```
    return 1 / (1 + np.exp(-t));
```

1、 逻辑回归的损失函数可以写成如下形式

$$cost = \begin{cases} -log(\hat{p}) & y = 1 \\ -log(1 - \hat{p}) & y = 0 \end{cases}$$

- [x] A、对
- [ ] B、错

2、 下列说法正确的是

- [x] A、损失值能够衡量模型在训练数据集上的拟合程度
- [ ] B、sigmoid函数不可导
- [x] C、sigmoid函数的输入越大，输出就越大
- [x] D、训练的过程，就是寻找合适的参数使得损失函数值最小的过程

3、 sigmoid 函数(**对数几率函数**)相对于单位阶跃函数有哪些好处?

- [x] A、sigmoid函数可微分
- [x] B、sigmoid函数处处连续
- [ ] C、sigmoid函数不是单调的
- [ ] D、sigmoid函数最多计算二阶导

4、 逻辑回归的优点有哪些?

- [ ] A、需要事先对数据的分布做假设
- [ ] B、可以得到“类别”的真正的概率预测
- [ ] C、可以用闭式解求解
- [x] D、可以用现有的数值优化算法求解

```
    return 3
# 这题真实一把子无语住了
```

```
# -*- coding: utf-8 -*-

import numpy as np
import warnings
warnings.filterwarnings("ignore")

def sigmoid(x):
    '''
    sigmoid函数
    :param x: 转换前的输入
    :return: 转换后的概率
    '''
    return 1/(1+np.exp(-x))


def fit(x,y,eta=1e-3,n_iters=10000):
    '''
    训练逻辑回归模型
    :param x: 训练集特征数据，类型为ndarray
    :param y: 训练集标签，类型为ndarray
    :param eta: 学习率，类型为float
    :param n_iters: 训练轮数，类型为int
    :return: 模型参数，类型为ndarray
    '''
    #    请在此添加实现代码    #
    #********** Begin *********#
    theta = np.zeros(x.shape[1])
    for _ in range (n_iters):
        gradient = (sigmoid(x.dot(theta)) - y).dot(x)
        theta = theta - eta*gradient
    return theta
    #********** End **********#
```

```
from sklearn.linear_model import LogisticRegression

def digit_predict(train_image, train_label, test_image):
```

```
    '''
    实现功能：训练模型并输出预测结果
    :param train_sample: 包含多条训练样本的样本集，类型为ndarray,shape为
[-1, 8, 8]
    :param train_label: 包含多条训练样本标签的标签集，类型为ndarray
    :param test_sample: 包含多条测试样本的测试集，类型为ndarry
    :return: test_sample对应的预测标签
    '''

    #************* Begin ************#
    flat_train_image = train_image.reshape((-1, 64))
    # 训练集标准化
    train_min = flat_train_image.min()
    train_max = flat_train_image.max()
    flat_train_image = (flat_train_image-train_min)/(train_max-
train_min)
    # 测试集变形
    flat_test_image = test_image.reshape((-1, 64))
    # 测试集标准化
    test_min = flat_test_image.min()
    test_max = flat_test_image.max()
    flat_test_image = (flat_test_image - test_min) / (test_max -
test_min)

    # 训练--预测
    rf = LogisticRegression(C=4.0)
    rf.fit(flat_train_image, train_label)
    return rf.predict(flat_test_image)
    #************* End **************#
```