SVM.1

1、$\because K(\vec{x_i},\vec{x_j})=(1+\vec{x_i}\cdot\vec{x_j})^2$，且 $\vec{x_i}=(x_{i1},x_{i2})$，$\vec{x_j}=(x_{j1},x_{j2})$

又 $\because \vec{x_1}=(0,0)$，$\vec{x_2}=(0,1)$，$\vec{x_3}=(1,0)$，$\vec{x_4}=(1,1)$

$\therefore$ $K(\vec{x_1},\vec{x_1})=(1+\vec{x_1}\cdot\vec{x_1})^2=(1+0+0)^2=1$

$K(\vec{x_1},\vec{x_2})=(1+\vec{x_1}\cdot\vec{x_2})^2=(1+0+0)^2=1$

$K(\vec{x_1},\vec{x_3})=(1+\vec{x_1}\cdot\vec{x_3})^2=(1+0+0)^2=1$

$K(\vec{x_1},\vec{x_4})=(1+\vec{x_1}\cdot\vec{x_4})^2=(1+0+0)^2=1$

同理，$K(\vec{x_2},\vec{x_1})=1$，$K(\vec{x_2},\vec{x_2})=4$，$K(\vec{x_2},\vec{x_3})=1$，$K(\vec{x_2},\vec{x_4})=4$

$K(\vec{x_3},\vec{x_1})=1$，$K(\vec{x_3},\vec{x_2})=1$，$K(\vec{x_3},\vec{x_3})=4$，$K(\vec{x_3},\vec{x_4})=4$

$K(\vec{x_4},\vec{x_1})=1$，$K(\vec{x_4},\vec{x_2})=4$，$K(\vec{x_4},\vec{x_3})=4$，$K(\vec{x_4},\vec{x_4})=9$

$\therefore$ 核矩阵为 $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 1 & 4 \\ 1 & 1 & 4 & 4 \\ 1 & 4 & 4 & 9 \end{bmatrix}$

对偶问题的目标函数为 $L_D(\vec{a})=\sum_i a_i-\frac{1}{2}\sum_{i,j}a_ia_jy_iy_j\phi(\vec{x_i})\cdot\phi(\vec{x_j})$

$=\sum_i a_i-\frac{1}{2}\sum_{i,j}a_ia_jy_iy_jK(\vec{x_i},\vec{x_j})$

对于XOR问题，$y_1=0$，$y_2=1$，$y_3=1$，$y_4=0$

$\therefore L_D(\vec{a})=(a_1+a_2+a_3+a_4)-\frac{1}{2}[a_2^2K(\vec{x_2},\vec{x_2})+a_2a_3K(\vec{x_2},\vec{x_3})+a_3a_2K(\vec{x_3},\vec{x_2})+a_3^2K(\vec{x_3},\vec{x_3})]$

$=(a_1+a_2+a_3+a_4)-\frac{1}{2}(4a_2^2+2a_2a_3+4a_3^2)$

SVM.2

2、对于多分类问题，SVM可有以下几种途径去解决：

① 利用 One Against All 的思想，训练M个SVM，第i个SVM用于区分第i类和其他类

② 利用 One Against One 的思想，训练 C_M^2 个SVM，每个SVM用与之相关的两类样本训练。每来一个新样本，用所有SVM去对其分类，根据投票结果得出这个新样本的最终预测类别。

③ 使用基于决策树的SVMs. 在树根节点训练一个SVM将样本分为类1和'其他'，对于在'其他'类中的样本，再训练一个SVM将其分为类2和'其他'，如此不断分支直到所有类都可被分出来。

SVM.3

题目要求使用$K(x_i, x_j) = (1 + x_i^T x_j)^p$作为核函数，而sklearn中的SVC类默认使用的核函数是RBF，因此需要先自编新的核函数my_poly()再将其作为参数传递给SVC()然后实例化一个SVM分类器。

由于SVC要求传入的核函数必须有且仅有两个参数，因此要想改变p值只能每次运行完程序后手动在my_poly()函数中改变。

程序源码如下：

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

# 自定义核函数
def my_poly(x, y):
    return np.power((np.dot(x, y.T) + 1), 2)

model_svm = SVC(C = 1, kernel = my_poly, tol = 0.001, max_iter = 100)


# 数据集
X_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
X_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

# 标签
y_train = np.array([0, 1, 1, 0])
y_test = np.array([0, 1, 1, 0])

# 训练SVM
model_svm.fit(X_train, y_train)

# 预测测试集
pred = model_svm.predict(X_test)
print('预测标签:', pred)
print('实际标签:', y_test)
print('预测准确率:{}%'.format(100 * model_svm.score(X_test, y_test)))

# 求决策函数的输出：大于0表示对应的样本为正样本的置信度高（即更可能是正样本）
Decision_Function = model_svm.decision_function(X_test)
print('决策函数输出:', Decision_Function)

# 绘制决策边界
arr1 = np.arange(-1, 2, 0.01)
arr2 = np.arange(-1, 2, 0.01)
xx, yy = np.meshgrid(arr1, arr2)

input_array = np.array([xx.ravel(), yy.ravel()]).T
# 对区域内每个点都预测它的类别，便于可视化
labels = model_svm.predict(input_array)
# 绘图
plt.figure(figsize =(10, 7))
plt.contourf(xx, yy, labels.reshape(xx.shape), alpha=0.4, cmap='Blues', linestyles='dashed')
plt.scatter(X_test[:, 0], X_test[:, 1], s=100, c=y_test, cmap='Blues', edgecolors='k')
plt.xlabel('f1')
plt.ylabel('f2')
plt.title('Decision Boundary')
```

运行结果及分析：

$p = 1$ 时：

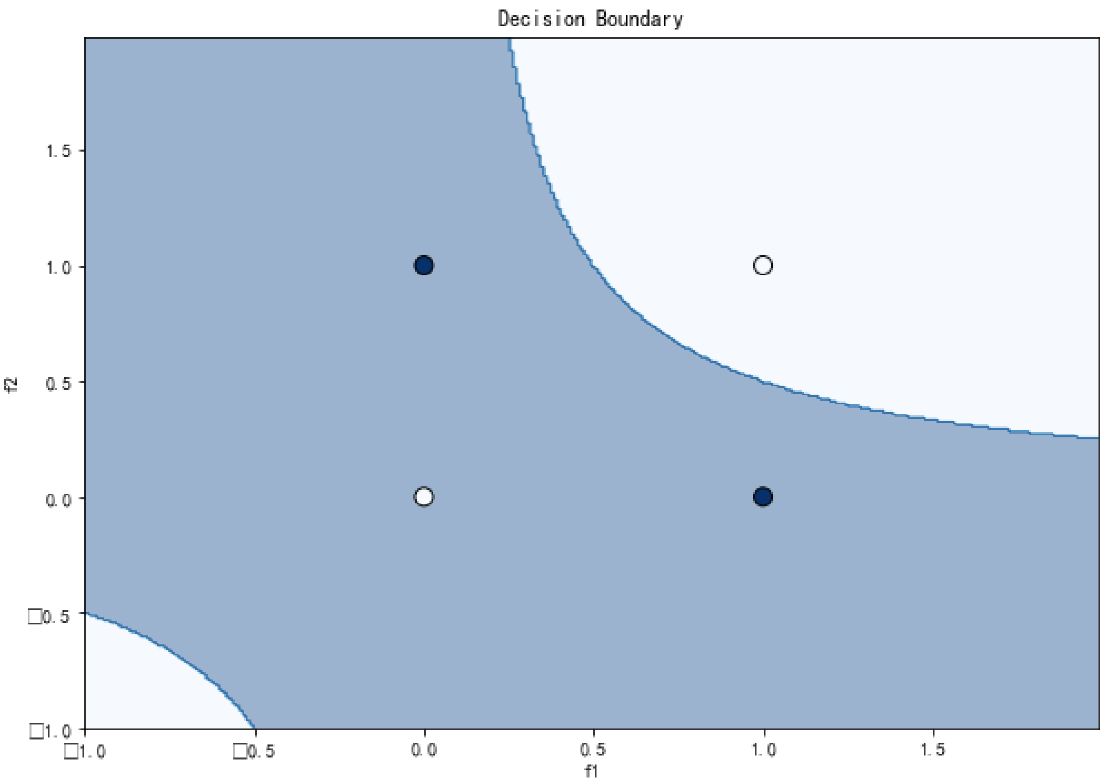
```
预测标签: [1 1 1 1]
实际标签: [0 1 1 0]
预测准确率:50.0%
决策函数输出: [-0. -0. -0. -0.]
```

分析：当p = 1时，SVM将其全部分为了正类，其实本质上是没能成功分类。原因是当p = 1时，核函数为 $K(x_i, x_j) = 1 + x_i^T x_j$ ，相当于输入数据的二维向量仍在原始二维空间中进行内积运算，并未被扩增到更高维空间中。我们知道在二维空间中，XOR问题是线性不可分的，因此此时使用SVM并不能对样本进行正确分类。并且此时决策函数输出4个样本对应的置信度值均为0，表示SVM无法判定这4个样本属于哪一类。

$p = 2$ 时：

```
预测标签: [1 1 1 0]
实际标签: [0 1 1 0]
预测准确率:75.0%
决策函数输出: [ 1.  1.  1. -1.]
```

可视化决策边界如下：

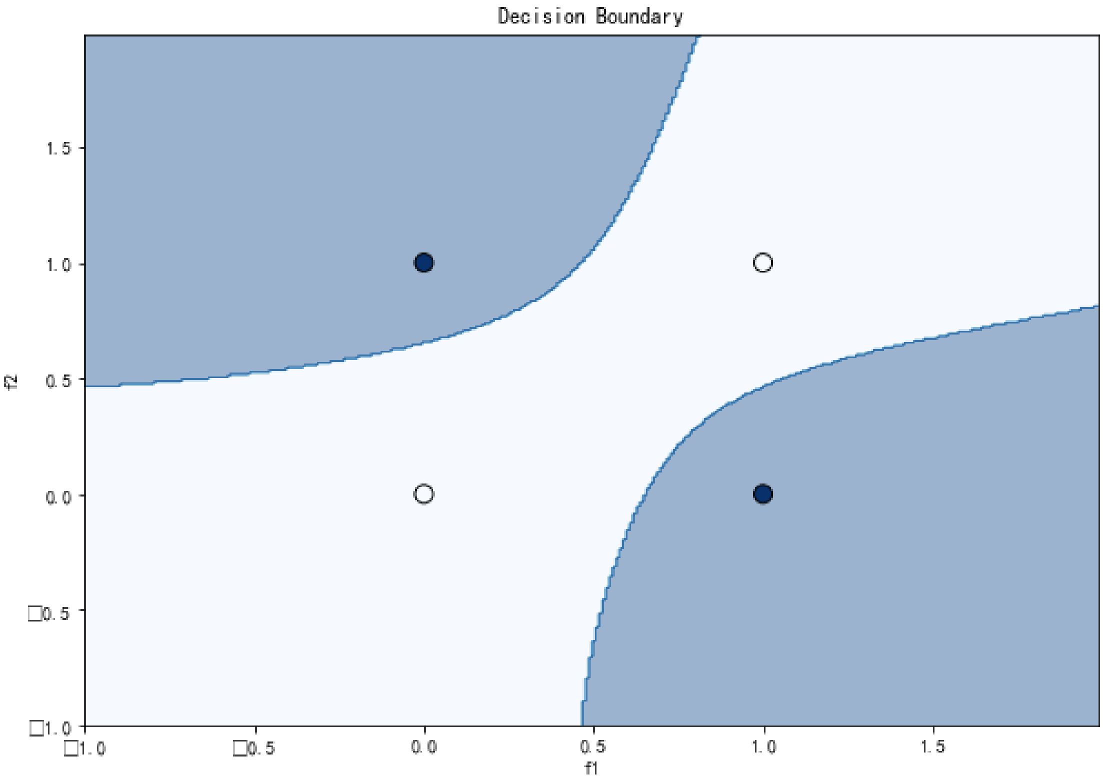


分析：当p = 2时，SVM将前3个样本分为正类，第4个样本分为了负类，分类准确率为75%。当p = 2时，核函数为 $K(x_i, x_j) = (1 + x_i^T x_j)^2 = [1 + (x_{i1}x_{j1} + x_{i2}x_{j2})]^2 = 1 + x_{i1}^2 x_{j1}^2 + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + 2x_{i1}x_{j1}x_{i2}x_{j2}$ ，相当于输入数据的二维向量已经被扩增到了6维空间中进行内积计算。可能由于在该空间中XOR问题仍然是难以被线性分开的，因此此时使用SVM也不能对样本进行正确分类。

$p = 3$ 时：

```
预测标签: [0 1 1 0]
实际标签: [0 1 1 0]
预测准确率:100.0%
决策函数输出: [-0.99976776  0.99930327  1.00023224 -0.99976776]
```

可视化决策边界如下:

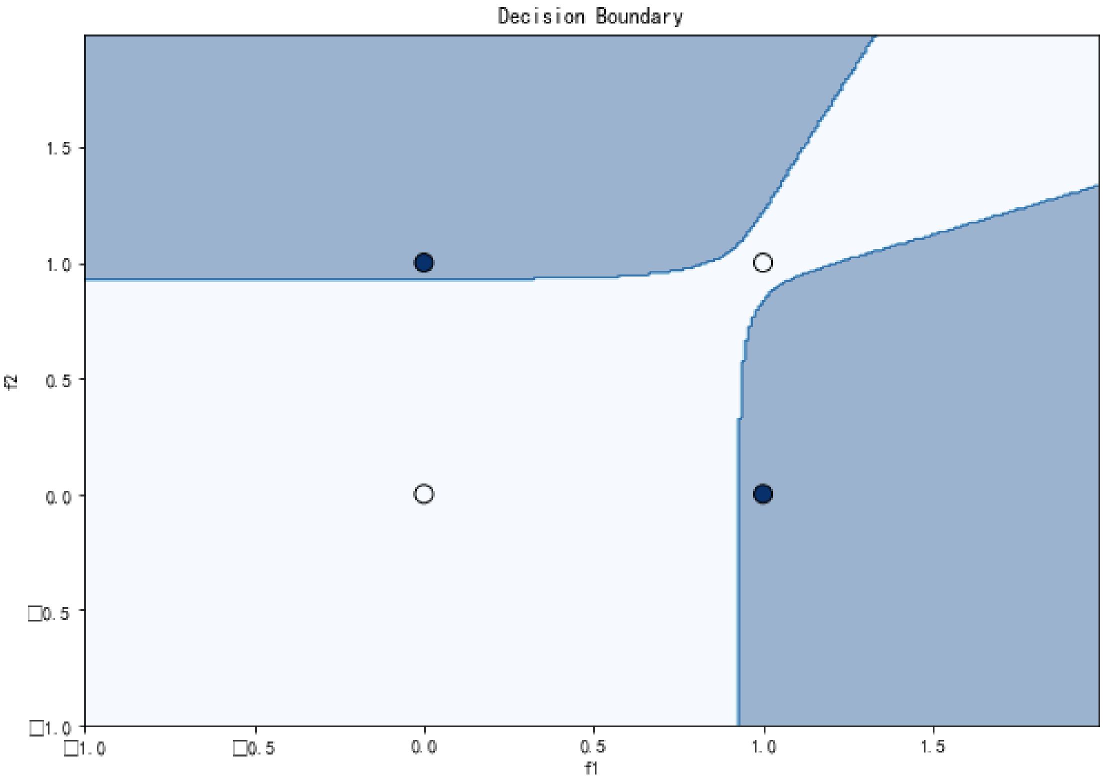


分析：当p = 3时，SVM已经可以将所有的样本正确分类，分类准确率为100%。当p = 3时，核函数为 $K(x_i, x_j) = (1 + x_i^T x_j)^3 = [1 + (x_{i1}x_{j1} + x_{i2}x_{j2})]^3$ ，相当于输入数据的二维向量已经被扩增到了10维空间中进行内积计算。应该是由于在该空间中XOR问题已经成为了一个线性可分问题，因此此时使用SVM能够对样本进行正确分类。

$p = 19$ 时:

```
预测标签: [0 1 1 0]
实际标签: [0 1 1 0]
预测准确率:100.0%
决策函数输出: [-0.99977423  0.99932277  1.00022577 -0.99977417]
```

可视化决策边界如下:

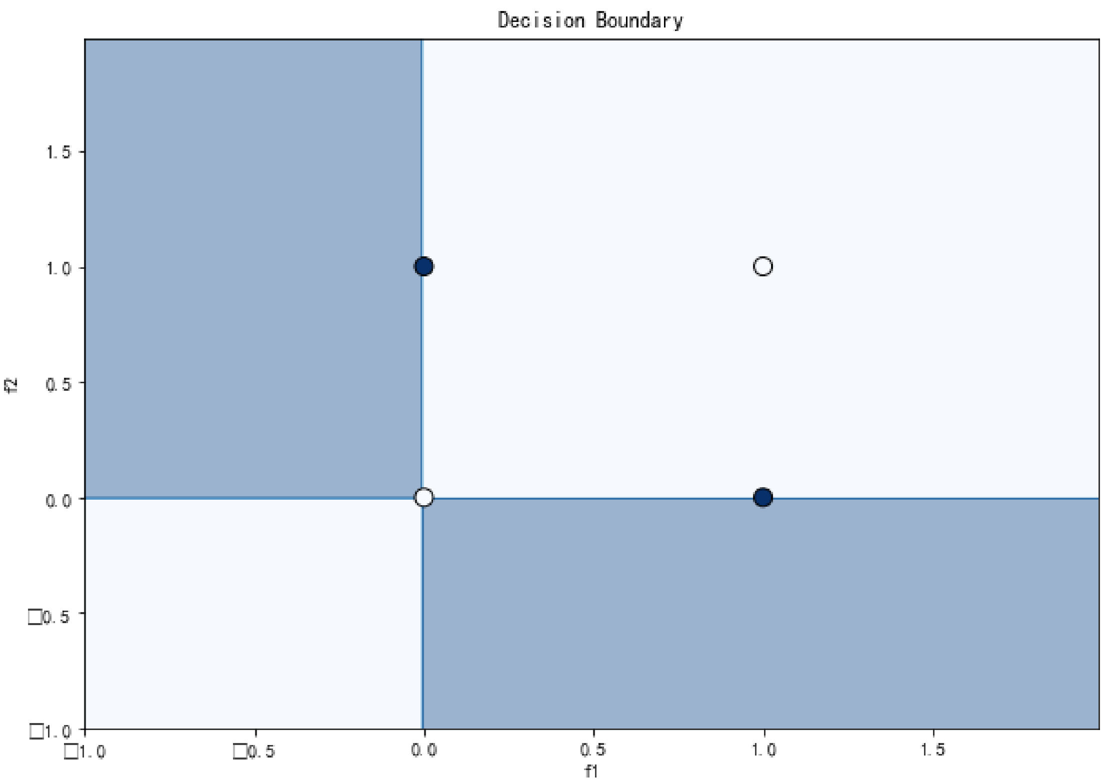


分析：通过观察决策边界，我们可以发现当p = 19时两个正类样本已经十分靠近决策边界线了，而SVM的目的是要让所有的正负样本均尽可能地远离决策边界，因此可以得知此时SVM的分类效果已经不是很好了。虽然仍能够正确分类，但得到的决策边界已经不是ideal的了（相比p = 3时而言）。

$p = 20$ 时:

```
预测标签: [1 1 1 1]
实际标签: [0 1 1 0]
预测准确率:50.0%
决策函数输出: [-0.00000000e+00 -0.00000000e+00 -0.00000000e+00  8.10280046e+08]
```

可视化决策边界如下:



分析：当p = 20时，SVM将所有的样本分为了正类（即分类失败），这可能是因为此时样本维度被扩增得太高，以至于SVM认为所有的样本均为同一类了。

结论：我们使用核函数是为了将原本在低维空间中线性不可分的样本投影到高维空间中试图解决。核函数能够代替对应的高维向量内积从而简化计算复杂度。但是在实际解决具体问题的时候，要结合问题本身的特性去选择合适的核函数及其参数。比如对于本题中的XOR问题，若采用多项式核，则取p=3即可成功分类。若p太小会导致投影到的空间维度不够高，使得样本仍旧线性不可分；若p太大则会导致投影之后维数太高，以至于SVM将所有样本分为同一类，这也是不可取的。因此我们要具体问题具体分析，多尝试几个核函数，最后选择一个最合适的核函数以及该核函数的参数去解决问题。