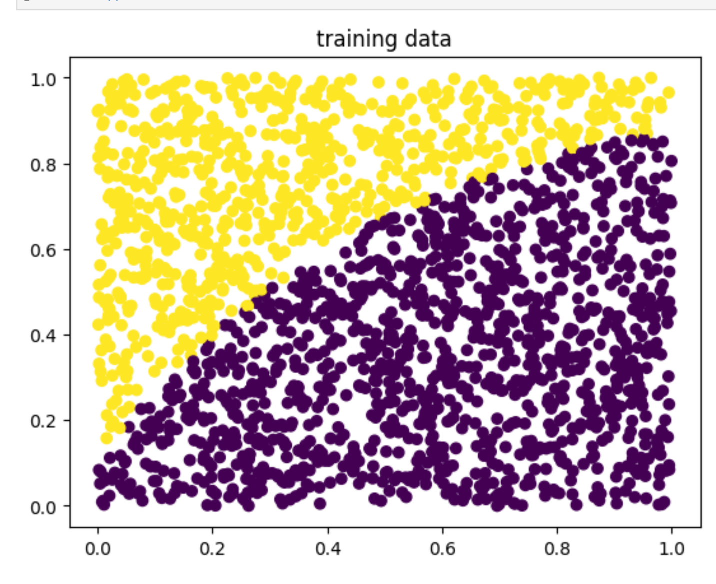
```python
import numpy as np
import matplotlib.pyplot as plt
import pickle

pkl_file = open('classifier_data.pkl', 'rb')
x_train, y_train = pickle.load(pkl_file)

n_train = np.size(y_train)

plt.scatter(x_train[:,0],x_train[:,1], c=y_train[:,0])
plt.title('training data')
plt.show()
```




```python
print(n_train)
```

```
2000
```

2a)

2b)

∇L(w)= ∑{i=1}^n -yi*xi*/(1+e**(yi*xi.T*w)) + 2λw

2c)

```python
## hey! I have a big question here! In this version, grad = top * denominator should be 2X1, but the result always reports bug
def logistic_graddescent_old_version(X, y, tau, w_init, it, lambda_val=1):
    W = np.zeros((w_init.shape[0], it+1))
    W[:, [0]] = w_init
    # w_init = np.zeros((2,1))
    for k in range (it):
        total_grad = np.zeros(w_init.shape)
        for i in range(X.shape[0]):
            denominator = 1/(1 + np.exp(y[i] * (X[[i],:].reshape(1, 2))@ W[:, k]))
            top = (-X[[i],:].T @ y[i])
            grad = top * denominator
            total_grad += grad
        total_grad = total_grad+ 2 * lambda_val * W[:, [k]]
        W[:, [k+1]] = W[:, [k]] - tau * total_grad
    return W
```

```python
## hey! another question here. I find there is a simplified version of the gradient ∇L(w)= (1/n)*X^T*(σ(Xw)-y)+2λw. But I do 
def logistic_graddescent_old_version_another_version(X,y,tau,w_init,it):
    # σ(z)= 1/(1+exp(-z))
    #L(w)= ∑_{i=1}^n log(1+exp(-y_i x_i^T w)) + λ||w||_2^2
    #∇L(w)= (1/n)*X^T*(σ(Xw)-y)+2λw
    W = np.zeros((w_init.shape[0],it))
    W[:,[0]] = w_init
    for k in range(it-1):
        W[:,[k+1]] = W[:,[k]] - tau * ((1/n_train)*X@(1/(1+np.exp(-X.T@W[:,[k]])-y)) + 2*W[:,[k]])
    return W
```

```python
def logistic_graddescent_final_version(X, y, tau, w_init, it, lambda_val=1):
    n, d = X.shape
    W = np.zeros((d, it + 1))
    W[:, 0] = w_init.flatten()

    for k in range(it):
        total_grad = np.zeros(w_init.shape)
        for i in range(n):
            xi = X[i, :].reshape(1, 2)
            yi = y[i]
            z = xi @ W[:, k]
            Part_1= 1 / (1 + np.exp(yi * z))
            grad = (-xi.T*yi * Part_1).reshape(2, 1)
            total_grad += grad

        reg_grad = 2 * lambda_val * W[:, [k]]
        total_grad = total_grad + reg_grad
        W[:, [k+1]] = W[:, [k]] - tau * total_grad

    return W
```

```python
w_init = np.array([[0],[0]])
it = 150
tau = .001
W = logistic_graddescent(x_train,y_train,tau,w_init,it)
W
```

```
array([[ 0.        , -0.2358214 , -0.44242964, -0.62673778, -0.79379238,
        -0.94724884, -1.08976174, -1.22327507, -1.34922957, -1.46870879,
        -1.58254138, -1.69137284, -1.79571576, -1.8959853 , -1.99252427,
        -2.08562096, -2.17552192, -2.26244126, -2.3465673 , -2.42806763,
        -2.50709278, -2.58377905, -2.65825068, -2.73062153, -2.80099643,
        -2.86947229, -2.93613894, -3.00107995, -3.06437316, -3.1260913 ,
        -3.18630241, -3.24507027, -3.30245473, -3.35851204, -3.41329515,
        -3.46685393, -3.51923544, -3.5704841 , -3.62064189, -3.66974856,
        -3.7178417 , -3.76495699, -3.81112824, -3.85638757, -3.90076549,
        -3.94429104, -3.98699182, -4.02889414, -4.07002307, -4.11040253,
        -4.15005534, -4.1890033 , -4.22726725, -4.26486711, -4.30182195,
        -4.33815002, -4.37386883, -4.40899515, -4.44354506, -4.47753401,
        -4.51097682, -4.54388775, -4.57628048, -4.60816821, -4.63956359,
        -4.67047885, -4.70092573, -4.73091556, -4.76045928, -4.78956741,
        -4.81825013, -4.84651724, -4.87437824, -4.90184228, -4.92891821,
        -4.95561459, -4.98193972, -5.00790159, -5.03350798, -5.05876639,
        -5.0836841 , -5.10826816, -5.13252542, -5.15646251, -5.18008585,
        -5.2034017 , -5.2264161 , -5.24913494, -5.27156394, -5.29370865,
        -5.31557447, -5.33716663, -5.35849025, -5.37955027, -5.40035153,
        -5.42089873, -5.44119644, -5.4612491 , -5.48106106, -5.50063654,
        -5.51997965, -5.53909441, -5.55798473, -5.57665442, -5.5951072 ,
        -5.6133467 , -5.63137645, -5.64919992, -5.66682047, -5.68424141,
        -5.70146593, -5.7184972 , -5.73533826, -5.75199213, -5.76846172,
        -5.78474992, -5.8008595 , -5.81679322, -5.83255376, -5.84814372,
        -5.86356568, -5.87882214, -5.89391556, -5.90884834, -5.92362283,
        -5.93824133, -5.95270611, -5.96701936, -5.98118326, -5.99519992,
        -6.00907143, -6.02279981, -6.03638707, -6.04983517, -6.06314601,
        -6.07632149, -6.08936344, -6.10227368, -6.11505399, -6.1277061 ,
        -6.14023172, -6.15263254, -6.1649102 , -6.17706632, -6.18910248,
        -6.20102024, -6.21282113, -6.22450665, -6.23607828, -6.24753748,
        -6.25888566],
       [ 0.        ,  0.07899054,  0.1733846 ,  0.27695953,  0.38538745,
         0.49571432,  0.60595703,  0.71481101,  0.82144295,  0.92534496,
         1.02623241,  1.12397227,  1.21853317,  1.30995058,  1.39830268,
         1.4836937 ,  1.56624252,  1.64607499,  1.72331875,  1.79809987,
         1.8705407 ,  1.94075858,  2.00886517,  2.07496607,  2.13916078,
         2.20154284,  2.26219997,  2.32121443,  2.37866325,  2.4346186 ,
         2.48914811,  2.54231513,  2.5941791 ,  2.64479579,  2.69421757,
         2.74249368,  2.78967043,  2.83579142,  2.88089776,  2.92502821,
         2.96821939,  3.01050588,  3.05192041,  3.09249398,  3.13225593,
         3.17123413,  3.209455  ,  3.24694366,  3.283724  ,  3.31981874,
         3.35524952,  3.39003696,  3.42420075,  3.45775966,  3.49073163,
         3.52313379,  3.55498255,  3.5862936 ,  3.61708197,  3.64736205,
         3.67714766,  3.70645205,  3.73528793,  3.76366753,  3.79160258,
         3.81910437,  3.84618377,  3.87285123,  3.89911682,  3.92499024,
         3.95048084,  3.97559765,  4.00034936,  4.02474438,  4.04879081,
         4.0724965 ,  4.09586902,  4.11891569,  4.1416436 ,  4.16405961,
         4.18617034,  4.20798224,  4.22950151,  4.2507342 ,  4.27168616,
         4.29236305,  4.31277039,  4.33291349,  4.35279756,  4.37242763,
         4.39180857,  4.41094515,  4.42984197,  4.44850352,  4.46693418,
         4.48513817,  4.50311963,  4.52088259,  4.53843095,  4.55576852,
         4.57289901,  4.58982605,  4.60655315,  4.62308374,  4.63942118,
         4.65556873,  4.67152957,  4.68730681,  4.70290348,  4.71832254,
         4.73356688,  4.74863931,  4.7635426 ,  4.77827944,  4.79285245,
         4.80726419,  4.82151719,  4.8356139 ,  4.84955671,  4.86334797,
         4.87698999,  4.890485  ,  4.9038352 ,  4.91704274,  4.93010972,
         4.94303822,  4.95583024,  4.96848776,  4.98101271,  4.99340699,
         5.00567246,  5.01781094,  5.02982421,  5.04171401,  5.05348206,
         5.06513004,  5.0766596 ,  5.08807236,  5.09936988,  5.11055374,
         5.12162546,  5.13258652,  5.14343841,  5.15418255,  5.16482037,
         5.17535326,  5.18578257,  5.19610965,  5.20633581,  5.21646234,
         5.22649051]])
```

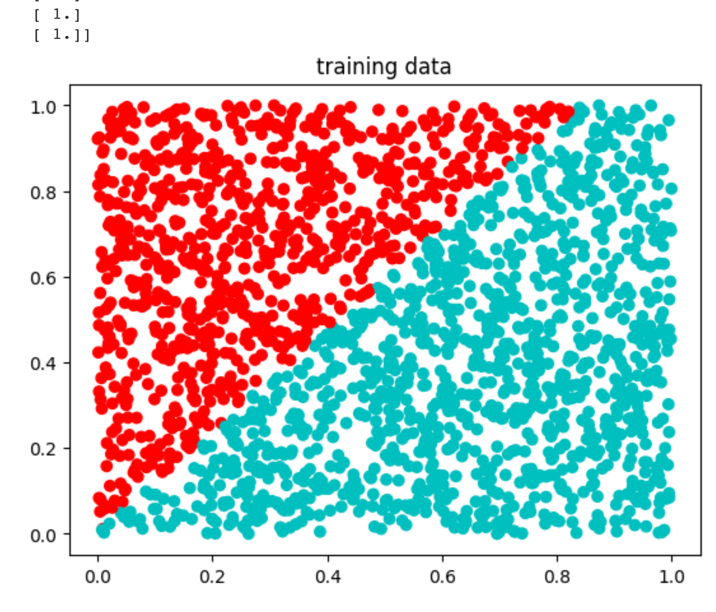
```python
W_op = W[:,-1]
W_op
```

```
array([-6.25888566,  5.22649051])
```

2d)

```python
y_hat = np.sign(x_train@W_op)
y_hat = y_hat.reshape((2000,1))
print(y_hat)
plt.scatter(x_train[:,0],x_train[:,1], color=['c' if i==-1 else 'r' for i in y_hat[:,0]])
plt.title('training data')
plt.show()
```

```
[[ 1.]
 [ 1.]
 [-1.]
 ...
 [-1.]
 [ 1.]
 [ 1.]]
```




```python
error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat, y_train))]
print('Errors: '+ str(sum(error_vec)))
Error_rate = str(sum(error_vec)/n_train)
Error_rate
```

```
Errors: 229
'0.1145'
```

2e)

```python
# I think both of the following versions work
def graddescent(X,y,tau,w_init,it):
    W = np.zeros((w_init.shape[0],it))
    W[:,[0]] = w_init
    for k in range(it-1):
        W[:,[k+1]] = W[:,[k]] - tau * ((X.T @ (X @ W[:,[k]] - y))+2*W[:,[k]])
    return W
```

```python
def graddescent(X,y,tau,w_init,it):
    W = np.zeros((w_init.shape[0], it+1))
    W[:, [0]] = w_init
    Z = np.zeros((w_init.shape[0], it+1))
    for k in range (it):
        Z[:, [k]] = W[:, [k]] - tau * ((X.T @ (X @ W[:, [k]] - y)))
        W[:, [k+1]] = 1/(1+tau) * Z[:, [k]]
    return W
```

```python
w_init = np.array([[0],[0]])
it = 150
tau = .001
W = graddescent(x_train,y_train,tau,w_init,it)
W
```

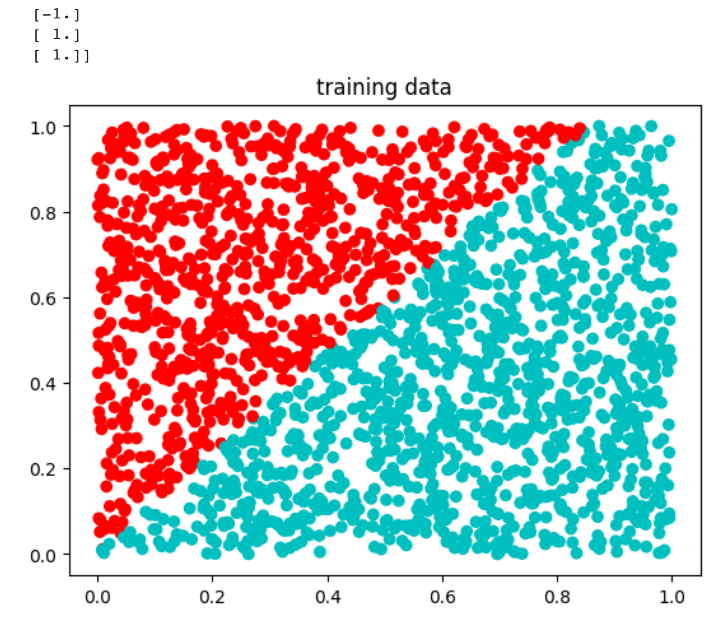
```
array([[ 0.        , -0.47117163, -0.71194428, -0.93287344, -1.11283416,
        -1.26237057, -1.38619638, -1.48879361, -1.57379285, -1.64421387,
        -1.70255678, -1.75089318, -1.79093928, -1.82411699, -1.85160431,
        -1.87437722, -1.89324429, -1.90887543, -1.92182563, -1.93255472,
        -1.94144363, -1.94880799, -1.95490927, -1.95996411, -1.96415197,
        -1.96762157, -1.97049609, -1.97287759, -1.97485064, -1.97648529,
        -1.97783957, -1.97896158, -1.97989114, -1.98066128, -1.98129933,
        -1.98182794, -1.9822659 , -1.98262873, -1.98292934, -1.98317839,
        -1.98338472, -1.98355567, -1.98369729, -1.98381463, -1.98391184,
        -1.98399238, -1.9840591 , -1.98411438, -1.98416018, -1.98419812,
        -1.98422956, -1.9842556 , -1.98427718, -1.98429506, -1.98430987,
        -1.98432214, -1.98433231, -1.98434073, -1.98434771, -1.98435349,
        -1.98435828, -1.98436224, -1.98436553, -1.98436826, -1.98437051,
        -1.98437238, -1.98437393, -1.98437521, -1.98437628, -1.98437716,
        -1.98437789, -1.98437849, -1.98437899, -1.98437941, -1.98437975,
        -1.98438004, -1.98438027, -1.98438047, -1.98438063, -1.98438076,
        -1.98438088, -1.98438097, -1.98438104, -1.98438111, -1.98438116,
        -1.9843812 , -1.98438124, -1.98438127, -1.98438129, -1.98438131,
        -1.98438133, -1.98438134, -1.98438136, -1.98438137, -1.98438137,
        -1.98438138, -1.98438139, -1.98438139, -1.98438139, -1.9843814 ,
        -1.9843814 , -1.9843814 , -1.9843814 , -1.98438141, -1.98438141,
        -1.98438141, -1.98438141, -1.98438141, -1.98438141, -1.98438141,
        -1.98438141, -1.98438141, -1.98438141, -1.98438141, -1.98438141,
        -1.98438141, -1.98438141, -1.98438141, -1.98438141, -1.98438141,
        -1.98438141, -1.98438141, -1.98438141, -1.98438141, -1.98438141,
        -1.98438141, -1.98438141, -1.98438141, -1.98438141, -1.98438141,
        -1.98438141, -1.98438141, -1.98438141, -1.98438141, -1.98438141,
        -1.98438141, -1.98438141, -1.98438141, -1.98438141, -1.98438141,
        -1.98438141, -1.98438141, -1.98438141, -1.98438141, -1.98438141,
        -1.98438141, -1.98438141, -1.98438141, -1.98438141, -1.98438141,
        -1.98438141],
       [ 0.        ,  0.15782327,  0.43993236,  0.65195085,  0.83071828,
         0.97837851,  1.10077721,  1.20217385,  1.28618103,  1.35577976,
         1.41344148,  1.4612135 ,  1.50079203,  1.53358235,  1.56074873,
         1.58325575,  1.60190253,  1.61735116,  1.63015016,  1.64075397,
         1.6495391 ,  1.65681748,  1.66284752,  1.66784333,  1.6719823 ,
         1.67541139,  1.67825235,  1.68060604,  1.68255605,  1.68417161,
         1.68551008,  1.68661899,  1.6875377 ,  1.68829885,  1.68892945,
         1.68945189,  1.68988473,  1.69024333,  1.69054042,  1.69078656,
         1.69099049,  1.69115944,  1.69129941,  1.69141538,  1.69151145,
         1.69159105,  1.69165699,  1.69171163,  1.69175689,  1.69179439,
         1.69182546,  1.6918512 ,  1.69187253,  1.6918902 ,  1.69190484,
         1.69191696,  1.69192701,  1.69193533,  1.69194223,  1.69194794,
         1.69195268,  1.6919566 ,  1.69195985,  1.69196254,  1.69196477,
         1.69196662,  1.69196815,  1.69196942,  1.69197047,  1.69197134,
         1.69197206,  1.69197266,  1.69197315,  1.69197356,  1.6919739 ,
         1.69197418,  1.69197442,  1.69197461,  1.69197477,  1.6919749 ,
         1.69197501,  1.6919751 ,  1.69197518,  1.69197524,  1.69197529,
         1.69197534,  1.69197537,  1.6919754 ,  1.69197543,  1.69197545,
         1.69197546,  1.69197548,  1.69197549,  1.6919755 ,  1.69197551,
         1.69197551,  1.69197552,  1.69197552,  1.69197553,  1.69197553,
         1.69197553,  1.69197553,  1.69197553,  1.69197554,  1.69197554,
         1.69197554,  1.69197554,  1.69197554,  1.69197554,  1.69197554,
         1.69197554,  1.69197554,  1.69197554,  1.69197554,  1.69197554,
         1.69197554,  1.69197554,  1.69197554,  1.69197554,  1.69197554,
         1.69197554,  1.69197554,  1.69197554,  1.69197554,  1.69197554,
         1.69197554,  1.69197554,  1.69197554,  1.69197554,  1.69197554,
         1.69197554,  1.69197554,  1.69197554,  1.69197554,  1.69197554,
         1.69197554,  1.69197554,  1.69197554,  1.69197554,  1.69197554,
         1.69197554,  1.69197554,  1.69197554,  1.69197554,  1.69197554,
         1.69197554]])
```

```python
W_op = W[:,-1]
W_op
```

```
array([-1.98438141,  1.69197554])
```

```python
y_hat = np.sign(x_train@W_op)
y_hat = y_hat.reshape((2000,1))
print(y_hat)
plt.scatter(x_train[:,0],x_train[:,1], color=['c' if i==-1 else 'r' for i in y_hat[:,0]])
plt.title('training data')
plt.show()
```

```
[[ 1.]
 [ 1.]
 [-1.]
 ...
 [-1.]
 [ 1.]
 [ 1.]]
```




```python
error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat, y_train))]
print('Errors: '+ str(sum(error_vec)))
Error_rate = str(sum(error_vec)/n_train)
Error_rate

# compare with a decision boundary when trained with logistic loss, it also works not bad.
```

```
Errors: 228
'0.114'
```

2f

```python
#Add 1000 points to x_train and y_train, with y = -1 and x = [10, 0]T
new_x = np.array([[10, 0]] * 1000)
new_y = np.array([-1] * 1000).reshape(-1, 1)

x_train = np.vstack((x_train, new_x))
y_train = np.concatenate((y_train, new_y))

print("New x_train shape:", x_train.shape)
print("New y_train shape:", y_train.shape)
```

```
New x_train shape: (3000, 2)
New y_train shape: (3000, 1)
```

```python
w_init = np.array([[0],[0]])
it = 150
tau = .001
W = logistic_graddescent(x_train,y_train,tau,w_init,it)
W
```

```
array([[ 0.        , -5.2358214 , -5.02872919, -4.83946155, -4.67086635,
        -4.52500387, -4.40282496, -4.30407426, -4.22742765, -4.17079144,
        -4.13165333, -4.10739238, -4.09550062, -4.09371224, -4.10005937,
        -4.11287917, -4.13079342, -4.1526754 , -4.177613  , -4.20487315,
        -4.23386986, -4.26413664, -4.29530342, -4.32707749, -4.359228  ,
        -4.39157343, -4.42397153, -4.45631128, -4.48850645, -4.5204905 ,
        -4.55221248, -4.58363383, -4.61472579, -4.64546735, -4.67584364,
        -4.7058446 , -4.73546402, -4.76469864, -4.79354758, -4.82201175,
        -4.85009351, -4.87779627, -4.90512429, -4.93208244, -4.95867604,
        -4.98491075, -5.01079243, -5.03632709, -5.06152083, -5.08637975,
        -5.11090995, -5.1351175 , -5.15900837, -5.18258849, -5.20586364,
        -5.22883954, -5.25152178, -5.27391581, -5.296027  , -5.31786056,
        -5.3394216 , -5.3607151 , -5.38174592, -5.40251882, -5.4230384 ,
        -5.44330919, -5.46333558, -5.48312187, -5.50267224, -5.52199078,
        -5.54108147, -5.55994819, -5.57859473, -5.5970248 , -5.61524201,
        -5.63324987, -5.65105183, -5.66865125, -5.6860514 , -5.7032555 ,
        -5.72026667, -5.73708797, -5.75372239, -5.77017285, -5.78644221,
        -5.80253326, -5.81844873, -5.8341913 , -5.84976357, -5.8651681 ,
        -5.88040739, -5.8954839 , -5.91040001, -5.92515808, -5.93976041,
        -5.95420924, -5.96850678, -5.98265519, -5.99665658, -6.01051303,
        -6.02422657, -6.03779919, -6.05123284, -6.06452945, -6.07769088,
        -6.09071898, -6.10361556, -6.11638239, -6.1290212 , -6.1415337 ,
        -6.15392157, -6.16618645, -6.17832995, -6.19035366, -6.20225913,
        -6.2140479 , -6.22572145, -6.23728126, -6.24872879, -6.26006544,
        -6.27129263, -6.28241171, -6.29342405, -6.30433096, -6.31513375,
        -6.3258337 , -6.33643206, -6.34693009, -6.357329  , -6.36762998,
        -6.37783421, -6.38794285, -6.39795704, -6.40787791, -6.41770655,
        -6.42744405, -6.43709148, -6.4466499 , -6.45612033, -6.4655038 ,
        -6.47480131, -6.48401384, -6.49314237, -6.50218786, -6.51115125,
        -6.52003346, -6.52883542, -6.53755802, -6.54620215, -6.55476869,
        -6.56325848],
       [ 0.        ,  0.07899054,  0.51435439,  0.91065302,  1.26517858,
         1.57738293,  1.84872146,  2.0822062 ,  2.28183821,  2.45205633,
         2.59729164,  2.72166642,  2.82883155,  2.92191087,  3.00351396,
         3.0757854 ,  3.14046842,  3.19896996,  3.25242055,  3.30172637,
         3.34761301,  3.39066142,  3.43133709,  3.4700135 ,  3.50699093,
         3.54251131,  3.57677017,  3.60992606,  3.64210799,  3.67342139,
         3.70395285,  3.73377383,  3.76294364,  3.79151184,  3.81952009,
         3.84700368,  3.87399275,  3.9005132 ,  3.92658752,  3.95223539,
         3.97747416,  4.00231927,  4.02678458,  4.05088263,  4.07462483,
         4.09802167,  4.12108282,  4.14381728,  4.16623346,  4.18833927,
         4.21014216,  4.23164918,  4.25286706,  4.27380219,  4.2944607 ,
         4.31484844,  4.33497107,  4.35483399,  4.37444245,  4.39380148,
         4.41291596,  4.43179062,  4.45043001,  4.46883858,  4.48702062,
         4.5049803 ,  4.52272167,  4.54024868,  4.55756514,  4.57467479,
         4.59158125,  4.60828806,  4.62479865,  4.64111637,  4.65724449,
         4.67318619,  4.68894458,  4.70452268,  4.71992346,  4.7351498 ,
         4.7502045 ,  4.76509034,  4.77980998,  4.79436606,  4.80876114,
         4.82299772,  4.83707825,  4.85100514,  4.86478072,  4.87840728,
         4.89188706,  4.90522226,  4.91841502,  4.93146744,  4.94438159,
         4.95715946,  4.96980304,  4.98231425,  4.99469499,  5.0069471 ,
         5.01907241,  5.03107269,  5.04294969,  5.05470512,  5.06634065,
         5.07785792,  5.08925856,  5.10054413,  5.1117162 ,  5.12277628,
         5.13372587,  5.14456642,  5.15529939,  5.16592618,  5.17644817,
         5.18686674,  5.19718321,  5.20739889,  5.21751508,  5.22753305,
         5.23745403,  5.24727924,  5.2570099 ,  5.26664717,  5.27619222,
         5.2856462 ,  5.29501021,  5.30428536,  5.31347273,  5.3225734 ,
         5.3315884 ,  5.34051877,  5.34936553,  5.35812966,  5.36681215,
         5.37541396,  5.38393605,  5.39237935,  5.40074478,  5.40903324,
         5.41724563,  5.42538282,  5.43344567,  5.44143504,  5.44935176,
         5.45719667,  5.46497056,  5.47267424,  5.4803085 ,  5.48787411,
         5.49537184]])
```

```python
W_op = W[:,-1]
W_op
```

```
array([-6.56325848,  5.49537184])
```

```python
y_hat = np.sign(x_train@W_op)
y_hat = y_hat.reshape((3000,1))
print(y_hat)
plt.scatter(x_train[:,0],x_train[:,1], color=['c' if i==-1 else 'r' for i in y_hat[:,0]])
plt.title('training data')
plt.show()
```

```
[[ 1.]
 [ 1.]
 [-1.]
 ...
 [-1.]
 [-1.]
 [-1.]]
```


```python
error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat, y_train))]
print('Errors: '+ str(sum(error_vec)))
Error_rate = str(sum(error_vec)/n_train)
Error_rate
```

```
Errors: 229
'0.1145'
```

```python
w_init = np.array([[0],[0]])
it = 150
tau = .001
W = graddescent(x_train,y_train,tau,w_init,it)
W
```

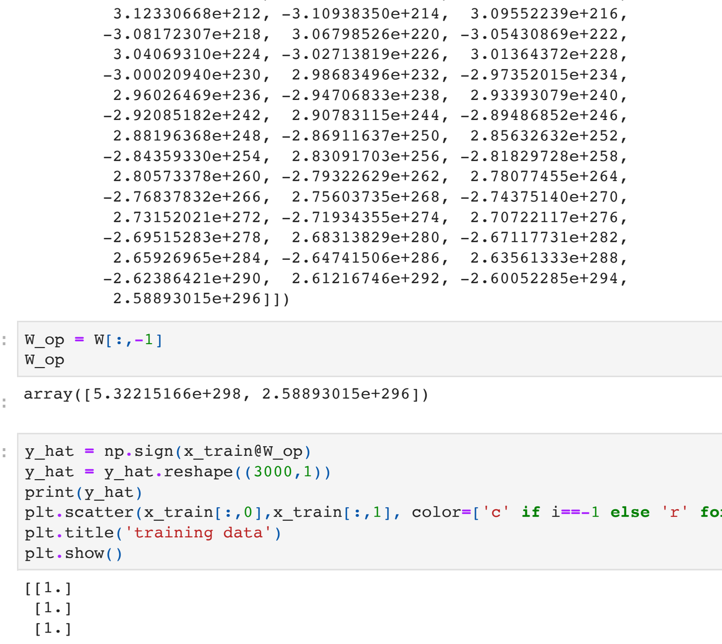
```
array([[ 0.00000000e+000, -1.04611816e+001,  1.03089214e+003,
        -1.02640257e+005,  1.02182598e+007, -1.01727086e+009,
         1.01273603e+011, -1.00822143e+013,  1.00372694e+015,
        -9.99252496e+016,  9.94797995e+018, -9.90363352e+020,
         9.85948477e+022, -9.81553284e+024,  9.77177683e+026,
        -9.72821588e+028,  9.68484912e+030, -9.64167568e+032,
         9.59869469e+034, -9.55590532e+036,  9.51330669e+038,
        -9.47089795e+040,  9.42867827e+042, -9.38664680e+044,
         9.34480269e+046, -9.30314512e+048,  9.26167325e+050,
        -9.22038626e+052,  9.17928331e+054, -9.13836360e+056,
         9.09762630e+058, -9.05707060e+060,  9.01669569e+062,
        -8.97650077e+064,  8.93648503e+066, -8.89664767e+068,
         8.85698790e+070, -8.81750492e+072,  8.77819796e+074,
        -8.73906622e+076,  8.70010892e+078, -8.66132528e+080,
         8.62271454e+082, -8.58427592e+084,  8.54600865e+086,
        -8.50791197e+088,  8.46998512e+090, -8.43222734e+092,
         8.39463788e+094, -8.35721598e+096,  8.31996091e+098,
        -8.28287191e+100,  8.24594825e+102, -8.20918919e+104,
         8.17259399e+106, -8.13616193e+108,  8.09989228e+110,
        -8.06378431e+112,  8.02783731e+114, -7.99205055e+116,
         7.95642332e+118, -7.92095492e+120,  7.88564462e+122,
        -7.85049173e+124,  7.81549555e+126, -7.78065538e+128,
         7.74597051e+130, -7.71144027e+132,  7.67706396e+134,
        -7.64284089e+136,  7.60877038e+138, -7.57485175e+140,
         7.54108432e+142, -7.50746743e+144,  7.47400039e+146,
        -7.44068255e+148,  7.40751322e+150, -7.37449177e+152,
         7.34161751e+154, -7.30888981e+156,  7.27630800e+158,
        -7.24387143e+160,  7.21157946e+162, -7.17943144e+164,
         7.14742674e+166, -7.11556470e+168,  7.08384470e+170,
        -7.05226610e+172,  7.02082828e+174, -6.98953060e+176,
         6.95837244e+178, -6.92735317e+180,  6.89647219e+182,
        -6.86572887e+184,  6.83512259e+186, -6.80465276e+188,
         6.77431875e+190, -6.74411997e+192,  6.71405581e+194,
        -6.68412567e+196,  6.65432895e+198, -6.62466506e+200,
         6.59513341e+202, -6.56573341e+204,  6.53646446e+206,
        -6.50732600e+208,  6.47831742e+210, -6.44943817e+212,
         6.42068765e+214, -6.39206529e+216,  6.36357053e+218,
        -6.33520280e+220,  6.30696152e+222, -6.27884614e+224,
         6.25085609e+226, -6.22299082e+228,  6.19524976e+230,
        -6.16763237e+232,  6.14013810e+234, -6.11276639e+236,
         6.08551670e+238, -6.05838848e+240,  6.03138119e+242,
        -6.00449430e+244,  5.97772727e+246, -5.95107956e+248,
         5.92455064e+250, -5.89813998e+252,  5.87184706e+254,
        -5.84567134e+256,  5.81961232e+258, -5.79366946e+260,
         5.76784224e+262, -5.74213017e+264,  5.71653271e+266,
        -5.69104936e+268,  5.66567961e+270, -5.64042296e+272,
         5.61527889e+274, -5.59024692e+276,  5.56532653e+278,
        -5.54051723e+280,  5.51581853e+282, -5.49122993e+284,
         5.46675095e+286, -5.44238108e+288,  5.41811986e+290,
        -5.39396678e+292,  5.36992138e+294, -5.34598317e+296,
         5.32215166e+298],
       [ 0.00000000e+000,  1.57823266e-001,  5.29422089e+000,
        -4.98984125e+002,  4.97064503e+004, -4.94845543e+006,
         4.92639636e+008, -4.90443531e+010,  4.88257215e+012,
        -4.86080646e+014,  4.83913780e+016, -4.81756573e+018,
         4.79608982e+020, -4.77470965e+022,  4.75342480e+024,
        -4.73223482e+026,  4.71113931e+028, -4.69013784e+030,
         4.66922998e+032, -4.64841534e+034,  4.62769348e+036,
        -4.60706399e+038,  4.58652647e+040, -4.56608050e+042,
         4.54572567e+044, -4.52546159e+046,  4.50528784e+048,
        -4.48520401e+050,  4.46520972e+052, -4.44530456e+054,
         4.42548814e+056, -4.40576005e+058,  4.38611991e+060,
        -4.36656732e+062,  4.34710189e+064, -4.32772323e+066,
         4.30843097e+068, -4.28922470e+070,  4.27010405e+072,
        -4.25106864e+074,  4.23211809e+076, -4.21325201e+078,
         4.19447004e+080, -4.17577179e+082,  4.15715690e+084,
        -4.13862498e+086,  4.12017568e+088, -4.10180863e+090,
         4.08352345e+092, -4.06531978e+094,  4.04719727e+096,
        -4.02915554e+098,  4.01119423e+100, -3.99331300e+102,
         3.97551148e+104, -3.95778931e+106,  3.94014615e+108,
        -3.92258163e+110,  3.90509542e+112, -3.88768715e+114,
         3.87035649e+116, -3.85310309e+118,  3.83592660e+120,
        -3.81882668e+122,  3.80180299e+124, -3.78485518e+126,
         3.76798293e+128, -3.75118589e+130,  3.73446373e+132,
        -3.71781612e+134,  3.70124271e+136, -3.68474319e+138,
         3.66831722e+140, -3.65196447e+142,  3.63568463e+144,
        -3.61947735e+146,  3.60334232e+148, -3.58727923e+150,
         3.57128773e+152, -3.55536753e+154,  3.53951829e+156,
        -3.52373971e+158,  3.50803147e+160, -3.49239325e+162,
         3.47682474e+164, -3.46132564e+166,  3.44589563e+168,
        -3.43053440e+170,  3.41524165e+172, -3.40001707e+174,
         3.38486036e+176, -3.36977122e+178,  3.35474934e+180,
        -3.33979443e+182,  3.32490619e+184, -3.31008431e+186,
         3.29532851e+188, -3.28063848e+190,  3.26601394e+192,
        -3.25145460e+194,  3.23696015e+196, -3.22253033e+198,
         3.20816482e+200, -3.19386336e+202,  3.17962565e+204,
        -3.16545141e+206,  3.15134036e+208, -3.13729221e+210,
         3.12330668e+212, -3.10938350e+214,  3.09552239e+216,
        -3.08172307e+218,  3.06798526e+220, -3.05430869e+222,
         3.04069310e+224, -3.02713819e+226,  3.01364372e+228,
        -3.00020940e+230,  2.98683496e+232, -2.97352015e+234,
         2.96026469e+236, -2.94706833e+238,  2.93393079e+240,
        -2.92085182e+242,  2.90783115e+244, -2.89486852e+246,
         2.88196368e+248, -2.86911637e+250,  2.85632632e+252,
        -2.84359330e+254,  2.83091703e+256, -2.81829728e+258,
         2.80573378e+260, -2.79322629e+262,  2.78077455e+264,
        -2.76837832e+266,  2.75603735e+268, -2.74375140e+270,
         2.73152021e+272, -2.71934355e+274,  2.70722117e+276,
        -2.69515283e+278,  2.68313829e+280, -2.67117731e+282,
         2.65926965e+284, -2.64741506e+286,  2.63561333e+288,
        -2.62386421e+290,  2.61216746e+292, -2.60052285e+294,
         2.58893015e+296]])
```

```python
W_op = W[:,-1]
W_op
```

```
array([5.32215166e+298, 2.58893015e+296])
```

```python
y_hat = np.sign(x_train@W_op)
y_hat = y_hat.reshape((3000,1))
print(y_hat)
plt.scatter(x_train[:,0],x_train[:,1], color=['c' if i==-1 else 'r' for i in y_hat[:,0]])
plt.title('training data')
plt.show()
```

```
[[1.]
 [1.]
 [1.]
 ...
 [1.]
 [1.]
 [1.]]
```




```python
error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat, y_train))]
print('Errors: '+ str(sum(error_vec)))
Error_rate = str(sum(error_vec)/n_train)
Error_rate

# Now the logistic loss classifier works well. But the squared error classifier works terribly.
```

```
Errors: 2230
'1.115'
```