第2章 感知机

1．感知机是根据输入实例的特征向量x对其进行二类分类的线性分类模型：

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

感知机模型对应于输入空间（特征空间）中的分离超平面$w\cdot x+b=0$。

2．感知机学习的策略是极小化损失函数：

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

损失函数对应于误分类点到分离超平面的总距离。

3．感知机学习算法是基于随机梯度下降法的对损失函数的最优化算法，有原始形式和对偶形式。算法简单且易于实现。原始形式中，首先任意选取一个超平面，然后用梯度下降法不断极小化目标函数。在这个过程中一次随机选取一个误分类点使其梯度下降。

4．当训练数据集线性可分时，感知机学习算法是收敛的。感知机算法在训练数据集上的误分类次数k满足不等式：

$$k\leqslant\left(\frac{R}{\gamma}\right)^2$$

当训练数据集线性可分时，感知机学习算法存在无穷多个解，其解由于不同的初值或不同的迭代顺序而可能有所不同。

二分类模型

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

$$\operatorname{sign}(x)=\begin{cases}+1, & x\geqslant 0\\ -1, & x<0\end{cases}$$

给定训练集：

$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$

定义感知机的损失函数

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

算法

随即梯度下降法 Stochastic Gradient Descent

随机抽取一个误分类点使其梯度下降。

$w=w+\eta y_i x_i$

$$b = b + \eta y_i$$

当实例点被误分类，即位于分离超平面的错误侧，则调整w, b的值，使分离超平面向该无分类点的一侧移动，直至误分类点被正确分类

拿出iris数据集中两个分类的数据和[sepal length，sepal width]作为特征

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
%matplotlib inline
```

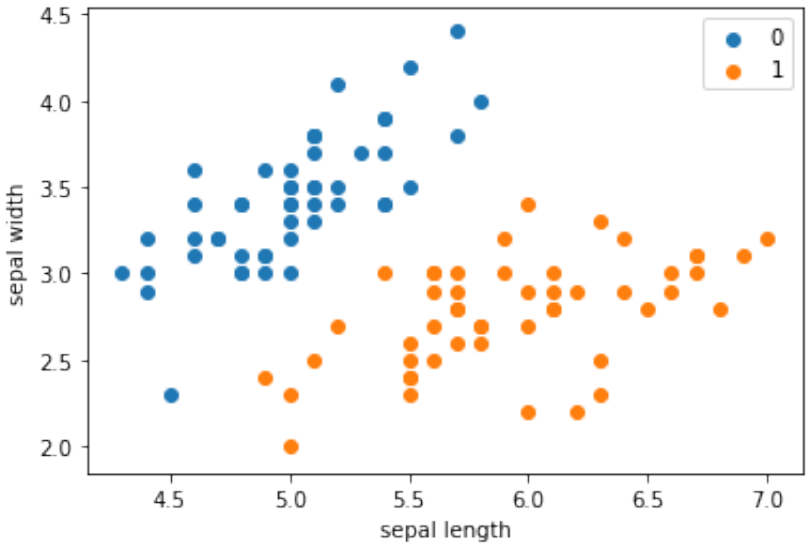
```
# load data
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['label'] = iris.target
```

```
df.columns = [
    'sepal length', 'sepal width', 'petal length', 'petal width', 'label'
]
df.label.value_counts()
```

```
2    50
1    50
0    50
Name: label, dtype: int64
```

```
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()
```

```
<matplotlib.legend.Legend at 0x1781578e588>
```




```
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])
```

Perceptron

```
# 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:
```

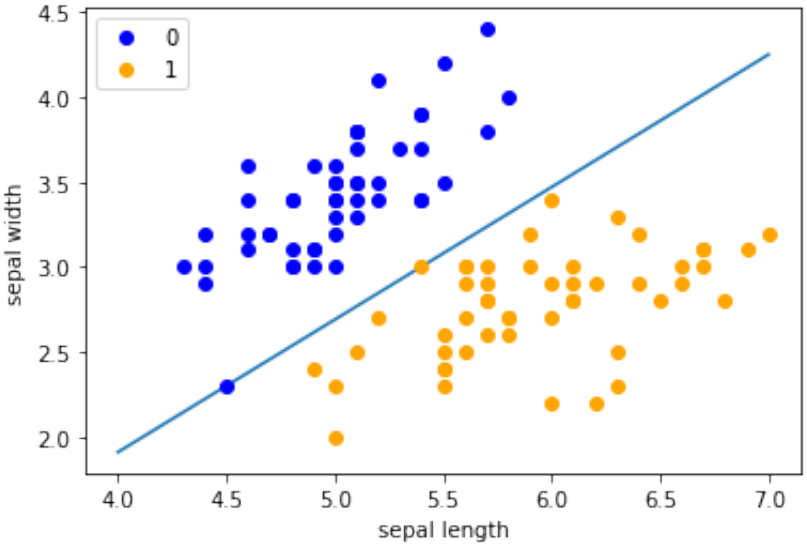
```
                    self.w = self.w + self.l_rate * np.dot(y, X)
                    self.b = self.b + self.l_rate * y
                    wrong_count += 1
            if wrong_count == 0:
                is_wrong = True
        return 'Perceptron Model!'

    def score(self):
        pass
```

```
perceptron = Model()
perceptron.fit(X, y)
```

```
'Perceptron Model!'
```

```
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()
```

```
<matplotlib.legend.Legend at 0x178158adfc8>
```



scikit-learn实例

```
import sklearn
from sklearn.linear_model import Perceptron
```

```
sklearn.__version__
```

```
'0.23.1'
```

```
clf = Perceptron(fit_intercept=True,
                 max_iter=1000,
                 shuffle=True)
clf.fit(X, y)
```

```
Perceptron()
```

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

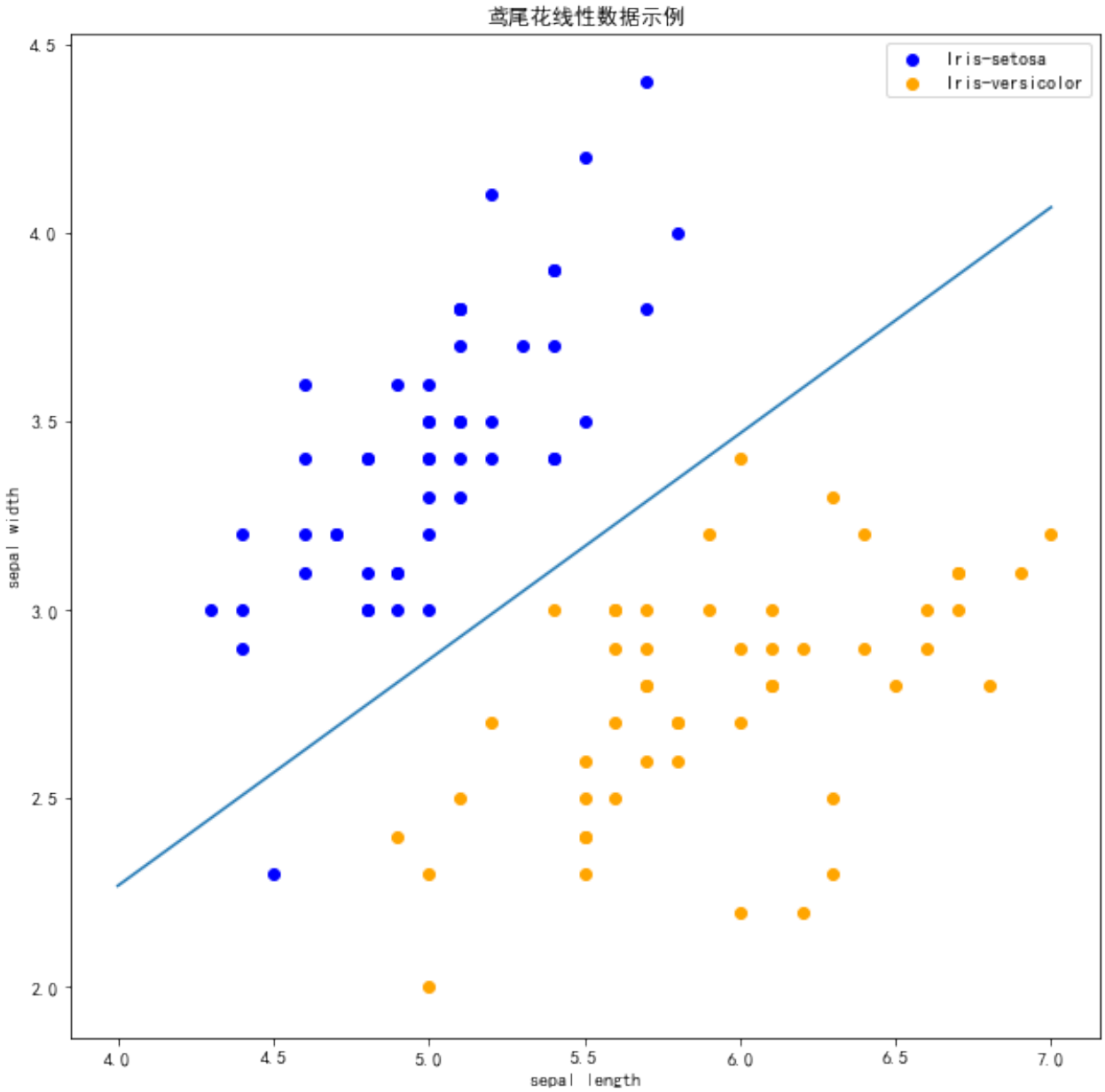
```
[[ 23.2 -38.7]]
```

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

```
[-5.]
```

```
# 画布大小
plt.figure(figsize=(10,10))

# 中文标题
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.title('鸢尾花线性数据示例')

plt.scatter(data[:50, 0], data[:50, 1], c='b', label='Iris-setosa',)
plt.scatter(data[50:100, 0], data[50:100, 1], c='orange', label='Iris-
versicolor')

# 画感知机的线
x_ponits = np.arange(4, 8)
y_ = -(clf.coef_[0][0]*x_ponits + clf.intercept_)/clf.coef_[0][1]
plt.plot(x_ponits, y_)

# 其他部分
plt.legend()  # 显示图例
plt.grid(False)  # 不显示网格
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x17815902248>
```



注意！

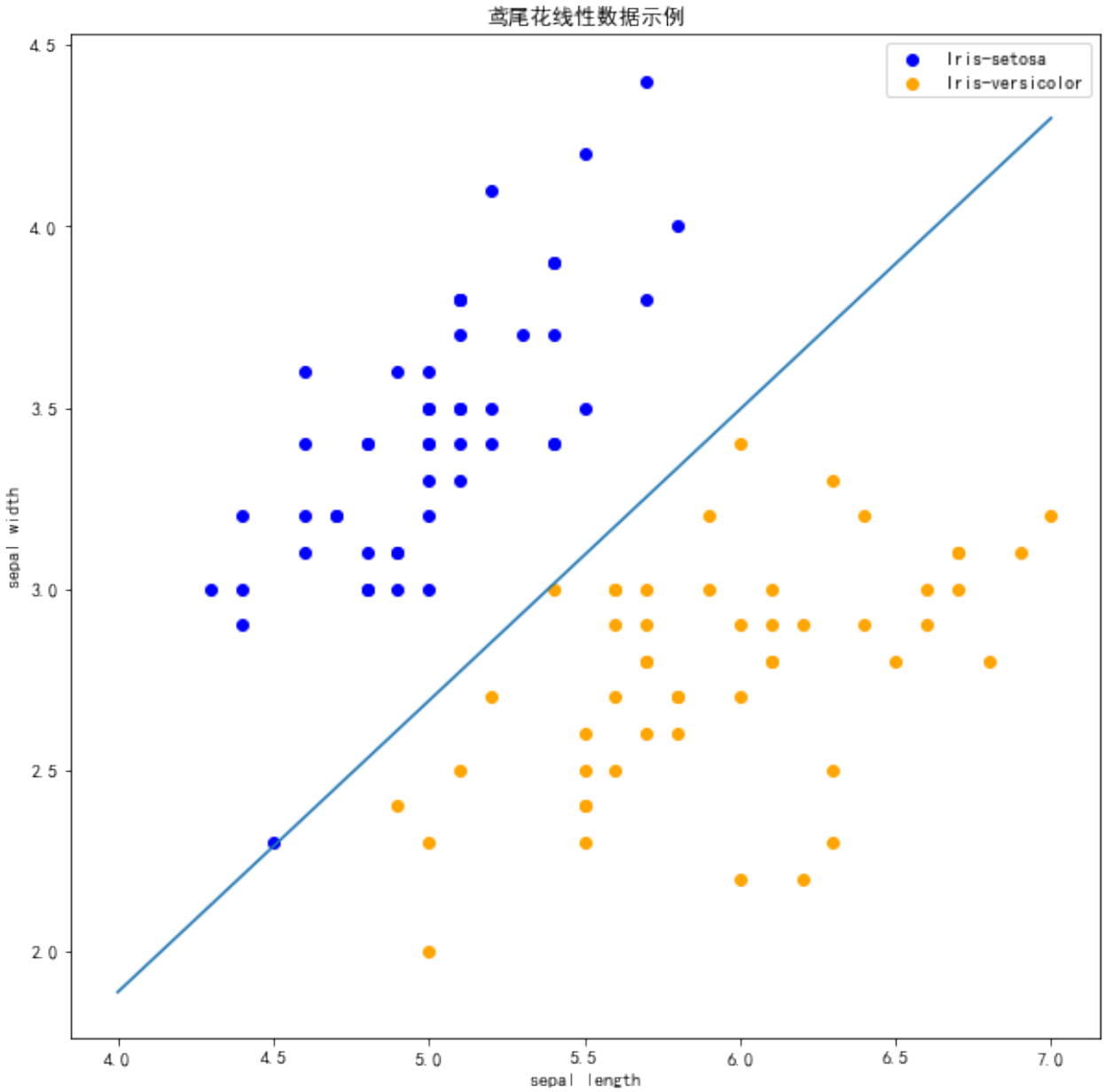
在上图中，有一个位于左下角的蓝点没有被正确分类，这是因为 SKlearn 的 Perceptron 实例中有一个`tol`参数。

`tol` 参数规定了如果本次迭代的损失和上次迭代的损失之差小于一个特定值时，停止迭代。所以我们需要设置 `tol=None` 使之可以继续迭代：

```
clf = Perceptron(fit_intercept=True,
                 max_iter=1000,
                 tol=None,
                 shuffle=True)
clf.fit(X, y)

# 画布大小
plt.figure(figsize=(10,10))

# 中文标题
```

```
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.title('鸢尾花线性数据示例')

plt.scatter(data[:50, 0], data[:50, 1], c='b', label='Iris-setosa',)
plt.scatter(data[50:100, 0], data[50:100, 1], c='orange', label='Iris-
versicolor')

# 画感知机的线
x_ponits = np.arange(4, 8)
y_ = -(clf.coef_[0][0]*x_ponits + clf.intercept_)/clf.coef_[0][1]
plt.plot(x_ponits, y_)

# 其他部分
plt.legend()  # 显示图例
plt.grid(False)  # 不显示网格
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1781588e608>
```



现在可以看到，所有的两种鸢尾花都被正确分类了。

第2章感知机-习题

习题2.1

Minsky 与 Papert 指出：感知机因为是线性模型，所以不能表示复杂的函数，如异或 (XOR)。验证感知机为什么不能表示异或。

解答：

对于异或函数XOR，全部的输入与对应的输出如下：

$x^{(1)}$	$x^{(2)}$	y
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

参考代码：https://github.com/wzyonggege/statistical-learning-method

本文代码更新地址：https://github.com/fengdu78/lihang-code

习题解答：https://github.com/datawhalechina/statistical-learning-method-solutions-manual

中文注释制作：机器学习初学者公众号：ID:ai-start-com

配置环境：python 3.5+

代码全部测试通过。