# <机器学习>课程 Lecture2 实验

## 感知器

给定一组数据,其输入维度为2,输出维度为1,完成二分类任务.

请参考之前的代码和PPT内容,手动实现感知器模型(仅可以使用numpy),并完成分类.

首先加载数据.

In [6]:
```
import numpy as np

x = np.array([
    [-0.5, 0.5],
    [-0.5, 0.5],
    [0.3, -0.5],
    [0.0, 1.0]
])
y = np.array([1, 1, -1, -1])
```

参考之前实验中梯度下降算法的实现,实现感知器模型.

In [7]:
```
import numpy as np

class Perceptron:
    """
    感知器模型
    """
    def __init__(self,
                 c_in: int,
                 init_mean: float, init_var: float,
                 with_bias: bool = True
                 ) -> None:
        self.c_in = c_in
        self.with_bias = with_bias

        if with_bias:
            weight_size = (c_in + 1, 1)
        else:
            weight_size = (c_in, 1)

        # 初始化参数
        self.weight = np.random.normal(
            init_mean, init_var,
            size=weight_size)

    def predict(self,
                x: np.ndarray,
                ) -> np.ndarray:
        # [C] -> [B, C]
        if len(x.shape) == 1:
            x = x[np.newaxis, :]
```

```
        # 对于存在bias的情况进行扩充,添加bias的指数1
        if x.shape[1] == self.c_in and self.with_bias:
            b = x.shape[0]
            x = np.concatenate([x, np.ones((b, 1))], axis=1)

        probs = np.matmul(x, self.weight)

        # 使用阶跃函数,大于0为+1,小于0为-1
        return np.sign(probs)

    def fit(self,
            x: np.ndarray, y: np.ndarray,
            step: float = 0.001,
            epochs: int = 100
            ) -> None:
        '''
        使用梯度下降法拟合感知器模型。
        :param x: 输入特征矩阵，形状为 (n_samples, n_features)
        :param y: 目标标签向量，形状为 (n_samples,)
        :param step: 学习率
        :param epochs: 训练轮数
        '''
        b = x.shape[0]  # 样本数
        if len(y.shape) == 1:
            y = y[:, np.newaxis]  # 将 y 转换为列向量

        # 如果使用偏置项，将偏置项添加到输入特征中
        if x.shape[1] == self.c_in and self.with_bias:
            x = np.concatenate([x, np.ones((b, 1))], axis=1)

        # 梯度下降训练
        for epoch_idx in range(epochs):
            for b_idx in range(b):
                # 预测结果
                preds = np.sign(np.dot(x[b_idx], self.weight))  # 计算预测值

                # 计算梯度更新值，并将其转换为列向量
                update_val = step * (y[b_idx] - preds) * x[b_idx][:, np.newaxis]

                # 更新权重
                self.weight += update_val

# 示例数据
x = np.array([[1, 2], [2, 3], [3, 4], [4, 5], [5, 6], [6, 7]])
y = np.array([-1, -1, -1, 1, 1, 1])

# 训练感知器
model = Perceptron(2, 0.5, 0, 0.5)
print("Initial weights:\n", model.weight)

model.fit(x, y, step=0.01, epochs=1000)
print("Trained weights:\n", model.weight)

# 预测
predictions = model.predict(x)[:, 0]
print("True labels:", y)
print("Predictions:", predictions)
```

```
Initial weights:
 [[0.5]
 [0.5]
 [0.5]]
Trained weights:
 [[ 0.24]
 [-0.2 ]
 [ 0.06]]
True labels: [-1 -1 -1  1  1  1]
Predictions: [-1. -1. -1.  1.  1.  1.]
```