Mustererkennung / Machine Learning   Josephine Mertens
3. Aufgabenzettel   Leonardo Balestrieri

Ziffernpaar	3&5	3&7	3&8	5&7	5&8	7&8
Klassifikationsgüte	**0.950920245399**	**0.993610223642**	**0.954819277108**	**0.990228013029**	**0.953987730061**	**0.977635782748**

Die Testdaten wurden wie in der Vorlesung besprochen mit Hilfe der Dichtewerten der zugehörigen multivariaten Normalverteilungen klassifiziert.

Codeauszug:Berechnung der Normalverteilungen

```
def computeMean(A):

    mean = np.asarray(A.mean(0)).reshape(-1)

    return mean


def computeCovarianceMatrix(A):

    CovarianceMatrix = np.cov(A,rowvar=0)
    n = len(CovarianceMatrix.T*CovarianceMatrix)
    eps = 0.0001
    i = 0
    while np.linalg.det(CovarianceMatrix)==0:
        i=i+1
        CovarianceMatrix =CovarianceMatrix +eps*2**i*np.identity(n)

    return  CovarianceMatrix

def computeNormalDistribution(data_mean, data_cov):
    normal = multivariate_normal(mean=data_mean, cov = data_cov)
    return normal
```

Codeauszug:Berechnung der Klassenzugehörigkeit und Klassifikationsgüte

```
def classifier(train_Set1, train_Set2, test_Set1, test_Set2):

    data_matrix1 = readTrainingSet(train_Set1)
    data_matrix2 = readTrainingSet(train_Set2)

    mean1 = computeMean(data_matrix1)
    mean2 = computeMean(data_matrix2)

    cov1 = computeCovarianceMatrix(data_matrix1)
    cov2 = computeCovarianceMatrix(data_matrix2)

    normal1 = computeNormalDistribution(mean1,cov1)
    normal2 = computeNormalDistribution(mean2,cov2)

    test_Matrix = composeMatrices((test_Set1, test_Set2))

    test_objects1 = len(test_Set1)
    test_objects2 = len(test_Set2)
    test_objects = test_objects1 + test_objects2

    classVector1 = np.matrix(np.ones(test_objects1)).T
    classVector2 = np.matrix((-1)*np.ones(test_objects2)).T

    classVector = np.matrix(composeMatrices((classVector1,classVector2)))

    classified_Objects = np.matrix(composeMatrices((classVector1,classVector2)))

    for k in range(test_objects):
        if normal1.pdf(test_Matrix[k])>=normal2.pdf(test_Matrix[k]):
            classified_Objects[k]=1
        else:
            classified_Objects[k]=-1

    cc_proportion = ((classVector.T*classified_Objects).item()+len(classVector))/
(2*len(classVector))

    return cc_proportion

print(classifier('train3.3','train5.5',test_Matrix_Digit3,test_Matrix_Digit5))
print(classifier('train3.3','train7.7',test_Matrix_Digit3,test_Matrix_Digit7))
print(classifier('train3.3','train8.8',test_Matrix_Digit3,test_Matrix_Digit8))
print(classifier('train5.5','train7.7',test_Matrix_Digit5,test_Matrix_Digit7))
print(classifier('train5.5','train8.8',test_Matrix_Digit5,test_Matrix_Digit8))
print(classifier('train7.7','train8.8',test_Matrix_Digit7,test_Matrix_Digit8))
```