

机器学习实验---安装

一、实验目的

1. 熟悉运行环境。

二、实验内容

1. 安装 Pycharm, 注册学生版。
2. 安装常见的机器学习库, 如 Scipy、Numpy、Pandas、Matplotlib, sklearn 等。
3. 熟悉 iris 数据集。

三、实验报告要求

1. 按实验内容撰写实验过程;
2. 报告中涉及到的代码, 每一行需要有详细的注释;
3. 按自己的理解重新组织, 禁止粘贴复制实验内容。

四、实验记录

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris # 导入数据
import matplotlib.pyplot as plt
%matplotlib inline
# Load data
iris = load_iris() #获取数据
df = pd.DataFrame(iris.data, columns=iris.feature_names)# 获取列的属性值
df['label'] = iris.target# 增加一个新列
df.columns = ['sepal length', 'sepal width', 'petal length', 'petal width',
'label'] # 重命名各个列
df.label.value_counts() # 计算label 列0、1、2 出现的次数
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
"""
绘制散点图
"""
plt.scatter(df[:50]['sepal length'], df[:50]['sepal width'], label='0')
plt.scatter(df[50:100]['sepal length'], df[50:100]['sepal width'],
label='1')
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.legend()

data = np.array(df.iloc[:100, [0, 1, -1]])# 按行索引, 取前 100 行, 取第 0, 1 列以及
最后 1 列
X, y = data[:, :-1], data[:, -1] #X: {ndarray: (100, 2)} y: {ndarray: (100, )}
y = np.array([1 if i == 1 else -1 for i in y])# 将存在 y 中的数据为 0 的值改为-1
# 数据线性可分, 二分类数据
# 此处为一元一次线性方程
```

```

class Model:
    def __init__(self):# 初始化数据
        self.w = np.ones(len(data[0]) - 1, dtype=np.float32)
        self.b = 0
        self.l_rate = 0.1
        # self.data = data
        """
        numpy.dot() 对于两个一维的数组，计算的是这两个数组对应下标元素的乘积和(数学上称之为内积)；对于二维数组，计算的是两个数组的矩阵乘积；对于多维数组，它的通用计算公式如下，即结果数组中的每个元素都是：数组 a 的最后一维上的所有元素与数组 b 的倒数第二位上的所有元素的乘积和： dot(a, b)[i,j,k,m] = sum(a[i,j,:] * b[k,:,m])。
        """
        def sign(self, x, w, b):
            y = np.dot(x, w) + b
            return y
        # 随机梯度下降法
        def fit(self, X_train, y_train):
            is_wrong = False
            while not is_wrong:
                wrong_count = 0# 初始设置错误次数为0
                for d in range(len(X_train)):
                    X = X_train[d]
                    y = y_train[d]
                    if y * self.sign(X, self.w, self.b) <= 0:
                        self.w = self.w + self.l_rate * np.dot(y, X)
                        self.b = self.b + self.l_rate * y
                        wrong_count += 1
                if wrong_count == 0:# 误分点数目为0 跳出循环
                    is_wrong = True
            return 'Perceptron Model!'
        def score(self):
            pass
perceptron = Model()
perceptron.fit(X, y)

x_points = np.linspace(4, 7,10)
y_ = -(perceptron.w[0]*x_points + perceptron.b)/perceptron.w[1]
plt.plot(x_points, y_)
plt.plot(data[:50, 0], data[:50, 1], 'bo', color='blue', label='0')
plt.plot(data[50:100, 0], data[50:100, 1], 'bo', color='orange', label='1')
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.legend()

```

```
from sklearn.linear_model import Perceptron# 使用 scikit-learn 自带的感知机模型
clf = Perceptron(fit_intercept=False, max_iter=1000, shuffle=False)# 配置
导入的感知机模型
clf.fit(X, y)# 使用上面的训练数据代入模型中进行训练

# Weights assigned to the features.
print(clf.coef_)

# 截距 Constants in decision function.
print(clf.intercept_)

x_ponits = np.arange(4, 8)
y_ = -(clf.coef_[0][0]*x_ponits + clf.intercept_)/clf.coef_[0][1]
plt.plot(x_ponits, y_)
plt.plot(data[:50, 0], data[:50, 1], 'bo', color='blue', label='0')
plt.plot(data[50:100, 0], data[50:100, 1], 'bo', color='orange', label='1')
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.legend()
```

五、运行结果

```

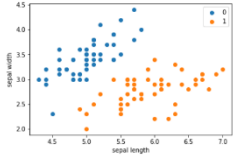
In [14]: import pandas as pd
import numpy as np
from sklearn.datasets import load_iris # 导入数据
import matplotlib.pyplot as plt
%matplotlib inline

In [17]: # load data
iris = load_iris() # 加载数据
df = pd.DataFrame(iris.data, columns=iris.feature_names) # 获取列的别名
df['label'] = iris.target # 增加一个数列

In [3]: df.columns = ['sepal length', 'sepal width', 'petal length', 'petal width', 'label'] # 重新命名各列
df.label.value_counts() # 统计label为0, 1, 2出现的次数

Out[3]:
0    50
1    50
2    50
Name: label, dtype: int64

In [4]: # 绘制散点图
plt.scatter(df[:50]['sepal length'], df[:50]['sepal width'], label='0')
plt.scatter(df[50:100]['sepal length'], df[50:100]['sepal width'], label='1')
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.legend()

Out[4]: Outplotlib.Legend.Legend at 0x4329868370:


In [5]: data = np.array(df.iloc[:100, [0, 1, 17]]) # 按行索引, 取前100行, 数据0, 1列以及最后一列

In [6]: %s, y = data[:, 1:1], data[:, 17] # %s: (ndarray: (100, 2)) y: (ndarray: (100, ))

In [7]: y = np.array([1 if i == 1 else 0 for i in y]) # 将y中的元素为0的值改为-1

In [8]: # 数据线性可分, 训练数据
# 定义一个感知器模型
class Model:
    def __init__(self): # 初始化数据
        self.w = np.zeros(len(data[0]) - 1, dtype=np.float32)
        self.b = 0
        self.lr = 0.1
        # self.data = data

    def sign(self, x, w, b):
        y = np.dot(x, w) + b
        return y

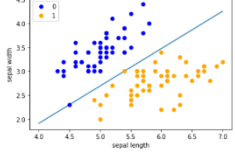
    # 随机梯度下降法
    def fit(self, X_train, y_train):
        is_wrong = False
        while not is_wrong:
            wrong_count = 0
            for i in range(len(X_train)):
                x = X_train[i]
                y = y_train[i]
                if y * self.sign(x, self.w, self.b) <= 0:
                    self.w = self.w + self.lr * np.dot(y, x)
                    self.b = self.b + self.lr * y
                    wrong_count += 1
            if wrong_count == 0: # 误分点数为0跳出循环
                is_wrong = True
            return 'Perceptron Model!'

    def score(self):
        pass

In [9]: perceptron = Model()
perceptron.fit(X, y)

Out[9]: 'Perceptron Model!'

In [10]: x_points = np.linspace(4, 7, 10)
y_ = -(perceptron.w[0]*x_points + perceptron.b)/perceptron.w[1]
plt.plot(x_points, y_)
plt.plot(data[:50, 0], data[:50, 1], 'bo', color='blue', label='0')
plt.plot(data[50:100, 0], data[50:100, 1], 'bo', color='orange', label='1')
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.legend()

Out[10]: Outplotlib.Legend.Legend at 0x432986d600:


In [11]: from sklearn.linear_model import Perceptron # 使用skikit-learn自带的感知器模型

In [12]: clf = Perceptron(fit_intercept=False, max_iter=1000, shuffle=False) # 新建导入的感知器模型
clf.fit(X, y) # 使用上面的训练数据代入模型中进行训练

Out[12]: Perceptron(fit_intercept=False, shuffle=False)

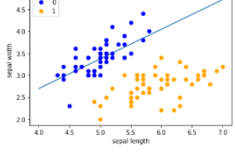
In [13]: # Weights assigned to the features.
print(clf.coef_)

Out[13]:
[[ 16.3 -24.9]]

In [14]: # 截距 Constants in decision function.
print(clf.intercept_)

Out[14]:
[0.]

In [15]: x_points = np.arange(4, 8)
y_ = -(clf.coef_[0][0]*x_points + clf.intercept_)/clf.coef_[0][1]
plt.plot(x_points, y_)
plt.plot(data[:50, 0], data[:50, 1], 'bo', color='blue', label='0')
plt.plot(data[50:100, 0], data[50:100, 1], 'bo', color='orange', label='1')
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.legend()

Out[15]: Outplotlib.Legend.Legend at 0x43298721f0:


```

六、实验小结

本次实验是理解感知机算法的原理并实现感知机算法，感知机称为单层感知机模型，其输入是实例的特征向量，输出为实例的分类类别。它是一种使用阶梯函数激活的人工神经元，以产生二分类输出，用于将数据分为两部分，因此也称为线性二分类器。实验中使用 `jupyterbook` 进行实验，并使用到了 `pandas`、`numpy`、`Matplotlib`、`sklearn` 等机器学习库，可对机器学习有初步理解。