华南理工大学

《深度学习与神经网络》课程实验报告

实验题目：<u>第四次作业</u>

姓名：<u>何宇航</u>　学号：<u>201830170110</u>

班级：<u>计科 2</u>　组别：________

合作者：________

指导教师：<u>马千里</u>

实验概述

【实验目的及要求】

(1) 数据集选择：在以上两个数据集中任意挑选一个感兴趣的数据集。

(2) 数据预处理：数据集预处理可参考相关项目，不做硬性要求。

(3) 模型：任意选择一个本门课接触到的神经网络进行以上分类任务（逻辑回 归、CNN、RNN...）

(4) 回答以下问题

① 模型有没有出现过拟合现象？引入任意一种正则化方法（如 L2 正则化、 Dropout 等）对结果是否有提升？

② 不同的优化算法对结果是否有影响？引入任意一种其他的优化算法进行比 较。（如 SGD、Adam 等）

【实验环境】

操作系统：Windows win 10　Google Colab

实验内容

【实验过程】

小结

本次实验使用了全连接的神经网络对数据集进行二分类的训练，整体训练的准确度达到**85%**左右。深入地对模型的参数和超参数进行了探讨，发现 **Optimizer**，正则化方法，以及有关的参数对模型都有很大的影响，通过实验我明白了哪些方法能够改进模型的拟合效果，防止拟合，迅速收敛等，对模型调参有了更深刻的认识。最后我根据讨论的结果，结合不同的方法，得到了一个较为优秀的神经网络模型。

指导教师评语及成绩

评语：

成绩：　　　指导教师签名：

批阅日期：

模型基本参数：

本次实验数据内容为第一个数据集，采用全连接 NN 训练

#已经根据作业提供的参考代码对数据进行预处理了

1.神经网络结构

```
model_simple.add(Dense(128,  activation='tanh',  input_shape=train_data[0].shape))
model_simple.add(Dense(64,  activation='tanh'))
model_simple.add(Dense(1,  activation='sigmoid'))
```

2.神经网络训练的参数：

```
history_simple = model_simple.fit(train_data, train_label, epochs=50, batch_size=16, validation_data=(test_data, test_label))
```

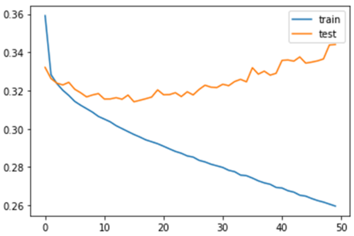
3.神经网络测试集和训练集的划分：

```
# Train - Test split
train_data, test_data, train_label, test_label = train_test_split(adult_data_1hot, yyyy, test_size = 0.25)
```

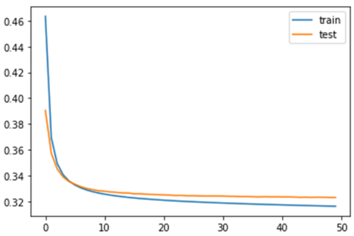
4.其他超参数的选择与模型的优化：

以下是不同模型优化和不同超参数运行得到的结果（如 dropout rate,l1_lambda 等）：

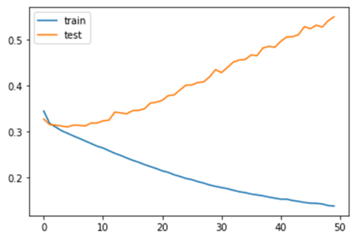
不使用正则化和Droupout
使用最基础的Optimizer = SGD
accuracy: 0.8787
val_accuracy: 0.8439



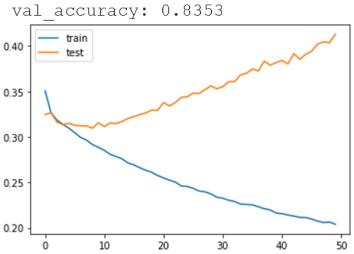
不使用正则化和Droupout
使用Optimizer = adagrad
accuracy: 0.8531
val_accuracy: 0.8496



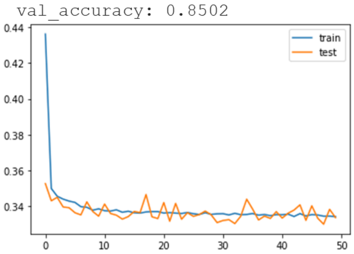
不使用正则化和Droupout
使用Optimizer = adam
accuracy: 0.8536
val_accuracy: 0.8508



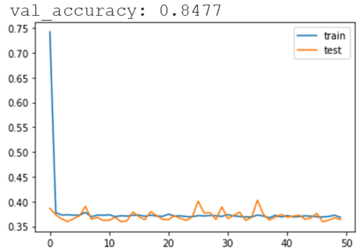
不使用正则化使用Dropout
使用Optimizer = adam
accuracy: 0.9067
val_accuracy: 0.8353



使用l2正则化不使用Dropout
使用Optimizer = adam
accuracy: 0.8484
val_accuracy: 0.8502

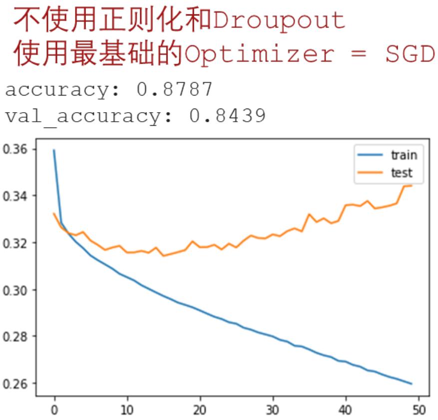


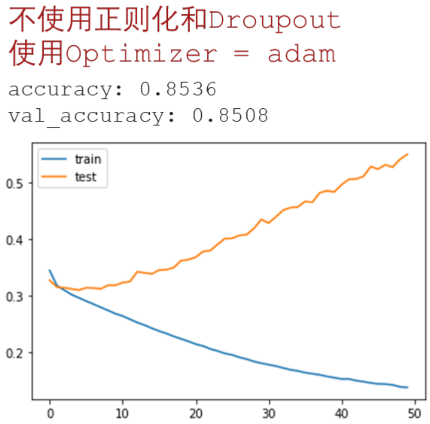
使用l1正则化不使用Dropout
使用Optimizer = adam
accuracy: 0.8486
val_accuracy: 0.8477



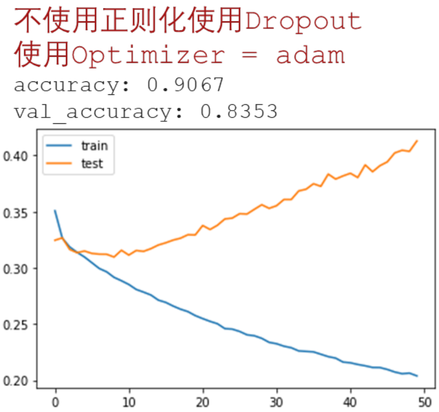
有关模型的讨论和问题回答

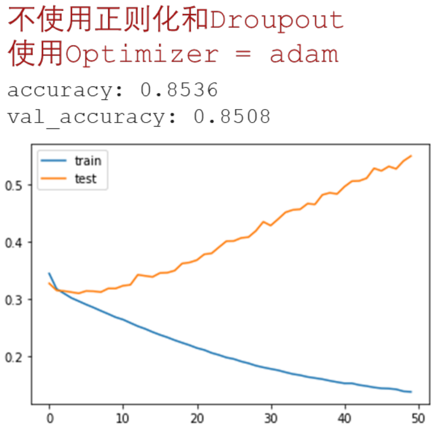
模型有没有出现过拟合现象？引入任意一种正则化方法（如 L2 正则化、 Dropout 等）对结果是否有提升？



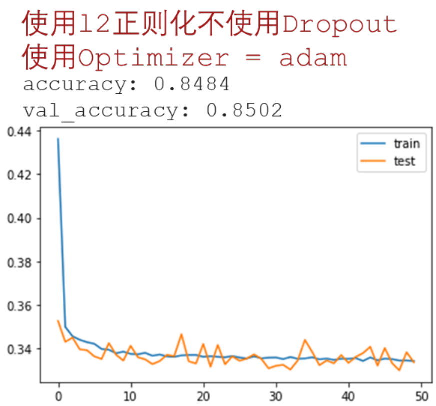


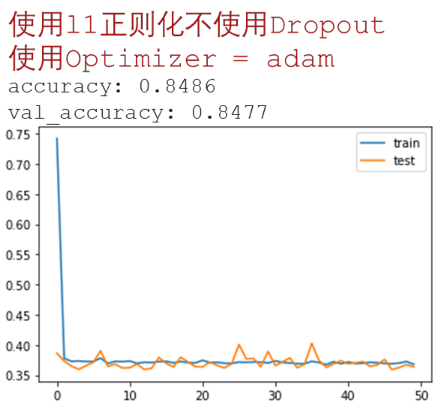
上述两个模型中出现了明显的过拟合现象，使用 **adam** 的过拟合现象要远高于 **SGD**。
所以①考虑引入 **Dropout** 来对提升结果：





上面左图是采用了 **Dropout** 的结果，对比右图可以发现，引入 **Dropout** 对结果有较为明显的提升，但是过拟合效果还是很严重
所以②考虑引入正则化来提升效果



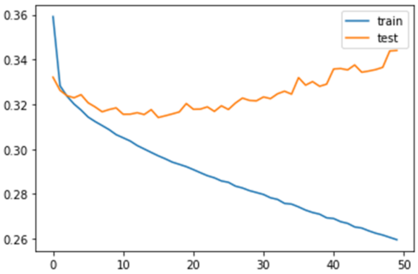


从上面四幅图的对比可以看出，通过引入 **l1 l2** 正则化对模型的收敛速度和防止过拟合的效果都有很大的提升，但是模型在测试集上的收敛效果并不好，通过后续的实验发现，调高 **l1 l2** 正则化的参数后，测试集上的收敛效果提升了很多。总而言之，引入 **l1 l2** 正则化效果比 **Dropout** 显著。

不同的优化算法对结果是否有影响？引入任意一种其他的优化算法进行比 较。（如 SGD、Adam 等）

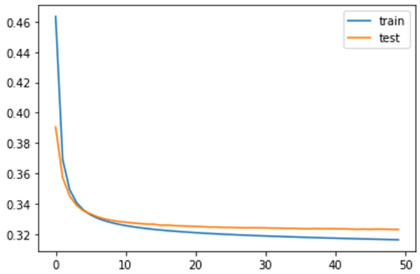
不使用正则化和Droupout
使用最基础的Optimizer = SGD

accuracy: 0.8787
val_accuracy: 0.8439



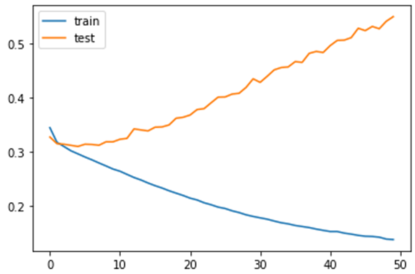
不使用正则化和Droupout
使用Optimizer = adagrad

accuracy: 0.8531
val_accuracy: 0.8496



不使用正则化和Droupout
使用Optimizer = adam

accuracy: 0.8536
val_accuracy: 0.8508

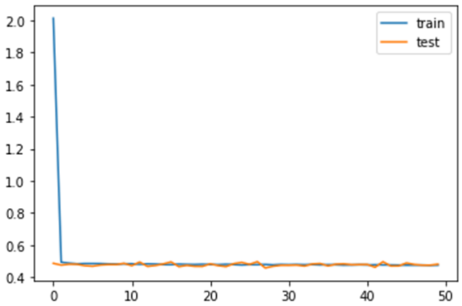


以上列举了不使用正则化和 **Dropout** 时，不同 **Optimizer** 对模型训练的影响，可以看出三种 **Opt** 训练模型时都会出现过拟合的效果，收敛速度：**adam>SGD>adagrad**；收敛效果：**adagrad>adam>SGD;** 过拟合程度：**SGD>adam>adagrad**。整体来看，**adagrad** 效果好，改变 **Opt** 对过拟合、收敛都有影响。

综合以上问题的探讨，我对模型同时采用较好的 **Opt** 和正则化方法对模型进行改善得到了大约 **85%**的准确率，较好的收敛和拟合效果，结果如下：

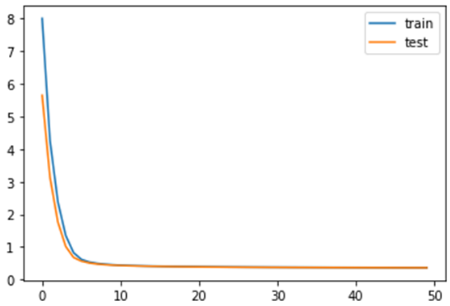
使用l2_l1正则化
使用Dropout
使用Optimizer = adam

accuracy: 0.8415
val_accuracy: 0.8454



使用l1正则化不使用Dropout
使用Optimizer = adagrad

accuracy: 0.8491
val_accuracy: 0.8510



In [35]:
```
import pandas as pd
from IPython.display import Markdown, display
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
import numpy as np
from sklearn import metrics

seed = 7
np.random.seed(seed)

def printmd(string):
    display(Markdown(string))


from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import cross_val_score

from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize


%  matplotlib inline
```

In [2]:
```
#adult = pd.read_csv('adult.csv')

column_names = ['age', 'workclass', 'fnlwgt', 'education', 'educational-num','marita

train = pd.read_csv('adult_data.txt', sep=",\s", header=None, names = column_names,
test = pd.read_csv('adult_test.txt', sep=",\s", header=None, names = column_names, e
test['income'].replace(regex=True,inplace=True,to_replace=r'\.',value=r'')


adult = pd.concat([test,train])
adult.reset_index(inplace = True, drop = True)
```

---

1. Preliminary Data Analysis

In [ ]:
```
# Setting all the categorical columns to type category
for col in set(adult.columns) - set(adult.describe().columns):
    adult[col] = adult[col].astype('category')

printmd('## 1.1. Columns and their types')
print(adult.info())
```

In [4]:
```
# Top 5 records
printmd('## 1.2. Data')
adult.head()
```

1.2. Data

Out[4]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	genc
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	M
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	M
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	M
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	M
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Fem

In [5]:
```
printmd('## 1.3. Summary Statistics')

adult.describe()
```

1.3. Summary Statistics

Out[5]:

	age	fnlwgt	educational-num	capital-gain	capital-loss	hours-per-week
count	48842.000000	4.884200e+04	48842.000000	48842.000000	48842.000000	48842.000000
mean	38.643585	1.896641e+05	10.078089	1079.067626	87.502314	40.422382
std	13.710510	1.056040e+05	2.570973	7452.019058	403.004552	12.391444
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.175505e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.781445e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.376420e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000

In [6]:
```
printmd('## 1.4. Missing values')
for i,j in zip(adult.columns,(adult.values.astype(str) == '?').sum(axis = 0)):
    if j > 0:
        printmd(str(i) + ': ' + str(j) + ' records')
```

1.4. Missing values

workclass: 2799 records

occupation: 2809 records

native-country: 857 records

Treating Missing Values by predicting them

I fill the missing values in each of the three columns by predicting their values. For each of the three columns, I use all the attributes (including 'income') as independent variables and treat that column as the dependent variable, making it a multi-class classification task. I use three classification algorithms, namely, logistic regression, decision trees and random forest to predict the class when the value is missing (in this case a '?'). I then take a majority vote amongst the three classifiers to be the class of the missing value. In case of a tie, I pick the majority class of that column using the entire dataset.

In [7]:

```
# Create one hot encoding of the categorical columns in the data frame.
def oneHotCatVars(df, df_cols):

    df_1 = adult_data = df.drop(columns = df_cols, axis = 1)
    df_2 = pd.get_dummies(df[df_cols])

    return (pd.concat([df_1, df_2], axis=1, join='inner'))
```

In [ ]:

```
printmd('### 1.4.1. Filling in missing values for Attribute workclass')

test_data = adult[(adult.workclass.values == '?')].copy()
test_label = test_data.workclass

train_data = adult[(adult.workclass.values != '?')].copy()
train_label = train_data.workclass

test_data.drop(columns = ['workclass'], inplace = True)
train_data.drop(columns = ['workclass'], inplace = True)

train_data = oneHotCatVars(train_data, train_data.select_dtypes('category').columns)
test_data = oneHotCatVars(test_data, test_data.select_dtypes('category').columns)

log_reg = LogisticRegression()
log_reg.fit(train_data, train_label)
log_reg_pred = log_reg.predict(test_data)


clf = tree.DecisionTreeClassifier()
clf = clf.fit(train_data, train_label)
clf_pred = clf.predict(test_data)

r_forest = RandomForestClassifier(n_estimators=10)
r_forest.fit(train_data, train_label)
r_forest_pred = r_forest.predict(test_data)

majority_class = adult.workclass.value_counts().index[0]

pred_df =  pd.DataFrame({'RFor': r_forest_pred, 'DTree' : clf_pred, 'LogReg' : log_r
overall_pred = pred_df.apply(lambda x: x.value_counts().index[0] if x.value_counts()

adult.loc[(adult.workclass.values == '?'),'workclass'] = overall_pred.values
print(adult.workclass.value_counts())
print(adult.workclass.unique())
```

In [ ]:

```
printmd('### 1.4.2. Filling in missing values for Occupation occupation')

test_data = adult[(adult.occupation.values == '?')].copy()
test_label = test_data.occupation

train_data = adult[(adult.occupation.values != '?')].copy()
train_label = train_data.occupation

test_data.drop(columns = ['occupation'], inplace = True)
```

```
train_data.drop(columns = ['occupation'], inplace = True)

train_data = oneHotCatVars(train_data, train_data.select_dtypes('category').columns)
test_data = oneHotCatVars(test_data, test_data.select_dtypes('category').columns)

log_reg = LogisticRegression()
log_reg.fit(train_data, train_label)
log_reg_pred = log_reg.predict(test_data)


clf = tree.DecisionTreeClassifier()
clf = clf.fit(train_data, train_label)
clf_pred = clf.predict(test_data)

r_forest = RandomForestClassifier(n_estimators=10)
r_forest.fit(train_data, train_label)
r_forest_pred = r_forest.predict(test_data)


majority_class = adult.occupation.value_counts().index[0]

pred_df =  pd.DataFrame({'RFor': r_forest_pred, 'DTree' : clf_pred, 'LogReg' : log_r
overall_pred = pred_df.apply(lambda x: x.value_counts().index[0] if x.value_counts()

adult.loc[(adult.occupation.values == '?'),'occupation'] = overall_pred.values
print(adult.occupation.value_counts())
print(adult.occupation.unique())
```

In [ ]:

```
printmd('### 1.4.3. Filling in missing values for Native Country')

test_data = adult[(adult['native-country'].values == '?')].copy()
test_label = test_data['native-country']

train_data = adult[(adult['native-country'].values != '?')].copy()
train_label = train_data['native-country']

test_data.drop(columns = ['native-country'], inplace = True)
train_data.drop(columns = ['native-country'], inplace = True)

train_data = oneHotCatVars(train_data, train_data.select_dtypes('category').columns)
test_data = oneHotCatVars(test_data, test_data.select_dtypes('category').columns)

log_reg = LogisticRegression()
log_reg.fit(train_data, train_label)
log_reg_pred = log_reg.predict(test_data)


clf = tree.DecisionTreeClassifier()
clf = clf.fit(train_data, train_label)
clf_pred = clf.predict(test_data)

r_forest = RandomForestClassifier(n_estimators=10)
r_forest.fit(train_data, train_label)
r_forest_pred = r_forest.predict(test_data)


majority_class = adult['native-country'].value_counts().index[0]

pred_df =  pd.DataFrame({'RFor': r_forest_pred, 'DTree' : clf_pred, 'LogReg' : log_r
overall_pred = pred_df.apply(lambda x: x.value_counts().index[0] if x.value_counts()

adult.loc[(adult['native-country'].values == '?'),'native-country'] = overall_pred.v
print(adult['native-country'].value_counts())
print(adult['native-country'].unique())
```

In [11]:
```
# Resetting the categories

adult['workclass'] = adult['workclass'].cat.remove_categories('?')
adult['occupation'] = adult['occupation'].cat.remove_categories('?')
adult['native-country'] = adult['native-country'].cat.remove_categories('?')
```

In [12]:
```
printmd('## 1.5. Correlation Matrix')

display(adult.corr())

printmd('We see that none of the columns are highly correlated.')
```

1.5. Correlation Matrix

	age	fnlwgt	educational-num	capital-gain	capital-loss	hours-per-week
age	1.000000	-0.076628	0.030940	0.077229	0.056944	0.071558
fnlwgt	-0.076628	1.000000	-0.038761	-0.003706	-0.004366	-0.013519
educational-num	0.030940	-0.038761	1.000000	0.125146	0.080972	0.143689
capital-gain	0.077229	-0.003706	0.125146	1.000000	-0.031441	0.082157
capital-loss	0.056944	-0.004366	0.080972	-0.031441	1.000000	0.054467
hours-per-week	0.071558	-0.013519	0.143689	0.082157	0.054467	1.000000

We see that none of the columns are highly correlated.

---

3. Data Transformations

3.1. Feature Selection

In [13]:
```
# Remove education and fnlwgt
#adult.drop(columns = ['education','fnlwgt','hours-per-week'], inplace = True)

printmd('* For education level, we have 2 features that convey the same meaning, \'e
        and \'educational-num\'. To avoid the effect of this attribute on the models
        overstated, I am not going to use the categorical education attribute.')
printmd('* I use the categorical Hours work column and drop the \'hour-per-week\' co
printmd('* Also, I chose not to use the \'Fnlwgt\' attribute that is used by the cen
        as the inverse of sampling fraction adjusted for non-response and over or un
        of particular groups. This attribute does not convey individual related mean
```

- For education level, we have 2 features that convey the same meaning, 'education' and 'educational-num'. To avoid the effect of this attribute on the models to be overstated, I am not going to use the categorical education attribute.
- I use the categorical Hours work column and drop the 'hour-per-week' column
- Also, I chose not to use the 'Fnlwgt' attribute that is used by the census, as the inverse of sampling fraction adjusted for non-response and over or under sampling of particular

groups. This attribute does not convey individual related meaning.

3.2 Normalization

In [ ]:
```
printmd('## Box plot')
adult.select_dtypes(exclude = 'category').plot(kind = 'box', figsize = (10,8))
```

In [15]:
```
printmd ('Normalization happens on the training dataset, by removing the mean and \
        scaling to unit variance. These values are stored and then later applied  \
        to the test data before the test data is passed to the model for prediction.
```

Normalization happens on the training dataset, by removing the mean and scaling to unit variance. These values are stored and then later applied to the test data before the test data is passed to the model for prediction.

---

4. Model Development & Classification

4.1. Data Preparation'

One-hot encoding is the process of representing multi-class categorical features as binary features, one for each class. Although this process increases the dimensionality of the dataset, classification algorithms tend to work better on this format of data.

I use one-hot encoding to represent all the categorical features in the dataset.

In [16]:
```
# Data Prep
adult_data = adult.drop(columns = ['income'])
adult_label = adult.income


adult_cat_1hot = pd.get_dummies(adult_data.select_dtypes('category'))
adult_non_cat = adult_data.select_dtypes(exclude = 'category')

adult_data_1hot = pd.concat([adult_non_cat, adult_cat_1hot], axis=1, join='inner')
```

In [17]:
```
def vectorize_sequences(squences, dimension=10000):
    """
    @函数功能:将序列向量化，初始化全0的序列，在单词索引对应的位置上置1
    """
    resluts = np.zeros((len(squences), dimension))
    for i, sequence in enumerate(squences):
        resluts[i, sequence] = 1
    return resluts
```

In [18]:
```
def encode_label(label):
  if label == '>50K':
    return 1
  return 0
```

In [19]:
```
adult_label.values
```

Out[19]:
```
['<=50K', '<=50K', '>50K', '>50K', '<=50K', ..., '<=50K', '>50K', '<=50K', '<=50K',
'>50K']
```

```
Length: 48842
Categories (2, object): ['<=50K', '>50K']
```

In [20]:
```
yyyy = np.array( [ encode_label(i) for i in adult_label.values ] )
```

In [21]:
```
# Train - Test split
train_data, test_data, train_label, test_label = train_test_split(adult_data_1hot, y
```

In [22]:
```
train_data
```

Out[22]:

	age	fnlwgt	educational-num	capital-gain	capital-loss	hours-per-week	workclass_Federal-gov	workclass_Local-gov	w
32272	28	192588	9	0	0	40	0	0	
42251	31	208881	10	0	0	40	0	0	
44314	61	197286	4	0	0	40	0	0	
10896	27	54897	9	0	0	40	0	0	
287	46	157857	10	0	0	40	0	0	
...	...	...	...	...	...	...	...	...	
23802	28	150309	9	0	0	45	0	0	
45851	38	27408	9	0	0	50	0	0	
29788	34	236543	9	0	0	40	0	0	
342	31	179415	3	0	0	40	0	0	
4933	23	273010	9	0	0	40	0	0	

36631 rows × 105 columns

In [23]:
```
# Normalization
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# Fitting only on training data
scaler.fit(train_data)
train_data = scaler.transform(train_data)

# Applying same transformation to test data
test_data = scaler.transform(test_data)
```

In [24]:
```
train_label
```

Out[24]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [25]:
```
train_data[0].shape
```

Out[25]:
```
(105,)
```

In [26]:
```
import tensorflow as tf
import numpy as np
import pandas as pd
# from tensorflow import keras
```

```
from keras.models import Sequential
from keras.layers import Dense,Dropout
```

In [45]:

```
""" 构建神经网络
不使用正则化和Droupout
使用最基础的Optimizer = SGD
loss = 'binary_crossentropy'
最后输出层使用sigmoid激活函数
"""
model_simple = Sequential()
# keras.regularizers.l1(lambda)
# keras.regularizers.l2(lambda)
# keras.regularizers.l1_l2(l1=lambda1, l2=lambda2)
model_simple.add(Dense(128, activation='tanh', input_shape=train_data[0].shape))
model_simple.add(Dense(64, activation='tanh'))
model_simple.add(Dense(1, activation='sigmoid'))
```

In [46]:

```
model_simple.compile(optimizer='SGD',loss = 'binary_crossentropy',metrics=['accuracy
history_simple = model_simple.fit(train_data, train_label, epochs=50, batch_size=16,
```

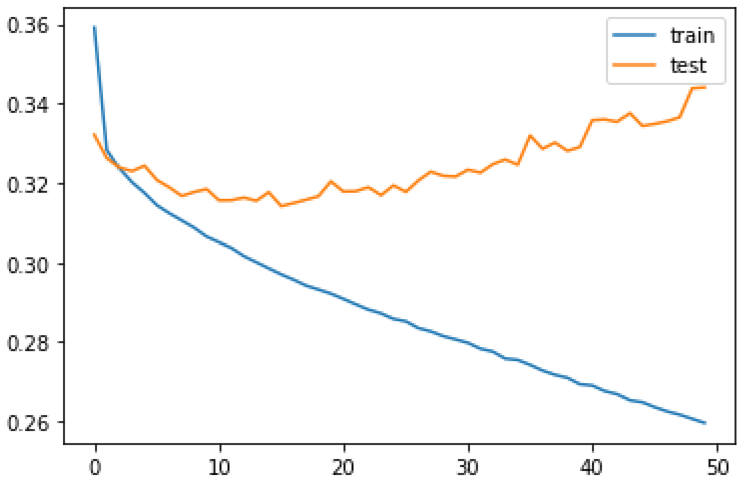
```
Epoch 1/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3591 - accuracy:
0.8336 - val_loss: 0.3321 - val_accuracy: 0.8455
Epoch 2/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3283 - accuracy:
0.8461 - val_loss: 0.3263 - val_accuracy: 0.8466
Epoch 3/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3236 - accuracy:
0.8489 - val_loss: 0.3238 - val_accuracy: 0.8473
Epoch 4/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3201 - accuracy:
0.8508 - val_loss: 0.3229 - val_accuracy: 0.8523
Epoch 5/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3174 - accuracy:
0.8508 - val_loss: 0.3243 - val_accuracy: 0.8467
Epoch 6/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3143 - accuracy:
0.8532 - val_loss: 0.3207 - val_accuracy: 0.8491
Epoch 7/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3123 - accuracy:
0.8532 - val_loss: 0.3188 - val_accuracy: 0.8500
Epoch 8/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3105 - accuracy:
0.8550 - val_loss: 0.3167 - val_accuracy: 0.8528
Epoch 9/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3087 - accuracy:
0.8560 - val_loss: 0.3177 - val_accuracy: 0.8509
Epoch 10/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3065 - accuracy:
0.8560 - val_loss: 0.3184 - val_accuracy: 0.8517
Epoch 11/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3051 - accuracy:
0.8563 - val_loss: 0.3156 - val_accuracy: 0.8537
Epoch 12/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3036 - accuracy:
0.8571 - val_loss: 0.3156 - val_accuracy: 0.8537
Epoch 13/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3016 - accuracy:
0.8595 - val_loss: 0.3163 - val_accuracy: 0.8543
Epoch 14/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3001 - accuracy:
0.8601 - val_loss: 0.3155 - val_accuracy: 0.8552
Epoch 15/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2985 - accuracy:
0.8611 - val_loss: 0.3176 - val_accuracy: 0.8530
Epoch 16/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2970 - accuracy:
```

```
0.8618 - val_loss: 0.3141 - val_accuracy: 0.8557
Epoch 17/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2957 - accuracy:
0.8627 - val_loss: 0.3149 - val_accuracy: 0.8541
Epoch 18/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2942 - accuracy:
0.8627 - val_loss: 0.3157 - val_accuracy: 0.8569
Epoch 19/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2932 - accuracy:
0.8624 - val_loss: 0.3166 - val_accuracy: 0.8550
Epoch 20/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2922 - accuracy:
0.8647 - val_loss: 0.3204 - val_accuracy: 0.8522
Epoch 21/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2909 - accuracy:
0.8656 - val_loss: 0.3178 - val_accuracy: 0.8526
Epoch 22/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2895 - accuracy:
0.8644 - val_loss: 0.3179 - val_accuracy: 0.8541
Epoch 23/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2882 - accuracy:
0.8663 - val_loss: 0.3189 - val_accuracy: 0.8535
Epoch 24/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2872 - accuracy:
0.8676 - val_loss: 0.3169 - val_accuracy: 0.8560
Epoch 25/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2858 - accuracy:
0.8663 - val_loss: 0.3193 - val_accuracy: 0.8534
Epoch 26/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2852 - accuracy:
0.8655 - val_loss: 0.3177 - val_accuracy: 0.8523
Epoch 27/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2835 - accuracy:
0.8685 - val_loss: 0.3206 - val_accuracy: 0.8557
Epoch 28/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2827 - accuracy:
0.8691 - val_loss: 0.3228 - val_accuracy: 0.8522
Epoch 29/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2815 - accuracy:
0.8692 - val_loss: 0.3218 - val_accuracy: 0.8533
Epoch 30/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2807 - accuracy:
0.8690 - val_loss: 0.3216 - val_accuracy: 0.8513
Epoch 31/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2798 - accuracy:
0.8708 - val_loss: 0.3233 - val_accuracy: 0.8516
Epoch 32/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2783 - accuracy:
0.8707 - val_loss: 0.3225 - val_accuracy: 0.8516
Epoch 33/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2776 - accuracy:
0.8710 - val_loss: 0.3247 - val_accuracy: 0.8524
Epoch 34/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2758 - accuracy:
0.8714 - val_loss: 0.3259 - val_accuracy: 0.8496
Epoch 35/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2755 - accuracy:
0.8742 - val_loss: 0.3245 - val_accuracy: 0.8518
Epoch 36/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2743 - accuracy:
0.8726 - val_loss: 0.3319 - val_accuracy: 0.8447
Epoch 37/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2728 - accuracy:
0.8740 - val_loss: 0.3286 - val_accuracy: 0.8504
Epoch 38/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2718 - accuracy:
0.8741 - val_loss: 0.3301 - val_accuracy: 0.8489
Epoch 39/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2710 - accuracy:
```

```
0.8738 - val_loss: 0.3280 - val_accuracy: 0.8502
Epoch 40/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2694 - accuracy:
0.8750 - val_loss: 0.3290 - val_accuracy: 0.8517
Epoch 41/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2691 - accuracy:
0.8767 - val_loss: 0.3358 - val_accuracy: 0.8464
Epoch 42/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2677 - accuracy:
0.8762 - val_loss: 0.3360 - val_accuracy: 0.8469
Epoch 43/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2669 - accuracy:
0.8754 - val_loss: 0.3354 - val_accuracy: 0.8476
Epoch 44/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2653 - accuracy:
0.8764 - val_loss: 0.3376 - val_accuracy: 0.8466
Epoch 45/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2648 - accuracy:
0.8772 - val_loss: 0.3343 - val_accuracy: 0.8494
Epoch 46/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2636 - accuracy:
0.8769 - val_loss: 0.3348 - val_accuracy: 0.8505
Epoch 47/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2625 - accuracy:
0.8778 - val_loss: 0.3355 - val_accuracy: 0.8531
Epoch 48/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2617 - accuracy:
0.8777 - val_loss: 0.3365 - val_accuracy: 0.8511
Epoch 49/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2607 - accuracy:
0.8797 - val_loss: 0.3438 - val_accuracy: 0.8453
Epoch 50/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.2596 - accuracy:
0.8787 - val_loss: 0.3440 - val_accuracy: 0.8439
```

In [47]:

```
plt.plot(history_simple.history['loss'], label='train')
plt.plot(history_simple.history['val_loss'], label='test')
plt.legend()
plt.show()
```



In [48]:

```
""" 构建神经网络
不使用正则化和Droupout
使用Optimizer = adagrad
loss = 'binary_crossentropy'
最后输出层使用sigmoid激活函数
"""
model_adagrad = Sequential()
# keras.regularizers.l1(lambda)
# keras.regularizers.l2(lambda)
# keras.regularizers.l1_l2(l1=lambda1, l2=lambda2)
model_adagrad.add(Dense(128, activation='tanh', input_shape=train_data[0].shape))
```

```
model_adagrad.add(Dense(64, activation='tanh'))
model_adagrad.add(Dense(1, activation='sigmoid'))
```

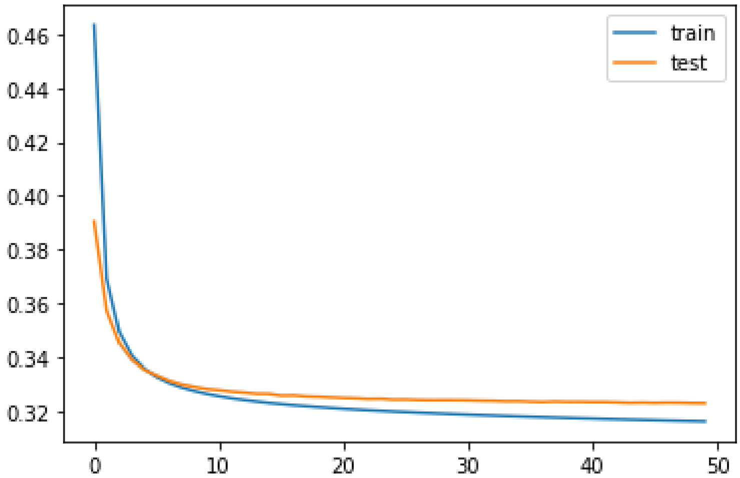
In [49]:
```
model_adagrad.compile(optimizer='Adagrad',loss = 'binary_crossentropy',metrics=['acc
history_adagrad = model_adagrad.fit(train_data, train_label, epochs=50, batch_size=1
```

```
Epoch 1/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.4634 - accuracy:
0.7786 - val_loss: 0.3905 - val_accuracy: 0.8258
Epoch 2/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3693 - accuracy:
0.8360 - val_loss: 0.3574 - val_accuracy: 0.8398
Epoch 3/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3494 - accuracy:
0.8419 - val_loss: 0.3452 - val_accuracy: 0.8428
Epoch 4/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3407 - accuracy:
0.8438 - val_loss: 0.3390 - val_accuracy: 0.8452
Epoch 5/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3358 - accuracy:
0.8461 - val_loss: 0.3354 - val_accuracy: 0.8464
Epoch 6/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3327 - accuracy:
0.8463 - val_loss: 0.3334 - val_accuracy: 0.8469
Epoch 7/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3305 - accuracy:
0.8469 - val_loss: 0.3314 - val_accuracy: 0.8475
Epoch 8/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3289 - accuracy:
0.8476 - val_loss: 0.3301 - val_accuracy: 0.8483
Epoch 9/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3276 - accuracy:
0.8482 - val_loss: 0.3292 - val_accuracy: 0.8487
Epoch 10/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3265 - accuracy:
0.8488 - val_loss: 0.3283 - val_accuracy: 0.8488
Epoch 11/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3257 - accuracy:
0.8491 - val_loss: 0.3279 - val_accuracy: 0.8487
Epoch 12/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3249 - accuracy:
0.8491 - val_loss: 0.3274 - val_accuracy: 0.8490
Epoch 13/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3243 - accuracy:
0.8493 - val_loss: 0.3271 - val_accuracy: 0.8490
Epoch 14/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3237 - accuracy:
0.8497 - val_loss: 0.3266 - val_accuracy: 0.8487
Epoch 15/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3232 - accuracy:
0.8497 - val_loss: 0.3266 - val_accuracy: 0.8485
Epoch 16/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3228 - accuracy:
0.8497 - val_loss: 0.3259 - val_accuracy: 0.8483
Epoch 17/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3223 - accuracy:
0.8505 - val_loss: 0.3260 - val_accuracy: 0.8484
Epoch 18/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3220 - accuracy:
0.8502 - val_loss: 0.3256 - val_accuracy: 0.8485
Epoch 19/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3216 - accuracy:
0.8509 - val_loss: 0.3255 - val_accuracy: 0.8483
Epoch 20/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3213 - accuracy:
0.8506 - val_loss: 0.3253 - val_accuracy: 0.8489
Epoch 21/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3210 - accuracy:
0.8509 - val_loss: 0.3251 - val_accuracy: 0.8488
```

```
Epoch 22/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3207 - accuracy:
0.8511 - val_loss: 0.3249 - val_accuracy: 0.8488
Epoch 23/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3205 - accuracy:
0.8515 - val_loss: 0.3247 - val_accuracy: 0.8492
Epoch 24/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3202 - accuracy:
0.8516 - val_loss: 0.3247 - val_accuracy: 0.8487
Epoch 25/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3200 - accuracy:
0.8516 - val_loss: 0.3244 - val_accuracy: 0.8490
Epoch 26/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3198 - accuracy:
0.8515 - val_loss: 0.3244 - val_accuracy: 0.8495
Epoch 27/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3196 - accuracy:
0.8514 - val_loss: 0.3243 - val_accuracy: 0.8495
Epoch 28/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3194 - accuracy:
0.8517 - val_loss: 0.3242 - val_accuracy: 0.8496
Epoch 29/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3192 - accuracy:
0.8518 - val_loss: 0.3242 - val_accuracy: 0.8493
Epoch 30/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3190 - accuracy:
0.8517 - val_loss: 0.3242 - val_accuracy: 0.8497
Epoch 31/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3188 - accuracy:
0.8518 - val_loss: 0.3242 - val_accuracy: 0.8494
Epoch 32/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3187 - accuracy:
0.8519 - val_loss: 0.3240 - val_accuracy: 0.8498
Epoch 33/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3185 - accuracy:
0.8522 - val_loss: 0.3240 - val_accuracy: 0.8496
Epoch 34/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3183 - accuracy:
0.8521 - val_loss: 0.3238 - val_accuracy: 0.8501
Epoch 35/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3182 - accuracy:
0.8521 - val_loss: 0.3238 - val_accuracy: 0.8496
Epoch 36/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3180 - accuracy:
0.8523 - val_loss: 0.3237 - val_accuracy: 0.8501
Epoch 37/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3179 - accuracy:
0.8523 - val_loss: 0.3235 - val_accuracy: 0.8499
Epoch 38/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3177 - accuracy:
0.8523 - val_loss: 0.3237 - val_accuracy: 0.8497
Epoch 39/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3176 - accuracy:
0.8523 - val_loss: 0.3236 - val_accuracy: 0.8494
Epoch 40/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3175 - accuracy:
0.8527 - val_loss: 0.3236 - val_accuracy: 0.8489
Epoch 41/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3174 - accuracy:
0.8524 - val_loss: 0.3235 - val_accuracy: 0.8496
Epoch 42/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3172 - accuracy:
0.8526 - val_loss: 0.3235 - val_accuracy: 0.8495
Epoch 43/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3171 - accuracy:
0.8528 - val_loss: 0.3234 - val_accuracy: 0.8498
Epoch 44/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3170 - accuracy:
0.8528 - val_loss: 0.3232 - val_accuracy: 0.8498
```

```
Epoch 45/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3169 - accuracy:
0.8533 - val_loss: 0.3233 - val_accuracy: 0.8497
Epoch 46/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3168 - accuracy:
0.8527 - val_loss: 0.3232 - val_accuracy: 0.8497
Epoch 47/50
2290/2290 [==============================] - 5s 2ms/step - loss: 0.3166 - accuracy:
0.8532 - val_loss: 0.3233 - val_accuracy: 0.8493
Epoch 48/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3165 - accuracy:
0.8530 - val_loss: 0.3232 - val_accuracy: 0.8495
Epoch 49/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3164 - accuracy:
0.8532 - val_loss: 0.3231 - val_accuracy: 0.8495
Epoch 50/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3163 - accuracy:
0.8531 - val_loss: 0.3231 - val_accuracy: 0.8496
```

In [50]:
```
plt.plot(history_adagrad.history['loss'], label='train')
plt.plot(history_adagrad.history['val_loss'], label='test')
plt.legend()
plt.show()
```



In [74]:
```
""" 构建神经网络
不使用正则化和Droupout
使用Optimizer = adam
loss = 'binary_crossentropy'
最后输出层使用sigmoid激活函数
"""
model_adam = Sequential()
# keras.regularizers.l1(lambda)
# keras.regularizers.l2(lambda)
# keras.regularizers.l1_l2(l1=lambda1, l2=lambda2)
model_adam.add(Dense(128, activation='tanh', input_shape=train_data[0].shape))
model_adam.add(Dense(64, activation='tanh'))
model_adam.add(Dense(1, activation='sigmoid'))
```

In [75]:
```
model_adam.compile(optimizer='Adam',loss = 'binary_crossentropy',metrics=['accuracy'
history_adam = model_adam.fit(train_data, train_label, epochs=50, batch_size=16, val
```

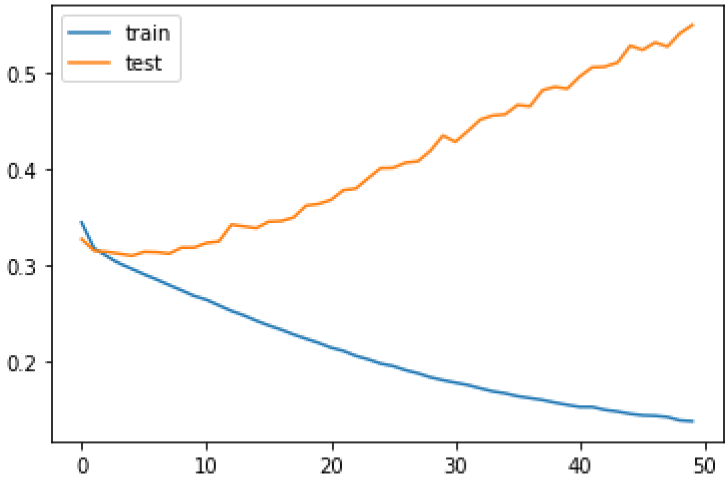
```
Epoch 1/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3439 - accuracy:
0.8386 - val_loss: 0.3267 - val_accuracy: 0.8447
Epoch 2/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3171 - accuracy:
0.8513 - val_loss: 0.3145 - val_accuracy: 0.8510
Epoch 3/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3093 - accuracy:
0.8543 - val_loss: 0.3134 - val_accuracy: 0.8519
```

```
Epoch 4/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3014 - accuracy:
0.8591 - val_loss: 0.3114 - val_accuracy: 0.8517
Epoch 5/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2957 - accuracy:
0.8610 - val_loss: 0.3094 - val_accuracy: 0.8547
Epoch 6/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2899 - accuracy:
0.8655 - val_loss: 0.3134 - val_accuracy: 0.8564
Epoch 7/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2846 - accuracy:
0.8682 - val_loss: 0.3130 - val_accuracy: 0.8540
Epoch 8/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2789 - accuracy:
0.8707 - val_loss: 0.3116 - val_accuracy: 0.8583
Epoch 9/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2734 - accuracy:
0.8725 - val_loss: 0.3178 - val_accuracy: 0.8527
Epoch 10/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2677 - accuracy:
0.8751 - val_loss: 0.3178 - val_accuracy: 0.8551
Epoch 11/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2636 - accuracy:
0.8773 - val_loss: 0.3226 - val_accuracy: 0.8510
Epoch 12/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2578 - accuracy:
0.8803 - val_loss: 0.3242 - val_accuracy: 0.8516
Epoch 13/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2521 - accuracy:
0.8819 - val_loss: 0.3417 - val_accuracy: 0.8379
Epoch 14/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2475 - accuracy:
0.8848 - val_loss: 0.3398 - val_accuracy: 0.8439
Epoch 15/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2420 - accuracy:
0.8869 - val_loss: 0.3380 - val_accuracy: 0.8498
Epoch 16/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2370 - accuracy:
0.8910 - val_loss: 0.3448 - val_accuracy: 0.8471
Epoch 17/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2327 - accuracy:
0.8931 - val_loss: 0.3454 - val_accuracy: 0.8469
Epoch 18/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2277 - accuracy:
0.8945 - val_loss: 0.3490 - val_accuracy: 0.8475
Epoch 19/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2232 - accuracy:
0.8977 - val_loss: 0.3613 - val_accuracy: 0.8402
Epoch 20/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2190 - accuracy:
0.8989 - val_loss: 0.3631 - val_accuracy: 0.8424
Epoch 21/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2140 - accuracy:
0.9019 - val_loss: 0.3673 - val_accuracy: 0.8425
Epoch 22/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2106 - accuracy:
0.9025 - val_loss: 0.3774 - val_accuracy: 0.8407
Epoch 23/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2055 - accuracy:
0.9056 - val_loss: 0.3791 - val_accuracy: 0.8415
Epoch 24/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2018 - accuracy:
0.9068 - val_loss: 0.3898 - val_accuracy: 0.8365
Epoch 25/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.1975 - accuracy:
0.9104 - val_loss: 0.4003 - val_accuracy: 0.8376
Epoch 26/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1949 - accuracy:
0.9120 - val_loss: 0.4006 - val_accuracy: 0.8356
```

```
Epoch 27/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1906 - accuracy:
0.9120 - val_loss: 0.4059 - val_accuracy: 0.8356
Epoch 28/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1874 - accuracy:
0.9136 - val_loss: 0.4074 - val_accuracy: 0.8342
Epoch 29/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1832 - accuracy:
0.9172 - val_loss: 0.4183 - val_accuracy: 0.8319
Epoch 30/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.1803 - accuracy:
0.9181 - val_loss: 0.4341 - val_accuracy: 0.8316
Epoch 31/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1777 - accuracy:
0.9176 - val_loss: 0.4275 - val_accuracy: 0.8410
Epoch 32/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1752 - accuracy:
0.9201 - val_loss: 0.4387 - val_accuracy: 0.8357
Epoch 33/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1718 - accuracy:
0.9219 - val_loss: 0.4505 - val_accuracy: 0.8292
Epoch 34/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1687 - accuracy:
0.9224 - val_loss: 0.4548 - val_accuracy: 0.8295
Epoch 35/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.1666 - accuracy:
0.9245 - val_loss: 0.4560 - val_accuracy: 0.8333
Epoch 36/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.1636 - accuracy:
0.9255 - val_loss: 0.4658 - val_accuracy: 0.8280
Epoch 37/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1617 - accuracy:
0.9262 - val_loss: 0.4645 - val_accuracy: 0.8304
Epoch 38/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1597 - accuracy:
0.9272 - val_loss: 0.4811 - val_accuracy: 0.8279
Epoch 39/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1570 - accuracy:
0.9291 - val_loss: 0.4846 - val_accuracy: 0.8322
Epoch 40/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1546 - accuracy:
0.9307 - val_loss: 0.4827 - val_accuracy: 0.8262
Epoch 41/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1524 - accuracy:
0.9312 - val_loss: 0.4953 - val_accuracy: 0.8305
Epoch 42/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1525 - accuracy:
0.9313 - val_loss: 0.5049 - val_accuracy: 0.8301
Epoch 43/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1495 - accuracy:
0.9328 - val_loss: 0.5054 - val_accuracy: 0.8279
Epoch 44/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.1476 - accuracy:
0.9336 - val_loss: 0.5101 - val_accuracy: 0.8314
Epoch 45/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.1453 - accuracy:
0.9344 - val_loss: 0.5273 - val_accuracy: 0.8286
Epoch 46/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1438 - accuracy:
0.9355 - val_loss: 0.5229 - val_accuracy: 0.8293
Epoch 47/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1433 - accuracy:
0.9364 - val_loss: 0.5306 - val_accuracy: 0.8225
Epoch 48/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.1420 - accuracy:
0.9350 - val_loss: 0.5265 - val_accuracy: 0.8252
Epoch 49/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.1385 - accuracy:
0.9384 - val_loss: 0.5402 - val_accuracy: 0.8279
```

```
Epoch 50/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.1376 - accuracy:
0.9383 - val_loss: 0.5488 - val_accuracy: 0.8236
```

In [76]:
```python
plt.plot(history_adam.history['loss'], label='train')
plt.plot(history_adam.history['val_loss'], label='test')
plt.legend()
plt.show()
```



In [52]:
```python
""" 构建神经网络
不使用正则化
使用Dropout
使用Optimizer = adam
loss = 'binary_crossentropy'
最后输出层使用sigmoid激活函数
"""
model_dropout = Sequential()
# keras.regularizers.l1(lambda)
# keras.regularizers.l2(lambda)
# keras.regularizers.l1_l2(l1=lambda1, l2=lambda2)
model_dropout.add(Dense(128, activation='tanh', input_shape=train_data[0].shape))
model_dropout.add(Dense(64, activation='tanh'))
model_dropout.add(Dropout(0.25))
model_dropout.add(Dense(1, activation='sigmoid'))
```

In [53]:
```python
model_dropout.compile(optimizer='adam',loss = 'binary_crossentropy',metrics=['accura
history_dropout = model_dropout.fit(train_data, train_label, epochs=50, batch_size=1
```

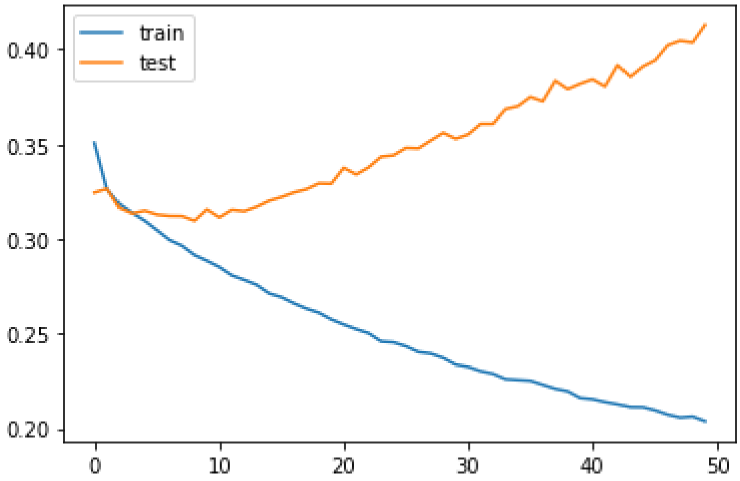
```
Epoch 1/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3508 - accuracy:
0.8347 - val_loss: 0.3246 - val_accuracy: 0.8505
Epoch 2/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3264 - accuracy:
0.8465 - val_loss: 0.3267 - val_accuracy: 0.8448
Epoch 3/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3186 - accuracy:
0.8525 - val_loss: 0.3164 - val_accuracy: 0.8529
Epoch 4/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.3138 - accuracy:
0.8541 - val_loss: 0.3135 - val_accuracy: 0.8509
Epoch 5/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3097 - accuracy:
0.8554 - val_loss: 0.3150 - val_accuracy: 0.8503
Epoch 6/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3046 - accuracy:
0.8583 - val_loss: 0.3129 - val_accuracy: 0.8556
Epoch 7/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2995 - accuracy:
0.8618 - val_loss: 0.3122 - val_accuracy: 0.8546
Epoch 8/50
```

```
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2965 - accuracy:
0.8616 - val_loss: 0.3121 - val_accuracy: 0.8578
Epoch 9/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2916 - accuracy:
0.8648 - val_loss: 0.3096 - val_accuracy: 0.8574
Epoch 10/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2885 - accuracy:
0.8639 - val_loss: 0.3157 - val_accuracy: 0.8501
Epoch 11/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2852 - accuracy:
0.8679 - val_loss: 0.3114 - val_accuracy: 0.8565
Epoch 12/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2808 - accuracy:
0.8694 - val_loss: 0.3154 - val_accuracy: 0.8573
Epoch 13/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2784 - accuracy:
0.8692 - val_loss: 0.3147 - val_accuracy: 0.8550
Epoch 14/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2758 - accuracy:
0.8717 - val_loss: 0.3171 - val_accuracy: 0.8546
Epoch 15/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2713 - accuracy:
0.8726 - val_loss: 0.3203 - val_accuracy: 0.8525
Epoch 16/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2693 - accuracy:
0.8749 - val_loss: 0.3223 - val_accuracy: 0.8479
Epoch 17/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2660 - accuracy:
0.8766 - val_loss: 0.3247 - val_accuracy: 0.8514
Epoch 18/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2633 - accuracy:
0.8777 - val_loss: 0.3264 - val_accuracy: 0.8534
Epoch 19/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2611 - accuracy:
0.8793 - val_loss: 0.3295 - val_accuracy: 0.8482
Epoch 20/50
2290/2290 [==============================] - 7s 3ms/step - loss: 0.2575 - accuracy:
0.8802 - val_loss: 0.3293 - val_accuracy: 0.8546
Epoch 21/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2549 - accuracy:
0.8812 - val_loss: 0.3377 - val_accuracy: 0.8520
Epoch 22/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2524 - accuracy:
0.8843 - val_loss: 0.3341 - val_accuracy: 0.8505
Epoch 23/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2503 - accuracy:
0.8834 - val_loss: 0.3379 - val_accuracy: 0.8510
Epoch 24/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2460 - accuracy:
0.8866 - val_loss: 0.3435 - val_accuracy: 0.8481
Epoch 25/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2455 - accuracy:
0.8854 - val_loss: 0.3443 - val_accuracy: 0.8533
Epoch 26/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2435 - accuracy:
0.8858 - val_loss: 0.3482 - val_accuracy: 0.8472
Epoch 27/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2405 - accuracy:
0.8897 - val_loss: 0.3479 - val_accuracy: 0.8476
Epoch 28/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2396 - accuracy:
0.8869 - val_loss: 0.3521 - val_accuracy: 0.8464
Epoch 29/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2373 - accuracy:
0.8902 - val_loss: 0.3561 - val_accuracy: 0.8493
Epoch 30/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2337 - accuracy:
0.8916 - val_loss: 0.3530 - val_accuracy: 0.8454
Epoch 31/50
```

```
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2325 - accuracy:
0.8921 - val_loss: 0.3553 - val_accuracy: 0.8454
Epoch 32/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2303 - accuracy:
0.8936 - val_loss: 0.3608 - val_accuracy: 0.8459
Epoch 33/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2289 - accuracy:
0.8932 - val_loss: 0.3608 - val_accuracy: 0.8459
Epoch 34/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2261 - accuracy:
0.8967 - val_loss: 0.3684 - val_accuracy: 0.8431
Epoch 35/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2257 - accuracy:
0.8959 - val_loss: 0.3700 - val_accuracy: 0.8445
Epoch 36/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2253 - accuracy:
0.8960 - val_loss: 0.3748 - val_accuracy: 0.8455
Epoch 37/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2232 - accuracy:
0.8972 - val_loss: 0.3725 - val_accuracy: 0.8446
Epoch 38/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2210 - accuracy:
0.8956 - val_loss: 0.3834 - val_accuracy: 0.8373
Epoch 39/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2197 - accuracy:
0.8975 - val_loss: 0.3790 - val_accuracy: 0.8402
Epoch 40/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2162 - accuracy:
0.8984 - val_loss: 0.3818 - val_accuracy: 0.8445
Epoch 41/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2156 - accuracy:
0.9013 - val_loss: 0.3842 - val_accuracy: 0.8391
Epoch 42/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2141 - accuracy:
0.9007 - val_loss: 0.3804 - val_accuracy: 0.8425
Epoch 43/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2129 - accuracy:
0.9015 - val_loss: 0.3916 - val_accuracy: 0.8397
Epoch 44/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2114 - accuracy:
0.9029 - val_loss: 0.3856 - val_accuracy: 0.8410
Epoch 45/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2113 - accuracy:
0.9017 - val_loss: 0.3909 - val_accuracy: 0.8428
Epoch 46/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2096 - accuracy:
0.9031 - val_loss: 0.3943 - val_accuracy: 0.8408
Epoch 47/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2074 - accuracy:
0.9029 - val_loss: 0.4022 - val_accuracy: 0.8365
Epoch 48/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2059 - accuracy:
0.9064 - val_loss: 0.4046 - val_accuracy: 0.8385
Epoch 49/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.2064 - accuracy:
0.9052 - val_loss: 0.4037 - val_accuracy: 0.8376
Epoch 50/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.2039 - accuracy:
0.9067 - val_loss: 0.4129 - val_accuracy: 0.8353
```

In [54]:

```
plt.plot(history_dropout.history['loss'], label='train')
plt.plot(history_dropout.history['val_loss'], label='test')
plt.legend()
plt.show()
```



In [55]:

```
""" 构建神经网络
使用l2正则化不使用Dropout
使用Optimizer = adam
loss = 'binary_crossentropy'
最后输出层使用sigmoid激活函数
"""
model_l2 = Sequential()
# keras.regularizers.l1(lambda)
# keras.regularizers.l2(lambda)
# keras.regularizers.l1_l2(l1=lambda1, l2=lambda2)
model_l2.add(Dense(128, activation='tanh', input_shape=train_data[0].shape,kernel_re
model_l2.add(Dense(64, activation='tanh'))
model_l2.add(Dense(1, activation='sigmoid'))
```

In [56]:

```
model_l2.compile(optimizer='adam',loss = 'binary_crossentropy',metrics=['accuracy'])
history_l2 = model_l2.fit(train_data, train_label, epochs=50, batch_size=16, validat
```

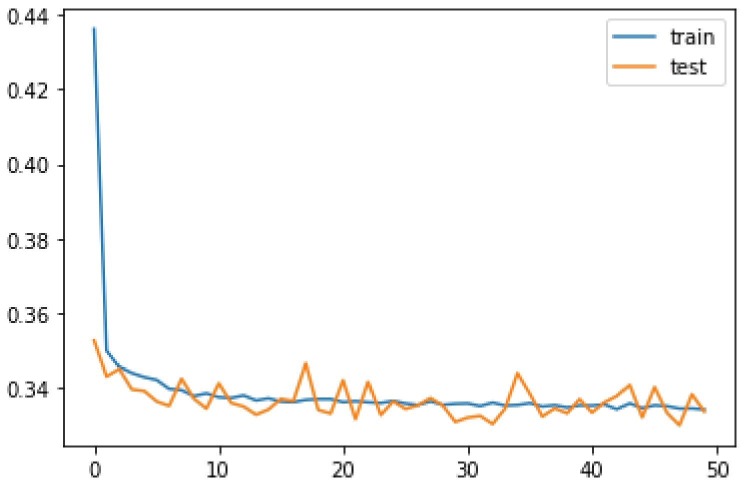
```
Epoch 1/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4362 - accuracy:
0.8370 - val_loss: 0.3525 - val_accuracy: 0.8446
Epoch 2/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3498 - accuracy:
0.8442 - val_loss: 0.3429 - val_accuracy: 0.8457
Epoch 3/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3454 - accuracy:
0.8466 - val_loss: 0.3448 - val_accuracy: 0.8478
Epoch 4/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3438 - accuracy:
0.8462 - val_loss: 0.3394 - val_accuracy: 0.8483
Epoch 5/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3427 - accuracy:
0.8468 - val_loss: 0.3390 - val_accuracy: 0.8505
Epoch 6/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3420 - accuracy:
0.8468 - val_loss: 0.3362 - val_accuracy: 0.8481
Epoch 7/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3396 - accuracy:
0.8482 - val_loss: 0.3350 - val_accuracy: 0.8501
Epoch 8/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3392 - accuracy:
0.8490 - val_loss: 0.3423 - val_accuracy: 0.8470
Epoch 9/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3376 - accuracy:
0.8476 - val_loss: 0.3369 - val_accuracy: 0.8498
Epoch 10/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3384 - accuracy:
0.8481 - val_loss: 0.3343 - val_accuracy: 0.8498
Epoch 11/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3373 - accuracy:
0.8489 - val_loss: 0.3411 - val_accuracy: 0.8451
```

```
Epoch 12/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3371 - accuracy:
0.8487 - val_loss: 0.3358 - val_accuracy: 0.8512
Epoch 13/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3378 - accuracy:
0.8481 - val_loss: 0.3348 - val_accuracy: 0.8492
Epoch 14/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3365 - accuracy:
0.8486 - val_loss: 0.3326 - val_accuracy: 0.8517
Epoch 15/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3370 - accuracy:
0.8471 - val_loss: 0.3340 - val_accuracy: 0.8501
Epoch 16/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3362 - accuracy:
0.8491 - val_loss: 0.3369 - val_accuracy: 0.8470
Epoch 17/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3360 - accuracy:
0.8499 - val_loss: 0.3363 - val_accuracy: 0.8522
Epoch 18/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3366 - accuracy:
0.8477 - val_loss: 0.3464 - val_accuracy: 0.8460
Epoch 19/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3368 - accuracy:
0.8483 - val_loss: 0.3339 - val_accuracy: 0.8502
Epoch 20/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3368 - accuracy:
0.8476 - val_loss: 0.3329 - val_accuracy: 0.8518
Epoch 21/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3360 - accuracy:
0.8490 - val_loss: 0.3419 - val_accuracy: 0.8455
Epoch 22/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3363 - accuracy:
0.8487 - val_loss: 0.3314 - val_accuracy: 0.8510
Epoch 23/50
2290/2290 [==============================] - 7s 3ms/step - loss: 0.3360 - accuracy:
0.8479 - val_loss: 0.3414 - val_accuracy: 0.8449
Epoch 24/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3358 - accuracy:
0.8490 - val_loss: 0.3325 - val_accuracy: 0.8519
Epoch 25/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3363 - accuracy:
0.8482 - val_loss: 0.3362 - val_accuracy: 0.8452
Epoch 26/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3356 - accuracy:
0.8483 - val_loss: 0.3342 - val_accuracy: 0.8518
Epoch 27/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3352 - accuracy:
0.8497 - val_loss: 0.3352 - val_accuracy: 0.8489
Epoch 28/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3362 - accuracy:
0.8483 - val_loss: 0.3371 - val_accuracy: 0.8502
Epoch 29/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3353 - accuracy:
0.8486 - val_loss: 0.3350 - val_accuracy: 0.8492
Epoch 30/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3356 - accuracy:
0.8487 - val_loss: 0.3307 - val_accuracy: 0.8517
Epoch 31/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3356 - accuracy:
0.8488 - val_loss: 0.3318 - val_accuracy: 0.8515
Epoch 32/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3349 - accuracy:
0.8494 - val_loss: 0.3323 - val_accuracy: 0.8495
Epoch 33/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3359 - accuracy:
0.8476 - val_loss: 0.3301 - val_accuracy: 0.8502
Epoch 34/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3351 - accuracy:
0.8478 - val_loss: 0.3342 - val_accuracy: 0.8496
```

```
Epoch 35/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3352 - accuracy:
0.8481 - val_loss: 0.3438 - val_accuracy: 0.8451
Epoch 36/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3357 - accuracy:
0.8480 - val_loss: 0.3382 - val_accuracy: 0.8471
Epoch 37/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3348 - accuracy:
0.8473 - val_loss: 0.3321 - val_accuracy: 0.8496
Epoch 38/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3352 - accuracy:
0.8481 - val_loss: 0.3343 - val_accuracy: 0.8482
Epoch 39/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3346 - accuracy:
0.8491 - val_loss: 0.3330 - val_accuracy: 0.8479
Epoch 40/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3351 - accuracy:
0.8492 - val_loss: 0.3368 - val_accuracy: 0.8478
Epoch 41/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3351 - accuracy:
0.8486 - val_loss: 0.3332 - val_accuracy: 0.8505
Epoch 42/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3354 - accuracy:
0.8494 - val_loss: 0.3359 - val_accuracy: 0.8484
Epoch 43/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3341 - accuracy:
0.8487 - val_loss: 0.3377 - val_accuracy: 0.8483
Epoch 44/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3357 - accuracy:
0.8487 - val_loss: 0.3406 - val_accuracy: 0.8445
Epoch 45/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3344 - accuracy:
0.8498 - val_loss: 0.3319 - val_accuracy: 0.8497
Epoch 46/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3351 - accuracy:
0.8487 - val_loss: 0.3400 - val_accuracy: 0.8447
Epoch 47/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3349 - accuracy:
0.8478 - val_loss: 0.3331 - val_accuracy: 0.8505
Epoch 48/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3343 - accuracy:
0.8488 - val_loss: 0.3298 - val_accuracy: 0.8510
Epoch 49/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3343 - accuracy:
0.8483 - val_loss: 0.3381 - val_accuracy: 0.8454
Epoch 50/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3340 - accuracy:
0.8484 - val_loss: 0.3335 - val_accuracy: 0.8502
```

In [57]:

```
plt.plot(history_l2.history['loss'], label='train')
plt.plot(history_l2.history['val_loss'], label='test')
plt.legend()
plt.show()
```



In [58]:

```
""" 构建神经网络
使用l1正则化不使用Dropout
使用Optimizer = adam
loss = 'binary_crossentropy'
最后输出层使用sigmoid激活函数
"""
model_l1 = Sequential()
# keras.regularizers.l1(lambda)
# keras.regularizers.l2(lambda)
# keras.regularizers.l1_l2(l1=lambda1, l2=lambda2)
model_l1.add(Dense(128, activation='tanh', input_shape=train_data[0].shape,kernel_re
model_l1.add(Dense(64, activation='tanh'))
model_l1.add(Dense(1, activation='sigmoid'))
```

In [59]:

```
model_l1.compile(optimizer='adam',loss = 'binary_crossentropy',metrics=['accuracy'])
history_l1 = model_l1.fit(train_data, train_label, epochs=50, batch_size=16, validat
```

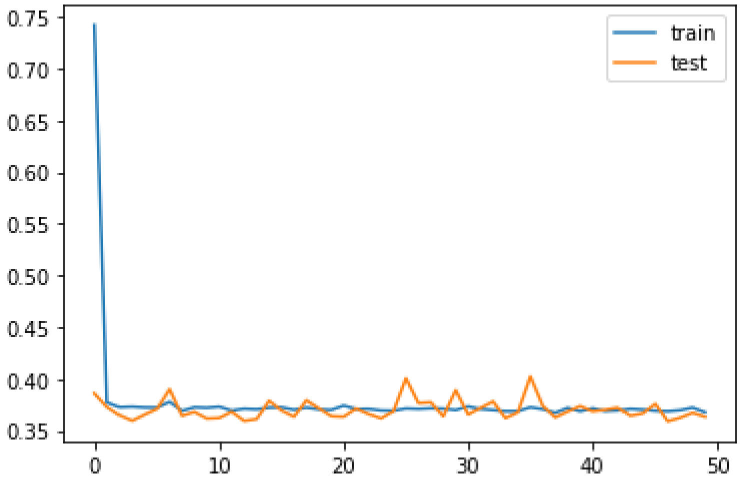
```
Epoch 1/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.7422 - accuracy:
0.8403 - val_loss: 0.3861 - val_accuracy: 0.8451
Epoch 2/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3775 - accuracy:
0.8475 - val_loss: 0.3730 - val_accuracy: 0.8451
Epoch 3/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3728 - accuracy:
0.8466 - val_loss: 0.3649 - val_accuracy: 0.8476
Epoch 4/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3733 - accuracy:
0.8472 - val_loss: 0.3595 - val_accuracy: 0.8494
Epoch 5/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3726 - accuracy:
0.8467 - val_loss: 0.3656 - val_accuracy: 0.8451
Epoch 6/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3725 - accuracy:
0.8472 - val_loss: 0.3712 - val_accuracy: 0.8486
Epoch 7/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3780 - accuracy:
0.8463 - val_loss: 0.3902 - val_accuracy: 0.8493
Epoch 8/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3693 - accuracy:
0.8472 - val_loss: 0.3644 - val_accuracy: 0.8466
Epoch 9/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3727 - accuracy:
0.8488 - val_loss: 0.3681 - val_accuracy: 0.8465
Epoch 10/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3722 - accuracy:
0.8470 - val_loss: 0.3617 - val_accuracy: 0.8497
Epoch 11/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3733 - accuracy:
0.8491 - val_loss: 0.3623 - val_accuracy: 0.8469
```

```
Epoch 12/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3695 - accuracy:
0.8482 - val_loss: 0.3683 - val_accuracy: 0.8441
Epoch 13/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3713 - accuracy:
0.8464 - val_loss: 0.3594 - val_accuracy: 0.8501
Epoch 14/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3707 - accuracy:
0.8467 - val_loss: 0.3611 - val_accuracy: 0.8475
Epoch 15/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3723 - accuracy:
0.8459 - val_loss: 0.3791 - val_accuracy: 0.8472
Epoch 16/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3727 - accuracy:
0.8481 - val_loss: 0.3696 - val_accuracy: 0.8497
Epoch 17/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3703 - accuracy:
0.8466 - val_loss: 0.3635 - val_accuracy: 0.8486
Epoch 18/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3724 - accuracy:
0.8477 - val_loss: 0.3795 - val_accuracy: 0.8515
Epoch 19/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3707 - accuracy:
0.8480 - val_loss: 0.3717 - val_accuracy: 0.8472
Epoch 20/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3702 - accuracy:
0.8471 - val_loss: 0.3641 - val_accuracy: 0.8446
Epoch 21/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3744 - accuracy:
0.8469 - val_loss: 0.3634 - val_accuracy: 0.8492
Epoch 22/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3710 - accuracy:
0.8475 - val_loss: 0.3715 - val_accuracy: 0.8502
Epoch 23/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3711 - accuracy:
0.8466 - val_loss: 0.3661 - val_accuracy: 0.8464
Epoch 24/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3696 - accuracy:
0.8463 - val_loss: 0.3619 - val_accuracy: 0.8519
Epoch 25/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3694 - accuracy:
0.8467 - val_loss: 0.3689 - val_accuracy: 0.8477
Epoch 26/50
2290/2290 [==============================] - 7s 3ms/step - loss: 0.3715 - accuracy:
0.8459 - val_loss: 0.4007 - val_accuracy: 0.8485
Epoch 27/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3711 - accuracy:
0.8483 - val_loss: 0.3768 - val_accuracy: 0.8494
Epoch 28/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3717 - accuracy:
0.8458 - val_loss: 0.3775 - val_accuracy: 0.8477
Epoch 29/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3714 - accuracy:
0.8468 - val_loss: 0.3639 - val_accuracy: 0.8451
Epoch 30/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3700 - accuracy:
0.8483 - val_loss: 0.3892 - val_accuracy: 0.8449
Epoch 31/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3736 - accuracy:
0.8484 - val_loss: 0.3656 - val_accuracy: 0.8482
Epoch 32/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3710 - accuracy:
0.8472 - val_loss: 0.3721 - val_accuracy: 0.8483
Epoch 33/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3700 - accuracy:
0.8487 - val_loss: 0.3783 - val_accuracy: 0.8488
Epoch 34/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3689 - accuracy:
0.8467 - val_loss: 0.3621 - val_accuracy: 0.8465
```

```
Epoch 35/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3689 - accuracy:
0.8466 - val_loss: 0.3678 - val_accuracy: 0.8492
Epoch 36/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3727 - accuracy:
0.8487 - val_loss: 0.4026 - val_accuracy: 0.8422
Epoch 37/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3708 - accuracy:
0.8464 - val_loss: 0.3740 - val_accuracy: 0.8489
Epoch 38/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3673 - accuracy:
0.8488 - val_loss: 0.3628 - val_accuracy: 0.8464
Epoch 39/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3721 - accuracy:
0.8458 - val_loss: 0.3685 - val_accuracy: 0.8448
Epoch 40/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3689 - accuracy:
0.8472 - val_loss: 0.3739 - val_accuracy: 0.8473
Epoch 41/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3717 - accuracy:
0.8472 - val_loss: 0.3688 - val_accuracy: 0.8466
Epoch 42/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3690 - accuracy:
0.8471 - val_loss: 0.3703 - val_accuracy: 0.8496
Epoch 43/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3696 - accuracy:
0.8456 - val_loss: 0.3723 - val_accuracy: 0.8504
Epoch 44/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3712 - accuracy:
0.8472 - val_loss: 0.3645 - val_accuracy: 0.8487
Epoch 45/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3705 - accuracy:
0.8460 - val_loss: 0.3668 - val_accuracy: 0.8464
Epoch 46/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3693 - accuracy:
0.8479 - val_loss: 0.3760 - val_accuracy: 0.8460
Epoch 47/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3688 - accuracy:
0.8468 - val_loss: 0.3590 - val_accuracy: 0.8485
Epoch 48/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3699 - accuracy:
0.8467 - val_loss: 0.3625 - val_accuracy: 0.8504
Epoch 49/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3724 - accuracy:
0.8473 - val_loss: 0.3673 - val_accuracy: 0.8510
Epoch 50/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3678 - accuracy:
0.8486 - val_loss: 0.3634 - val_accuracy: 0.8477
```

In [60]:

```
plt.plot(history_l1.history['loss'], label='train')
plt.plot(history_l1.history['val_loss'], label='test')
plt.legend()
plt.show()
```



In [77]:

```
""" 构建神经网络
使用l1正则化不使用Dropout
使用Optimizer = adagrad
loss = 'binary_crossentropy'
最后输出层使用sigmoid激活函数
"""
model_l1_Adagrad = Sequential()
# keras.regularizers.l1(lambda)
# keras.regularizers.l2(lambda)
# keras.regularizers.l1_l2(l1=lambda1, l2=lambda2)
model_l1_Adagrad.add(Dense(128, activation='tanh', input_shape=train_data[0].shape,k
model_l1_Adagrad.add(Dense(64, activation='tanh'))
model_l1_Adagrad.add(Dense(1, activation='sigmoid'))
```

In [78]:

```
model_l1_Adagrad.compile(optimizer='Adagrad',loss = 'binary_crossentropy',metrics=['
history_l1_Adagrad = model_l1_Adagrad.fit(train_data, train_label, epochs=50, batch_
```

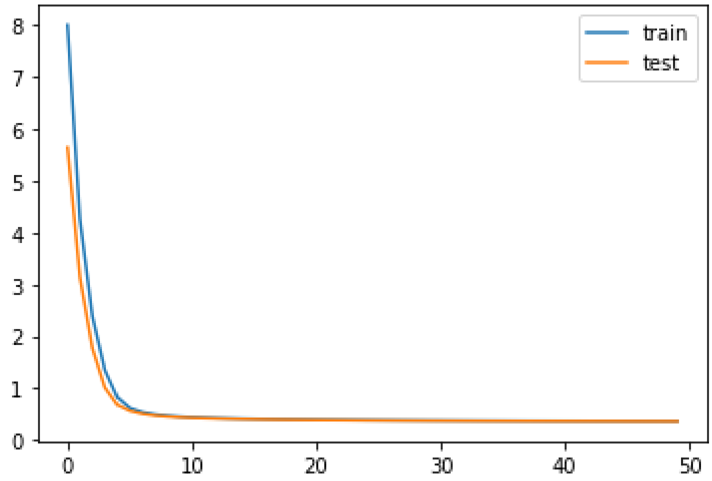
```
Epoch 1/50
2290/2290 [==============================] - 7s 3ms/step - loss: 8.0088 - accuracy:
0.7891 - val_loss: 5.6487 - val_accuracy: 0.8382
Epoch 2/50
2290/2290 [==============================] - 6s 3ms/step - loss: 4.2595 - accuracy:
0.8371 - val_loss: 3.1335 - val_accuracy: 0.8398
Epoch 3/50
2290/2290 [==============================] - 6s 3ms/step - loss: 2.3868 - accuracy:
0.8398 - val_loss: 1.7649 - val_accuracy: 0.8410
Epoch 4/50
2290/2290 [==============================] - 6s 3ms/step - loss: 1.3524 - accuracy:
0.8396 - val_loss: 1.0147 - val_accuracy: 0.8410
Epoch 5/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.8219 - accuracy:
0.8391 - val_loss: 0.6764 - val_accuracy: 0.8416
Epoch 6/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.6153 - accuracy:
0.8385 - val_loss: 0.5624 - val_accuracy: 0.8414
Epoch 7/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.5354 - accuracy:
0.8387 - val_loss: 0.5065 - val_accuracy: 0.8424
Epoch 8/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4932 - accuracy:
0.8392 - val_loss: 0.4741 - val_accuracy: 0.8429
Epoch 9/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4675 - accuracy:
0.8394 - val_loss: 0.4539 - val_accuracy: 0.8434
Epoch 10/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4505 - accuracy:
0.8399 - val_loss: 0.4392 - val_accuracy: 0.8439
Epoch 11/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4381 - accuracy:
0.8399 - val_loss: 0.4288 - val_accuracy: 0.8440
```

```
Epoch 12/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.4290 - accuracy:
0.8408 - val_loss: 0.4209 - val_accuracy: 0.8448
Epoch 13/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4218 - accuracy:
0.8408 - val_loss: 0.4144 - val_accuracy: 0.8447
Epoch 14/50
2290/2290 [==============================] - 6s 2ms/step - loss: 0.4159 - accuracy:
0.8415 - val_loss: 0.4090 - val_accuracy: 0.8460
Epoch 15/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4109 - accuracy:
0.8423 - val_loss: 0.4044 - val_accuracy: 0.8464
Epoch 16/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4065 - accuracy:
0.8425 - val_loss: 0.4004 - val_accuracy: 0.8466
Epoch 17/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4027 - accuracy:
0.8429 - val_loss: 0.3967 - val_accuracy: 0.8473
Epoch 18/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3993 - accuracy:
0.8433 - val_loss: 0.3935 - val_accuracy: 0.8470
Epoch 19/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3963 - accuracy:
0.8437 - val_loss: 0.3910 - val_accuracy: 0.8475
Epoch 20/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3936 - accuracy:
0.8441 - val_loss: 0.3883 - val_accuracy: 0.8469
Epoch 21/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3911 - accuracy:
0.8447 - val_loss: 0.3862 - val_accuracy: 0.8478
Epoch 22/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3888 - accuracy:
0.8448 - val_loss: 0.3839 - val_accuracy: 0.8473
Epoch 23/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3867 - accuracy:
0.8450 - val_loss: 0.3824 - val_accuracy: 0.8483
Epoch 24/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3848 - accuracy:
0.8454 - val_loss: 0.3801 - val_accuracy: 0.8488
Epoch 25/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3831 - accuracy:
0.8454 - val_loss: 0.3785 - val_accuracy: 0.8487
Epoch 26/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3814 - accuracy:
0.8461 - val_loss: 0.3771 - val_accuracy: 0.8487
Epoch 27/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3799 - accuracy:
0.8460 - val_loss: 0.3758 - val_accuracy: 0.8492
Epoch 28/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3785 - accuracy:
0.8465 - val_loss: 0.3743 - val_accuracy: 0.8494
Epoch 29/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3772 - accuracy:
0.8465 - val_loss: 0.3733 - val_accuracy: 0.8488
Epoch 30/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3761 - accuracy:
0.8472 - val_loss: 0.3721 - val_accuracy: 0.8492
Epoch 31/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3750 - accuracy:
0.8472 - val_loss: 0.3713 - val_accuracy: 0.8496
Epoch 32/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3740 - accuracy:
0.8476 - val_loss: 0.3705 - val_accuracy: 0.8500
Epoch 33/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3731 - accuracy:
0.8475 - val_loss: 0.3693 - val_accuracy: 0.8500
Epoch 34/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3722 - accuracy:
0.8481 - val_loss: 0.3685 - val_accuracy: 0.8499
```

```
Epoch 35/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3713 - accuracy:
0.8479 - val_loss: 0.3676 - val_accuracy: 0.8501
Epoch 36/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3705 - accuracy:
0.8479 - val_loss: 0.3671 - val_accuracy: 0.8498
Epoch 37/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3697 - accuracy:
0.8483 - val_loss: 0.3665 - val_accuracy: 0.8501
Epoch 38/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3690 - accuracy:
0.8486 - val_loss: 0.3656 - val_accuracy: 0.8501
Epoch 39/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3683 - accuracy:
0.8488 - val_loss: 0.3650 - val_accuracy: 0.8507
Epoch 40/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3676 - accuracy:
0.8487 - val_loss: 0.3644 - val_accuracy: 0.8507
Epoch 41/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3670 - accuracy:
0.8487 - val_loss: 0.3639 - val_accuracy: 0.8505
Epoch 42/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3663 - accuracy:
0.8487 - val_loss: 0.3630 - val_accuracy: 0.8507
Epoch 43/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3657 - accuracy:
0.8489 - val_loss: 0.3628 - val_accuracy: 0.8508
Epoch 44/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3651 - accuracy:
0.8492 - val_loss: 0.3621 - val_accuracy: 0.8506
Epoch 45/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3646 - accuracy:
0.8492 - val_loss: 0.3621 - val_accuracy: 0.8508
Epoch 46/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3640 - accuracy:
0.8487 - val_loss: 0.3609 - val_accuracy: 0.8507
Epoch 47/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3635 - accuracy:
0.8490 - val_loss: 0.3605 - val_accuracy: 0.8507
Epoch 48/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3630 - accuracy:
0.8489 - val_loss: 0.3602 - val_accuracy: 0.8514
Epoch 49/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3625 - accuracy:
0.8491 - val_loss: 0.3598 - val_accuracy: 0.8511
Epoch 50/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.3621 - accuracy:
0.8491 - val_loss: 0.3593 - val_accuracy: 0.8510
```

In [80]:

```
plt.plot(history_l1_Adagrad.history['loss'], label='train')
plt.plot(history_l1_Adagrad.history['val_loss'], label='test')
plt.legend()
plt.show()
```



In [71]:
```
""" 构建神经网络
使用l2_l1正则化
使用Dropout
使用Optimizer = adam
loss = 'binary_crossentropy'
最后输出层使用sigmoid激活函数
"""
model = Sequential()
# keras.regularizers.l1(lambda)
# keras.regularizers.l2(lambda)
# keras.regularizers.l1_l2(l1=lambda1, l2=lambda2)
model.add(Dense(128, activation='tanh', input_shape=train_data[0].shape,kernel_regul
model.add(Dense(64, activation='tanh'))
model.add(Dropout(0.25))
model.add(Dense(1, activation='sigmoid'))
```

In [72]:
```
model.compile(optimizer='adam',loss = 'binary_crossentropy',metrics=['accuracy'])
history = model.fit(train_data, train_label, epochs=50, batch_size=16, validation_da
```

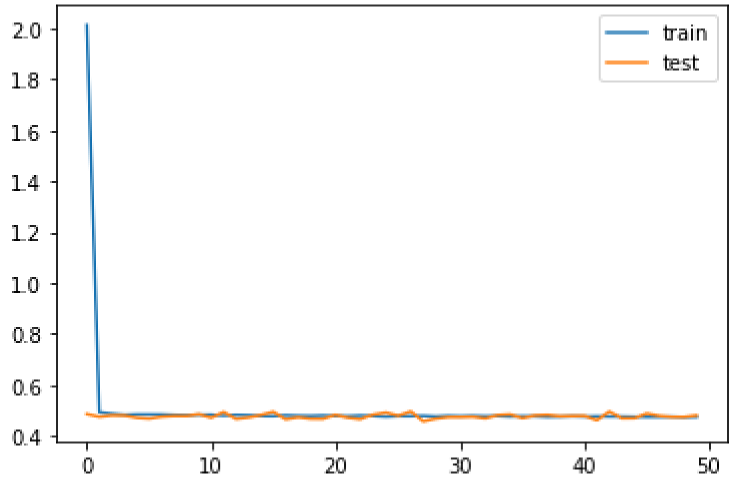
```
Epoch 1/50
2290/2290 [==============================] - 6s 3ms/step - loss: 2.0142 - accuracy:
0.8287 - val_loss: 0.4856 - val_accuracy: 0.8451
Epoch 2/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4928 - accuracy:
0.8436 - val_loss: 0.4749 - val_accuracy: 0.8457
Epoch 3/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4870 - accuracy:
0.8411 - val_loss: 0.4802 - val_accuracy: 0.8432
Epoch 4/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4826 - accuracy:
0.8435 - val_loss: 0.4801 - val_accuracy: 0.8473
Epoch 5/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4841 - accuracy:
0.8416 - val_loss: 0.4710 - val_accuracy: 0.8443
Epoch 6/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4841 - accuracy:
0.8434 - val_loss: 0.4677 - val_accuracy: 0.8465
Epoch 7/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4839 - accuracy:
0.8432 - val_loss: 0.4752 - val_accuracy: 0.8442
Epoch 8/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4822 - accuracy:
0.8421 - val_loss: 0.4783 - val_accuracy: 0.8458
Epoch 9/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4809 - accuracy:
0.8426 - val_loss: 0.4780 - val_accuracy: 0.8482
Epoch 10/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4827 - accuracy:
0.8423 - val_loss: 0.4852 - val_accuracy: 0.8444
Epoch 11/50
```

```
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4828 - accuracy:
0.8420 - val_loss: 0.4713 - val_accuracy: 0.8443
Epoch 12/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4797 - accuracy:
0.8415 - val_loss: 0.4936 - val_accuracy: 0.8392
Epoch 13/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4820 - accuracy:
0.8426 - val_loss: 0.4665 - val_accuracy: 0.8460
Epoch 14/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4808 - accuracy:
0.8425 - val_loss: 0.4722 - val_accuracy: 0.8482
Epoch 15/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4779 - accuracy:
0.8424 - val_loss: 0.4818 - val_accuracy: 0.8471
Epoch 16/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4777 - accuracy:
0.8435 - val_loss: 0.4943 - val_accuracy: 0.8424
Epoch 17/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4806 - accuracy:
0.8417 - val_loss: 0.4651 - val_accuracy: 0.8471
Epoch 18/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4789 - accuracy:
0.8431 - val_loss: 0.4730 - val_accuracy: 0.8444
Epoch 19/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4783 - accuracy:
0.8432 - val_loss: 0.4673 - val_accuracy: 0.8470
Epoch 20/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4799 - accuracy:
0.8428 - val_loss: 0.4668 - val_accuracy: 0.8461
Epoch 21/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4779 - accuracy:
0.8434 - val_loss: 0.4825 - val_accuracy: 0.8451
Epoch 22/50
2290/2290 [==============================] - 7s 3ms/step - loss: 0.4774 - accuracy:
0.8432 - val_loss: 0.4715 - val_accuracy: 0.8465
Epoch 23/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4797 - accuracy:
0.8432 - val_loss: 0.4655 - val_accuracy: 0.8444
Epoch 24/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4780 - accuracy:
0.8433 - val_loss: 0.4835 - val_accuracy: 0.8460
Epoch 25/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4756 - accuracy:
0.8425 - val_loss: 0.4914 - val_accuracy: 0.8456
Epoch 26/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4775 - accuracy:
0.8430 - val_loss: 0.4790 - val_accuracy: 0.8438
Epoch 27/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4775 - accuracy:
0.8426 - val_loss: 0.4960 - val_accuracy: 0.8377
Epoch 28/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4784 - accuracy:
0.8419 - val_loss: 0.4569 - val_accuracy: 0.8461
Epoch 29/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4762 - accuracy:
0.8427 - val_loss: 0.4675 - val_accuracy: 0.8453
Epoch 30/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4784 - accuracy:
0.8444 - val_loss: 0.4735 - val_accuracy: 0.8424
Epoch 31/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4772 - accuracy:
0.8424 - val_loss: 0.4739 - val_accuracy: 0.8414
Epoch 32/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4783 - accuracy:
0.8417 - val_loss: 0.4749 - val_accuracy: 0.8417
Epoch 33/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4771 - accuracy:
0.8430 - val_loss: 0.4687 - val_accuracy: 0.8454
Epoch 34/50
```

```
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4779 - accuracy:
0.8421 - val_loss: 0.4814 - val_accuracy: 0.8443
Epoch 35/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4759 - accuracy:
0.8437 - val_loss: 0.4842 - val_accuracy: 0.8431
Epoch 36/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4773 - accuracy:
0.8423 - val_loss: 0.4696 - val_accuracy: 0.8455
Epoch 37/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4767 - accuracy:
0.8413 - val_loss: 0.4801 - val_accuracy: 0.8383
Epoch 38/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4746 - accuracy:
0.8438 - val_loss: 0.4823 - val_accuracy: 0.8449
Epoch 39/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4754 - accuracy:
0.8436 - val_loss: 0.4771 - val_accuracy: 0.8439
Epoch 40/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4770 - accuracy:
0.8423 - val_loss: 0.4780 - val_accuracy: 0.8445
Epoch 41/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4757 - accuracy:
0.8421 - val_loss: 0.4785 - val_accuracy: 0.8460
Epoch 42/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4767 - accuracy:
0.8412 - val_loss: 0.4612 - val_accuracy: 0.8463
Epoch 43/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4759 - accuracy:
0.8421 - val_loss: 0.4954 - val_accuracy: 0.8386
Epoch 44/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4758 - accuracy:
0.8426 - val_loss: 0.4690 - val_accuracy: 0.8443
Epoch 45/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4749 - accuracy:
0.8440 - val_loss: 0.4695 - val_accuracy: 0.8492
Epoch 46/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4742 - accuracy:
0.8433 - val_loss: 0.4877 - val_accuracy: 0.8423
Epoch 47/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4741 - accuracy:
0.8428 - val_loss: 0.4785 - val_accuracy: 0.8456
Epoch 48/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4751 - accuracy:
0.8431 - val_loss: 0.4742 - val_accuracy: 0.8421
Epoch 49/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4735 - accuracy:
0.8425 - val_loss: 0.4741 - val_accuracy: 0.8410
Epoch 50/50
2290/2290 [==============================] - 6s 3ms/step - loss: 0.4742 - accuracy:
0.8415 - val_loss: 0.4802 - val_accuracy: 0.8454
```

In [73]:

```python
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
plt.show()
```


train
test
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
10
20
30
40
50