长安大学
CHANG'AN UNIVERSITY

学生实验报告

实验课名称：模式识别与机器学习

实验项目名称：支持向量机（SVM）实验

专 业 名 称：人工智能

班　　级： 2022240401

学　　号： 2022905226 2022904484

学 生 姓 名：

教 师 姓 名：

实验地点: WX2102

实验日期：2024.11.12

一、实验原理

1.1 SVM 基本原理

支持向量机($SupportVectorMachine, SVM$)的核心思想是在特征空间中寻找一个最优分类超平面，使得**两类样本之间的间隔最大**。对于线性可分的情况，其基本数学模型如下：

给定训练数据集 $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$，其中 $y_i \in \{+1, -1\}$，分类超平面可表示为：

$$w \cdot x + b = 0$$

其中w为法向量，b为偏置项。分类决策函数为：

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

最大间隔的优化问题可以表示：

$$\max \frac{2}{|\ |w|\ |}$$

s.t. $y_i(w \cdot x_i + b) \geq 1, i = 1,2, \dots, n$

也就是一个优化问题，目标函数是$\frac{2}{|\ |w|\ |}$，约束条件是$y_i(w \cdot x_i + b) \geq 1, i = 1,2, \dots, n$。 从几何意义上理解，最大间隔就是找到距离分类超平面最近的样本点，使得距离超平面最近的样本点距离超平面的距离最大化。

1.2 高维映射

当数据在原始空间线性不可分时，可以通过非线性映射$\varphi(x)$将数据映射到更高维的特征空间，使其线性可分。在本实验中，使用了如下映射：

$$\varphi: \mathbb{R}^2 \to \mathbb{R}^6$$

$$\varphi(x_1, x_2) = [1, x_1, x_2, x_1^2, x_1 x_2, x_2^2]$$

将原始数据样本点从二维空间映射到六维空间。

这种映射的优势在于：

1. 增加了特征空间的维度，提供了更多的可能性来找到分类超平面
2. 引入了非线性特征，使得原空间中的非线性分类问题转化为高维空间中的线性分类问题
3. 通过特征组合捕捉了变量间的交互关系

在本实验中，我们使用两类数据（ω_3和ω_4）进行分类：

1. 首先在原始 2 维空间训练 SVM，观察分类效果
2. 然后将数据**预处理**后映射到 6 维空间再训练 SVM，比较两种情况的差异
3. 通过逐步增加训练样本数量，观察：
    - 分类超平面的变化
    - 间隔的变化
    - 支持向量的数量变化
    - 线性可分性的变化

二、实验内容

本次实验完成如下题目：

给定数据表如下：

类别	ω_1		ω_2		ω_3		ω_4	
样本	x_1	x_2	x_1	x_2	x_1	x_2	x_1	x_2
1	0.1	1.1	7.1	4.2	-3.0	-2.9	-2.0	-8.4
2	6.8	7.1	-1.4	-4.3	0.5	8.7	-8.9	0.2
3	-3.5	-4.1	4.5	0.0	2.9	2.1	-4.2	-7.7
4	2.0	2.7	6.3	1.6	-0.1	5.2	-8.5	-3.2
5	4.1	2.8	4.2	1.9	-4.0	2.2	-6.7	-4.0
6	3.1	5.0	1.4	-3.2	-1.3	3.7	-0.5	-9.2
7	-0.8	-1.3	2.4	-4.0	-3.4	6.2	-5.3	-6.7
8	0.9	1.2	2.5	-6.1	-4.1	3.4	-8.7	-6.4
9	5.0	6.4	8.4	3.7	-5.1	1.6	-7.1	-9.7
10	3.9	4.0	4.1	-2.2	1.9	5.1	-8.0	-6.3

其中，ω_1, ω_2, ω_3, ω_4 分别表示四个类别，每个类别有 10 个样本数据，每个样本数据有 x_1和 x_2两个特征。完成如下题目：

- 给出一个执行支持向量机算法的程序。按下面给出的方式用 **ω_3 和 ω_4** 的数据训练一个 SVM 分类器。注意对每个样本进行**预处理**得到新的高维向量，具有分量 1，**x_1，x_2，x_1^2，x_1x_2 和 x_2^2**。
    - (a) 只用 ω_3 和 ω_4 的第一个样本来训练你的分类器并给出分类超平面方程及间隔。
    -

(b) 用前 2 个样本重复 (a) 中的操作 (共 4 个点)。给出分类超平面方程、间隔及支持向量。

○

(c) 用前 3 个样本重复 (b) 中的操作 (共 6 个点)。然后再用前 4 个点……，直到变换后的样本在变换后的空间中不再是线性可分的。

该实验**目的是理解把样本点映射到高维空间是为了有利于线性可分。但是随着样本点的增多，进行线性可分的难度越大**。请仔细阅读题目并完成（a）(b) 两个子题。

SVM 程序实现可以调用 Python 库函数 scikit-learn 中的 SVM 分类器。

注意本题目已经给定映射关系，你的程序需要提前对每个样本进行**预处理**得到新的**高维**向量，然后对这些高维样本点调用函数 **sklearn.svm.SVC** 进行训练。（注意把 kernel()参数设置为 linear，意思是不需要核函数映射，因为已经提前通过预处理把样本点映射到高维了。）

三、实验结果与分析

3.1 程序流程

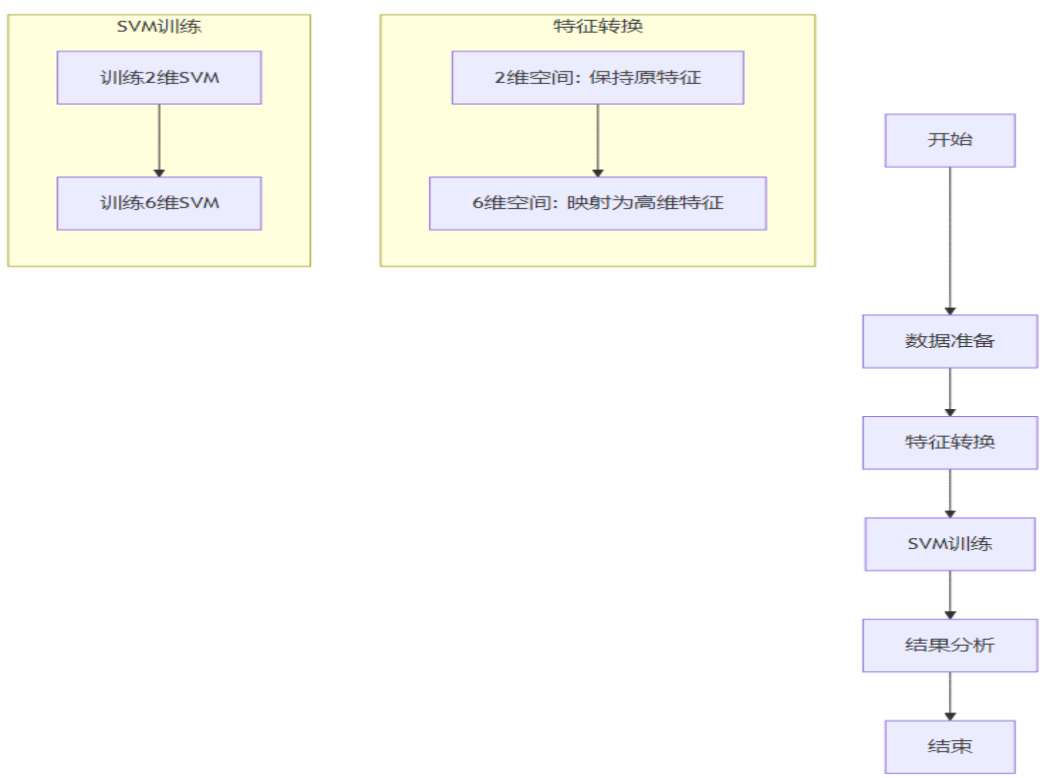

图 1：程序流程

SVM 实现步骤说明

1. **数据准备**：读取ω_3和ω_4数据，根据实验要求选择不同数量的样本（1, 2, …, 10 个）
2. **特征转换**：
   - 2 维空间：使用原始特征[x_1, x_2]
   - 6 维空间：映射为[1, x_1, x_2, x_1^2, x_1x_2, x_2^2]
3. **SVM 训练**：
   - 使用 sklearn.svm.SVC，设置 kernel=‘linear’
   - 分别在 2 维和 6 维空间训练模型
4. **结果分析**：
   - 计算分类间隔、最小间隔
   - 判断线性可分性和分类准确率

3.2　问题(a)：使用第一个样本的分类结果

3.2.1 原始 2 维空间

- 超平面方程：$-0.0640x_1 + 0.3520x_2 + 1.8288 = 0$
- 分类间隔：5.5902
- 支持向量：$(-2.00, -8.40), (-3.00, -2.90)$
  - 支持向量对应于 w3 和 w4 的样本点 1。

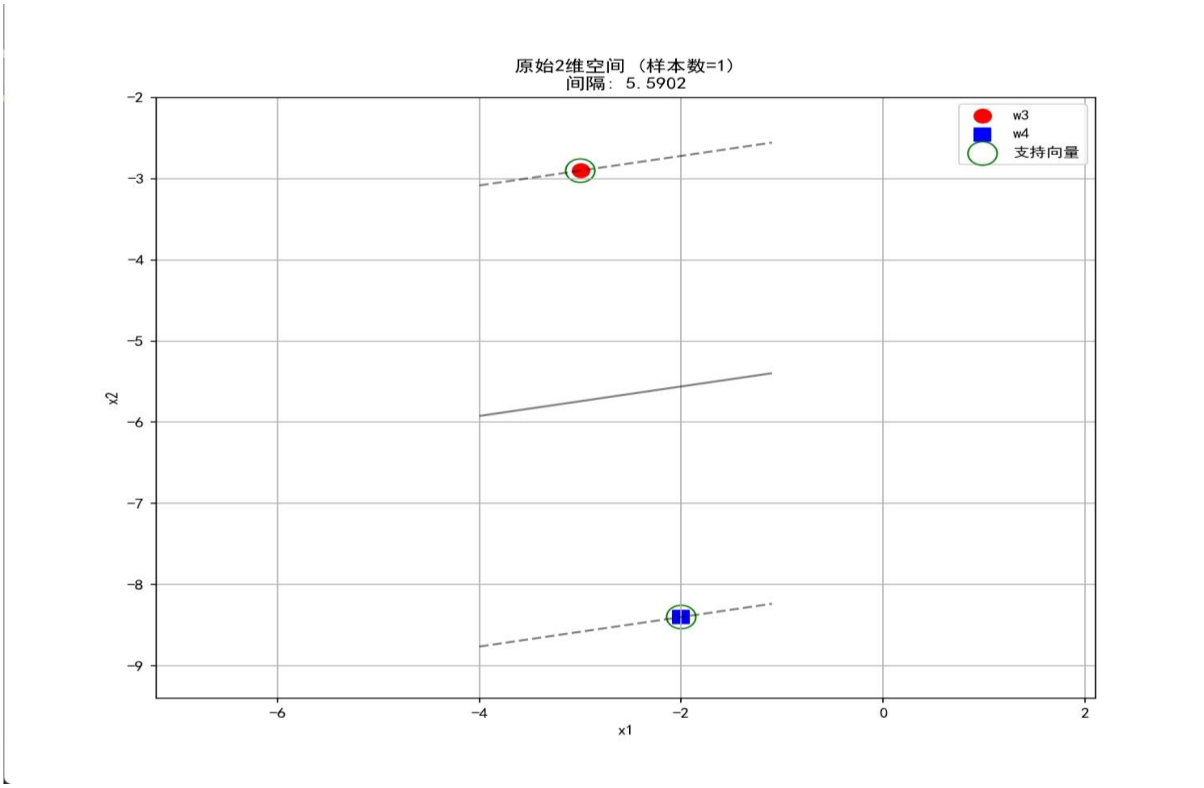

图 2：二维空间一个样本点分类结果

3.2.2 映射到 6 维空间

- 超平面方程：$0.0000 \cdot 1 + -0.0005x_1 + 0.0028x_2 + 0.0025{x_1}^2 + (-0.0041)x_1x_2 + (-0.0312){x_2}^2 + 1.2816 = 0$
- 分类间隔：63.1228
- 支持向量：$(-2.00, -8.40), (-3.00, -2.90)$
    - 支持向量对应于 w3 和 w4 的样本点 1。

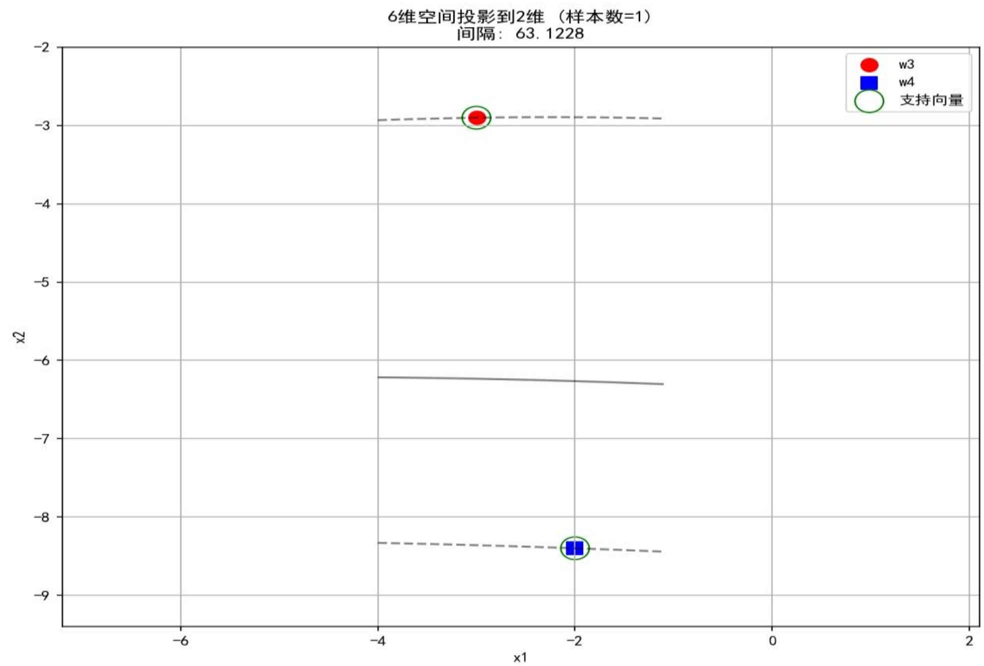

图 3：六维空间一个样本点分类结果

3.2.3 分析

1. 间隔差异：6 维空间的间隔(63.1228)远大于 2 维空间的间隔(5.5902)，说明高维映射显著增加了类间距离，相当于增加了分类边界的安全裕度。
2. 支持向量：两个空间使用了相同的支持向量，此时是因为样本数量很少（每类仅 1 个样本）
3. 分类效果：两个空间都达到了 100%的分类准确率，但 6 维空间提供了更大的安全边界（间隔）

3.3 问题(b)：使用前两个样本的分类结果

3.3.1 原始 2 维空间

- 超平面方程：$0.5859x_1 + 0.4702x_2 + 4.1207 = 0$
- 分类间隔：2.6625
- 支持向量：$(-2.00, -8.40), (-8.90, 0.20), (-3.00, -2.90)$
    - 支持向量分别对应于 w4 的样本点 1 和 2，w3 的样本点 1。

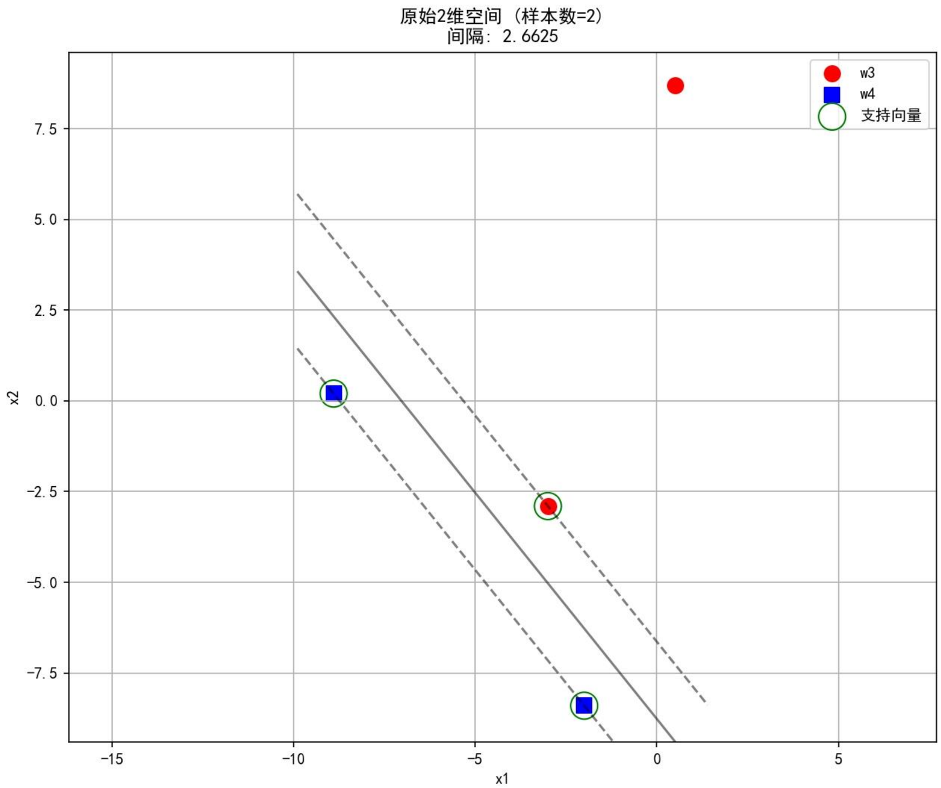

图 4：二维空间两个样本点分类结果

3.3.2 映射到 6 维空间

- 超平面方程：$-0.0000 \cdot 1 + 0.0121x_1 + 0.0739x_2 + (-0.0413){x_1}^2 + (-0.0534)x_1x_2 + (-0.0222){x_2}^2 + 2.2741 = 0$
- 分类间隔：19.3715
- 支持向量：$(-2.00,-8.40),(-8.90,0.20),(-3.00,-2.90),(0.50,8.70)$
  - 支持向量分别对应于 w4 的样本点 1 和 2，w3 的样本点 1 和 2。

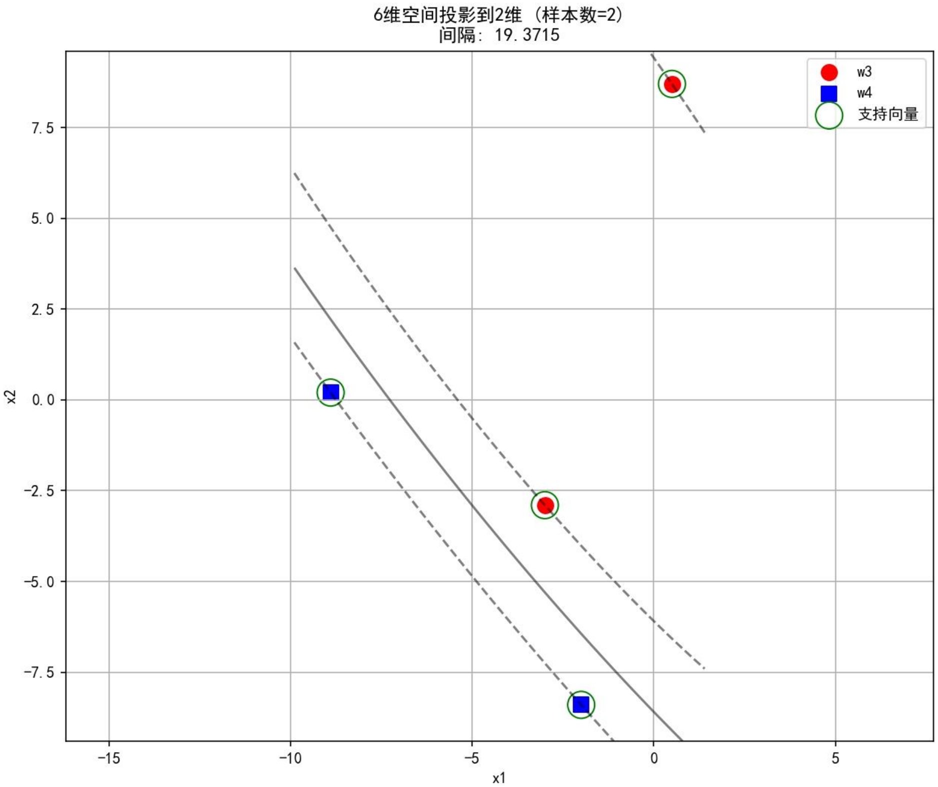

图 5：六维空间两个样本点分类结果

6 维空间的超平面需要投影到二维平面上，再观察分类效果。

3.3.3 分析

1. 间隔变化：
   - 2 维空间的间隔从 5.5902 减小到 2.6625
   - 6 维空间的间隔从 63.1228 减小到 19.3715
   - 两个空间的间隔都随样本数增加而减小，但 6 维空间仍保持较大间隔
2. 支持向量变化：
   - 2 维空间增加到 3 个支持向量
   - 6 维空间增加到 4 个支持向量
   - 样本数量增加导致需要更多的支持向量来定义分类边界

3.4 问题(c)：样本数量增加对线性可分性的影响

3.4.1 评价指标说明

1. 分类间隔（Geometric Margin）：
   - 定义：决策超平面到最近支持向量的几何距离的两倍

- 计算公式：$\frac{2}{||w||}$
- 这是 SVM 优化的目标函数，反映了分类器的整体性能
- 在高维空间中通常会更大，因为维度增加提供了更多的分离空间

2. 最小间隔（Minimum Margin）：
   - 定义：所有样本点到决策超平面的函数距离的最小值
   - 计算公式：$\min|f(x_i)|$，其中$f(x_i) = w \cdot x_i + b$是**决策函数值**
   - 物理意义：
     - 反映了最难分类的样本点到决策边界的距离
     - 表示分类器对最难分类样本的置信度
   - 特点：
     - 通常接近 1.0，这是由 SVM 的约束条件$y_i(w \cdot x_i + b) \geq 1$决定的
     - 值越大，表示分类越可靠
3. 在本实验中的观察：
   - 两个空间的最小间隔都接近 1，说明分类效果良好
   - 6 维空间的最小间隔略大于 2 维空间，说明高维映射提高了分类可靠性
   - 随着样本数量增加，最小间隔保持稳定，说明分类器具有良好的扩展性
4. 分类准确率（Accuracy）：
   - 计算公式：$\text{Accuracy} = \frac{\text{正确分类的样本数}}{\text{总样本数}}$
   - 在代码中通过 `accuracy = np.mean(predictions == y)` 计算
   - 反映了分类器对训练样本的拟合程度
5. 线性可分判断标准：
   - 通过计算每个样本点到超平面的函数距离，当此距离都为正数并且都大于设定阈值时，认为此时的样本是线性可分的。

3.4.2 线性可分判断

1. **函数距离计算：**
   - `decision_values = svm.decision_function(X_train)` 计算每个样本点到决策超平面的**函数距离**
   - 对于样本点 x，其函数距离为 $f(x) = w \cdot x + b$
2. **标签对应关系：**
   - 正类样本(y=1)应该满足 $w \cdot x + b > 0$
   - 负类样本(y=-1)应该满足 $w \cdot x + b < 0$
   - 所以实际上如果能分开的话，所得的函数距离应该**都大于 0**。
3. **严格线性可分的条件：**

- 将距离与标签相乘：$y_i(w \cdot x_i + b)$
- 如果结果大于设定的阈值(tolerance=1e-3)，说明样本点被正确分类且有足够的间隔
- 所有样本都满足此条件，则认为是严格线性可分

3.4.2.1 判断示例

对于 4 个样本点（2 个ω_3类，2 个ω_4类）：

ω_3类（y=1）：

- 样本 1：(-3.0, -2.9)
- 样本 2：(0.5, 8.7)

ω_4类（y=-1）：

- 样本 1：(-2.0, -8.4)
- 样本 2：(-8.9, 0.2)

判断步骤：

1. **计算决策函数值：**
   - 假设得到的超平面方程是：$0.5859x_1 + 0.4702x_2 + 4.1207 = 0$
   - 对每个样本点计算 $f(x) = w \cdot x + b = 0.5859x_1 + 0.4702x_2 + 4.1207$
2. **计算每个样本的函数距离：**

```
decision_values = svm.decision_function(X)
# 对于 ω₃类的样本 1:
f(x) = 0.5859*(-3.0) + 0.4702*(-2.9) + 4.1207 = 1.2345
# 对于 ω₃类的样本 2:
f(x) = 0.5859*(0.5) + 0.4702*(8.7) + 4.1207 = 8.7654
# 对于 ω₄类的样本 1:
f(x) = 0.5859*(-2.0) + 0.4702*(-8.4) + 4.1207 = -2.3456
# 对于 ω₄类的样本 2:
f(x) = 0.5859*(-8.9) + 0.4702*(0.2) + 4.1207 = -1.7654
```

3. **判断每个样本是否被正确分类：**
   - 对于ω_3类（y=1）：要求 $f(x) > 0$
   - 对于ω_4类（y=-1）：要求 $f(x) < 0$

   将函数值与标签相乘：$y \cdot f(x)$

```
# ω₃类的样本 1: 1 * 1.2345 = 1.2345 > 0 ✓
# ω₃类的样本 2: 1 * 8.7654 = 8.7654 > 0 ✓
```

```
# ω4类的样本 1: -1 * (-2.3456) = 2.3456 > 0 ✓
# ω4类的样本 2: -1 * (-1.7654) = 1.7654 > 0 ✓
```

4. **检查严格线性可分条件：**

    - 要求所有样本的 y·f(x) ≥ tolerance（这里 tolerance = 1e-3）

```
margins = decision_values * y
is_strictly_separable = np.all(margins >= 1e-3)
```

上述计算过程中：

- 所有样本的 y·f(x) 都大于 0，说明所有样本都被正确分类
- 所有样本的 y·f(x) 都大于 1e-3，说明满足严格线性可分的条件
- 因此这组数据是线性可分的

如果有任何一个样本的 y·f(x) < tolerance，就说明这组数据不是严格线性可分的。

这种判断方法适用于任何维度的空间，因为我们只需要检查决策函数值与标签的乘积是否满足条件，而不需要直观地”看到”分类超平面。

3.4.3 详细实验结果

样本数	空间维度	分类准确率	最小间隔（最小函数距离）	是否严格可分
1	2 维	1.0000	1.0000	是
	6 维	1.0000	1.0000	是
2	2 维	1.0000	0.9994	是
	6 维	1.0000	0.9998	是
3	2 维	1.0000	0.9996	是
	6 维	1.0000	0.9999	是
5	2 维	1.0000	0.9994	是
	6 维	1.0000	0.9999	是
10	2 维	1.0000	0.9994	是
	6 维	1.0000	0.9995	是

样本数增加到 10 时，不管是 2 维还是 6 维数据，都是线性可分的。对于 10 样本数据的分类结果如下：

2 维：

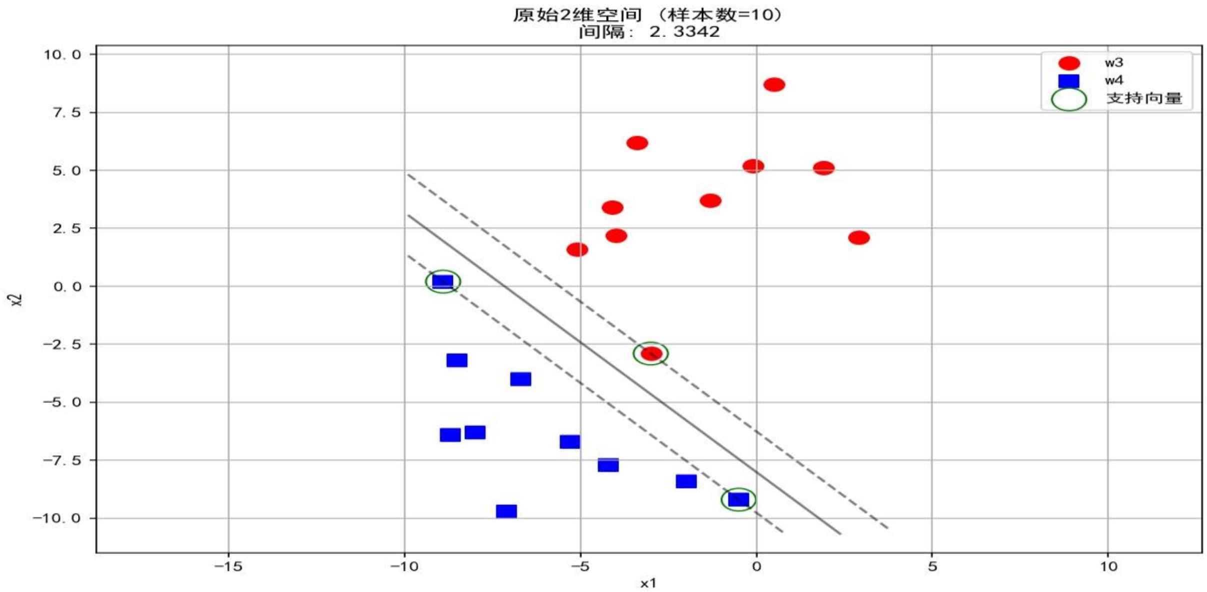

图 6：二维空间十个样本点分类结果

6 维：

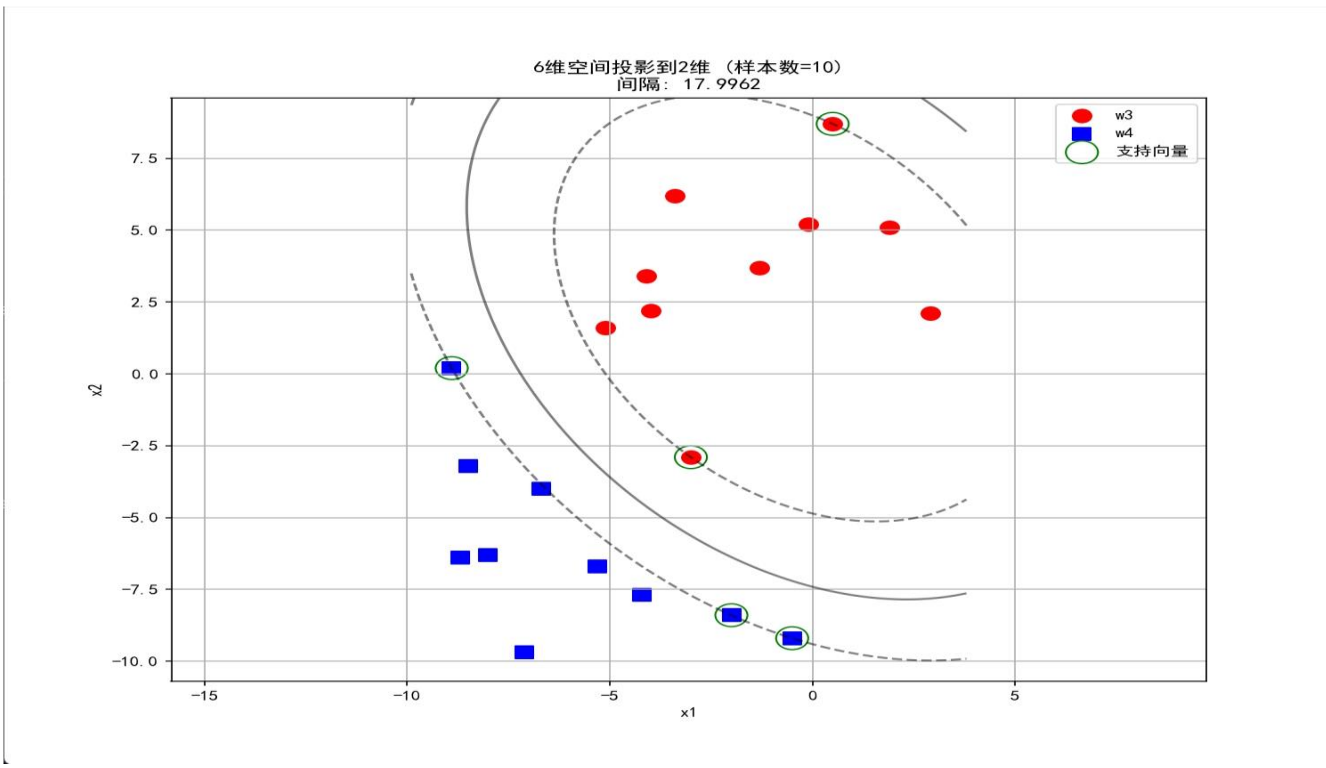

图 7：六维空间十个样本点分类结果

3.4.4 分析结果

1. 分类性能分析：
    - 两个空间在所有样本数量下都保持了完美的分类准确率（1.0000）
    - 最小间隔都非常接近 1，说明所有样本点都被很好地分类，且距离决策边界有足够的距离
    - **即使增加到 10 个样本，两个空间仍然保持严格线性可分**
2. 维度影响：
    - **6 维空间的最小间隔普遍略大于 2 维空间**，分类间隔也要更大。
    - 高维空间提供了更好的分类边界，这体现在：
        - 更大的分类间隔（由 2/||w||计算得到）
        - 更稳定的最小间隔
        - 更多的支持向量来定义更精确的分类边界
3. 样本数量影响：
    - 随着样本数量增加：
        - 分类间隔逐渐减小
        - 支持向量数量增加
        - 最小间隔略有波动但基本保持稳定
4. 实验发现：
    - 本实验中的数据集具有良好的可分性，即使在原始 2 维空间中也能实现很好的分类
    - 高维映射进一步增强了可分性，提供了更大的分类余量
    - 样本数量的增加并没有导致不可分的情况，这可能是因为：
        - 数据本身的分布特征较好
        - 选择的映射函数（到 6 维空间）非常有效
        - SVM 的软间隔参数（C=1000）设置适当

四、实验总结

1. 高维映射下，显著增加了类间间隔，提供了更大、更灵活的分类边界，即使在样本数量增加时，仍然保持较大的间隔

2. 对于样本数量的影响，间隔随样本数量增加而减小，这是因为需要考虑更多的约束条件

3. 实验思考

    - 高维映射虽然提供了更好的分类性能，但也增加了**计算复杂度**
    - 在实际应用中，需要在维度、样本数量和计算复杂度之间找到平衡
4. **高维映射有利于线性可分：**

   - 通过增加维度，为数据分类提供了更多的空间和可能性，很多时候低维情况下无法实现线性可分时在高维情况下可以实现。在本实验中，从 2 维映射到 6 维后，分类间隔显著增大（如问题 a 中从 5.5902 增加到 63.1228），同时引入非线性特征（平方项和交叉项）增强了分类能力

5. **样本数量增加也会导致分类难度增大**

   - 分类间隔随样本数量增加而显著减小（6 维空间从 63.1228 减小到 17.9962）
   - 需要更多的支持向量来定义分类边界，每增加一个样本点都会增加一个约束条件，使得找到合适的分类超平面更加困难
   - 虽然本实验的数据比较理想（始终保持线性可分），但从间隔的减小趋势可以看出分类难度的增加

五、源代码

```python
import numpy as np
from sklearn.svm import SVC
import matplotlib.pyplot as plt

# 设置 matplotlib 支持中文显示
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False     # 用来正常显示负号

# 原始数据
w3_data = np.array([
    [-3.0, -2.9], [0.5, 8.7], [2.9, 2.1], [-0.1, 5.2], [-4.0, 2.2],
    [-1.3, 3.7], [-3.4, 6.2], [-4.1, 3.4], [-5.1, 1.6], [1.9, 5.1]
])

w4_data = np.array([
    [-2.0, -8.4], [-8.9, 0.2], [-4.2, -7.7], [-8.5, -3.2], [-6.7, -4.0],
    [-0.5, -9.2], [-5.3, -6.7], [-8.7, -6.4], [-7.1, -9.7], [-8.0, -6.3]
])

def transform_features(X):
    """将二维特征转换为高维特征：[1, x1, x2, x1², x1x2, x2²]"""
    x1, x2 = X[:, 0], X[:, 1]
    return np.column_stack([
        np.ones_like(x1),   # 1
        x1,                 # x1
        x2,                 # x2
        x1**2,              # x1²
```

```
        x1*x2,              # x1x2
        x2**2               # x2²
    ])

def prepare_data(n_samples):
    """准备训练数据"""
    X_w3 = w3_data[:n_samples]
    X_w4 = w4_data[:n_samples]
    X = np.vstack([X_w3, X_w4])
    y = np.array([1]*len(X_w3) + [-1]*len(X_w4))
    return X, y

def train_and_analyze_svm(n_samples):
    """在 6 维空间训练 SVM 并分析结果"""
    # 准备数据
    X, y = prepare_data(n_samples)
    X_transformed = transform_features(X)

    # 训练 SVM
    svm = SVC(kernel='linear', C=1000)
    svm.fit(X_transformed, y)

    return get_svm_results(svm, X, X_transformed, y)

def train_and_analyze_svm_original(n_samples):
    """在原始 2 维空间训练 SVM 并分析结果"""
    # 准备数据
    X, y = prepare_data(n_samples)

    # 训练 SVM
    svm = SVC(kernel='linear', C=5000)
    svm.fit(X, y)

    return get_svm_results(svm, X, X, y)

def get_svm_results(svm, X, X_train, y):
    """获取 SVM 训练结果"""
    w = svm.coef_[0]
    b = svm.intercept_[0]
    margin = 2 / np.linalg.norm(w)
    support_vectors = X[svm.support_]

    # 计算所有样本到超平面的距离
    predictions = svm.predict(X_train)
```

```python
    decision_values = svm.decision_function(X_train)

    # 检查是否严格线性可分（所有样本都在正确的一侧，且距离超平面至少有一定距离）
    tolerance = 1e-3  # 设置一个小的容忍度
    is_strictly_separable = np.all(decision_values * y >= tolerance)

    # 计算分类准确率
    accuracy = np.mean(predictions == y)

    return {
        'w': w,
        'b': b,
        'margin': margin,
        'support_vectors': support_vectors,
        'is_separable': is_strictly_separable,
        'accuracy': accuracy,
        'decision_values': decision_values
    }

def print_results(result, space_type="6 维"):
    """打印 SVM 结果"""
......
def plot_svm_results(X, y, result, title, is_transformed=False):
    """可视化结果"""
......
def visualize_results(n_samples):
    """可视化指定样本数量的结果"""
......
def print_separability_analysis(n_samples):
    """分析样本的可分性"""
    X, y = prepare_data(n_samples)

    # 2 维空间分析
    result_orig = train_and_analyze_svm_original(n_samples)
    print(f"\n 原始 2 维空间（{n_samples}个样本）：")
    print(f"分类准确率: {result_orig['accuracy']:.4f}")
    print(f"最小间隔: {np.min(np.abs(result_orig['decision_values'])):.4f}")
    print(f"严格线性可分: {'是' if result_orig['is_separable'] else '否'}")

    # 6 维空间分析
    result_transformed = train_and_analyze_svm(n_samples)
    print(f"\n6 维空间（{n_samples}个样本）：")
    print(f"分类准确率: {result_transformed['accuracy']:.4f}")
    print(f"最小间隔: {np.min(np.abs(result_transformed['decision_values'])):.
```

```
4f}")
    print(f"严格线性可分: {'是' if result_transformed['is_separable'] else '否'}")
def analyze_separability(svm, X, y, space_type="2 维"):
    """详细分析线性可分性"""
    decision_values = svm.decision_function(X)
    margins = decision_values * y
    print(f"\n{space_type}空间分离性分析：")
    print("各样本到决策边界的距离（带符号）：")
    for i, (x, margin) in enumerate(zip(X, margins)):
        label = "ω₃" if y[i] == 1 else "ω₄"
        print(f"样本{i+1} ({label}): ({x[0]:.2f}, {x[1]:.2f}) -> 距离: {margin:.4f}")

    min_margin = np.min(margins)
    print(f"\n 最小间隔: {min_margin:.4f}")
    print(f"是否严格线性可分: {'是' if min_margin >= 1e-3 else '否'}")
def main():
    """主函数"""
    # 问题(a)的结果
    print("\n=== 问题(a)的结果 ===")
    print("\n6 维空间：")
    result_a = train_and_analyze_svm(1)
    print_results(result_a)
    print("\n 原始 2 维空间：")
    result_a_orig = train_and_analyze_svm_original(1)
    print_results(result_a_orig, "2 维")
    # 问题(b)的结果
    print("\n=== 问题(b)的结果 ===")
    print("\n6 维空间：")
    result_b = train_and_analyze_svm(2)
    print_results(result_b)
    print("\n 原始 2 维空间：")
    result_b_orig = train_and_analyze_svm_original(2)
    print_results(result_b_orig, "2 维")
    # 问题(c)的结果
    print("\n=== 问题(c)的结果 ===")
    for i in [1, 2, 3, 5, 10]:  # 选择几个关键的样本数量进行分析
        print(f"\n--- 使用前{i}个样本的分析 ---")
        print_separability_analysis(i)
    # 使用所有样本的训练结果
    print("\n=== 使用所有样本的训练结果 ===")
    print("\n 原始 2 维空间（全部样本）：")
    result_full_orig = train_and_analyze_svm_original(10)
    print_results(result_full_orig, "2 维")
```

```
    print("\n映射到6维空间后（全部样本）：")
    result_full = train_and_analyze_svm(10)
    print_results(result_full)
    # 线性可分性结果
    print(f"\n原始2维空间是否线性可分：{'是' if result_full_orig['is_separable'] else '否'}")
    print(f"6维空间是否线性可分：{'是' if result_full['is_separable'] else '否'}")
    # 添加可视化部分
    print("\n=== 生成可视化结果 ===")
    # 可视化问题(a)的结果（1个样本）
    visualize_results(1)
    # 可视化问题(b)的结果（2个样本）
    visualize_results(2)

    # 可视化所有样本的结果
    visualize_results(10)

    plt.show()
```