localhost:8888/notebooks/Untitled.ipynb?kernel_name=python3

jupyter **Untitled** Last Checkpoint: an hour ago (unsaved changes) — Logout

File Edit View Insert Cell Kernel Widgets Help — Trusted — Python 3

Run Code

```
        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]
```

In [13]:
```
# 리스트 6-2-(2)
print(X[:5, :])
```

```
[[-0.14173827  0.86533666]
 [-0.86972023 -1.25107804]
 [-2.15442802  0.29474174]
 [ 0.75523128  0.92518889]
 [-1.10193462  0.74082534]]
```

In [14]:
```
# 리스트 6-2-(3)
print(T2[:5,:])
```

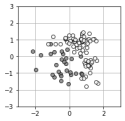
```
[[0 1]
 [1 0]
 [1 0]
 [0 1]
 [1 0]]
```

In [15]:
```
# 리스트 6-2-(4)
print(T3[:5,:])
```

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

In [17]:
```
# 리스트 6-2-(5)
# 데이터 표시 ----------------------
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.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.show()
```

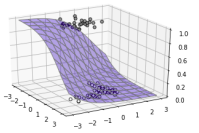


In [24]:
```
# 리스트 6-2-(6)
# 로지스틱 회귀 모델 ------------------
def logistic2(x0, x1, w):
    y = 1 / (1+np.exp(-(w[0] * x0 + w[1] *x1 + w[2])))
    return y
```

In [26]:
```
# 리스트 6-2-(7)
# 모델 3D 보기 ----------------------------
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=0.3)

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=5, a
    Ax.view_init(elev = 25, azim=-30)
```

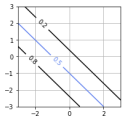

```
    for i in range(2):
        ax.plot(x[t[:,i] == 1, 0], x[t[:,i] == 1, 1], 1-i, marker='o', color=c[i], markeredgecolor='black', linestyle='none', markersize=5, a
    Ax.view_init(elev = 25, azim=-30)
```


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



In [29]:
```
# 리스트 6-2-(8)
# 모델 등고선 2D 표시 ------------------
def show_contour_logistic2(w):
    xn = 30 # 매개 변수의 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)
```


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

In [30]:
```
# 리스트 6-2-(9)
# 크로스 엔트로피 오차 ------------------
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
```

In [33]:
```
# 리스트 6-2-(10)
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)
```

Out[33]:
```
array([ 0.10272008,  0.04450983, -0.06307245])
```

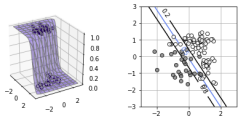
In [34]:
```
# 리스트 6-2-(11)
```

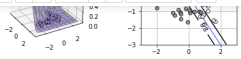
```
dcee_logistic2(W, X, T2)
```

Out[33]:
```
array([ 0.10272008,  0.04450983, -0.06307245])
```

In [34]:
```
# 리스트 6-2-(11)
from scipy.optimize import  minimize

# 로지스틱 회귀 모델의 매개 변수 검색 -----
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(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 = -3.75, w1 = -2.58, w2 = -0.27
CEE = 0.00
```





In [35]:
```
# 리스트 6-2-(12)
# 3클래스용 로지스틱 회귀 모델 --------------------
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

# test ---
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  0.005]]
```

In [36]:
```
# 리스트 6-2-(13)
# 교차 엔트로피 오차 -----------------
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

# test ----
W = np.array([1,2,3,4,5,6,7,8,9])
cee_logistic3(W, X, T3)
```

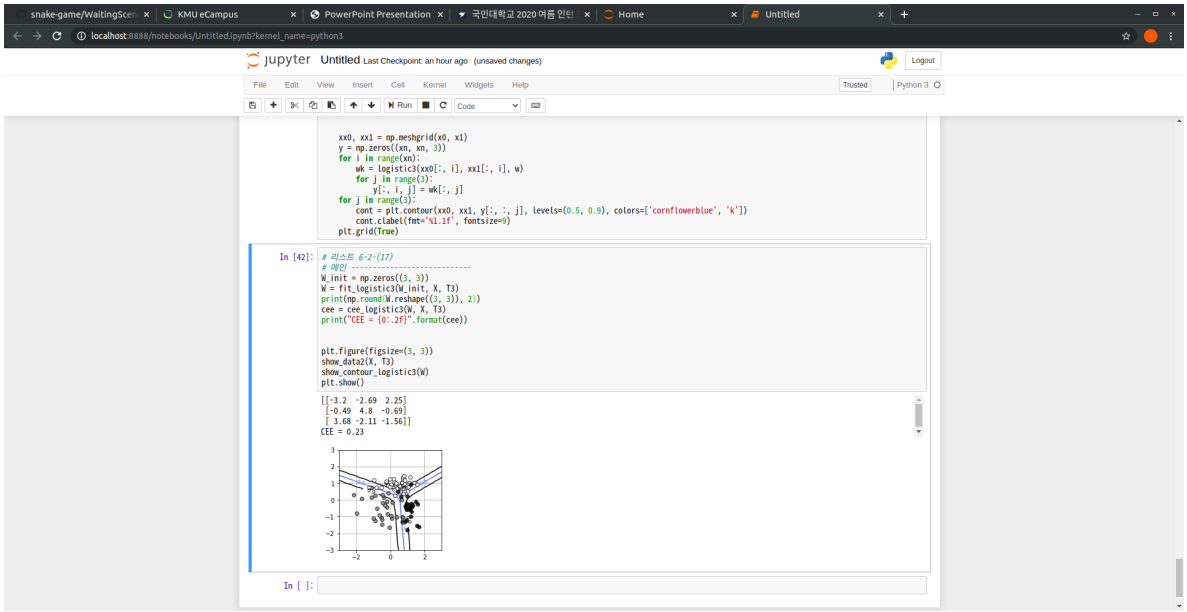
Out[36]:
```
3.9824582404787288
```

In [37]:
```
# 리스트 6-2-(14)
# 교차 엔트로피 오차의 미분 -----------
def dcee_logistic3(w, x, t):
    X_n = x.shape[0]
    y = logistic3(x[:3, 0], x[:3, 1], w)
    dcee = np.zeros((3, 3))
    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)

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

Out[37]:
```
array([ 0.00192633,  0.00021094, -0.00110022, -0.00039233, -0.00881633,
       -0.00890776, -0.001534  ,  0.00860539,  0.01000798])
```

In [38]:
```
# 리스트 6-2-(15)
# 매개 변수 검색 --------------------
def fit_logistic3(w_init, x, t):
    res = minimize(cee_logistic3, w_init, args=(x, t))
    return res.x
```

In [39]:
```
# 리스트 6-2-(16)
# 모델 등고서 2D 표시 ----------------
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)
```



3. 소감

그래프를 그려보면서, 분류 문제에 대해서 공부할 수 있었습니다. 또한, 확률 개념이 도입된 부분을 보면서 예측의 '불확실성'을 정량적으로 다룰 수 있어 좋았습니다. 예전에 고등학교 때 확률 공부했던 부분을 그래프로 표현하면서 가시적인 부분이 보여 신기하고 재밌었습니다.