20160040_assignment 11

June 11, 2019

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import copy
from sklearn.metrics import confusion_matrix

#my file data path
file_data = "C:\\Users\\recognize_data\\mnist_train.csv"
handle_file = open(file_data, "r")

#read data with line
data = handle_file.readlines()
handle_file.close()

#image size
size_row = 28    # height of the image
size_col = 28    # width of the image

num_image = len(data)
count = 0     # count for the number of images
```

In [2]:
```
#
# make a matrix each column of which represents an images in a vector form
#
list_image  = np.zeros((num_image, size_row * size_col), dtype=float)
list_label  = np.zeros(num_image, dtype=int)

count = 0
for line in data:
    #the number of lables is at the front. so split and put it into lable value.
    line_data   = line.split(',')
    list_label[count] = line_data[0]
    list_image[count] = np.asfarray(line_data[1:])
    count += 1
```

In [3]:
```
#my file data path
file_data = "C:\\Users\\recognize_data\\mnist_test.csv"
```

```
handle_file = open(file_data, "r")

#read data with line
data = handle_file.readlines()
handle_file.close()

t_num_image = len(data)

t_list_image  = np.zeros((t_num_image, size_row * size_col), dtype=float)
t_list_label  = np.zeros(t_num_image, dtype=int)


count = 0
for line in data:
    #the number of lables is at the front. so split and put it into lable value.
    line_data   = line.split(',')
    t_list_label[count] = line_data[0]
    t_list_image[count] = np.asfarray(line_data[1:])
    count += 1
```

1 Random vector and make matrix

```
In [4]: #random vector
        def random_vector():
            mean = 0
            std = 0.1

            rv = np.random.normal(mean, std, size_row*size_col)

            return rv
        #make matrix
        def make_matrix(num_image, rv, list_image):
            matrix = np.zeros((num_image, size_row * size_col+1), dtype=float)
            for i in range(num_image):
                for j in range(size_row * size_col+1):
                    if(j == 0):
                        matrix[i,j] = 1
                    else:
                        temp = list_image[i,j-1]*rv[j-1]

                        #g_k = max( inner production( r_k, x ), 0 )
                        if(temp<0):
                            matrix[i,j] = 0
                        else:
                            matrix[i,j] = temp
            return matrix
```

```
In [5]: # assign y value
        def assign_y_value(index):
            y = np.zeros((num_image,1), dtype=float)
            count = 0
            for i in list_label:
                if(i == index):
                    y[count] = 1
                else:
                    y[count] = -1
                count += 1
            return y
```

2 Compute an optimal model parameter using the training dataset(seta)

```
In [6]: def make_seta(y, matrix):
            seta = np.zeros((size_row * size_col+1, 1), dtype=float)
            values = copy.deepcopy(np.linalg.pinv((np.mat(matrix.T)*np.mat(matrix)))*np.mat(mat
            seta = np.ravel(values)
            seta = np.reshape(np.array(seta),(size_row*size_col +1,1))

            return seta

        def estimation(seta, matrix, num_image):

            estimation = np.zeros((num_image,1), dtype=float)
            for i in range(num_image):
                for j in range(size_row*size_col +1):
                    estimation[i] += matrix[i,j]*seta[j]
            return estimation

In [7]: rv = copy.deepcopy(random_vector())
        matrix = copy.deepcopy(make_matrix(num_image, rv, list_image))
```

3 Result of Random vector(mean:0, std:0.1)

```
In [21]: print(rv)
```

```
[ 2.51946164e-02  9.41490768e-02  2.63030483e-02 -7.72868626e-02
 -2.63984918e-01 -3.12818974e-02 -6.04858018e-02  8.39090170e-02
 -1.76170165e-01 -2.33774324e-01  7.96356214e-02  1.06351817e-01
  1.53234184e-01 -6.61126267e-02  1.64575348e-01 -7.40240414e-02
  4.32008889e-03  1.92516142e-02  5.19154493e-02 -7.70465091e-02
  8.53547148e-02  8.81247068e-03 -2.38463843e-02 -2.01682580e-02
 -1.89570031e-02  1.92516978e-01  1.45026004e-02  1.93408006e-01
  1.31892827e-01 -9.26730853e-02  5.57417025e-02  5.42220612e-02
 -1.73915498e-02 -6.34038924e-02 -1.41813118e-01 -2.23597004e-01
```

```
 2.85951334e-02 -1.68829692e-01  1.50940266e-01 -1.59469274e-01
-4.56179651e-02  1.68436730e-01 -9.78453939e-02  8.39339170e-02
-9.38326888e-02  1.53130252e-01  4.79184679e-02 -1.36860413e-01
 7.48410814e-02 -3.27094538e-02  3.01113866e-02  2.01877088e-02
-2.45613867e-01  1.19384400e-01 -2.82916893e-03 -3.37881437e-02
 3.33423714e-02 -1.82628970e-01 -2.22621764e-02 -9.09944656e-02
 4.93607447e-02  3.83052465e-02  1.32129931e-01 -3.76046589e-02
 1.79950925e-02  7.73579635e-02 -5.71548885e-02  1.69498077e-01
-3.24426856e-02 -1.01544385e-01  3.82006347e-02  1.27990990e-02
 1.24750891e-01  1.53511517e-01 -1.64902459e-01 -1.00578995e-01
 8.44723184e-02 -1.14887344e-01  3.60283903e-02  2.28080953e-02
 5.85701134e-02  9.26925521e-02 -2.12508903e-01  6.58207962e-03
 6.83981255e-02  1.73840626e-01  1.37228983e-01  1.97092695e-02
 2.15839103e-01  1.12178828e-01 -6.17000553e-02  2.88986043e-02
 9.24679294e-02  1.19994677e-01  8.34315443e-02 -2.22254635e-01
 5.09215245e-02  2.80096970e-02  1.15965028e-02 -3.83880275e-02
-1.71599434e-01 -3.11910947e-02 -8.99891434e-03 -9.59496189e-02
 4.51264475e-04 -1.03609014e-01 -7.98592212e-02 -4.89799511e-02
 5.78784539e-02  1.39834408e-01 -1.16232028e-01  3.68120461e-02
-5.88287299e-03  6.39982611e-02  1.80382419e-01  6.57474598e-03
 2.43035026e-02 -8.82645976e-02  4.69168364e-02  5.85726181e-02
 6.60156351e-02  6.87511171e-02 -3.76705349e-02 -1.02577176e-01
 4.06755658e-02 -3.93355286e-02 -2.08648289e-01 -4.06794811e-02
-8.94643307e-02 -6.14368567e-02  1.85762081e-02 -4.21936966e-02
 1.97971966e-02 -5.67471855e-02 -9.02265504e-02 -3.52005794e-02
-9.48759933e-02 -2.04035289e-01  1.27072507e-01  9.92309039e-03
-1.52828329e-02 -2.96350612e-01 -1.13525213e-01  1.76332283e-01
-5.20830012e-03 -8.53909136e-02 -1.35390950e-01 -1.25563339e-02
-2.22100757e-01  3.55571248e-02  4.83223008e-02  2.80591974e-02
 1.74126221e-01 -5.06195974e-02  8.26913474e-02  3.71811731e-02
-1.22691014e-01  1.22767274e-01  2.59451235e-02  2.16361292e-02
 7.55267569e-02  6.50291160e-02  1.38230979e-01 -3.93836076e-02
 1.21177976e-01 -1.81508221e-02  6.27545659e-02 -5.47890286e-02
-1.67145766e-01 -1.86084000e-01 -3.48662629e-02 -4.12968307e-02
-6.05460090e-02  3.85299120e-02 -1.44485334e-01 -3.81528368e-02
-1.66163791e-01  1.17411742e-01 -1.78621381e-01 -8.01152057e-02
-3.45600739e-02 -4.65771974e-02  8.34572746e-02  5.39690957e-02
-6.40941303e-02  9.02625056e-02  2.64141349e-02  9.48292863e-02
-9.37925412e-02 -1.27131518e-02 -1.76666864e-01  3.48712759e-02
 1.69630897e-01 -1.62268332e-02 -7.97872243e-02  2.50951353e-01
 4.81662249e-02 -1.12966504e-01 -1.43368886e-01 -2.23446456e-02
-2.84604399e-03 -3.74538645e-02 -6.33701110e-02 -1.14265106e-01
 1.11860468e-01  2.35783621e-01 -9.83341892e-02 -4.55098727e-02
-4.26674576e-02 -8.37855569e-02 -1.34837184e-01 -2.21977927e-02
 3.80792286e-02  1.91942745e-02 -6.81493369e-02  1.26519400e-01
 3.81508703e-02  4.59176739e-02  1.24144970e-01 -4.26045925e-02
-9.69546604e-02  2.10929390e-01  9.46670069e-03  3.54792563e-02
-1.37784504e-01  6.04218656e-02  7.64636614e-02  1.21104355e-01
```

```
 6.16433062e-02 -1.15117916e-01 -4.70929468e-02 -6.78145990e-02
 4.09358359e-02  1.14977393e-01 -1.41375823e-02 -1.49866802e-01
-1.46164469e-01  1.83296709e-02  5.65782062e-02 -1.01924197e-01
 2.29175537e-02 -1.04769079e-01  8.65158311e-02  7.33815481e-02
 1.05465849e-01  7.69686213e-03 -7.02713779e-02 -1.21355205e-01
 1.32236922e-01  1.14441045e-01 -9.68186121e-03 -1.08064891e-01
 2.95516312e-02 -2.47374725e-02  8.34434907e-02 -5.74969373e-02
 5.25306090e-02  2.74277056e-01 -3.18847741e-02  2.29952830e-02
 3.39827827e-02 -3.14129966e-02 -5.48299887e-02  1.84585325e-02
 5.82127316e-02 -7.53181734e-02  4.60849694e-02 -1.69698302e-01
 5.81956084e-02 -9.24563621e-02  4.32706275e-03  6.43808347e-02
-1.90932914e-01 -4.67411113e-02  3.32843655e-02  1.08964441e-01
-3.66466954e-02  2.64667021e-02 -1.26886720e-02 -1.73406245e-01
-5.50402745e-02  2.44547840e-01 -1.18177849e-01 -8.09303135e-02
-2.27748220e-01 -1.47260214e-01  1.47914406e-01 -1.27758191e-01
-5.05275780e-02 -1.29282089e-02 -5.73709547e-02  9.55398199e-02
-6.71552649e-02  9.26151704e-02 -5.16462222e-02 -6.09267994e-02
 3.53053751e-02  1.31264912e-02 -6.37170680e-02  1.60999261e-01
-2.07552049e-02 -6.19738524e-02 -4.99288372e-02 -6.60877188e-02
-1.78320799e-02 -3.98910831e-02 -1.47681127e-01  1.92537596e-01
-2.67058421e-02  1.20705784e-02  4.11947678e-02 -7.09753494e-02
-1.07066543e-01  1.44587046e-01 -5.84518780e-02  1.05154920e-01
 7.22923854e-02 -9.76734970e-02 -1.79432622e-02  1.61360247e-02
 9.77494712e-02 -1.19664293e-01  1.54147178e-01  1.43547194e-01
-7.91501209e-02 -9.84768055e-02  6.94795475e-02  6.93899738e-02
 3.29901711e-02 -2.83926042e-01 -3.31511571e-02  8.55717937e-02
-3.95335971e-02 -4.77299991e-03  1.06399513e-01 -2.91725594e-02
 9.75145760e-02  1.40588580e-01  2.38971984e-01 -7.56755592e-02
 3.66533956e-02 -1.46261936e-01 -4.42536235e-02 -9.92401084e-02
 1.32967710e-02 -3.80764239e-03 -5.30524244e-02  6.43310968e-02
-1.34362775e-01  1.98818239e-02 -5.27135108e-02  2.01180432e-02
 2.95649786e-02 -1.45369018e-03 -1.25992130e-01 -8.51419428e-02
 1.52096469e-01  8.60808834e-02  1.10587607e-01 -2.51354662e-01
 2.32453015e-02  6.04359727e-02  1.37915728e-01 -2.56651370e-02
-8.20860879e-02  1.19118449e-01  1.42745462e-01 -1.07067221e-01
-5.37648119e-02 -8.42269922e-02  7.19877640e-02  6.40360506e-03
-2.08124401e-01  6.03473771e-03 -6.25627882e-02 -8.51360366e-02
-4.74728205e-02 -1.39065445e-01  1.12010409e-03  1.21142613e-01
-1.32917750e-01 -5.18543325e-02  2.45325025e-02 -2.07915938e-02
 2.83883693e-02  5.08685055e-03  1.85455297e-01  3.17284887e-02
 1.14496488e-02 -1.81974951e-01  8.68568430e-02 -5.79431289e-02
 2.84989286e-02 -2.01042690e-01 -1.90747848e-02 -7.51350079e-02
 1.04600588e-01 -6.29132986e-03 -1.35052518e-02 -1.50891175e-01
 1.49617748e-01  3.15887666e-02  1.31034712e-01 -2.95514944e-02
-1.89825537e-03 -1.31235592e-01  8.44209940e-02  2.49093656e-02
 3.38016530e-02 -7.34312441e-02 -1.13714296e-01  7.67187065e-02
 6.43727873e-02  1.20162963e-02 -9.51057641e-02  1.57109339e-01
 2.04488390e-01  1.03741845e-01  5.63756191e-02 -4.33320938e-02
```

```
 6.44793376e-02 -4.45035118e-02  8.96400491e-02  5.62696275e-02
 6.78464243e-02  1.62711317e-02 -4.73800756e-02 -1.70706773e-02
 5.34904724e-02  1.13377019e-01  2.47235577e-02 -1.09364547e-01
 1.14270525e-01  1.08497140e-01 -1.51790677e-01  4.04118141e-02
-1.79731334e-02  2.43928570e-02  1.75254834e-02 -8.78142896e-02
 7.49078480e-02  3.69871120e-02  1.28083497e-01 -1.14828533e-01
 4.48879030e-02  1.35253107e-01  1.20630177e-01 -4.64524400e-02
-1.10590476e-01  6.54821528e-02 -4.80461228e-02  1.19386146e-01
-5.04504142e-03 -1.04154951e-01  8.74859303e-02 -6.45643393e-02
-3.27691447e-02  1.82458702e-02  5.24775958e-02  3.65509248e-02
 2.37912719e-01 -1.68129071e-01 -1.70011607e-03  3.86627286e-02
-1.87300445e-02  6.79552789e-03  3.68621745e-02 -2.32169511e-01
-8.53354278e-02  5.00013037e-02  8.70045335e-02 -1.31766211e-02
-4.49803756e-02 -8.65354076e-03 -6.57762240e-03  4.11305697e-02
-7.13730264e-02  2.19099383e-02  6.83546944e-02 -3.45060793e-02
-1.23696308e-01 -1.34848994e-01  2.00560460e-01  5.52957255e-02
-1.20699712e-02 -3.11818117e-02 -7.94528596e-03  2.10069897e-01
-2.22620384e-02  1.45801010e-02 -9.88781545e-02 -6.19422935e-02
 1.43765681e-01 -2.94571487e-02  3.99096243e-02  1.66763743e-01
 1.59923772e-01  7.07330695e-02 -1.42560594e-01 -1.06821642e-01
-1.40036939e-01  1.83731387e-01 -5.30510787e-02 -1.61125323e-01
-1.24709632e-01  7.81427637e-02  1.45812804e-01 -1.85623992e-01
-7.31074473e-02  3.22695061e-02  6.24754068e-02 -6.29461966e-02
-1.47825874e-01 -5.73116437e-02  8.17091710e-02  7.93577886e-02
-2.45106841e-02  7.08793295e-02 -1.03503497e-02  5.80650656e-02
-4.88571659e-02  1.12841119e-01 -1.01176812e-01  1.97188717e-01
 1.11557934e-01 -6.77708270e-03  2.36170298e-02 -1.25224307e-02
 1.45158996e-01  9.12829968e-02 -2.60216181e-02  3.17122384e-02
 1.18570854e-01 -6.97130848e-02 -5.49451817e-02 -6.61749911e-02
 2.77823755e-02  1.50356668e-02 -6.57901012e-02 -9.84622659e-02
-1.34175054e-01  4.89821406e-02 -1.65923536e-01  4.55398454e-02
 3.21202711e-02  2.10650807e-01  4.11464445e-02 -1.78068951e-01
-1.00928990e-02  1.08831160e-01 -1.40746438e-02  1.95268397e-02
-1.57017545e-02  1.26330648e-01 -8.45819405e-02  7.85190305e-02
 5.75589744e-02  3.04225302e-02  1.02443877e-01 -1.17580235e-01
 8.21583866e-03 -1.67038140e-01 -1.24094092e-01  1.65970489e-01
-8.98022142e-02  1.70256642e-02  3.29631261e-02 -7.74111313e-02
 1.20564639e-02 -8.11148003e-02 -1.70612269e-01 -3.83997876e-02
 1.39677000e-01 -3.94367484e-02  1.07932449e-01 -1.94541995e-02
-1.76718360e-02 -1.15722768e-01 -8.00258488e-02 -3.95909399e-02
-4.50947827e-02  1.16323291e-02  1.22597163e-01 -1.61186266e-01
-1.78343911e-02  1.20614641e-01  1.47546723e-02  3.09930085e-02
-3.40520953e-02  5.39344964e-02  1.57340771e-01 -3.26099691e-02
 9.10448119e-02 -1.84993522e-01  1.70703914e-01 -2.44518951e-03
 6.87802699e-02 -1.30394181e-01 -2.92532466e-02  6.04028155e-02
-6.94563825e-02 -1.87870607e-03 -8.24228303e-02 -3.99022492e-02
-1.81543583e-02 -6.52208158e-02 -3.78668354e-02 -3.15848227e-02
-4.20966067e-02 -1.15795674e-01  4.86175798e-03  7.97837758e-02
```

```
 8.35001235e-02 -1.30272695e-01 -1.94701337e-01  6.16875542e-02
-7.33936092e-02  7.81324825e-02 -7.83647149e-02  7.82391336e-03
-2.10586072e-01 -1.52903029e-02 -8.39737671e-02  2.42200798e-02
-1.34966900e-01 -1.79173656e-01 -1.57647530e-02  8.28976792e-02
 1.72744484e-03 -4.28930218e-02 -1.23980147e-01  1.05998884e-02
-1.58486414e-02 -8.30062904e-02 -9.66896835e-02 -2.26750464e-01
-6.49684641e-02 -4.87906309e-02 -9.89116684e-02  1.45439987e-01
 3.61536241e-02  1.82098668e-01 -1.25250334e-01  8.90782497e-03
-1.26142728e-01 -6.07296891e-03  5.60138205e-02 -4.21780679e-02
-8.94322573e-02 -6.65074099e-02 -7.79360708e-02  1.10662060e-02
-1.02075775e-01 -1.21191458e-02 -6.32905957e-02  4.77791717e-02
-3.36354811e-02 -3.36518296e-02  6.77493572e-02  1.46616220e-01
 9.28803899e-02 -7.25487852e-02  8.96035088e-02  1.34262573e-01
-2.95418801e-02 -1.95440085e-01 -1.81058102e-01  8.11890475e-02
 4.50884145e-02 -3.80306408e-02  3.74288199e-02  1.83039072e-01
-8.72023877e-02  1.02793472e-01 -7.16764816e-02  3.16930571e-02
-1.35319422e-01 -4.35159762e-02 -6.48969951e-02 -1.06619530e-02
-2.69373940e-02  1.41366021e-01  2.99750365e-02 -3.45003417e-01
-6.87032210e-02  9.33733479e-03 -5.55867207e-02  3.17257318e-02
-1.53389583e-01  6.69946502e-02  8.69554613e-02  2.58476888e-02
-2.57489973e-02 -3.45386629e-02 -5.53662303e-02  1.03528723e-01
-8.03345883e-02 -9.51800378e-02 -1.10171034e-01  3.55630684e-02
-4.15639066e-02  5.48496950e-02  1.56116603e-02  8.97927936e-02
 2.41546892e-02 -2.61954382e-02  2.95187042e-02  3.62971472e-03
-9.79505183e-05  1.60557168e-02 -1.92755393e-02  1.11648105e-01
-1.03252118e-01 -1.36601845e-01 -1.08936665e-02  6.46108947e-02
 1.33350913e-01  3.80428344e-02 -5.85043006e-02  9.02957867e-02
-1.53741049e-01  5.68965252e-03 -1.26976260e-02  3.82548959e-02
-2.05754976e-01 -1.08440792e-01  6.79635404e-02  9.21504172e-02
 4.96075886e-02 -8.89718153e-02  1.05639797e-02 -2.03495865e-01
 1.11545115e-01 -1.76400916e-01  1.13268131e-01  4.71198574e-02
-2.07192930e-02  8.75856575e-02  4.50659397e-02  7.80860688e-03
 1.72765490e-01 -7.02154811e-02  1.99818283e-02 -6.48869205e-02
 2.66394024e-02 -3.64408387e-02  7.01665586e-02 -1.57539049e-01
 1.44739180e-01 -4.91598620e-02 -1.47500179e-01 -3.94737890e-02
-5.46105243e-02 -2.78653083e-03 -3.23926632e-02 -1.67678850e-02
 2.51428835e-02 -1.69809944e-01 -9.70064981e-02  8.34819415e-02
-2.83958186e-01  1.65234517e-02 -9.43193818e-02 -1.14409628e-01
 1.00304381e-02  1.06051749e-01 -1.28871681e-01 -3.96695446e-02
 1.26314693e-01  8.17767609e-04 -5.21398400e-02  1.19900772e-01
 5.00277179e-02 -1.14257623e-01 -1.85512871e-01  1.07884397e-03
 1.74550424e-02  3.34071117e-04 -1.53675156e-01  3.34096387e-02
-4.62821376e-02 -2.18337142e-02 -5.13552078e-02 -5.43110760e-02]
```

```
In [8]: y_values = np.zeros((10, num_image,1), dtype=float)
        setas = np.zeros((10, size_row * size_col+1, 1), dtype=float)
        for i in range(10):
```

```
            y_values[i] = copy.deepcopy(assign_y_value(i))
            setas[i] = copy.deepcopy(make_seta(y_values[i], matrix))
```

```
In [9]: estimations = np.zeros((10, num_image,1), dtype=float)
        for i in range(10):
            estimations[i] = copy.deepcopy(estimation(setas[i], matrix, num_image))
```

```
In [10]: esti_label = np.zeros((num_image,1), dtype=int)
         for i in range(num_image):
             esti_label[i] = np.argmax(estimations[:,i])
```

4 Compute TPR, ERR- Traning dataset

```
In [11]: c_matrix = np.zeros((10,10), dtype = int)
         c_matrix = copy.deepcopy(confusion_matrix(list_label,esti_label))
         print(c_matrix)
```

```
[[5678   11   19   16   28   33   83    3   48    4]
 [   2 6566   33   11   13   23   16   19   53    6]
 [ 104  303 4717  151  118   11  224  101  210   19]
 [  51  194  183 5095   34  120   59  122  151  122]
 [  12  111   45    4 5145   63   53   31   52  326]
 [ 182  126   34  511  128 3836  195   51  222  136]
 [ 124  103   64    2   67   82 5443    1   29    3]
 [  57  206   47   40  190    8    6 5380   16  315]
 [  76  560   76  257  122  232   55   33 4272  168]
 [  74   77   22  115  397   17    8  546   36 4657]]
```

```
In [12]: tpr_cnt = 0;
         err_cnt = 0;
         for i in range(10):
             for j in range(10):
                 if(i==j):
                     tpr_cnt += c_matrix[i][j]

                 else:
                     err_cnt += c_matrix[i][j]


         tpr_rate = tpr_cnt/num_image
         err_rate = err_cnt/num_image
```

5 Result

```
In [13]: print("true positive rate:", tpr_rate)
         print("error rate:", err_rate)
```

```
true positive rate: 0.8464833333333334
error rate: 0.15351666666666666
```

6 test

```
In [15]: t_matrix = copy.deepcopy(make_matrix(t_num_image, rv, t_list_image))
         t_estimations = np.zeros((10, t_num_image,1), dtype=float)
         for i in range(10):
             t_estimations[i] = copy.deepcopy(estimation(setas[i], t_matrix, t_num_image))

In [16]: t_esti_label = np.zeros((t_num_image,1), dtype=int)

         for i in range(t_num_image):
             t_esti_label[i] = np.argmax(t_estimations[:,i])
```

7 Compute TPR, ERR - Testing dataset

```
In [17]: t_c_matrix = copy.deepcopy(confusion_matrix(t_list_label,t_esti_label))

In [18]: t_tpr_cnt = 0;
         t_err_cnt = 0;
         for i in range(10):
             for j in range(10):
                 if(i==j):
                     t_tpr_cnt += t_c_matrix[i][j]

                 else:
                     t_err_cnt += t_c_matrix[i][j]


         t_tpr_rate = t_tpr_cnt/t_num_image
         t_err_rate = t_err_cnt/t_num_image

In [19]: print(t_c_matrix)
```

```
[[ 949    0    2    1    2    5   14    1    6    0]
 [   0 1110    2    2    2    1    4    1   13    0]
 [  19   69  792   31   21    1   29   21   44    5]
 [   5   20   23  882    3   12    8   22   22   13]
 [   1   20    7    1  881    6   12    2    7   45]
 [  15   16    6   92   31  629   28   17   41   17]
 [  30   14   10    0   21   14  869    0    0    0]
 [   5   44   13    7   29    1    1  884    3   41]
 [  11   57    8   41   26   39   19   11  743   19]
 [  20   15    4   16   85    2    1   75    9  782]]
```

8 Result

```
In [20]: print("true positive rate:", t_tpr_rate)
         print("error rate:", t_err_rate)
```

```
true positive rate: 0.8521
error rate: 0.1479
```