一、 实验目的

理解逻辑回归模型，掌握逻辑回归的参数估计算法。测试一下满足和不满足朴素贝叶斯的情况下逻辑回归的效果，并找一组 UCI 上的数据集观测效果。

二、 实验要求及实验环境

2.1 实验要求

1. 使用梯度下降，牛顿法，共轭梯度法一种或多种方法求解；
2. 手动生成符合多元高斯分布的数据，要求有满足和不满足朴素贝叶斯两种；
3. 在 UCI 网站上找一组数据集观测效果；
4. 使用正则项防止过拟合；

2.2 实验环境

- 硬件：Intel i5-8265U、512G SSD、8G RAM；
- 系统：Windows 11；
- IDE：Pycharm。

三、 设计思想

3.1 算法原理

对某一个样本 $<\boldsymbol{x}, y>$，其标记为 1 的概率为

$$\begin{aligned}
P(y=1|\boldsymbol{x}) &= \frac{P(y=1)P(\boldsymbol{x}|y=1)}{P(y=1)P(\boldsymbol{x}|y=1)+P(y=0)P(\boldsymbol{x}|y=0)} \\
&= \frac{1}{1+\dfrac{P(y=0)P(\boldsymbol{x}|y=0)}{P(y=1)P(\boldsymbol{x}|y=1)}} \\
&= \frac{1}{1+\exp(\ln\dfrac{P(y=0)}{P(y=1)}+\ln\dfrac{P(\boldsymbol{x}|y=0)}{P(\boldsymbol{x}|y=1)})}.
\end{aligned}$$

令 $P(y=1)=\pi$，假设此时样本各维度间互相独立。则有

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\exp(\ln\dfrac{1-\pi}{\pi}+\sum_{i=1}^{m}\ln\dfrac{P(x_i|y=0)}{P(x_i|y=1)})}, \tag{1}$$

其中 m 为样本 $\boldsymbol{x}$ 的属性个数。

不妨设 $P(x_i|y=1) \sim N(\mu_{i1}, \sigma_i)$，$P(x_i|y=0) \sim N(\mu_{i0}, \sigma_i)$，则：

$$P(x_i|y=1) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x_i-\mu_{i1})^2}{2\mu_i^2}},$$

$$P(x_i|y=0) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x_i-\mu_{i0})^2}{2\mu_i^2}},$$

$$\ln \frac{P(x_i|y=0)}{P(x_i|y=1)}) = -\frac{(x_i-\mu_{i0})^2}{2\mu_i^2} + \frac{(x_i-\mu_{i1})^2}{2\mu_i^2} = \frac{2(\mu_{i0}-\mu_{i1})x_i + \mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2}.$$

对式（1）进一步化简可得

$$\begin{aligned} P(y=1|\boldsymbol{x}) &= \frac{1}{1+\exp(\ln \dfrac{1-\pi}{\pi} + \sum_{i=1}^m (\frac{2(\mu_{i0}-\mu_{i1})}{2\sigma_i^2} x_i + \frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}))} \\ &= \frac{1}{1+exp(w_0 + \sum_{i=1}^m w_i x_i)}. \end{aligned}$$

构造

$$\boldsymbol{w}' = \begin{pmatrix} w_0 & w_1 & \cdots & w_m \end{pmatrix},$$

$$\boldsymbol{x}' = \begin{pmatrix} 1 & x_1 & \cdots & x_m \end{pmatrix},$$

则式（1）可进一步简化为

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\exp(\boldsymbol{w}'\boldsymbol{x})}.$$

由上易得

$$P(y=0|\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}'\boldsymbol{x})}{1+\exp(\boldsymbol{w}'\boldsymbol{x})},$$

$$\frac{P(y=0|\boldsymbol{x})}{P(y=1|\boldsymbol{x})} = \exp(\boldsymbol{w}'\boldsymbol{x}),$$

$$\ln \frac{P(y=0|\boldsymbol{x})}{P(y=1|\boldsymbol{x})} = \boldsymbol{w}'\boldsymbol{x}.$$

现在有一组样本点 $\{<\boldsymbol{x_1}, y_1>, <\boldsymbol{x_2}, y_2> \cdots <\boldsymbol{x_n}, y_n>\}$，应找到一组 $\boldsymbol{w}$ 使得上面这一组数据出现的概率最大，即

$$\begin{aligned} W_{MLE} &= \arg\max P(<\boldsymbol{x_1}, y>, <\boldsymbol{x_2}, y> \cdots <\boldsymbol{x_n}, y> |\boldsymbol{w}) \\ &= \arg\max \prod_{i=1}^n P(<\boldsymbol{x_i}, y_i> |\boldsymbol{w}) \end{aligned} \tag{2}$$

但之前仅推出 $P(y_i|\boldsymbol{x_i})$，因此将式（2）变化为

$$W_{MCLE} = \arg\max \prod_{i=1}^{n} P(y_i|\boldsymbol{x_i}, \boldsymbol{w}).$$

而加入先验概率，采用贝叶斯估计之后

$$W_{MCAP} = \arg\max \prod_{i=1}^{n} P(\boldsymbol{w})P(y_i|\boldsymbol{x_i}, \boldsymbol{w}).$$

具体表现为损失函数则为

$$\begin{aligned} L(\boldsymbol{w})_{MCLE} &= \sum_{i=1}^{n} y_i \ln P(y_i = 1|\boldsymbol{x_i}) + (1 - y_i) \ln P(y_i = 0|\boldsymbol{x_i}) \\ &= \sum_{i=1}^{n} (1 - y_i)\boldsymbol{w'x_i} - \ln[1 + \exp(\boldsymbol{w'x_i})]. \end{aligned}$$

对其求导有

$$\begin{aligned} \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} &= \sum_{i=1}^{n} [\frac{1}{1 + \exp(\boldsymbol{w'x_i})} - y_i]\boldsymbol{x_i} \\ &= \sum_{i=1}^{n} [sigmoid(-\boldsymbol{w'x_i}) - y_i]\boldsymbol{x_i}. \end{aligned} \tag{3}$$

为后续编程方便，现将其改写为矩阵乘法的方式。首先构造

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1(m+1)} \\ x_{21} & x_{22} & \cdots & x_{2(m+1)} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{n(m+1)} \end{bmatrix} \in \boldsymbol{R}^{n*(m+1)},$$

$$\boldsymbol{y'} = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \end{pmatrix},$$

$\boldsymbol{X}$,$\boldsymbol{y}$ 中每一行表示一个样本，$x_{i0} = 1$，一个样本有 m 个属性，一个标记，一共有 n 个样本。

则式（3）可以改写为

$$\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = k_1\boldsymbol{x_1} + k_2\boldsymbol{x_2} + \cdots + k_n\boldsymbol{x_n} = \boldsymbol{X'K},$$

$$\boldsymbol{X'} = \begin{pmatrix} \boldsymbol{x_1} & \boldsymbol{x_2} & \cdots & \boldsymbol{x_n} \end{pmatrix},$$

$$\boldsymbol{K}=\begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix},$$

$$k_i=\frac{1}{1+\exp(\boldsymbol{w'x_i})}-y_i=sigmoid(-\boldsymbol{w'x_i})-y_i.$$

即

$$\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}}=\boldsymbol{X'}[sigmoid(-\boldsymbol{Xw})-\boldsymbol{Y}].$$

其二阶导数为

$$\frac{\partial^2 L(\boldsymbol{w})}{\partial \boldsymbol{w}\partial \boldsymbol{w'}}=\boldsymbol{X'}\frac{\partial sigmoid(-\boldsymbol{Xw})}{\partial \boldsymbol{w'}}.$$

其中

$$sigmoid(-\boldsymbol{Xw})=\begin{pmatrix} sigmoid(-\boldsymbol{x_1'w}) \\ sigmoid(-\boldsymbol{x_2'w}) \\ \vdots \\ sigmoid(-\boldsymbol{x_n'w}) \end{pmatrix}=\begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix}$$

$$\boldsymbol{w'}=\begin{pmatrix} w_1 & w_2 & \cdots & w_{m+1} \end{pmatrix}$$

列向量对行向量求导有

$$\frac{\partial sigmoid(-\boldsymbol{Xw})}{\partial \boldsymbol{w'}}=\begin{bmatrix} \frac{\partial z_1}{\partial w_1} & \frac{\partial z_1}{\partial w_2} & \cdots & \frac{\partial z_1}{\partial w_{m+1}} \\ \frac{\partial z_2}{\partial w_1} & \frac{\partial z_2}{\partial w_2} & \cdots & \frac{\partial z_2}{\partial w_{m+1}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial z_n}{\partial w_1} & \frac{\partial z_n}{\partial w_2} & \cdots & \frac{\partial z_n}{\partial w_{m+1}} \end{bmatrix} \tag{4}$$

而

$$\frac{\partial z_i}{\partial w_j}=\frac{\partial sigmoid(-\boldsymbol{x_1'w})}{\partial w_j}=z_i(1-z_i)x_{ij}$$

则式（4）求解得

$$\frac{\partial sigmoid(-\boldsymbol{Xw})}{\partial \boldsymbol{w'}}=\begin{bmatrix} z_1(1-z_1)x_{11} & z_1(1-z_1)x_{12} & \cdots & z_1(1-z_1)x_{1(m+1)} \\ z_2(1-z_2)x_{21} & z_2(1-z_2)x_{22} & \cdots & z_2(1-z_2)x_{2(m+1)} \\ \vdots & \vdots & & \vdots \\ z_n(1-z_n)x_{n1} & z_n(1-z_n)x_{n2} & \cdots & z_n(1-z_n)x_{n(m+1)} \end{bmatrix}$$

构造

$$\boldsymbol{Z}=\begin{pmatrix} sigmoid(-\boldsymbol{x_1'w})(1-sigmoid(-\boldsymbol{x_1'w})) \\ sigmoid(-\boldsymbol{x_2'w})(1-sigmoid(-\boldsymbol{x_2'w})) \\ \vdots \\ sigmoid(-\boldsymbol{x_n'w})(1-sigmoid(-\boldsymbol{x_n'w})) \end{pmatrix}$$

则二阶导数可写作

$$\frac{\partial^2 L(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w'}}=\boldsymbol{X'}(\boldsymbol{Z}\bullet\boldsymbol{X})$$

3.2 梯度下降

参数更新的方法为

$$\boldsymbol{w}+=\eta\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}}-\lambda\boldsymbol{w}$$

3.3 牛顿法

参数更新的方法为

$$\boldsymbol{w}+=(\frac{\partial^2 L(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w'}})^{-1}\frac{\partial L\boldsymbol{w}}{\partial \boldsymbol{w}}-\lambda\boldsymbol{w}$$

3.4 数据结构

使用 ndarray 作为主要的数据结构，运用 pandas 库处理 UCI 数据集。

四、 实验结果与分析

4.1 梯度下降

表 **1**　梯度下降分类准确率

正则项系数	混合高斯分布		iris 数据集	adult 数据集
	独立	不独立		
0	0.97	0.98	1.0	0.8069
e^{-10}	0.99	1.0	1.0	0.8069
e^{-8}	0.96	1.0	1.0	0.8080
e^{-6}	0.99	0.97	1.0	0.8069
e^{-4}	0.99	1.0	1.0	0.8080

梯度下降此时分类效果较好，在 adult 数据集上准确率均在 0.80 左右，符合数据集附带文件上提到的错误率。

4.2 牛顿法

表 **2**　牛顿法分类准确率

正则项系数	混合高斯分布		iris 数据集
	独立	不独立	
0	0.99	0.93	1.0
e^{-6}	0.99	0.99	1.0
e^{-4}	1.0	0.96	1.0
e^{-3}	0.97	0.96	1.0
e^{-2}	0.99	0.98	1.0

表 **3**　牛顿法分类 **adult** 数据集

正则项系数	0	e^{-12}	e^{-10}	e^{-8}
正确率	0.7519	0.7519	0.7519	0.7519

牛顿法在处理 adult 数据集时，在求逆矩阵的过程中若正则项系数较大会在求逆过程中出现错误。

五、 结论

混合高斯分布中，即使不满足朴素贝叶斯的条件，但只要各属性间的相关性适当，仍然可以保证模型的效果。iris 数据集，两个类别非常容易区分，符合数据集附带文件的说明。adult 数据集，数据维数多，训练用时长，数据噪声大，根据附带文件的说明，使用朴素贝叶斯的思想分类，误差大约为 0.16，上述训练数据较为合理。

参考文献

[1] CSDN, https://blog.csdn.net/alanwalker1/article/details/112688367, last accessed 2021/10/24

A 主程序

```
import copy

import numpy as np

from data_guass import get_data_guass
from data_uci import get_data_uci

gradient_regulation_radio = 0
newton_regulation_radio = 0


def sigmoid_trick(m):
    for i in range(len(m)):
        a = m[i]
        if a >= 0:
            m[i] = 1 / (1 + np.exp(-1 * a))
        else:
            temp = np.exp(a)
            m[i] = temp / (1 + temp)
    return m


def gradient_descent(train_X, train_Y, validation_X, validation_Y, regulation_radio=0.0):
    dimension_num = len(train_X[0])

    # adam中的超参数，一般使用以下值
    adam_beta_1 = 0.9
    adam_beta_2 = 0.99
    inf_small = 1e-8
    learning_rate = 0.01

    # 计算过程中的累计值
    m = np.zeros(dimension_num)
    v = np.zeros(dimension_num)
    accumulative_adam_beta_1 = adam_beta_1
    accumulative_adam_beta_2 = adam_beta_2

    w = np.array([0.1 for _ in range(dimension_num)])
    temp_w = copy.deepcopy(w)
    accuracy = test(validation_X, w, validation_Y)
    accuracy_list = [accuracy]
    iteration_times = 0

    while True:
        for i in range(10000):
            gradient = np.dot(train_X.T, sigmoid_trick(-1 * np.dot(train_X, w)) - train_Y) -
                regulation_radio * w

            m = adam_beta_1 * m + (1 - adam_beta_1) * gradient
            v = adam_beta_2 * v + (1 - adam_beta_2) * (gradient ** 2)
            M = m / (1 - accumulative_adam_beta_1)
            V = v / (1 - accumulative_adam_beta_2)
```

```python
            accumulative_adam_beta_1 *= adam_beta_1
            accumulative_adam_beta_2 *= adam_beta_2
            w += learning_rate * M / (V ** (1 / 2) + inf_small)

        iteration_times += 1
        accuracy = test(validation_X, w, validation_Y)

        if accuracy < accuracy_list[-1] or iteration_times >= 15:
            break
        else:
            temp_w = copy.deepcopy(w)
            accuracy_list.append(accuracy)

    return temp_w, iteration_times


def Newton_method(train_X, train_Y, validation_X, validation_Y, regulation_radio=0.0):
    dimension_num = len(train_X[0])
    w = np.array([0.1 for _ in range(dimension_num)])
    temp_w = copy.deepcopy(w)
    accuracy = test(validation_X, w, validation_Y)
    accuracy_list = [accuracy]
    iteration_times = 0

    while True:
        for i in range(10000):
            sigmoid = sigmoid_trick(-1 * np.dot(train_X, w))
            a = np.dot(train_X.T, sigmoid - train_Y)
            b = train_X.T @ (np.reshape(sigmoid * (1 - sigmoid), (len(sigmoid), 1)) * train_X)
            delta_w = np.dot(np.linalg.pinv(b), a) - w * regulation_radio
            w += delta_w

        iteration_times += 1
        accuracy = test(validation_X, w, validation_Y)

        if iteration_times >= 15 or accuracy < accuracy_list[-1]:
            break
        else:
            temp_w = copy.deepcopy(w)
            accuracy_list.append(accuracy)

    return temp_w, iteration_times


def test(X, w, Y):
    X = np.array(X)
    Y = np.array(Y)

    predict_result = np.dot(X, w)
    right_num = 0
    for i in range(len(X)):
        if (predict_result[i] >= 0 and Y[i] == 0) or (predict_result[i] < 0 and Y[i] == 1):
            right_num += 1
```

```python
    return float(right_num) / len(X)


if __name__ == '__main__':
    train_X_list, train_Y_list, validation_X_list, validation_Y_list, test_X_list, test_Y_list
        = get_data_guass(100, 20, 100, 0.7, False)
    # train_X_list, train_Y_list, validation_X_list, validation_Y_list, test_X_list,
        test_Y_list = get_data_uci("iris")
    # train_X_list, train_Y_list, validation_X_list, validation_Y_list, test_X_list,
        test_Y_list = get_data_uci("adult")

    train_X = np.array(train_X_list)
    train_Y = np.array(train_Y_list)
    validation_X = np.array(validation_X_list)
    validation_Y = np.array(validation_Y_list)
    test_X = np.array(test_X_list)
    test_Y = np.array(test_Y_list)

    w1, iteration_times_1 = gradient_descent(train_X, train_Y, validation_X, validation_Y,
        gradient_regulation_radio)
    print((test(test_X_list, w1, test_Y_list)), w1, iteration_times_1)

    w2, iteration_times_1 = Newton_method(train_X, train_Y, validation_X, validation_Y,
        newton_regulation_radio)
    print(test(test_X, w2, test_Y), w2, iteration_times_1)
```

B 生成混合高斯分布

```python
import numpy as np


def get_data_guass(train_size, validation_size, test_size, positive_sample_radio, naive=True):
    if naive:
        cov12 = 0.0
    else:
        cov12 = 0.2

    mean_array_positive = [1, 1]
    cov_matrix_positive = [[0.4, cov12], [cov12, 0.3]]

    mean_array_negative = [-1, -1]
    cov_matrix_negative = [[0.4, cov12], [cov12, 0.3]]

    train_X_list, train_Y_list = get_guass_distribution(train_size, int(train_size *
        positive_sample_radio), mean_array_positive, cov_matrix_positive,
                                                        mean_array_negative, cov_matrix_negative)
    validation_X_list, validation_Y_list = get_guass_distribution(validation_size,
        int(validation_size * positive_sample_radio), mean_array_positive,
                                                                  cov_matrix_positive,
                                                                      mean_array_negative,
                                                                      cov_matrix_negative)
    test_X_list, test_Y_list = get_guass_distribution(test_size, int(test_size *
```

```
        positive_sample_radio), mean_array_positive, cov_matrix_positive,
                                        mean_array_negative, cov_matrix_negative)
    return train_X_list, train_Y_list, validation_X_list, validation_Y_list, test_X_list,
        test_Y_list


def get_guass_distribution(total_size, positive_size, mean_array_positive,
     cov_matrix_positive, mean_array_negative, cov_matrix_negative):
    X_list = [(0.0, 0.0)] * total_size
    X_list[:positive_size] = np.random.multivariate_normal(mean_array_positive,
        cov_matrix_positive, size=positive_size)
    X_list[positive_size:] = np.random.multivariate_normal(mean_array_negative,
        cov_matrix_negative, size=total_size - positive_size)
    X_list = [list(sample) for sample in X_list]
    Y_list = [1] * positive_size + [0] * (total_size - positive_size)
    return X_list, Y_list
```

C 读取 uci 数据集

```
import copy

import numpy as np
import pandas as pd

from data.config.adult_config import adult_config
from data.config.iris_config import iris_config


def get_data_uci(uci_data_name):
    if uci_data_name == "iris":
        data_config = iris_config
        train_nrows = 100
        test_nrows = 100
        validation_index = [25, 76]
    elif uci_data_name == "adult":
        data_config = adult_config
        train_nrows = 1000
        test_nrows = 1000
        validation_index = [0, 500]
    else:
        print("尚无此数据集")
        return None

    train_filename = data_config[0]
    test_filename = data_config[1]
    field_list = data_config[2]
    discrete_field_dict = data_config[3]
    label_dict = data_config[4]
    get_log_list = data_config[5]

    train_X_list, train_Y_list = read_uci_data(train_filename, train_nrows, field_list,
        discrete_field_dict, label_dict, get_log_list)
```

```
    if test_filename == train_filename:
        test_X_list, test_Y_list = train_X_list, train_Y_list
    else:
        test_X_list, test_Y_list = read_uci_data(test_filename, test_nrows, field_list,
             discrete_field_dict, label_dict, get_log_list)
    validation_X_list = test_X_list[validation_index[0]:validation_index[1]]
    validation_Y_list = test_Y_list[validation_index[0]:validation_index[1]]

    return train_X_list, train_Y_list, validation_X_list, validation_Y_list, test_X_list,
         test_Y_list


def read_uci_data(filename, nrows, field_list, discrete_field_dict, label_dict, get_log_list):
    columns = copy.deepcopy(field_list)
    columns.append("label")
    raw_data = pd.read_csv(filename, names=columns, nrows=nrows)

    processed_data = []
    processed_data_label = []

    for i in range(len(raw_data)):
        processed_sample = []
        raw_sample = raw_data.iloc[i]

        try:
            for field in field_list:
                if field in discrete_field_dict:
                    processed_sample.append(discrete_field_dict[field][raw_sample.loc[field]])
                else:
                    processed_sample.append(float(raw_sample.loc[field]))
        except KeyError:
            continue
        except ValueError:
            continue

        processed_data_label.append(label_dict[raw_sample.loc["label"]])
        processed_data.append(processed_sample)

    if len(get_log_list) != 0:
        for sample in processed_data:
            for index in get_log_list:
                if sample[index] <= 10:
                    sample[index] = 1
                else:
                    sample[index] = np.log10(sample[index])

            sample.insert(0, 1)
    else:
        for sample in processed_data:
            sample.insert(0, 1)

    return processed_data, processed_data_label
```

D 关于 iris 数据集的配置文件

```
train_filename = "./data/iris/iris.data"
test_filename = "./data/iris/iris.data"
field_list = ["sepal_length", "sepal_width", "petal_length", "petal_width"]
discrete_field_dict = {}
label_dict = {"Iris-setosa": 0, "Iris-versicolor": 1}
get_log_list = []

iris_config = [train_filename, test_filename, field_list, discrete_field_dict, label_dict,
    get_log_list]
```

E 关于 adult 数据集的配置文件

```
work_class_dict = {' Private': 0, ' Self-emp-not-inc': 1, ' Self-emp-inc': 2, ' Federal-gov':
    3, ' Local-gov': 4, ' State-gov': 5, ' Without-pay': 6,
                ' Never-worked': 7, }
education_dict = {' Bachelors': 0, ' Some-college': 1, ' 11th': 2, ' HS-grad': 3, '
    Prof-school': 4, ' Assoc-acdm': 5, ' Assoc-voc': 6, ' 9th': 7,
               ' 7th-8th': 8, ' 12th': 9, ' Masters': 10, ' 1st-4th': 11, ' 10th': 12, '
                   Doctorate': 13, ' 5th-6th': 14, ' Preschool': 15, }
marital_status_dict = {' Married-civ-spouse': 0, ' Divorced': 1, ' Never-married': 2, '
    Separated': 3, ' Widowed': 4, ' Married-spouse-absent': 5,
                    ' Married-AF-spouse': 6, }
occupation_dict = {' Tech-support': 0, ' Craft-repair': 1, ' Other-service': 2, ' Sales': 3, '
    Exec-managerial': 4, ' Prof-specialty': 5,
                ' Handlers-cleaners': 6, ' Machine-op-inspct': 7, ' Adm-clerical': 8, '
                    Farming-fishing': 9, ' Transport-moving': 10,
                ' Priv-house-serv': 11, ' Protective-serv': 12, ' Armed-Forces': 13, }
relationship_dict = {' Wife': 0, ' Own-child': 1, ' Husband': 2, ' Not-in-family': 3, '
    Other-relative': 4, ' Unmarried': 5, }
race_dict = {' White': 0, ' Asian-Pac-Islander': 1, ' Amer-Indian-Eskimo': 2, ' Other': 3, '
    Black': 4, }
sex_dict = {' Female': 0, ' Male': 1, }
native_country_dict = {' United-States': 0, ' Cambodia': 1, ' England': 2, ' Puerto-Rico': 3,
    ' Canada': 4, ' Germany': 5,
                    ' Outlying-US(Guam-USVI-etc)': 6, ' India': 7, ' Japan': 8, ' Greece': 9, '
                        South': 10, ' China': 11, ' Cuba': 12, ' Iran': 13,
                    ' Honduras': 14, ' Philippines': 15, ' Italy': 16, ' Poland': 17, '
                        Jamaica': 18, ' Vietnam': 19, ' Mexico': 20,
                    ' Portugal': 21, ' Ireland': 22, ' France': 23, ' Dominican-Republic': 24,
                        ' Laos': 25, ' Ecuador': 26, ' Taiwan': 27,
                    ' Haiti': 28, ' Columbia': 29, ' Hungary': 30, ' Guatemala': 31, '
                        Nicaragua': 32, ' Scotland': 33, ' Thailand': 34,
                    ' Yugoslavia': 35, ' El-Salvador': 36, ' Trinadad&Tobago': 37, ' Peru': 38,
                        ' Hong': 39, ' Holand-Netherlands': 40, }

train_filename = "./data/adult/adult.data"
test_filename = "./data/adult/adult.test"
field_list = ['age', 'work_class', 'fnlwgt', 'education', 'education_num', 'marital_status',
    'occupation', 'relationship', 'race', 'sex',
            'capital_gain', 'capital_loss', 'hours_per_week', 'native_country']
```

```
discrete_field_dict = {'work_class': work_class_dict, 'education': education_dict,
    'marital_status': marital_status_dict,
                   'occupation': occupation_dict, 'relationship': relationship_dict, 'race':
                       race_dict, 'sex': sex_dict,
                   'native_country': native_country_dict}
label_dict = {" <=50K": 0, " >50K": 1, " <=50K.": 0, " >50K.": 1}
get_log_list = [2, 10, 11]

adult_config = [train_filename, test_filename, field_list, discrete_field_dict, label_dict,
    get_log_list]
```