一、代码

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
%matplotlib inline
# load data
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['label'] = iris.target
df.columns = [
    'sepal length', 'sepal width', 'petal length', 'petal width', '
label'
]
df.label.value_counts()

plt.scatter(df[:50]['sepal length'], df[:50]['sepal width'], label
='0') plt.scatter(df[50:100]['sepal length'], df[50:100]['sepal wi
dth'], label='1') plt.xlabel('sepal length') plt.ylabel('sepal wid
th') plt.legend()
data = np.array(df.iloc[:100, [0, 1, -1]])
X, y = data[:,:-1], data[:,-1]
y = np.array([1 if i == 1 else -1 for i in y])

# 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):
        self.w = np.ones(len(data[0]) - 1, dtype=np.float32)
        self.b = 0
        self.l_rate = 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 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:
                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', l
abel='1')
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.legend()
```

二、计算机基础知识

感知机：

$$f(x) = sign(w \cdot x + b)$$

距离公式：空间中任意一点x_0到超平面S的距离：

$$d = \frac{1}{||w||}|w \cdot x_0 + b|$$

其中$||w||$为w的L_2范式。（L_2范式指向量各元素平方和的平方根）

推导：

取空间中任意一点x_0，超平面 S：$w \cdot x + b = 0$，其中x_0、w、x均为 N 维向量。设点x_0到平面S的距离为d，点x_0在平面 S 上的投影点为x_1，则x_1满足$w \cdot x_1 + b = 0$。因为向量$\overrightarrow{x_0x_1}$平行于S平面的法向量w，故有

$$|w \cdot \overrightarrow{x_0x_1}| = |w||\overrightarrow{x_0x_1}| = \sqrt{(w_1)^2 + \cdots + (w_N)^2} \cdot d = ||w|| \cdot d$$

$$w \cdot \overrightarrow{x_0x_1} = w \cdot \overrightarrow{x_1} - w \cdot \overrightarrow{x_0} = -b - w \cdot \overrightarrow{x_0}$$

$$|w \cdot \overrightarrow{x_0 x_1}| = |w \cdot x_0 + b| = ||w|| \cdot d$$

损失函数：误分类点到超平面的距离之和。

$$L(w,b) = -\sum_{x_i \in M} y_i(w \cdot x_i + b)$$

梯度：

$$\nabla_w L(w,b) = -\sum_{x_i \in M} y_i x_i$$

$$\nabla_b L(w,b) = -\sum_{x_i \in M} y_i$$

更新：随机选择一个误分类点(x_i, y_i)，对w, b进行更新：

$$w \longleftarrow w + \eta y_i x_i$$

$$b \longleftarrow b + \eta y_i$$

其中y_i为样本标签

对偶感知机

初始时设置w, b均为 0，逐步修改 n 次，则w, b关于(x_i, y_i)的增量分别是$\alpha_i y_i x_i$和$\alpha_i y_i$，这里$\alpha_i = n_i \eta$。即最后学习到的w, b分别表示为：

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$b = \sum_{i=1}^{N} \alpha_i y_i$$

这里$\alpha_i \geq 0$，当$\eta = 1$时，表示第i个实例点由于误分而进行更新的次数。实例点更新次数越多，意味着它距离分离超平面越近，也就越难正确分类。换句话说，这样的实例对学习结果影响最大。

误分条件：

$$y_i\left(\sum_{j=1}^{N} \alpha_j y_j x_j \cdot x_i + b\right) \leq 0$$

其中$\sum_{j=1}^{N} \alpha_j y_j x_j \cdot x_i = (\alpha_1 y_1 x_1 + \cdots + \alpha_N y_N x_N) \cdot x_i$，$\alpha_j y_j$为常数，因此需要计算

$$\alpha_1 y_1 x_1 \cdot x_1 + \cdots + \alpha_N y_N x_N \cdot x_1$$

$$\ldots$$

$$\alpha_1 y_1 x_1 \cdot x_N + \cdots + \alpha_N y_N x_N \cdot x_N$$

引入 **Gram** 矩阵：$G = [x_i \cdot x_j]_{N \times N} = \begin{bmatrix} x_1 x_1 & \cdots & x_1 x_N \\ \vdots & \ddots & \vdots \\ x_N x_1 & \cdots & x_N x_N \end{bmatrix}$

即上述计算为：

$$[\alpha_1 y_1 \quad \ldots \quad \alpha_N y_N] \begin{bmatrix} x_1 x_1 & \cdots & x_1 x_N \\ \vdots & \ddots & \vdots \\ x_N x_1 & \cdots & x_N x_N \end{bmatrix}$$

定理 2.1(2)：感知机误分类的次数是有上界的。（证明跳过）

三、开源项目

1、建立了每日作业和力扣打卡的专属目录，每日打卡后上传文件。
2、完成一次 PR 同步各成员项目。