人工智能导论

实验报告

实验题目：\_ \_使用kmeans算法压缩图片&三次实验的相关探索\_\_

姓 名：\_\_\_\_\_ \_\_\_ \_ \_\_\_ \_\_\_魏靖\_\_\_\_\_ \_\_\_\_\_ \_\_\_ \_\_\_\_\_

学 号：\_\_\_\_\_\_\_\_ \_\_\_ \_\_\_\_2021213513\_\_\_\_\_\_\_ \_\_\_ \_\_\_\_\_\_\_\_\_\_

日 期：\_\_\_\_\_\_\_ \_\_\_ \_\_\_\_\_2022.8.29\_\_\_\_\_\_\_\_ \_\_\_ \_\_\_\_\_\_\_\_\_\_

自我评分： 【A】

自我评分说明：A+，A，B+，B，B-，C，D，分别对应分数 95、90、85、80、75、70、60

**诚信声明**

本人郑重承诺：本实验程序和实验报告均是本人独立学习和工作所获得的成果。尽我所知，实验报告中除特别标注的地方外，不包含其他同学已经发表或撰写过的成果；实验程序中对代码工作的任何帮助者所作的贡献均做了明确的说明，并表达了谢意。

如有抄袭，本人愿意承担因此而造成的任何后果。

特此声明。

签名：\_\_\_\_\_\_\_\_魏靖\_\_\_\_\_\_\_\_\_\_

日期：\_\_\_\_\_2022.8.29\_\_\_\_\_\_\_\_

程序引用说明

总代码行数\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; 引用占比\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

目录

1 实验简介 ………………………………………………………… 1

2 程序框架 ………………………………………………………… 1

3 关键代码实现 …………………………………………………… 1

3-1 k均值聚类算法 ………………………………………………… 1

3-2 利用sklearn的K-means算法压缩图像 ……………………… 3

4 不足 ……………………………………………………………… 5

5 心得体会 ………………………………………………………… 5

5-1 实验1 ………………………………………………………………5

5-2 实验2 ………………………………………………………………5

5-3 实验3 ………………………………………………………………7

6 运行截图 ………………………………………………………… 8

6-1 实验1 ………………………………………………………………8

6-2 实验2 ………………………………………………………………9

6-3 实验3 …………………………………………………………… 11

6-4 图片压缩 ………………………………………………………… 12

1、实验简介

对实验算法内在结构进行分析和贯彻认知，达到理解运用的实践效果。

1. 程序框架

加载原图并归一化

训练kmeans模型

获取簇中心与数据所属簇

得到压缩后的图片

显示

3、关键代码实现

3-1 k均值聚类算法

k均值聚类算法就是通过k个均值将数据集划分为k个簇、实现类内差异最小且类间差异尽可能大的算法。首先选择k个中心（可以是人为指定，也可以是随机选择），第二步：每个样本点离哪个中心更近，就把他归为对应的类，第三步：接着每个聚类中心使用分配到对应类的样本点重新计算出中心值，然后重复二、三步直到最终收敛。因而此算法要尤其注意几个while和if的范围。

思考：k值能否更新？即能否通过目标函数最小化的方法找到一个最合适的k（当然碍于最后对颜色的设置和相关复杂度的限制，1<k<13）。

思路：先设置k的初始值为2，在聚类的同时计算目标函数的值，聚类完毕++k，如果++k的目标函数值小于k，则更新并记录，直到k=12.因而需要再有一个goalcentroids、goalclusterAssment和goalk。

def kmeans(dataSet):

k=2

while k<13:

#设置迭代次数终止条件，避免无效k

ss=1

numSamples=dataSet.shape[0]

clusterAssment=mat(zeros((numSamples,2)))

#i-of-K编码

r=mat(zeros((numSamples,k)))

clusterChanged=True

centroids=initCentroids(dataSet,k)

while clusterChanged:

clusterChanged=False

for i in range(numSamples):

minDist=10000.0

minIndex=0

for j in range(k):

distance=euclDistance(centroids[j,:],dataSet[i,:])

if distance<minDist:

minDist=distance

minIndex=j

r[i,:]=0

r[i,j]=1

#还可以考虑增加其他终止条件比如误差变化等等

if clusterAssment[i,0]!=minIndex or ss==len(dataSet)\*\*2:

clusterChanged=True

clusterAssment[i,:]=minIndex,minDist\*\*2

for j in range(k):

pointsInCluster=dataSet[nonzero(clusterAssment[:,0].A==j)[0]]

if len(pointsInCluster)!=0:

centroids[j,:]=mean(pointsInCluster,axis=0)

ss=ss+1

for i in range(numSamples):

for j in range(k):

#目标函数

J=r[i,j]\*((dataSet[i:]-centroids[j,:])\*\*2)

#第一次聚类完毕存一个goalJ的初始值

if k=2:

goalcentroids=centroids

goalclusterAssment=clusterAssment

goalk=k

goalJ=J

#如果当前聚类的J值小于goalJ了，则更新goalk

if J<goalJ:

#增加k-goalk行到原来的goalcentroids矩阵

numSamples,dim=dataSet,shape

b=zeros((k-goalk,dim))

goalcentroids=row\_stack((goalcentroids,b))

goalcentroids=centroids

goalclusterAssment=clusterAssment

goalk=k

k=k+1

print("聚类完毕！")

return goalcentroids,goalclusterAssment,goalk

然而，改进后的算法时间复杂度较大，需要不断运算来推出最优值，因而在实际应用中，对一些较明显的样本数据，我们可以先展示散点图，人为感觉可以分几类（或者说需要把它分几类）则输入相应的k，再交给机器聚类（这样实现的是满足人为需求下的最优聚类而不是最优分类）。照此思路完成实验1-2的最终输入输出。

3-2 利用sklearn的K-means算法压缩图像

图像压缩是为了减少表示数字图像时需要的数据量，而kmeans聚类中心可以看作是某个类的中心表示，如果我们将图像数据进行聚类，比如找到十个类，便可用这十个样本中心来表示该图像，实现了图片的压缩。

应当注意的是，在k的选取和模型训练次数选择时：k越大会越接近原图片，压缩质量越好，但是相对压缩率却不高，且训练过程中随机性可能会更高，需要选择的模型训练次数就应该相应增加以此提高准确度，如此一来时间复杂度和空间复杂度又增加了；而若k太小，压缩质量就极有可能赶不上。

from skimage import io

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

# 加载图片并进行归一化处理

#目标：①把数据转换为（0,1）区间的小数；②把有量纲（物理量的大小与单位有关）表达式变为无量纲表达式（此处只需实现第一个目标即可）

#优点：①加快梯度下降求最优解的速度；②有可能提高精度，特别是在需要计算样本之间距离的时候（如果一个特征值域范围非常大，那么距离计算就主要取决于这个特征，接着就极可能与实际情况相悖）。

p = io.imread('sample.jpg') / 255.

# 获得图像数据形状信息：宽、高、通道

w,h,d=p.shape

# 重置图像大小

data = p.reshape(w\*h, d)

# 构建kmeans算法模型，咱们这里设置簇的个数为32，构建100个模型，选择其中最好的输出（默认的n\_clusters为16、n\_init为100）

model = KMeans(n\_clusters=32, n\_init=100)

# 训练模型

model.fit(data)

# 得到各簇中心

centroids = model.cluster\_centers\_

# 获取每条数据所属簇

C = model.predict(data)

# 使用kmeans算法得到的数据得到压缩后的图片

compressed\_p = centroids[C].reshape((w,h,d))

# 绘制原图和压缩图片

fig, ax = plt.subplots(1, 2)

ax[0].imshow(p)

ax[1].imshow(compressed\_pic)

plt.show()

4、不足

在大多数实际应用当中其实k均值聚类的k值很难人为确定。Sklearn的kmeans虽然实现了多个模型取最优，但仍需要视具体情况而反复改变n\_clusters值，取最好的一次结果。

5、心得体会

5-1 实验1

相关库的安装和环境的调试过程中，学习使用了cmd、anaconda promote等处理，特别是后来在运行第一次实验代码时，两次出错，一次找不到sklearn的module，一次找不到matplotlib的module，通过网络查询了解到sklearn其实是[scikit-learn](https://so.csdn.net/so/search?q=scikit-learn&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_35687547/article/details/_blank)的简称，这个库中集成了一些常用的机器学习方法，在进行机器学习任务时，调用sklearn库中提供的模块对大多数的机器学习任务来说作用很大。检索发现有[scikit-learn](https://so.csdn.net/so/search?q=scikit-learn&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_35687547/article/details/_blank)的pyc文件但是但是存在问题，于是换成国内的源利用cmd重试了一下，就可以运行了。至于matplotlib的用处更是令人震惊：调用其中的函数，只需简单的代码就可以生成相应的图表。特别是在代码运行的时候，看着图上一步步逼近散点图回归的线性模型，体会到了matplotlib和Python结合在数据分析方面的实用性，期待学习更多。

对于一些Runtime Warning的处理：Mean of empty slice的warning出现在1-2的运行当中，解决方法为：

dataSet[i][1]和dataSet[i,1]的区别：C++中arr[i][j]在Python中应该表示为中括号内直接加逗号分隔的形式arr[i,j]，就譬如arr[i,:]表示第i行。

5-2 实验2

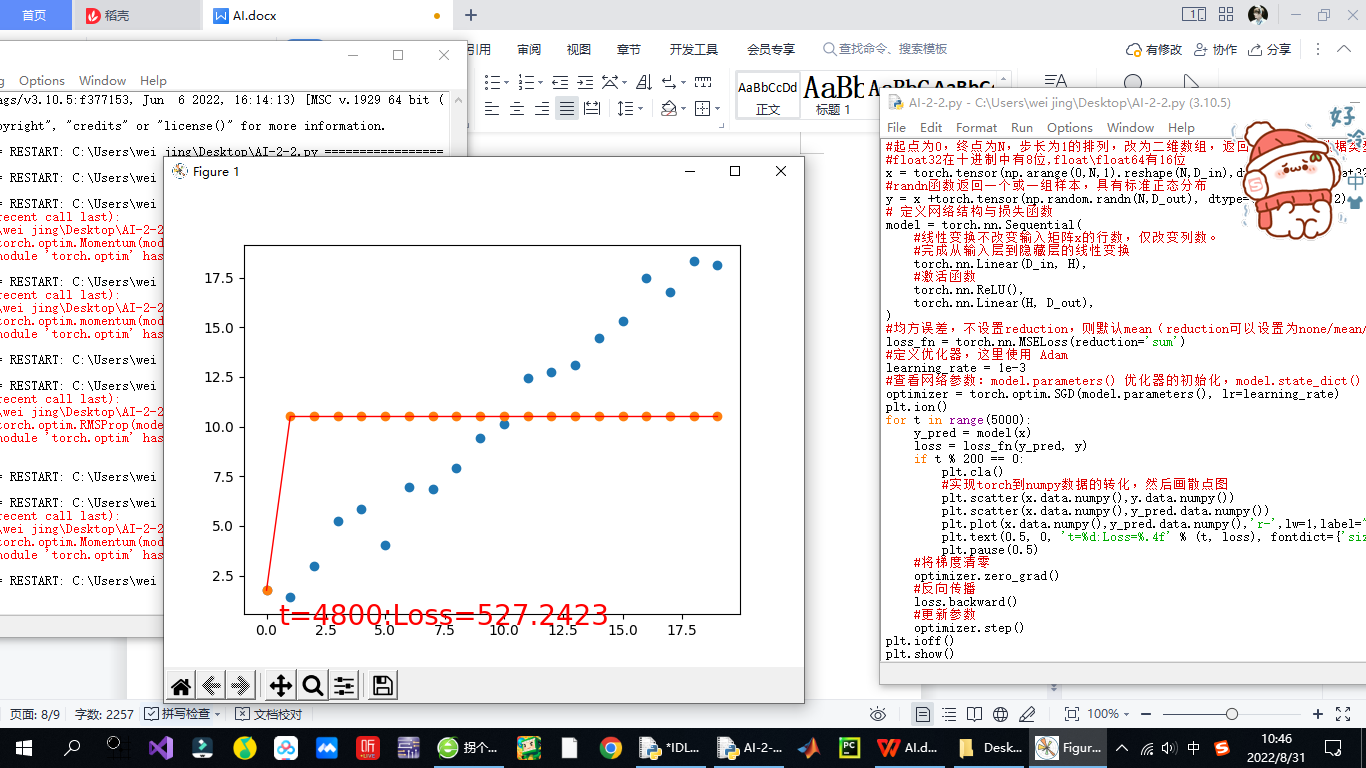
优化器的差别与选择：优化算法是根据损失函数得到的损失值来更新参数的算法，在2-2和2-3中我们分别选择了Adam和SGD算法。

SGD作为优化器相对简单和基础，其核心是随机梯度下降，学习率是指定不变的，使用SGD时会把数据拆分后再分批不断放入NN中计算，这对神经网络训练速度提高有很大的帮助，但是不能反映整体数据情况。

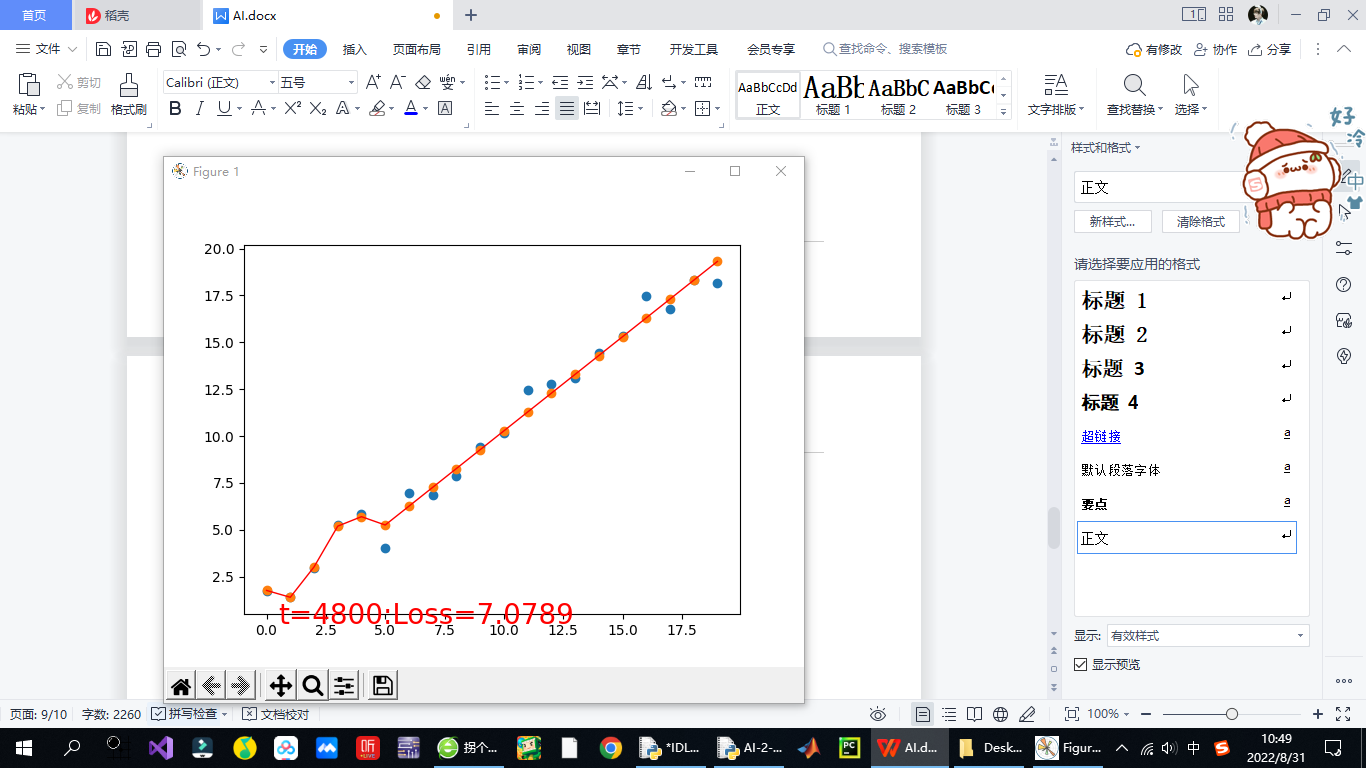
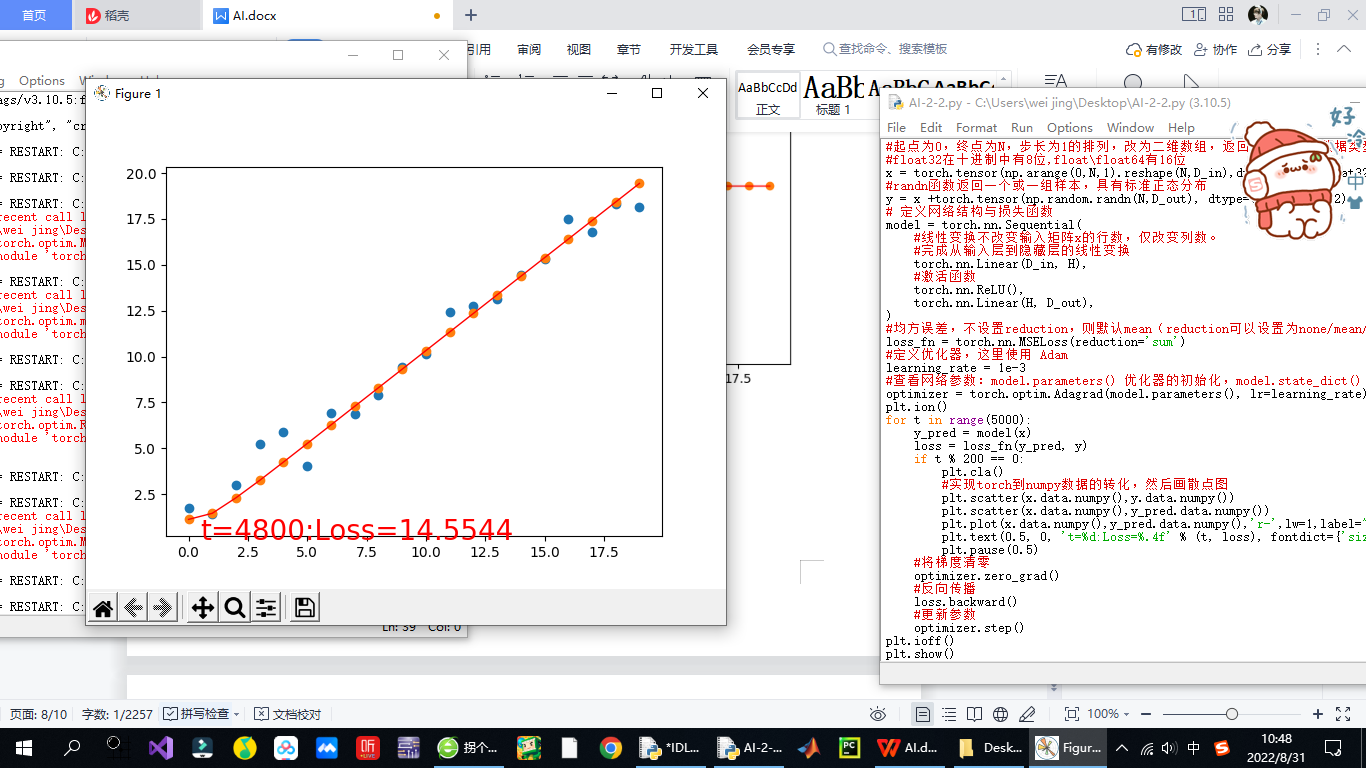
Adam算法通过计算梯度的一阶矩（期望值）估计和二阶矩（对变量的平方/随机变量与均值(期望)的差求期望）估计而为不同的参数设计独立的自适应性学习率，计算m 时有momentum下坡的属性（momentum参数的更新是把原始参数累加一个负的学习率，乘以校正值。好比把一个人从平地放到了一个斜坡上，只要他往下坡的方向走一点点，由于向下的惯性，他不自觉地就一直往下走, 这样走的弯路就变少了），计算v时有Adagrad阻力的属性（优化学习率，使得每一个参数更新都会有自己与众不同的学习率）, 然后在更新参数时把m和v都考虑进去。（虽然RMSprop也是结合了momentum和Adagrad，但是它只有一阶矩估计）

但是，并不是越先进的优化器结果越佳。我们在试验中可以尝试不同的优化器，找到那个最适合当下数据/网络的优化器。

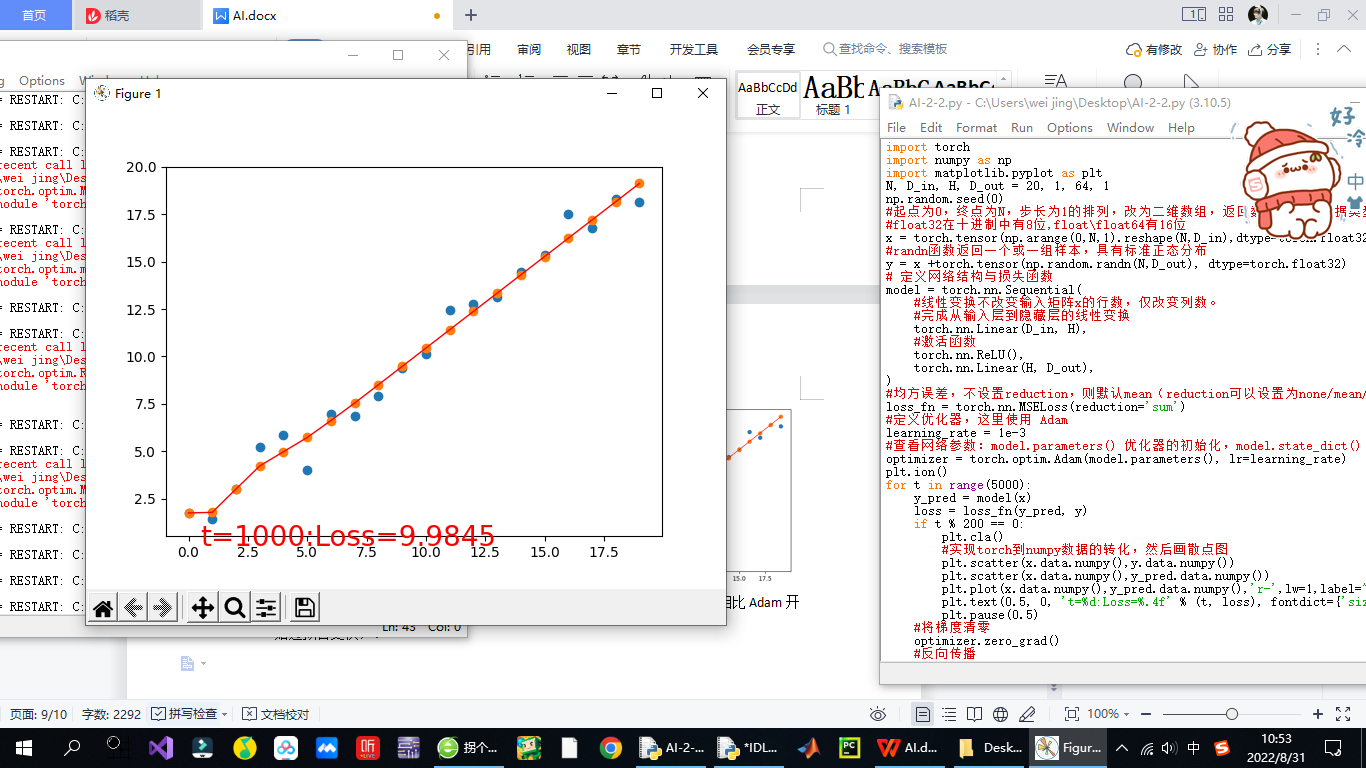
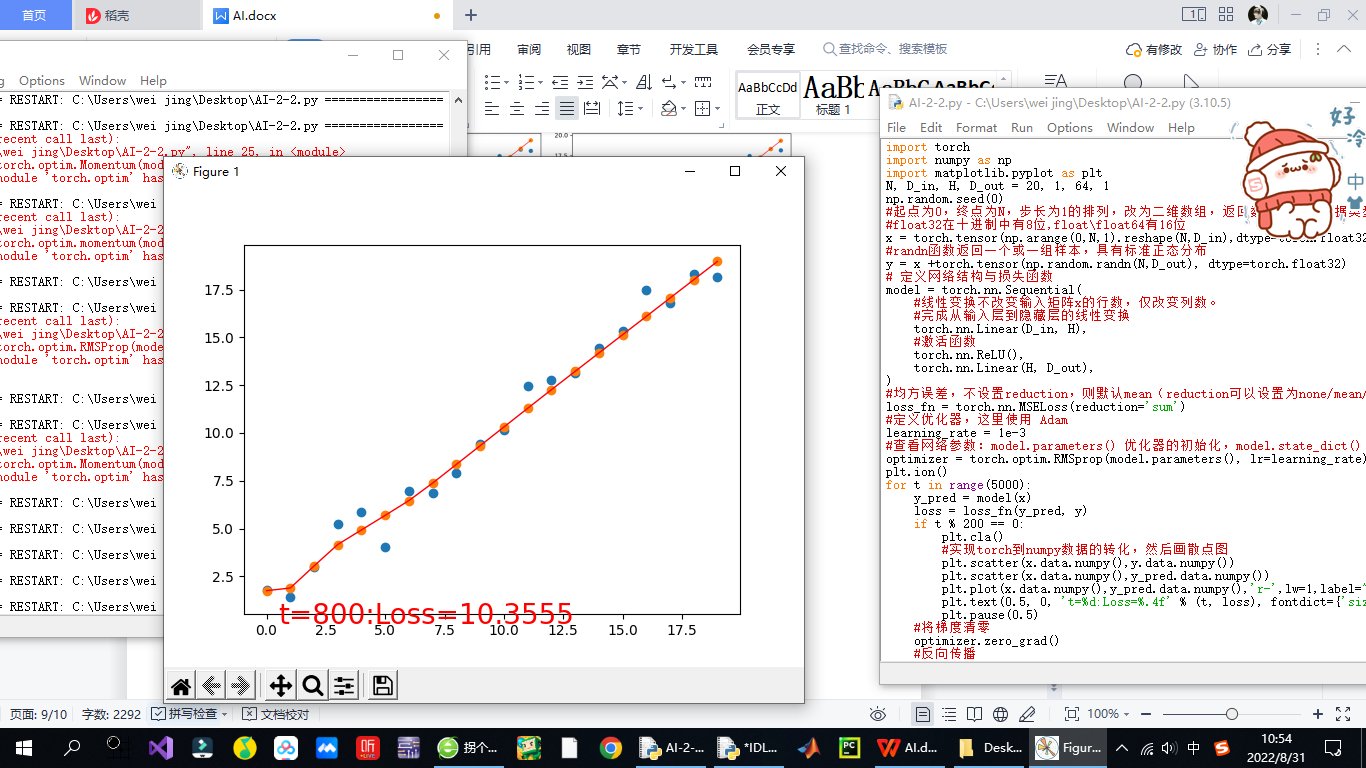
比如，在2-2中，对某组生成的随机数据，用SGD的拟合效果及其不佳：



Adagrad（左）的拟合和Adam效果很好，但不及Adam速度快（即使Adam后期出现了过拟合）：



RMSprop（左）在前期拟合效果很好，但很快出现了过拟合的现象（相比Adam开始过拟合更快）：



基于此次数据，最好的优化器当是Adagrad。但是只针对这一组数据，真正要确定这几种优化器哪个优化更适合此处的回归模型，还需要不同的数据观察对比分析。

确定隐藏层的神经元个数：隐藏层的神经元在一定程度上与输入输出层神经元总和成反比，与训练样本数成正比，但还与输入输出层大小、模型和数据本身等有关，需要不断试验得到。从较小开始，如果欠拟合就适当增加，过拟合就适当减少。原则上隐藏层神经元数应在输入输出层大小之间。隐藏层神经元数应为输入层大小的2/3加上输出层大小的2/3。隐藏层神经元数量应小于输入层大小两倍。

5-3 实验3

如果直接调用arr=im.flatten()，会出现错误，原因在于：flatten是numpy的一个函数，即返回一个一维数组。只能适用于numpy对象，即array或者mat。通过PIL的Image读入后直接进行直方图分析时则图片未能转化为数组。

应优化为：

im\_a=np.array(im,'f')

arr=im\_a.flatten()

图片二值化时自定义灰度table的操作：

#原代码

table = []

for i in range(256):

if i < threshold:

table.append(0)

else:

table.append(1)

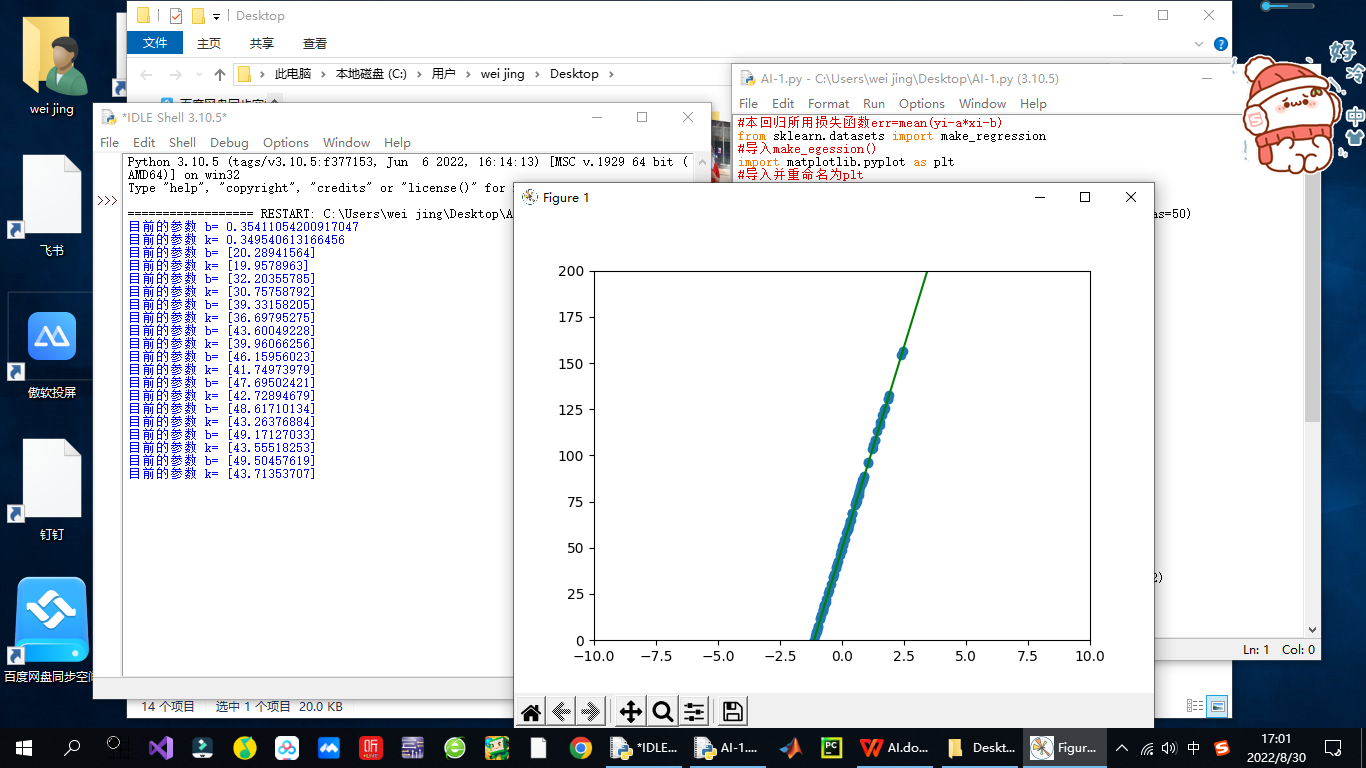
#改进后

table=lambda x:0 if x<threshold else 1

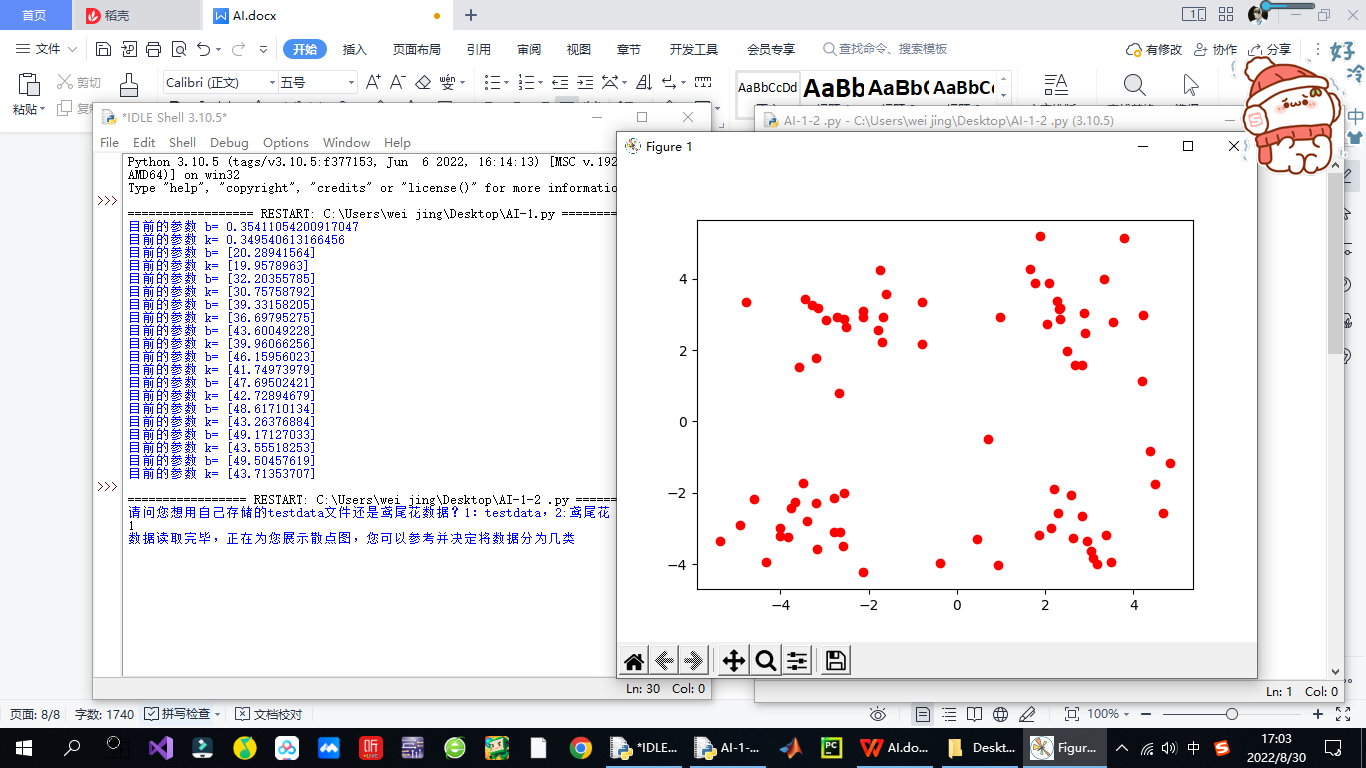
6、运行截图

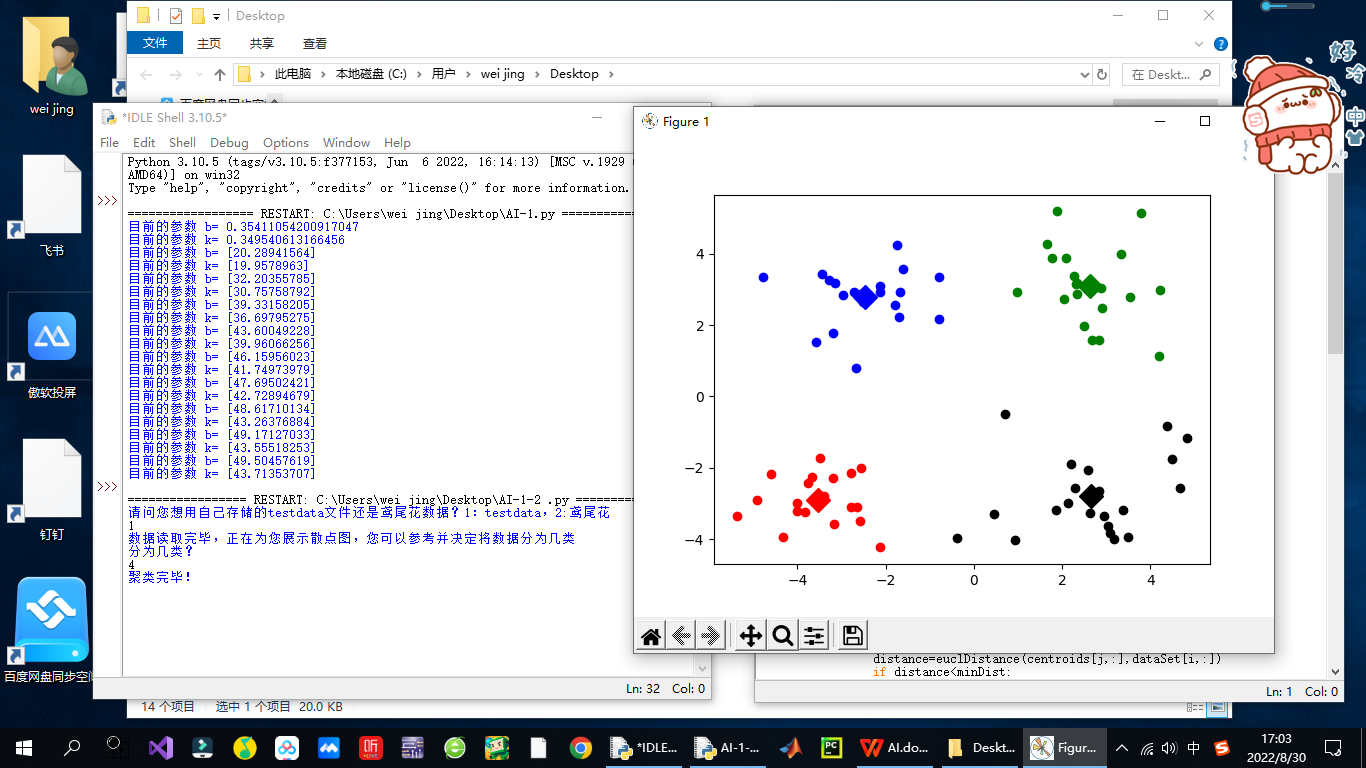
6-1 实验1

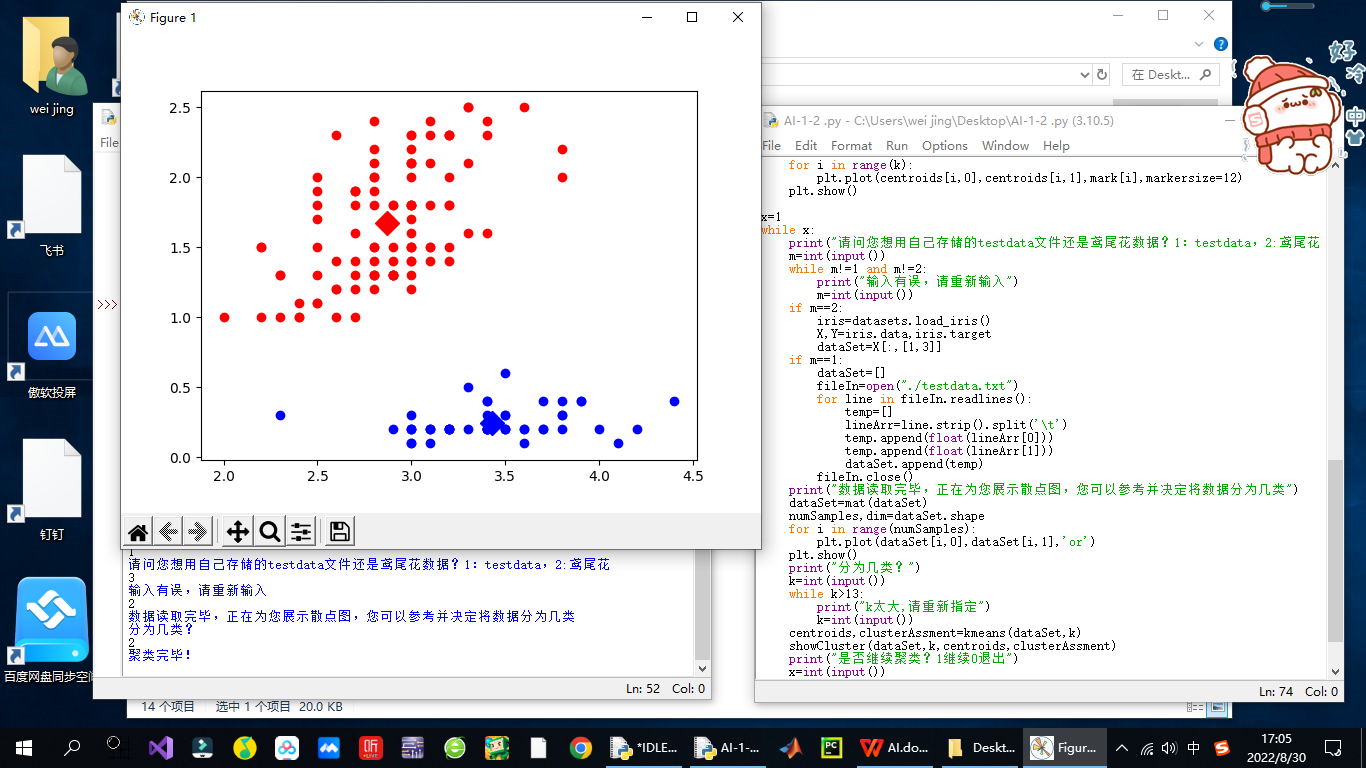
6-1-1



6-1-2

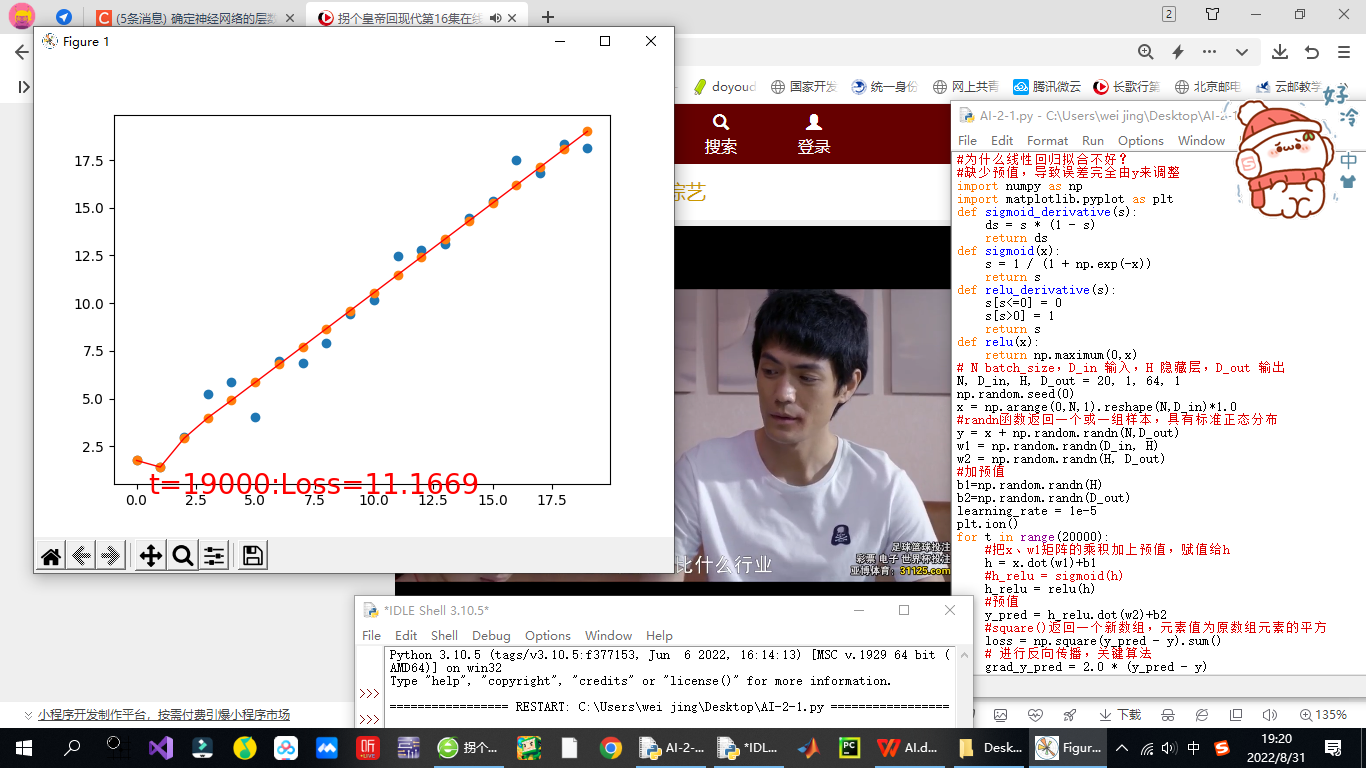
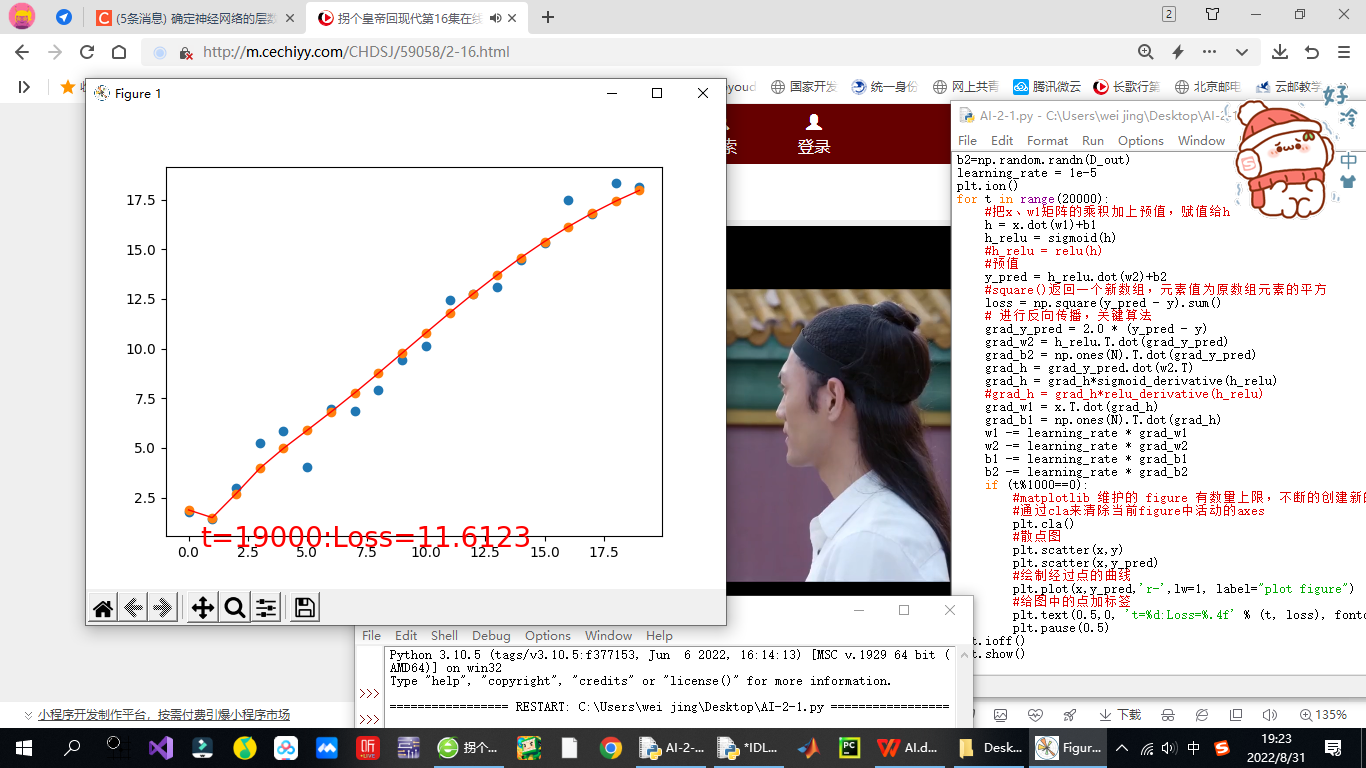




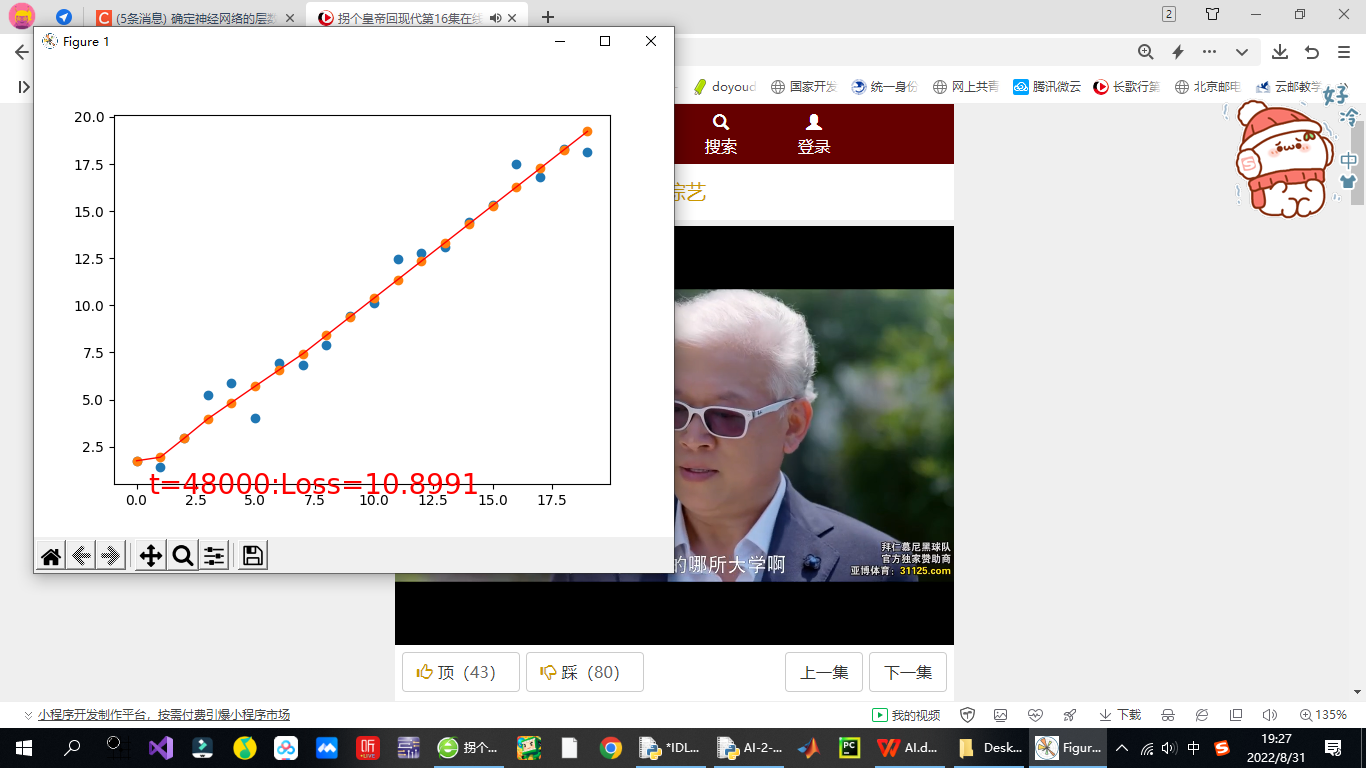
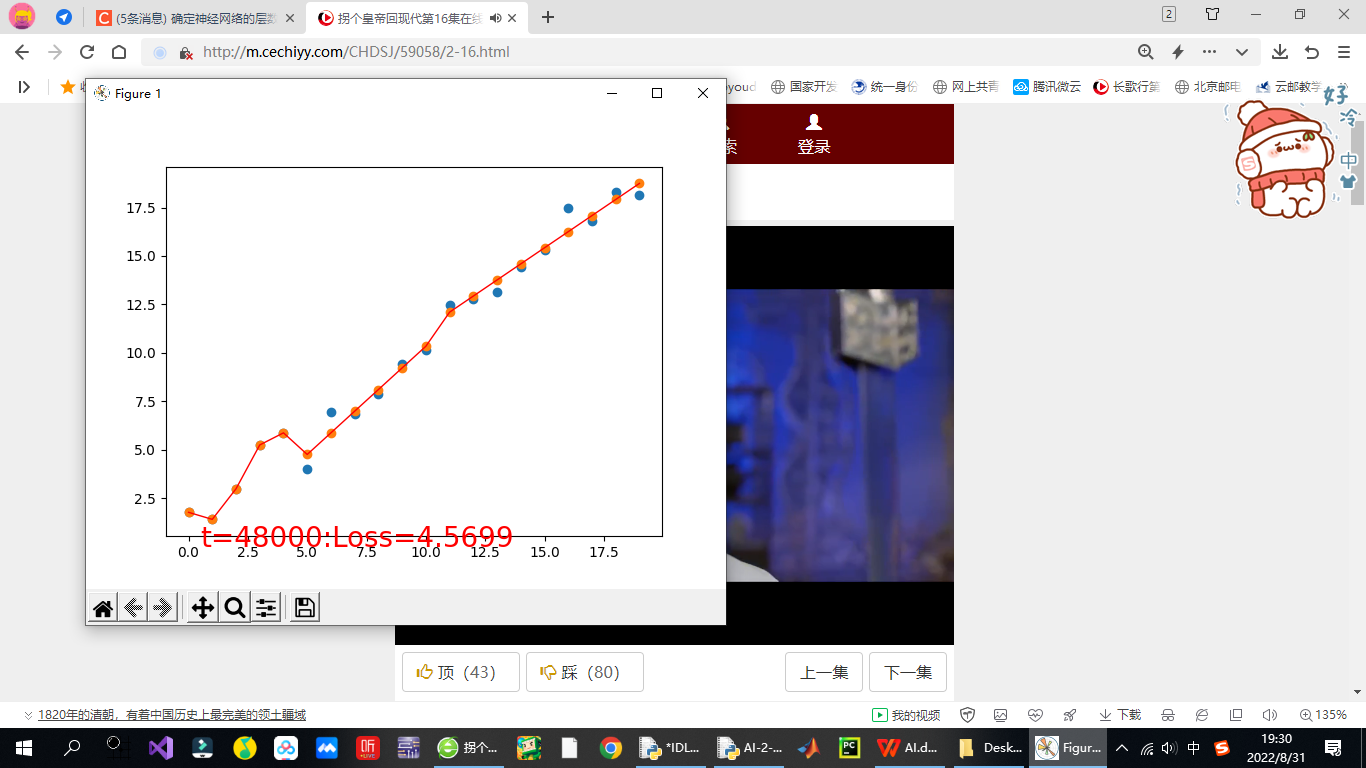


6-2 实验2

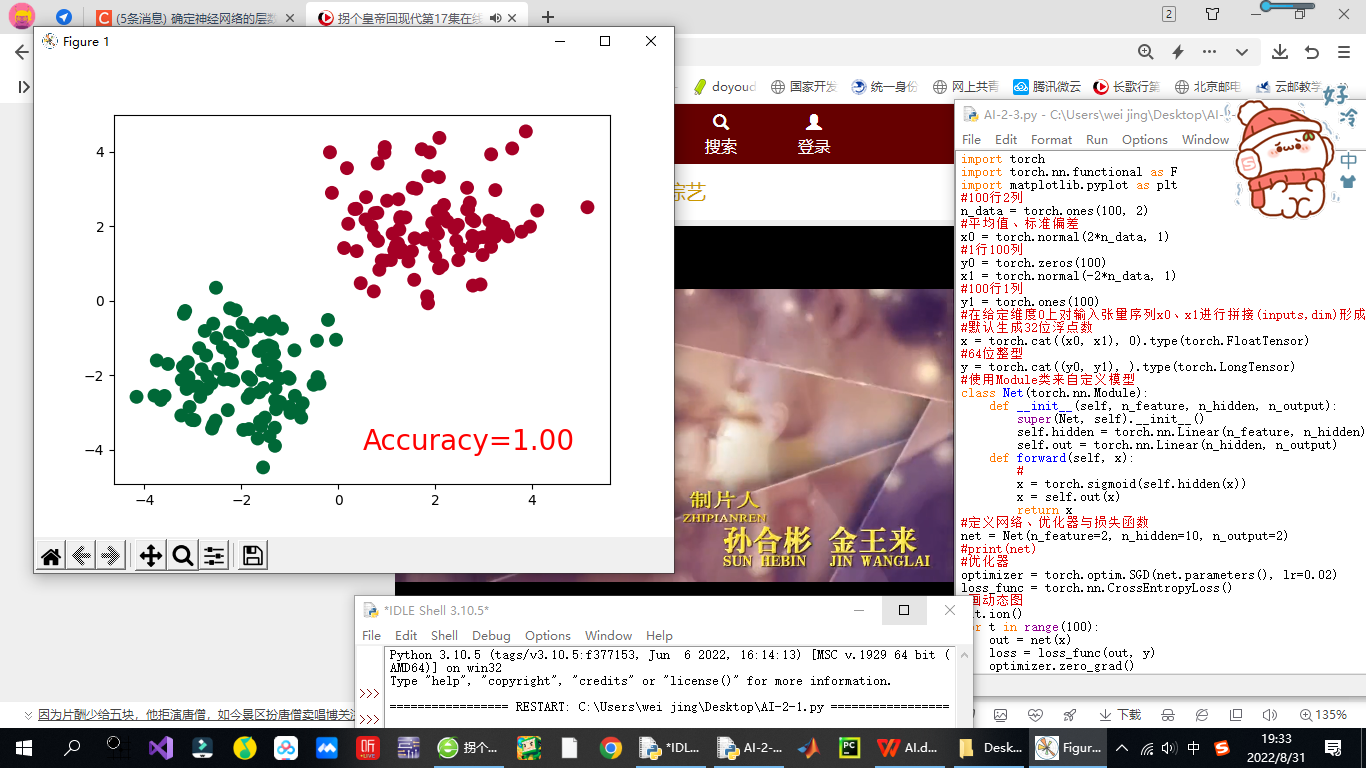
6-2-1 sigmoid作激活函数（左） relu作激活函数（右）



6-2-2 Adam作优化器（左） Adagrad作优化器（右）

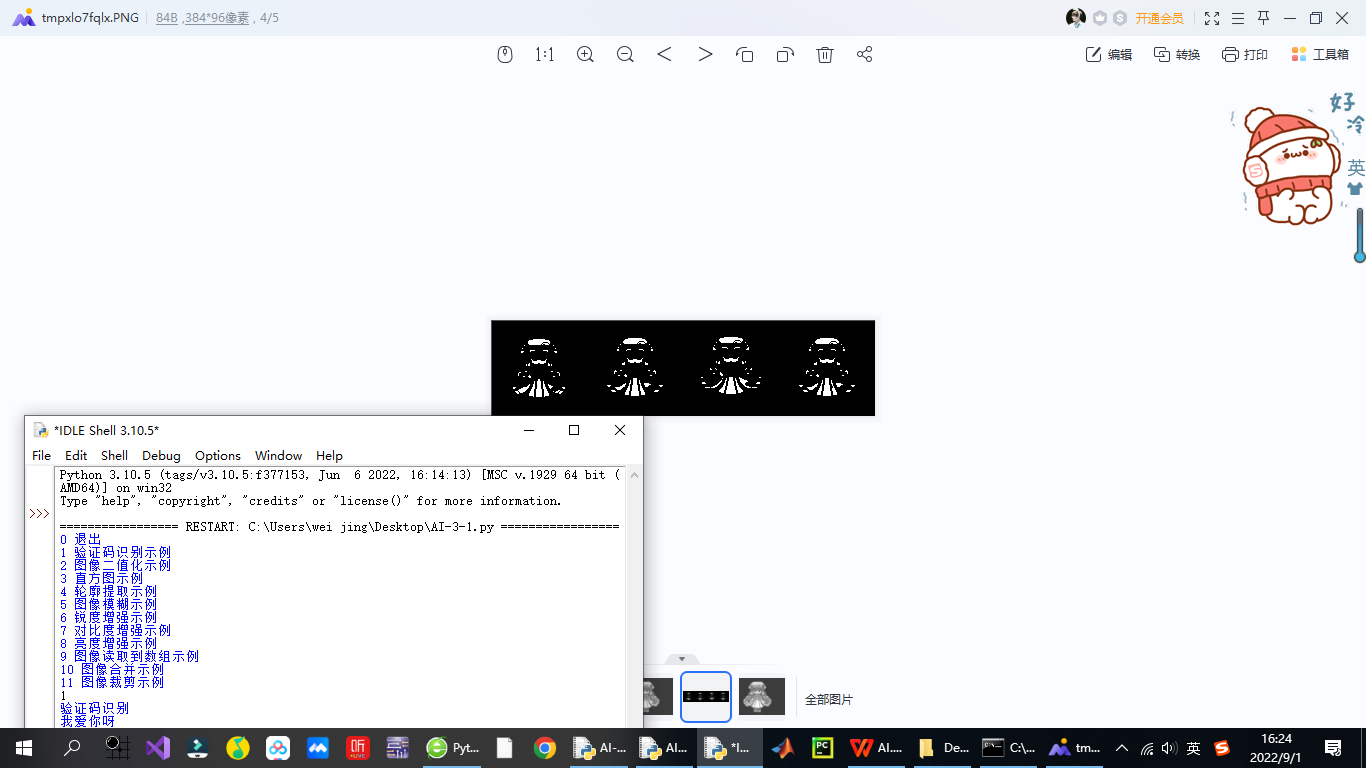


6-2-3

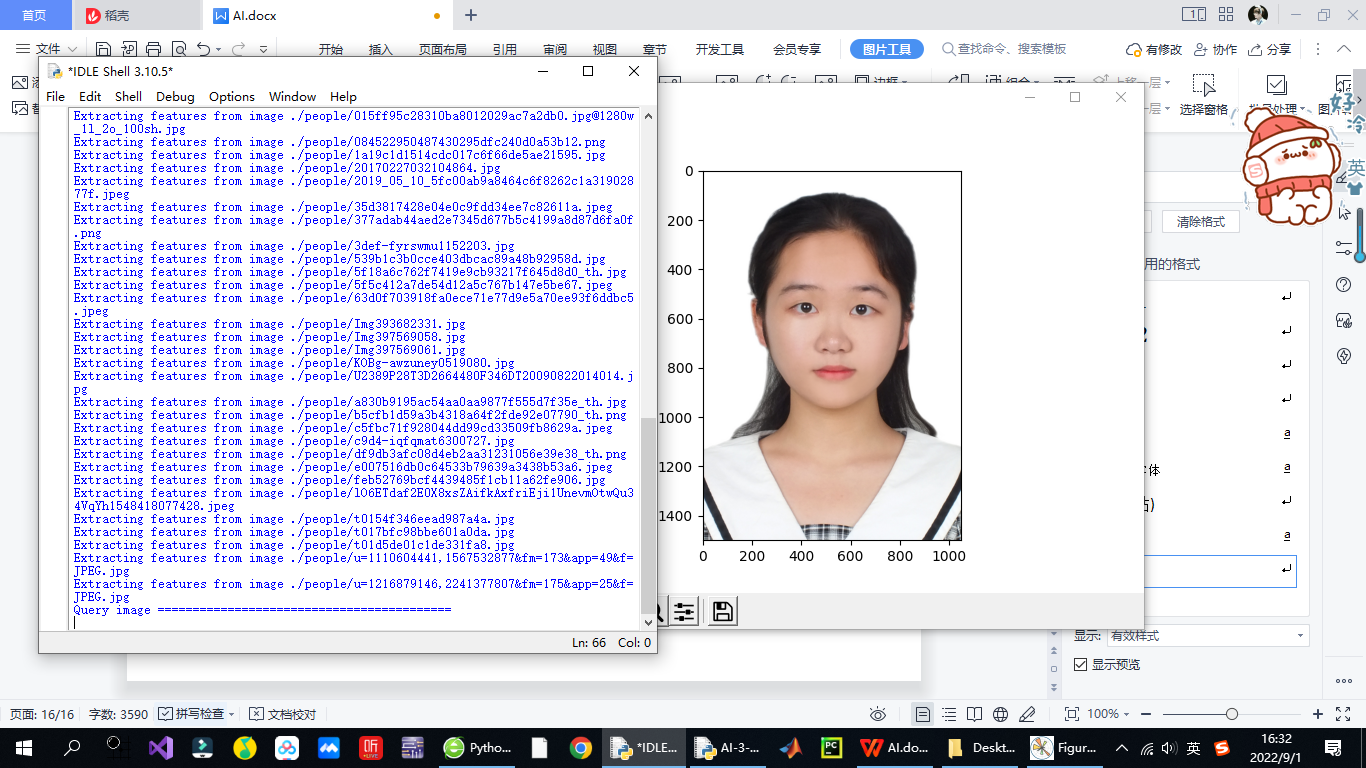


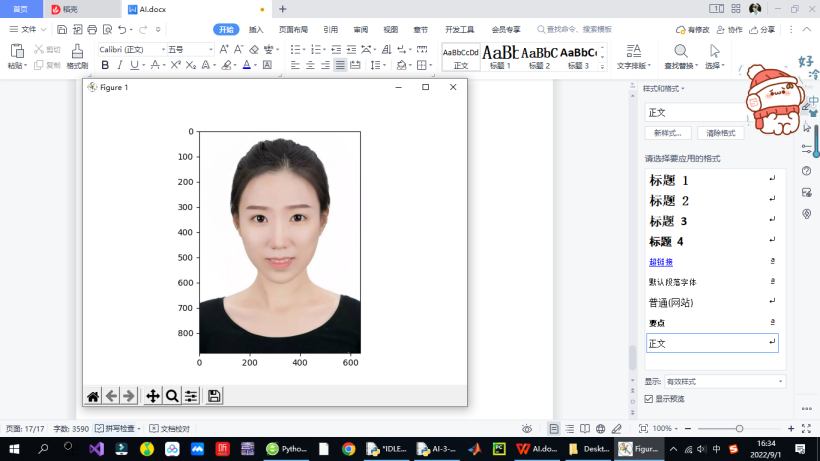
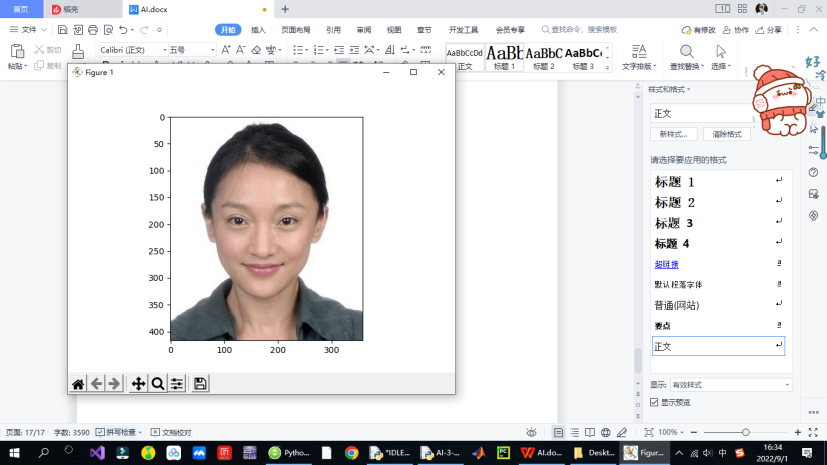
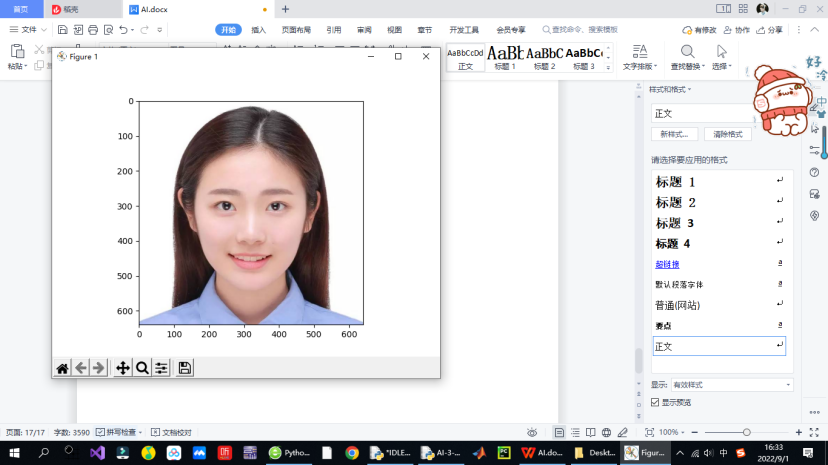
6-3 实验3

6-3-1

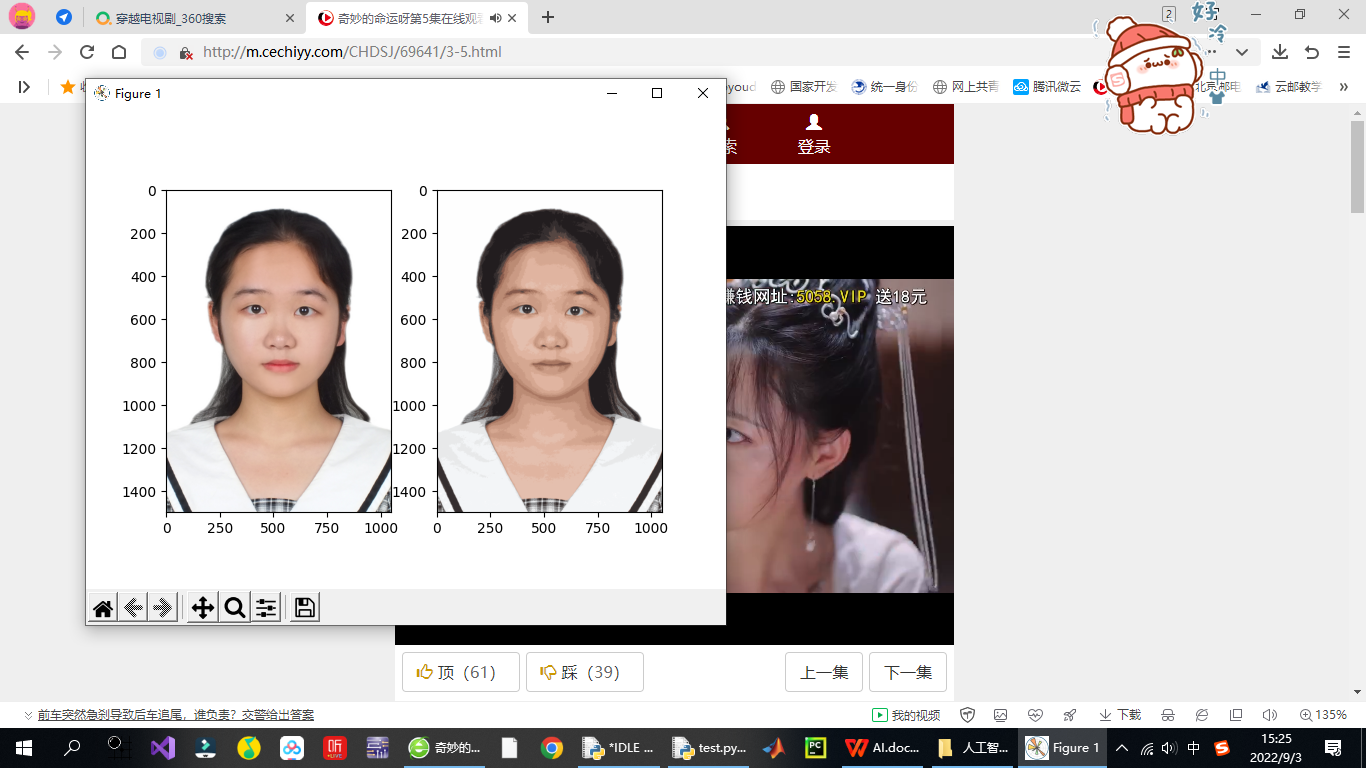


6-3-2





6-4 图片压缩



K=16 K=32 K=160

可以直观感知到K=32时压缩质量就相对不错了，运行的时间也可以接受（K=160的时候运行了越莫两个小时）。如果想获得更精确结果可以计算峰值信噪比。