```
##This is a notebook
```

```
#numpy.dot 二维数组是矩阵乘积，  一维数组是內积
import numpy as np
a = np.array([[1, 2], [3, 4]])
b = np.array([[11, 12], [13, 14]])
print(np.dot(a,b))  #matlab里通常用到的是 disp(['a1=' num2str(a(1)) ])
```

```
#numpy.linalg.inv()矩阵的逆矩阵
x = np.array([[1,2], [3, 4]]);
y = np.linalg.inv(x)
print (x)
print (y)
print (np.dot(x, y)) #This will be the I matrix
```

```
#range(start, stop, step)
#整数列表，通常用于for 循环
range(10)
#>>[0, 1 ..... 9]
range(0, 30, 5)
```

```
#tqdm 是一个可扩展，快速的Python进度条，  通常将for循环写在里面
import time
from tqdm import tqdm_notebook as tqdm
for t in tqdm(range(2 * n), total = 2*n, leave = False)
```

```
#numpy.matmul 返回矩阵的乘积，如果任一参数是一维数组，则在其维度上加一提升为矩阵，并在乘法之后去除
a = [[1,0], [0, 1]]
```

```
b = [1, 2]
print(np.matmul(a, b))
print(np.matmul(b, a))
#>>[1, 2]
#>>[1, 2]

# // 在python 里面相当于floor
err[t // 2] = np.amax(abs(v - vstar))

#numpy.amax() 相当于取数组中元素沿着指定轴的最大值,0是按照列，  1是按照行
a = ny.array([[3, 7, 5], [8, 4, 3], [2, 4, 9]])
print(np.amax(a))
print(np.amax(a, 0))

#plt.semilog() 将y轴对数坐标，  x轴正常
#另外还有plt.semilogx()

#下面是综合起来的例子
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import rand, randn
from scipy.linalg import inv, svd
from scipy.optimize import linprog
from tqdm import tqdm_notebook as tqdm

def null_space(A, rcond=None):
    u, s, vh = svd(A, full_matrices=True)
    M, N = u.shape[0], vh.shape[1]
    if rcond is None:
        rcond = np.finfo(s.dtype).eps * max(M, N)
    tol = np.amax(s) * rcond
    num = np.sum(s > tol, dtype=int)
    Q = vh[num:,:].T.conj()
    return Q

#

def altproj(A, B, v0, n):
    PU = A.dot((np.linalg.inv((A.T).dot(A))).dot(A.T))
    PW = B.dot((np.linalg.inv((B.T).dot(B))).dot(B.T))
    basis_UintW = np.hstack([A, B]) @ null_space(np.hstack([A, -B]))
    P_UintW = basis_UintW.dot((np.linalg.inv((basis_UintW.T).dot(basis_UintW))).dot(basis_UintW.T))
    v_star = P_UnitW.dot(v0)
    #Appy n sweeps of alternating projection
    v, err = v0, np.zeros(n)
    for t in tqmd(range(2* n), total = 2* n, leave = False):
    if (t % 2 == 0):
        v = np.matmul(PU, v)
        else:
        v = np.matmul(PW, v)
    if (t + 1) % 2 == 0:
        err[t // 2] = np.amax(abs(v - v_star))
    return v, err
#

A = np.array([[3, 1, 3, 1, 5], [2, 5, 11, 17, 23], [3, 7, 13, 19, 29]]).T
B = np.array([[1, 2, 2, 2, 6], [1, 0, 1, 0, -3], [2.5, 6, 12, 18, 26]]).T
v0 = np.array([1, 2, 3, 4, 5]) #numpy 自动判断行向量和列向量
```

```
n = 20
v, err = altproj(A, B, v0, n)

plt.figure(figsize = (8,6))
plt.semilogy(np.arange(1, n+1), err)

#Python里面取余% 取整（向下）// matlab 取整向下floor（） 取余mod（）

def kaczmarz(A, b, I):
    m, n = A.shape
    v, X, err = np.zeros(n), np.zeros((n, I)), np.zeros(I)
    v_star = ((A.T).dot(np.linalg.inv(A.dot(A.T))).dot(b)
    for i in tqdm(range(I *m), total = I * m, leave = False): #注意在Python里数组和矩阵的下标是从0开始的
    ai = A[i%m]
    bi = b[i%m]
    v = np.substract(v, np.multiply(ai, (v.dot(ai) - bi)/(ai.dot(ai))))
    if (i+1) % m == 0:
        err[i //m] = np.amax(abs(A.dot(v) - b))
        X[:, i//m] = v.T
return X, err


A = np.array([[2, 5, 11, 17, 23], [3, 7, 13, 19, 29]])
b = np.array([228, 227])
I = 500
X, err = kaczmarz(A, b, I)

plt.figure(figsize=(8, 6))
plt.semilogy(np.arange(1, I + 1), err)

#Python 中@也相当于矩阵乘法
A = randn(500, 1000)
b = A @ randn(1000)
I = 100
X, err = kaczmarz(A, b, I)

plt.figure(figsize=(8, 6))
plt.semilogy(np.arange(1, I + 1), err)

x_hat = ((A.T).dot(np.linalg.inv(A.dot(A.T))).dot(b)
plt.title(f'norm of difference between xhat and Kaczmarz is {np.linalg.norm(x_hat - X[:, -1]):.2e}');

def lp_altproj(A, b, I, s=1):
    """
    Find a feasible solution for A v >= b using alternating projection
    with every entry of v0 obeying Uniform[0,1]
    Arguments:
        A {numpy.ndarray} -- matrix defines the LHS of linear equation
        b {numpy.ndarray} -- vector defines the RHS of linear equation
        I {int} -- number of full passes through the alternating projection
        s {numpy.float} -- step size of projection (defaults to 1)
    Returns:
        v {numpy.ndarray} -- the output after I full passes
        err {numpy.ndarray} -- the error after each full pass
    """

    # Add code here
    m, n = A.shape
    v = np.zeros(n)
    err = np.zeros(I)
    res = linprog(c, A_ub=-A, b_ub=-b, bounds=[(0, None)] * c.size, method='interior-point')
    print(res)
    for t in tqdm(range(I * m), total = I * m, leave = False):
        ai = A[t%m]
        bi = b[t%m]
        if (v.dot(ai) < bi:
            np.substract(v, np.multiply(ai, (v.dot(ai) - bi)/(ai.dot(ai))))
        if (t + 1)%m == 0:
            err[t // m] = np.amax(abs(res.x - v))
    return v, err

I = 500  #Use more iteration to meet the requirement.
# Do not forget constraint xi >= 0
A1 = np.array([[2, -1, 1], [1, 0, 2], [-7, 4, -6],[-3,1,-2],[1,0,0],[0,1,0],[0,0,1]])
b1 = np.array([-1, 2, 1,0,0,0,0])
x, err = lp_altproj(A1, b1, I, s = 1)

plt.figure(figsize=(8, 6))
plt.semilogy(np.arange(1, I + 1), err)

print(np.all(A @ x - b > 0), np.all(x > 0))

#np.concatent() join sequence of arrays along an existing axis
#np.hstack() stack arrays in sequence horizontally(column wise)
#np.vstack() --- vertically---
```

```
#this is exercise 5
np.random.seed(0)
c = randn(1000)
A = np.vstack([-np.ones((1, 1000), randn(500, 1000)])
b = np.concatenate([[-1000], A[1:]@rand(1000)])
I, ep = 1000, le -6
#Do not forget constraint xi >= 0 and c^T x < = -1000
A1 = np.vstack([A, -c, np.identity(1000)])
b1 = np.concatenate([b, [1000], np.zeros(1000)])

x, err = lp_altproj(A1, b1 + ep, I, s =1)
plt.figure(figsize=(8, 6))
plt.semilogy(np.arange(1, I + 1), err)
print(np.all(A @ x - b > 0), np.all(x > 0))


import pandas as pd
# read mnist csv file to a dataframe
df = pd.read_csv('mnist_train.csv')
# append feature column by merging all pixel columns
df['feature'] = df.apply(lambda row: row.values[1:], axis=1)
# only keep feature and label column
df = df[['feature', 'label']]
df.head()

from sklearn.model_selection import train_test_split
def extract_and_split(df, d, test_size=0.5):
    """
    Extract the samples with given lables and randomly separate the samples into equal-sized training and testing groups, extend each vector to length 785 by appending a −1
    Arguments:
        df {dataframe} -- the dataframe of MNIST dataset
        d {int} -- digit needs to be extracted, can be 0, 1, ..., 9
        test_size {float} -- the fraction of testing set, default value is 0.5
    Returns:
        X_tr {numpy.ndarray} -- training set features, a matrix with 785 columns
                            each row corresponds the feature of a sample
        y_tr {numpy.ndarray} -- training set labels, 1d-array
                            each element corresponds the label of a sample
        X_te {numpy.ndarray} -- testing set features, a matrix with 785 columns
                            each row corresponds the feature of a sample
        y_te {numpy.ndarray} -- testing set labels, 1d-array
                            each element corresponds the label of a sample
    """
    df_d=df.loc[df['label']==d]   #extract digit d
    X=df_d['feature']
    y=df_d['label']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=0)
    X_train=X_train.values   #convert datafram to array
    X_test=X_test.values
    y_train=y_train.values
    y_test=y_test.values
    X_tr=np.zeros((X_train.shape[0],784))
    y_tr=np.zeros(y_train.shape[0])
    for i in range(X_train.shape[0]):
        for j in range(784):
            X_tr[i,j]=X_train[i][j]
    for i in range(y_train.shape[0]):
        y_tr[i]=y_train[i]
    X_tr=np.insert(X_tr,784,-1,axis=1)

    X_te=np.zeros((X_test.shape[0],784))
    y_te=np.zeros(y_test.shape[0])
    for i in range(X_test.shape[0]):
        for j in range(784):
            X_te[i,j]=X_test[i][j]
    for i in range(y_test.shape[0]):
        y_te[i]=y_test[i]
        X_te=np.insert(X_te,784,-1,axis=1)
    return X_tr, X_te, y_tr, y_te


def remove_outlier(x, thresh=3.5):
    """
    Returns points that are not outliers to make histogram prettier
    reference: https://stackoverflow.com/questions/11882393/matplotlib-disregard-outliers-when-plotting/11886564
    Arguments:
        x {numpy.ndarray} -- 1d-array, points to be filtered
        thresh {float} -- the modified z-score to use as a threshold. Observations with
                          a modified z-score (based on the median absolute deviation) greater
                          than this value will be classified as outliers.
    Returns:
        x_filtered {numpy.ndarray} -- 1d-array, filtered points after dropping outlier
    """
    if len(x.shape) == 1: x = x[:,None]
    median = np.median(x, axis=0)
    diff = np.sqrt(((x - median)**2).sum(axis=-1))
    modified_z_score = 0.6745 * diff / np.median(diff)
    x_filtered = x[modified_z_score <= thresh]
    return x_filtered

#numpy.multiply指的是矩阵的点乘
#Python format格式化函数
#>>>"{} {}".format("hello", "world")    # 不设置指定位置，按默认顺序
#'hello world'

#>>> "{0} {1}".format("hello", "world")  # 设置指定位置
#'hello world'

#>>> "{1} {0} {1}".format("hello", "world")  # 设置指定位置
'world hello world'
def mnist_pairwise_altproj（df, a, b, solver, test_size = 0.5, verbose = False):
"""
    Pairwise experiment for applying alternating projection to classify digit a and digit b
    Arguments:
        df {dataframe} -- the dataframe of MNIST dataset
        a, b {int} -- digits to be classified
        test_size {float} -- the fraction of testing set, default value is 0.5
        solver {function} -- function to compute linear classifier
        verbose {bool} -- whether to print and plot results
    Returns:
        z_hat {numpy.ndarray} -- coefficients for linear classifier
        res {numpy.ndarray} -- numpy.array([traing error, testing error])
```

```
"""
Xa_tr, Xa_te, ya_tr, ya_te = extract_and_split(df, a, test_size)
Xb_tr, Xb_te, yb_tr, yb_te = extract_and_split(df, b, test_size)

for i in range (ya_tr.shape[0]):
    ya_tr[i] = -1;

for i in range(ya_te.shape[0]):
    ya_te[i]=-1;

for i in range(yb_tr.shape[0]):
    yb_tr[i]=1;

for i in range(yb_te.shape[0]):
    yb_te[i]=1;
#Construct full set
X_tr = np.concatenate((Xa_tr, Xb_tr), axis = 0)
y_tr = np.concatenate((ya_tr, yb_tr), axis = 0)
X_te = np.concatenate((Xa_te, Xb_te), axis=0)
y_te = np.concatenate((ya_te, yb_te), axis=0)

#Run solver on training set to get the linear classifier
A_tilde = np.multiply(X_tr, y_tr[:, np.newaxis])
z_hat, err = solver(A_tilde, np.ones(y_tr.shape[0]))

#compute estimation and misclassification on training set
y_hat_tr = X_tr.dot(z_hat)
for i in range (y_hat_tr.shape[0]):
    if y_hat_tr[i] >= 0:
        y_hat_tr[i] = 1
    else:
        y_hat_tr[i] = -1

cm_tr = np.array([[0, 0], [0, 0]])

for i in range(y_hat_tr.shape(0))
     if (y_tr[i]==-1 and y_hat_tr[i]==-1):
        cm_tr[0,0]=cm_tr[0,0]+1
    elif (y_tr[i]==-1 and y_hat_tr[i]==1):
        cm_tr[0,1]=cm_tr[0,1]+1
    elif (y_tr[i]==1 and y_hat_tr[i]==-1):
        cm_tr[1,0]=cm_tr[1,0]+1
    elif (y_tr[i]==1 and y_hat_tr[i]==1):
        cm_tr[1,1]=cm_tr[1,1]+1
err_tr = cm_tr[0,1] + cm_tr[1, 0])/y_hat_tr.shape[0]

#compute estimation and misclassification on testing set
y_hat_te = X_te.dot(z_hat)
for i in range(y_hat_te.shape[0]):
    if y_hat_te[i]>=0:
        y_hat_te[i]=1
    else:
        y_hat_te[i]=-1

cm_te=np.array([[0,0],[0,0]])

for i in range(y_hat_te.shape[0]):
    if (y_te[i]==-1 and y_hat_te[i]==-1):
            cm_te[0,0]=cm_te[0,0]+1
        elif (y_te[i]==-1 and y_hat_te[i]==1):
            cm_te[0,1]=cm_te[0,1]+1
        elif (y_te[i]==1 and y_hat_te[i]==-1):
            cm_te[1,0]=cm_te[1,0]+1
        elif (y_te[i]==1 and y_hat_te[i]==1):
            cm_te[1,1]=cm_te[1,1]+1

    err_te = (cm_te[0,1]+cm_te[1,0])/y_hat_te.shape[0]

    if verbose:
        print('Pairwise experiment, mapping {0} to -1, mapping {1} to 1'.format(a, b))
        print('training error = {0: .2f}%, testing error = {1: .2f}%'.format(100 * err_tr, 100* err_te))
        print('Training set confusion matrix: \n {0}'.format(cm_tr))
        print('Testing set confusion matrix: \n {0}' .format(cm_te))

        #plot the two histogram together
        #plt.hist(-,bins) bins指的是在整个区间划分的小区间的个数
        ya_te_hat = Xa_te.dot(z_hat)
        yb_te_hat = Xb_te.dot(z_hat)
        output = [remove_outlier(ya_te_hat), remove_outlier(yb_te_hat)]
        plt.figure(figsize = (8, 4))
        plt.hist(output, bins = 50)
    res = np.array([err_tr,m err_te])
    return z_hat, res

solver = lambda A, b: lp_altproj(A, b + le-6, 100)
z_hat , res = mnist_pairwise_altproj （df, 0, 1,solver, verbose = True)
```

```
#This is excercise 7
def mnist_multiclass_altproj(df, solver, test_size=0.5):
    """
    Experiment for applying least-square to classify all digits using one-hot encoding
    Arguments:
        df {dataframe} -- the dataframe of MNIST dataset
        solver {function} -- function to compute linear classifier
        test_size {float} -- the fraction of testing set, default value is 0.5
    Returns:
        Z {numpy.ndarray} -- coefficients for linear classifier
        res {numpy.ndarray} -- numpy.array([traing error, testing error])
    """

    #Split training and testing sets
    X =df['feature']
    y =df[' label' ]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = test_size, random_state = 0)
    X_train = X_train.values #convert data frame to array
    X_test = X_test.values
    y_train = y_train.values
    y_test = y_test.values

    X_tr = np.zeros((X_train.shape[0], 784))
    y_tr = np.zeros((X_te,shape[0])
    for i in range(X_train.shape[0]):
        for j in range(784):
            X_tr[i, j] = X_train[i, j]
    for i in range(y_train.shape[0]) :
        y_tr[i] = y_train[i]
```

```
X_tr = np.insert(X_tr, 784, -1, axis = 1)


X_te=np.zeros((X_test.shape[0],784))
y_te=np.zeros(y_test.shape[0])
for i in range(X_test.shape[0]):
   for j in range(784):
       X_te[i,j]=X_test[i][j]
for i in range(y_test.shape[0]):
   y_te[i]=y_test[i]
X_te=np.insert(X_te,784,-1,axis=1)


for i in range(10):
   if i==0:
       A_tilde = np.multiply(X_tr, Y[:,0][:, np.newaxis])
   else:
       A_tilde = np.concatenate((A_tilde, np.multiply(X_tr, Y[:,i][:, np.newaxis])),axis=1)
A_new=np.zeros((10*y_tr.shape[0],7850))
for i in range(10*y_tr.shape[0]):
   for j in range((i%10)*785,(i%10)*785+785):
       A_new[i][j]=A_tilde[i//10][j]



b_tilde = np.ones(10*y_tr.shape[0])
Z, err = solver(A_new, b_tilde)
# Reshape z
Z = (Z.reshape((10,785))).T
y_hat_tr = X_tr.dot(Z)
y_hat_tr = Y_hat_tr.argmax(axis=1)
#how to find the maximum index of rows in maylab
```

```
#[M, I] = max(A, [], 2)
cm_tr = np.zeros((10,10))
for m in range(y_tr.shape[0]):
    for i in range(10):
     if (y_tr[m] == i and y_hat_tr[m] == j):
        cm[i, j] = cm[i, j] + 1
err_tr = 0
for i in range (10)
    for j in range (10)
        if i != j:
            err_tr = err_tr + cm_tr[i, j]
err_tr = err_tr/ y_tr.shape[0]

# Compute estimation and misclassification on testing set
Y_hat_te=X_te.dot(Z)
y_hat_te = Y_hat_te.argmax(axis=1)
cm_te=np.zeros((10,10))
for m in range(y_te.shape[0]):
    for i in range(10):
        for j in range(10):
            if (y_te[m]==i and y_hat_te[m]==j):
                cm_te[i,j]=cm_te[i,j]+1

err_te = 0
for i in range(10):
    for j in range(10):
        if i!=j:
            err_te=err_te+cm_te[i,j]
err_te=err_te/y_te.shape[0]
```

```
print('training error = {0:,2f}%,testing error = {1:, 2f}%' .format(100 * err_tr, 100 * err_te))

print('Training set condusion matrix: \n {0}' .format(cm_tr))

print('Testing set confusion matrix: \n {0}' .format(cm_te))


res = np.array([err_tr, err_te])

return Z, res


solver = lambda A, b: lp_altproj(A, b +le-6, 100)

Z ,res = mnist_multiclass_altproj(df, solver)
```