第一题

我将用两种不同的方法，logistic回归多分类方法和直接使用神经网络进行判别。完成玻璃类型的预测和判别。

In [1]:

```
import pandas as pd
data=pd.read_csv("glass.csv",encoding="UTF-8",header=None)
data
```

Out[1]:

	0	1	2	3	4	5	6	7	8	9
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0	1
...	...	...	...	...	...	...	...	...	...	...
209	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0	7
210	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0	7
211	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0	7
212	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0	7
213	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0	7

214 rows × 10 columns

In [2]:

```
import numpy as np
data=np.array(data)
x, y = data[:,:-1],data[:,-1]
```

In [3]:

```
from sklearn.neural_network import MLPClassifier
m=MLPClassifier(hidden_layer_sizes=(10,),activation='logistic',solver="sgd",max_iter=1000)
m.fit(x,y)
```

Out[3]:

```
MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
              beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(10,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=1000,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='sgd',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

In [4]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y, m.predict(x)))
```

0.35514018691588783

我们可以看到这个结果的准确性是非常差的，于是我在后面打算用多种不同的方法进行分析，并且设置不同的节点，最终画出图象进行分析，在这里我先用logistic回归以及OVR多分类的方法来得到准确率。

In [5]:

```
#logistic多分类
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier
log = LogisticRegression(solver='liblinear')
ovr = OneVsRestClassifier(log)
ovr.fit(x, y)
predict = ovr.predict(x)
print(accuracy_score(y,predict))
```

0.6448598130841121

可以看到，相比较于神经网络，logistic多分类回归的结果还算可以，但准确率也不是特别高。下面我利用多个不同的神经网络方法建立单层感知以及多个感知节点。

In [6]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pylab import *
import matplotlib.cm as cm
import warnings
warnings.filterwarnings(action = 'ignore')
%matplotlib inline
plt.rcParams['font.sans-serif']=['SimHei']  #解决中文显示乱码问题
plt.rcParams['axes.unicode_minus']=False
from mpl_toolkits.mplot3d import Axes3D
from sklearn.datasets import make_circles
from sklearn.model_selection import train_test_split
from sklearn.metrics import zero_one_loss,r2_score,mean_squared_error
import sklearn.neural_network as net
```

In [7]:

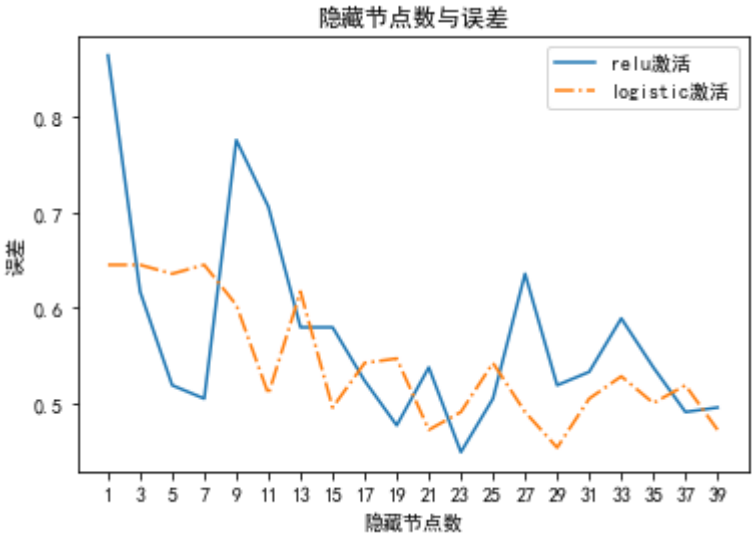
```
nodes=np.arange(1,40,2)
acts=['relu','logistic']
errTrain=np.zeros((len(nodes),2))
for i,node in enumerate(nodes):
    for j,act in enumerate(acts):
        NeuNet=net.MLPClassifier(hidden_layer_sizes=(node,),activation=act,random_state=1)
        NeuNet.fit(x,y)
        errTrain[i,j]=1-NeuNet.score(x,y)
```

In [8]:

```
plt.plot(nodes,errTrain[:,0],label="relu激活",linestyle='-')
plt.plot(nodes,errTrain[:,1],label="logistic激活",linestyle='-.')
plt.title('隐藏节点数与误差')
plt.xlabel('隐藏节点数')
plt.ylabel('误差')
plt.xticks(nodes)
plt.legend()
```

Out[8]:

<matplotlib.legend.Legend at 0x20dc7caaf08>



可以看到，整体来说神经网络的结果并不好。虽然随着隐藏节点数的增加，误差在不断地下降，但是整体地误差基本上是大于0.5的，所以其实效果并不是特别好的。

第二题

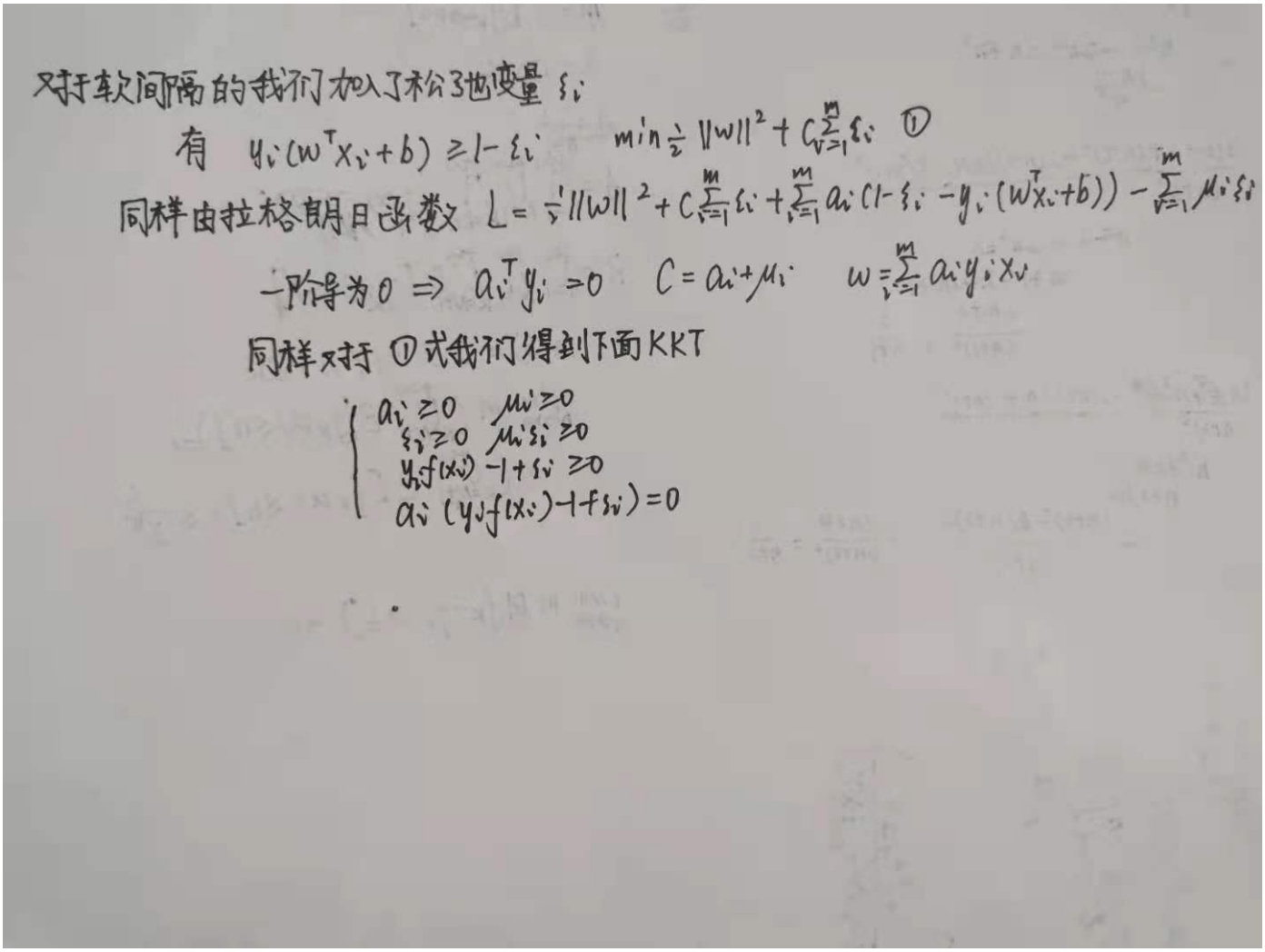
如果是以线性函数作为激活函数，我们可以发现，无论是在输入层还是隐藏层，所递推的式子依旧是原来的线性组合。这样就导致了一个问题，即当最终的预测结果到达最后一层输出层是，其结果依旧是特征x的线性组合。那么此时如果我们对标签y进行无论是梯度下降还是反向传播算法进行分析计算的时候，我们最终得到的其实是一个线性回归的结果，这与普通的线性回归基本等价。所以没有必要使用线性函数作为激活函数。

第三题

因为只有在计算b_h的时候，我们用到了v_{ih}，因此我们有:$\frac{\partial E_k}{\partial v_{ih}} = \frac{\partial E_k}{\partial b_h}\frac{\partial b_h}{\partial v_{ih}}$，而又:$\frac{\partial b_h}{\partial v_{ih}} = \frac{\partial b_h}{\partial a_h}\frac{\partial a_h}{\partial vih} = \frac{\partial b_h}{\partial a_h}x_i$，

故有:$\frac{\partial E_k}{\partial v_{ih}} = \frac{\partial E_k}{\partial b_h}\frac{\partial b_h}{\partial a_h}x_i = -e_h x_i$，此时可以得到最终的式子。

第四题



第五题

当原来的样本是线性可分的，我们基本上可以认为，线性判别分析和线性核的支持向量机的分类效果是相同的。我这里所说的分类效果相同指的是两种算法都会产生一个线性边界，依照这个线性边界，可以将两类清晰的划分。但是它们所产生的线性边界不一定重合，支持向量机要求使得线性边界与数据点离得尽可能地远，这样产生支持向量，可以更加彻底地分清楚。而线性判别分析还依据样本均值点的位置差，因此不一定可以得到分的效果最好的结果，没有支持向量机可以产生的边界的结果。