

支持向量机

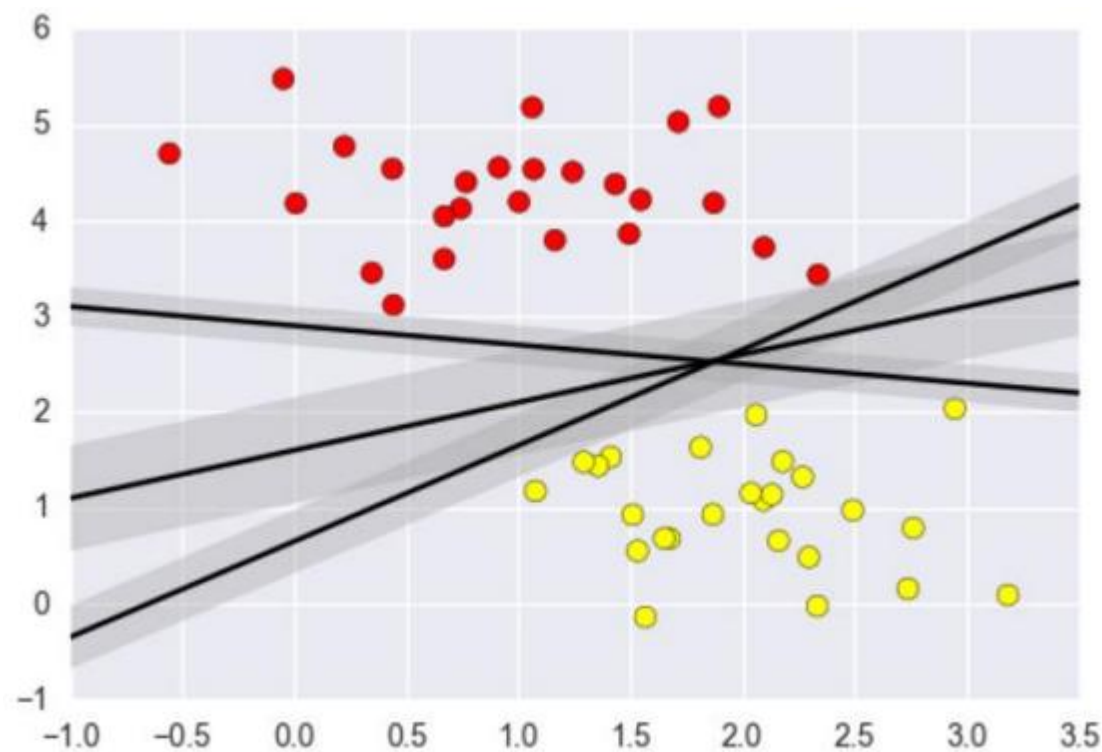
✓ Support Vector Machine

✎ 要解决的问题：什么样的决策边界才是最好的呢？

✎ 特征数据本身如果就很难分，怎么办呢？

✎ 计算复杂度怎么样？能实际应用吗？

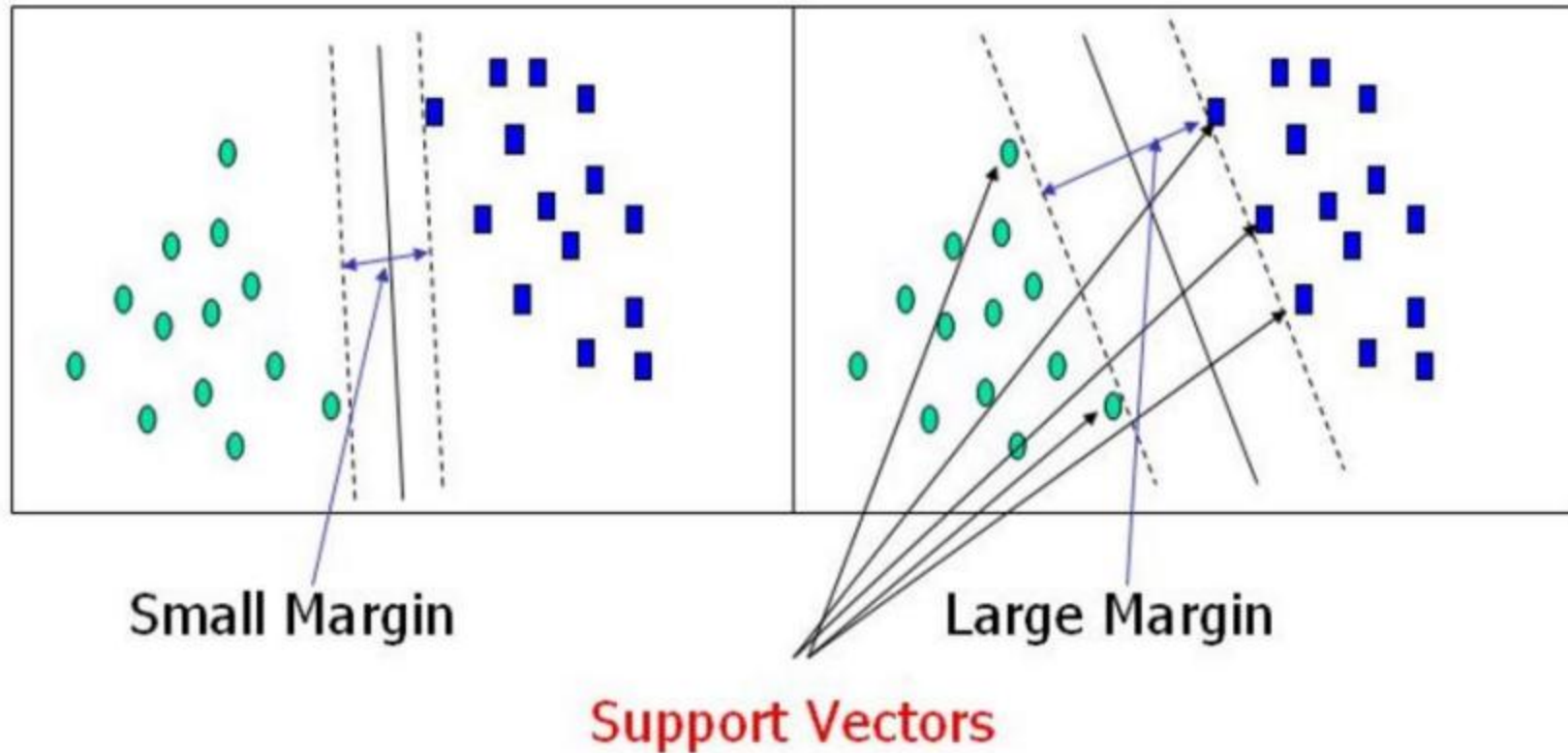
✎ 目标：基于上述问题对SVM进行推导



支持向量机

✓ Support Vector Machine

✎ 决策边界：选出来离雷区最远的（雷区就是边界上的点，要Large Margin）



支持向量机

✓ 距离的计算

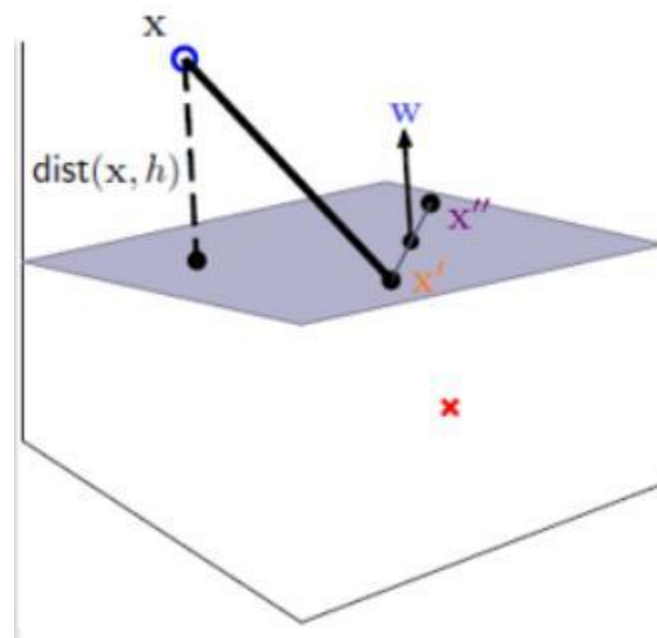
consider \mathbf{x}' , \mathbf{x}'' on hyperplane

① $\mathbf{w}^T \mathbf{x}' = -b$, $\mathbf{w}^T \mathbf{x}'' = -b$

② $\mathbf{w} \perp$ hyperplane:

$$\begin{pmatrix} \mathbf{w}^T & \underbrace{(\mathbf{x}'' - \mathbf{x}')}_{\text{vector on hyperplane}} \end{pmatrix} = 0$$

③ distance = project $(\mathbf{x} - \mathbf{x}')$ to \perp hyperplane



$$\text{distance}(\mathbf{x}, b, \mathbf{w}) = \left| \frac{\mathbf{w}^T}{\|\mathbf{w}\|} (\mathbf{x} - \mathbf{x}') \right| \stackrel{\textcircled{1}}{=} \frac{1}{\|\mathbf{w}\|} |\mathbf{w}^T \mathbf{x} + b|$$

支持向量机

✓ 数据标签定义

✎ 数据集： $(X_1, Y_1)(X_2, Y_2) \dots (X_n, Y_n)$

✎ Y为样本的类别： 当X为正例时候 $Y = +1$ 当X为负例时候 $Y = -1$

✎ 决策方程： $y(x) = w^T \Phi(x) + b$ (其中 $\Phi(x)$ 是对数据做了变换，后面继续说)

$$\begin{aligned} & y(x_i) > 0 \Leftrightarrow y_i = +1 \\ \Rightarrow & y(x_i) < 0 \Leftrightarrow y_i = -1 \quad \Rightarrow y_i \cdot y(x_i) > 0 \end{aligned}$$

支持向量机

✓ 优化的目标

✎ 通俗解释：找到一个条线（ w 和 b ），使得离该线最近的点（雷区）能够最远

✎ 将点到直线的距离化简得：
$$\frac{y_i \cdot (w^T \cdot \Phi(x_i) + b)}{\|w\|}$$

（由于 $\underline{y_i \cdot (w^T \cdot \Phi(x_i) + b)} > 0$ 所以将绝对值展开原始依旧成立）

支持向量机

✓ 目标函数

✎ 放缩变换：对于决策方程 (w, b) 可以通过放缩使得其结果值 $|Y| \geq 1$

$$\Rightarrow y_i \cdot (w^T \cdot \Phi(x_i) + b) \geq 1 \quad (\text{之前我们认为恒大于0, 现在严格了些})$$

✎ 优化目标： $\arg \max_{w, b} \left\{ \frac{1}{\|w\|} \min_i [y_i \cdot (w^T \cdot \Phi(x_i) + b)] \right\}$

由于 $y_i \cdot (w^T \cdot \Phi(x_i) + b) \geq 1$, 只需要考虑 $\arg \max_{w, b} \frac{1}{\|w\|}$ (目标函数搞定!)

支持向量机

✓ 目标函数

✎ 当前目标： $\max_{w,b} \frac{1}{\|w\|}$ ，约束条件： $y_i(w^T \cdot \Phi(x_i) + b) \geq 1$

✎ 常规套路：将求解极大值问题转换成极小值问题 $\Rightarrow \min_{w,b} \frac{1}{2}w^2$

✎ 如何求解：应用拉格朗日乘子法求解

支持向量机

✓ 拉格朗日乘子法

✎ 带约束的优化问题:

$$\begin{aligned} \min_x \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, i = 1, \dots, m \\ & h_i(x) = 0, i = 1, \dots, q \end{aligned}$$

✎ 原式转换:

$$\min L(x, \lambda, v) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^q v_i h_i(x)$$

✎ 我们的式子:

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i (y_i (w^T \cdot \Phi(x_i) + b) - 1)$$

(约束条件不要忘: $y_i (w^T \cdot \Phi(x_i) + b) \geq 1$)

支持向量机

✓ SVM求解

✎ 分别对 w 和 b 求偏导,分别得到两个条件 (由于对偶性质)

$$\square \max_{w,b} L(w,b,a) \rightarrow \max_{\alpha} \min_{w,b} L(w,b,\alpha)$$

✎ 对 w 求偏导: $\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^n \alpha_i y_i \Phi(x_i)$

对 b 求偏导: $\frac{\partial L}{\partial b} = 0 \Rightarrow 0 = \sum_{i=1}^n \alpha_i y_i$

支持向量机

✓ SVM求解

✎ 带入原始: $L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i (y_i (w^T \Phi(x_i) + b) - 1)$

其中 $w = \sum_{i=1}^n \alpha_i y_i \Phi(x_i)$ $0 = \sum_{i=1}^n \alpha_i y_i$

$$= \frac{1}{2} w^T w - w^T \sum_{i=1}^n \alpha_i y_i \Phi(x_i) - b \sum_{i=1}^n \alpha_i y_i + \sum_{i=1}^n \alpha_i$$

$$= \sum_{i=1}^n \alpha_i - \frac{1}{2} \left(\sum_{i=1}^n \alpha_i y_i \Phi(x_i) \right)^T \sum_{i=1}^n \alpha_i y_i \Phi(x_i)$$

$$= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \Phi^T(x_i) \Phi(x_j)$$

➡ 完成了第一步求解 $\min_{w, b} L(w, b, \alpha)$

支持向量机

✓ SVM求解



继续对 α 求极大值:

$$\max_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\Phi(x_i) \cdot \Phi(x_j))$$

条件: $\sum_{i=1}^n \alpha_i y_i = 0$

$$\alpha_i \geq 0$$



极大值转换成求极小值:

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\Phi(x_i) \cdot \Phi(x_j)) - \sum_{i=1}^n \alpha_i$$

条件: $\sum_{i=1}^n \alpha_i y_i = 0$

$$\alpha_i \geq 0$$

支持向量机

✓ SVM求解实例

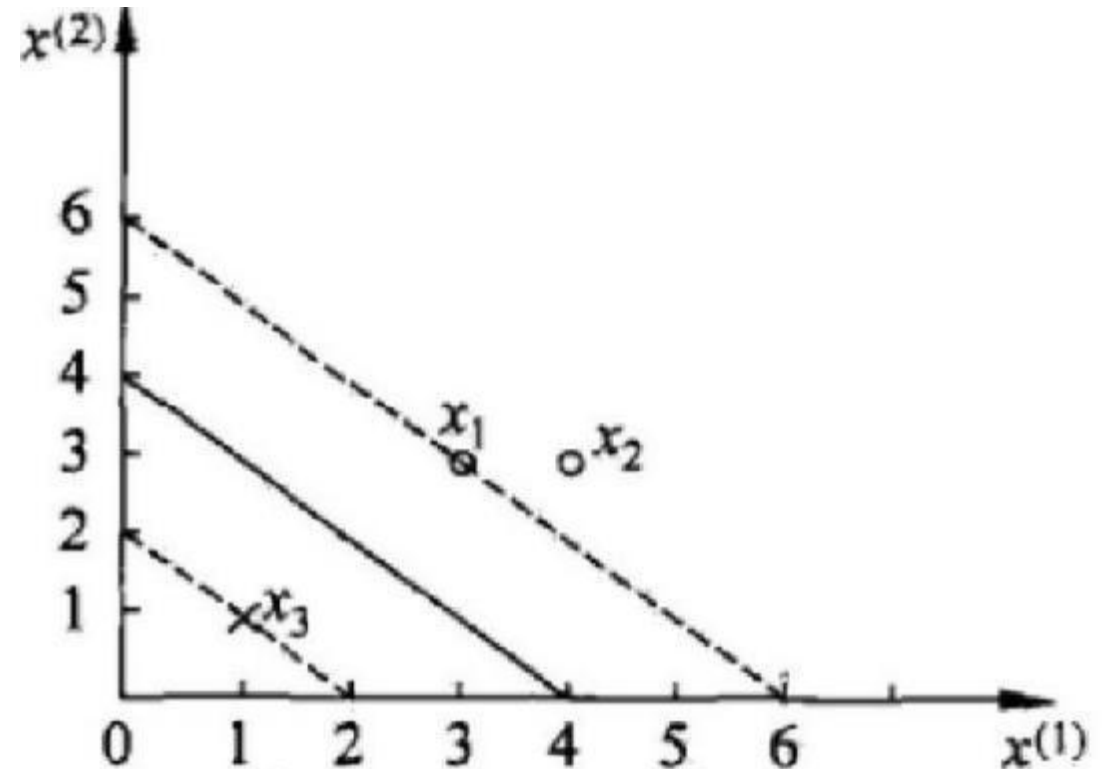
✎ 数据：3个点，其中正例 $X_1(3,3)$ ， $X_2(4,3)$ ，负例 $X_3(1,1)$

✎ 求解：

$$\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^n \alpha_i$$

约束条件：

$$\alpha_1 + \alpha_2 - \alpha_3 = 0$$
$$\alpha_i \geq 0, \quad i = 1, 2, 3$$



支持向量机

✓ SVM求解实例

✎ 原式: $\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^n \alpha_i$, 将数据代入

$$\frac{1}{2} (18\alpha_1^2 + 25\alpha_2^2 + 2\alpha_3^2 + 42\alpha_1\alpha_2 - 12\alpha_1\alpha_3 - 14\alpha_2\alpha_3) - \alpha_1 - \alpha_2 - \alpha_3$$

由于: $\alpha_1 + \alpha_2 = \alpha_3$ 化简可得: $4\alpha_1^2 + \frac{13}{2}\alpha_2^2 + 10\alpha_1\alpha_2 - 2\alpha_1 - 2\alpha_2$

支持向量机

✓ SVM求解实例

✎ 分别对 α_1 和 α_2 求偏导，偏导等于0可得：
 $\alpha_1 = 1.5$
 $\alpha_2 = -1$

(并不满足约束条件，所以解应在边界上)

$\alpha_1 = 0$
 $\alpha_2 = -2/13$ \rightarrow 带入原式 = -0.153 (不满足约束)

$\alpha_1 = 0.25$
 $\alpha_2 = 0$ \rightarrow 带入原式 = -0.25 (满足啦!)

最小值在(0.25,0,0.25)处取得

支持向量机

✓ SVM求解实例

✎ 将 α 结果带入求解 $w = \sum_{i=1}^n \alpha_i y_i \Phi(x_n)$

$$w = \frac{1}{4} * 1 * (3,3) + \frac{1}{4} * (-1) * (1,1) = \left(\frac{1}{2}, \frac{1}{2}\right)$$

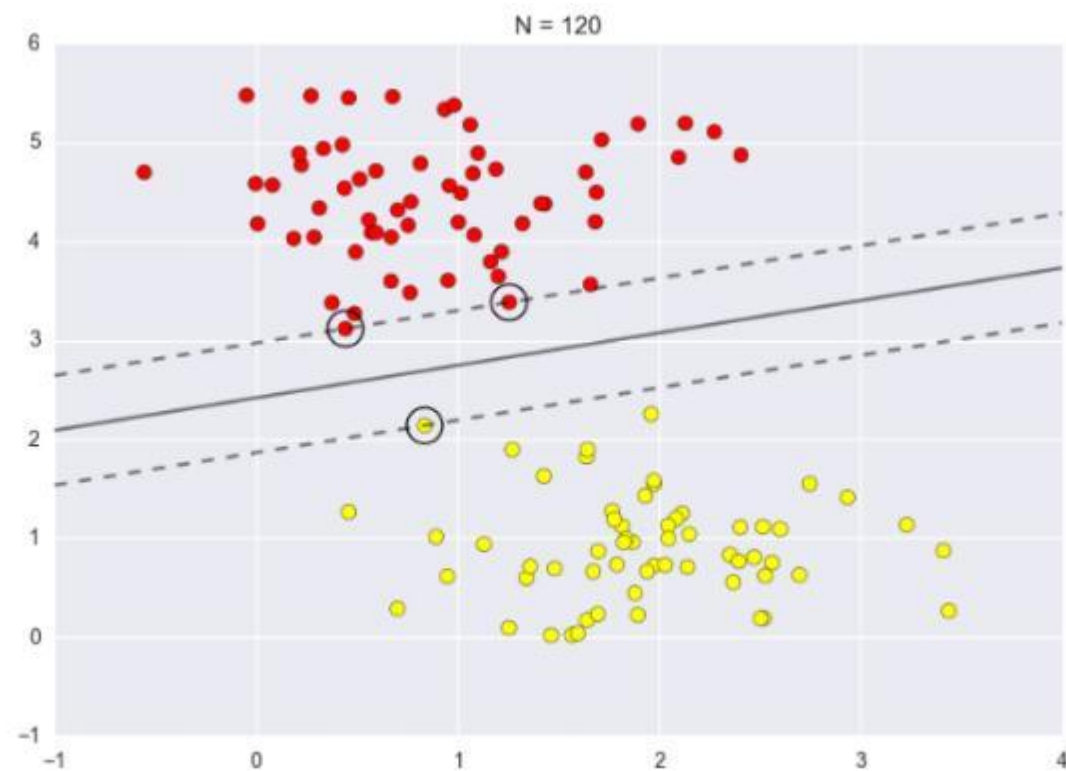
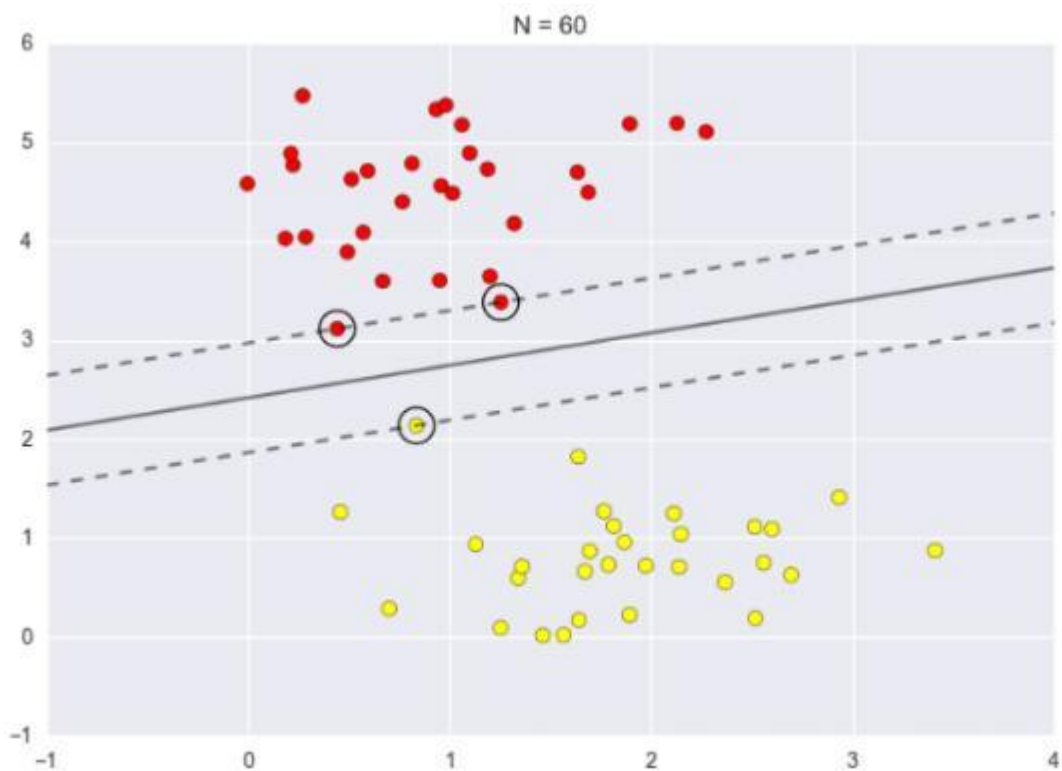
$$b = y_i - \sum_{i=1}^n \alpha_i y_i (x_i x_j) = 1 - \left(\frac{1}{4} * 1 * 18 + \frac{1}{4} * (-1) * 6\right) = -2$$

✎ 平面方程为： $0.5x_1 + 0.5x_2 - 2 = 0$

支持向量机

✓ SVM求解实例

📎 支持向量：真正发挥作用的数据点， α 值不为0的点



支持向量机

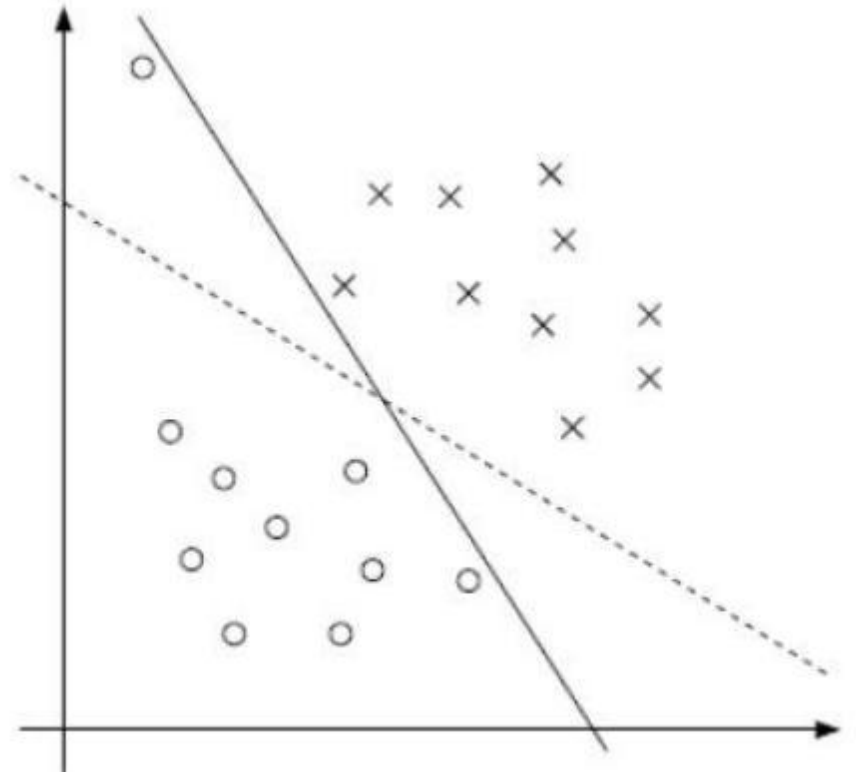
✓ soft-margin

✎ 软间隔：有时候数据中有一些噪音点，如果考虑它们咱们的线就不太好了

✎ 之前的方法要求要把两类点完全分得开，这个要求有点过于严格了，我们来放松一点！

✎ 为了解决该问题，引入松弛因子

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i$$



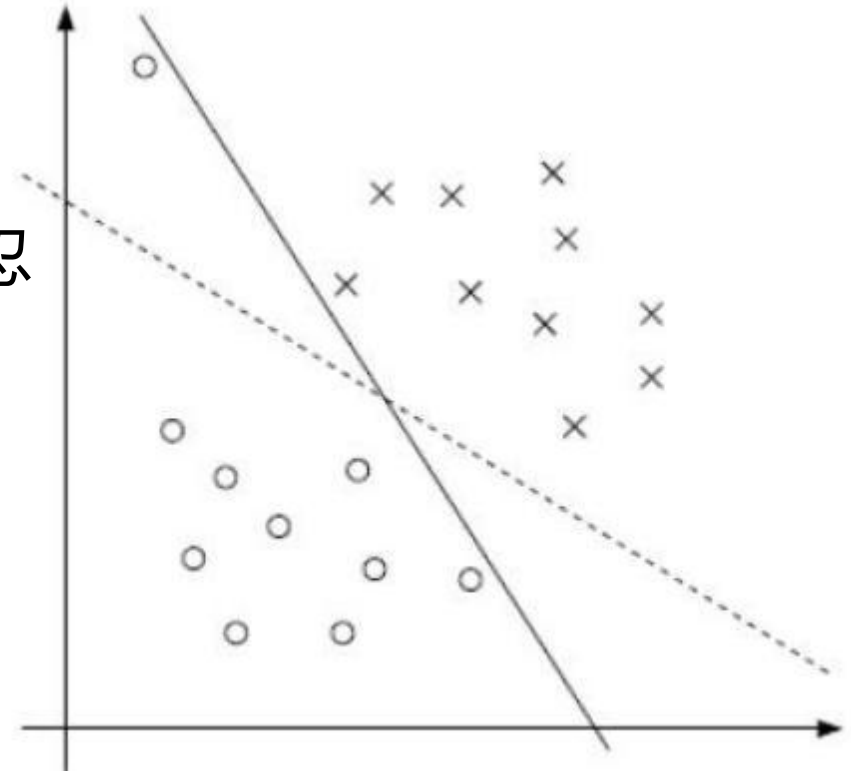
支持向量机

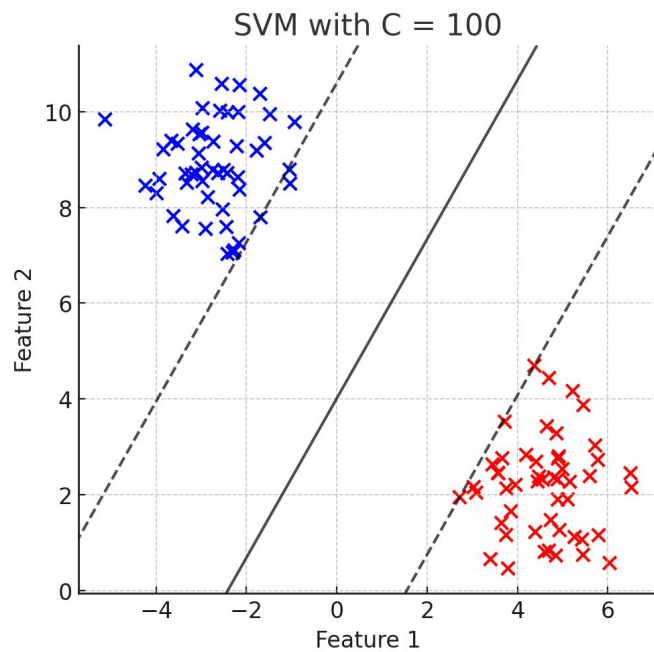
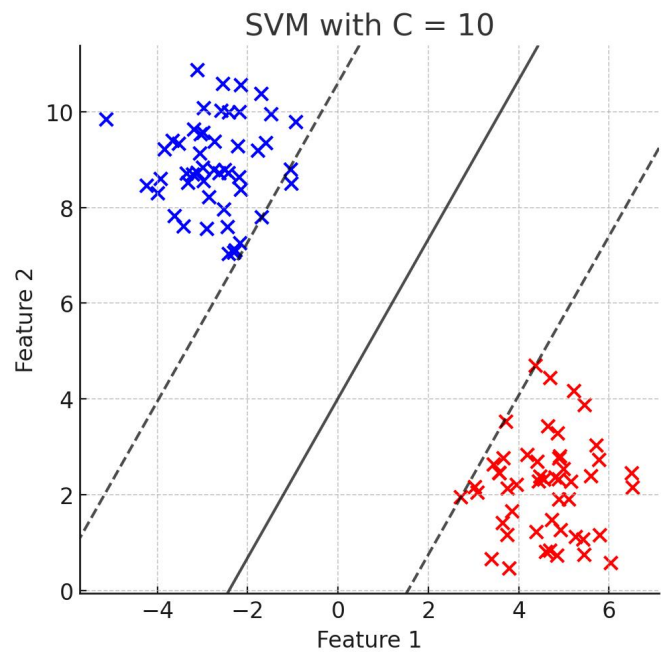
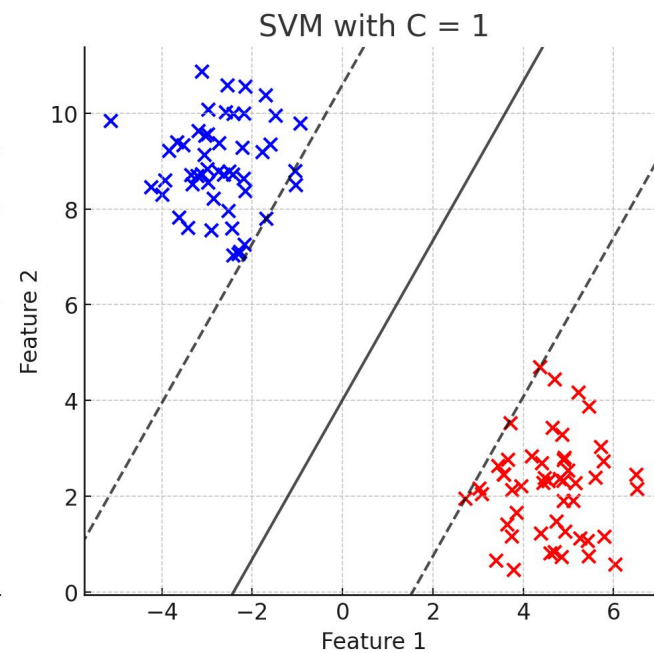
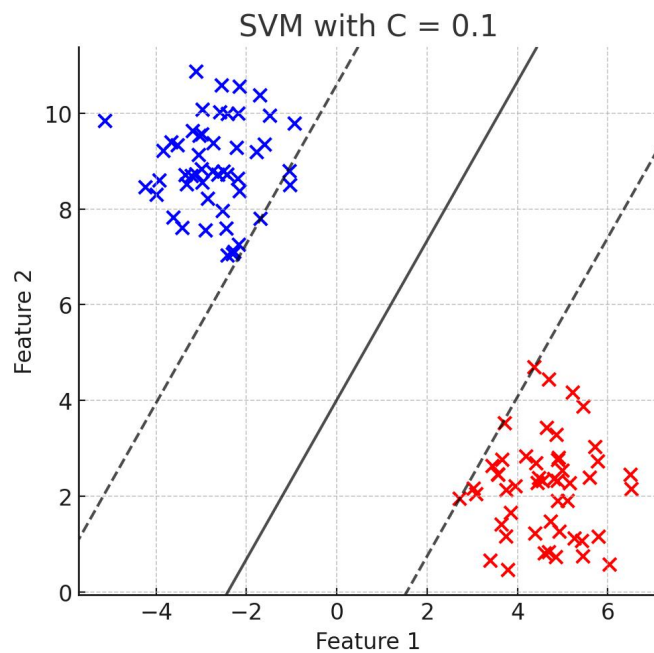
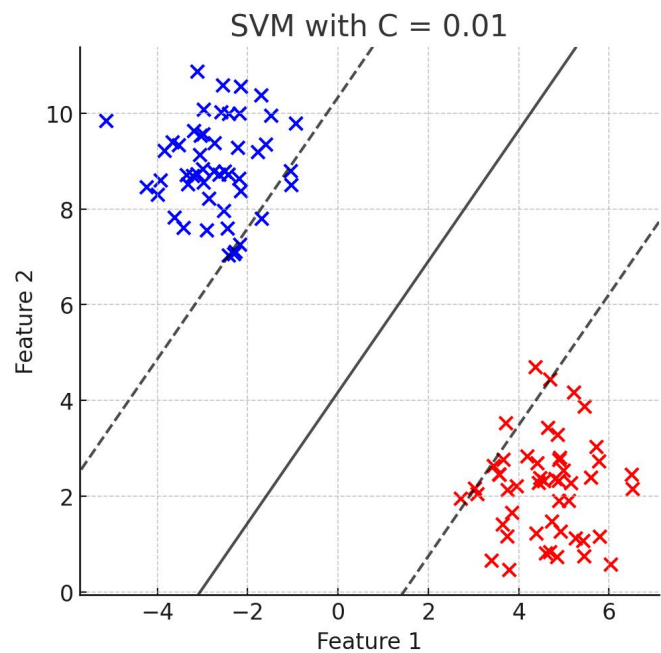
✓ soft-margin

✎ 新的目标函数: $\min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i$

✎ 当C趋近于很大时：意味着分类严格不能有错误
当C趋近于很小时：意味着可以有更大的错误容忍

✎ C是我们需要指定的一个参数！





支持向量机

✓ soft-margin

✎ 拉格朗日乘子法:

$$L(w, b, \xi, \alpha, \mu) \equiv \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i (y_i (w \cdot x_i + b) - 1 + \xi_i) - \sum_{i=1}^n \mu_i \xi_i$$

$$w = \sum_{i=1}^n \alpha_i y_i \phi(x_i)$$

约束: $0 = \sum_{i=1}^n \alpha_i y_i$

$$C - \alpha_i - \mu_i = 0$$

$$\alpha_i \geq 0 \quad \mu_i \geq 0$$

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^n \alpha_i$$

同样的解法: $\sum_{i=1}^n \alpha_i y_i = 0$

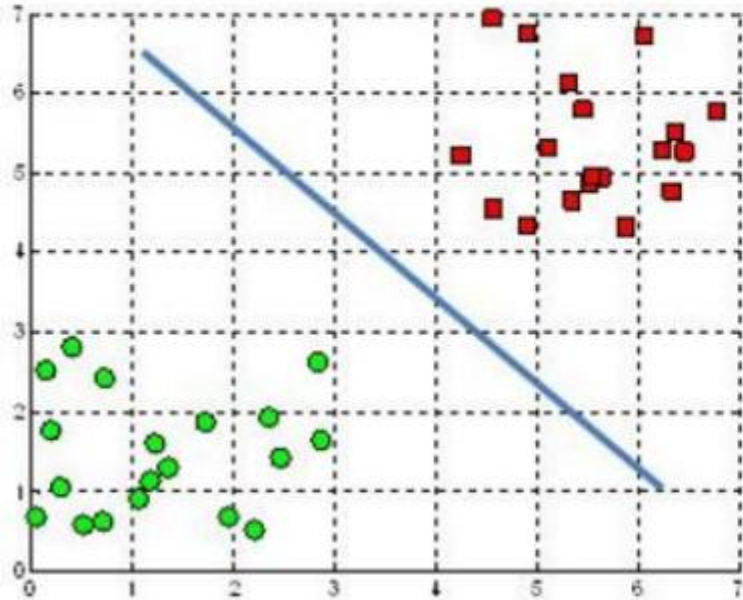
$$0 \leq \alpha_i \leq C$$

支持向量机

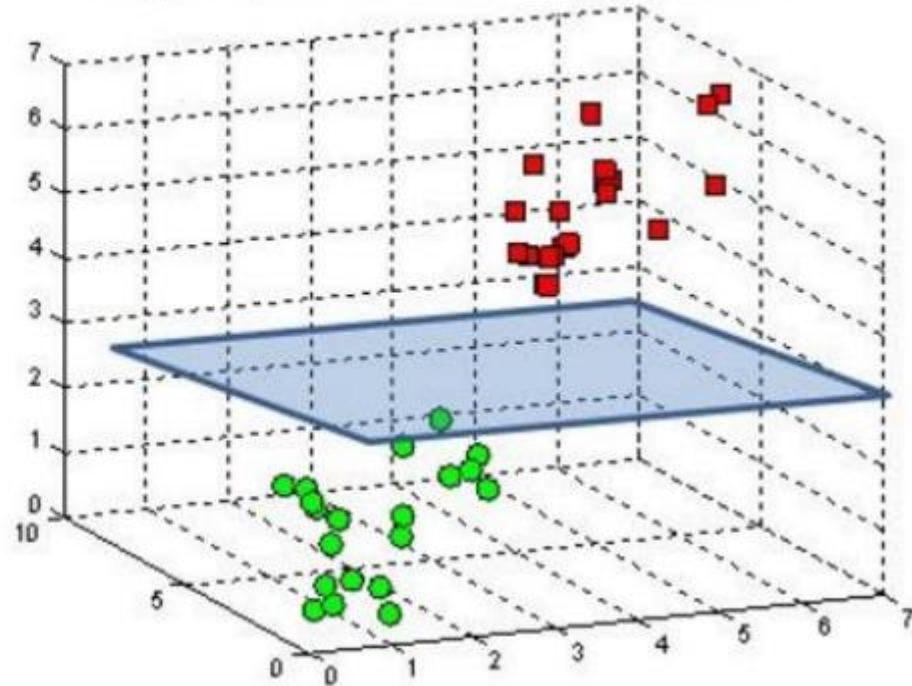
✓ 低维不可分问题

✎ 核变换：既然低维的时候不可分，那我给它映射到高维呢？

A hyperplane in \mathbb{R}^2 is a line



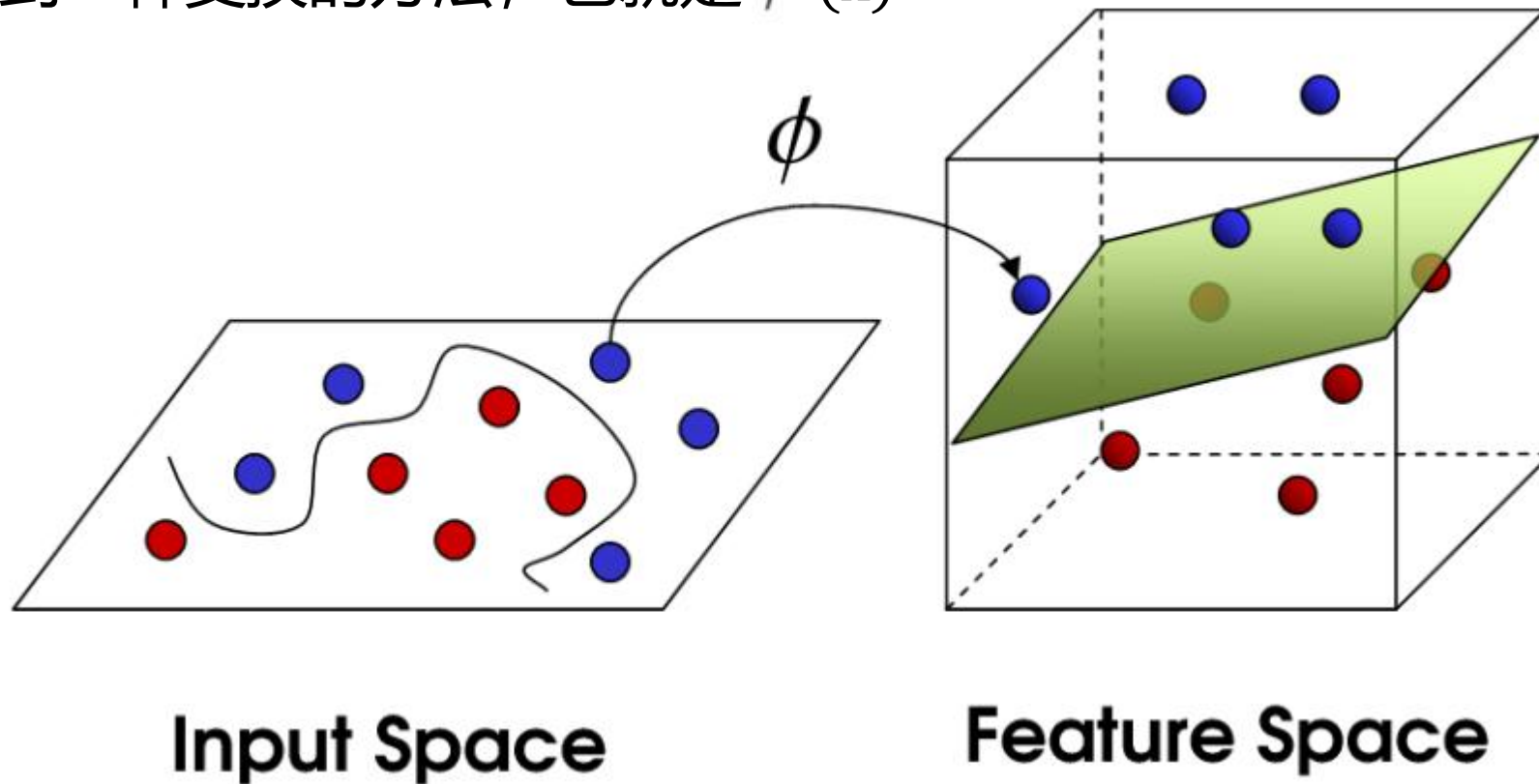
A hyperplane in \mathbb{R}^3 is a plane



支持向量机

✓ 低维不可分问题

✎ 目标：找到一种变换的方法，也就是 $\phi(x)$



支持向量机

还是先从小例子来阐述问题。假设我们有两个数据， $x = (x_1, x_2, x_3)$; $y = (y_1, y_2, y_3)$ ，此时在3D空间已经不能对其经行线性划分了，那么我们通过一个函数将数据映射到更高维的空间，比如9维的话，那么 $f(x) = (x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3)$ ，由于需要计算内积，所以在新的数据在9维空间，需要计算 $\langle f(x), f(y) \rangle$ 的内积，需要花费 $O(n^2)$ 。

在具体点，令 $x = (1, 2, 3)$; $y = (4, 5, 6)$ ，那么 $f(x) = (1, 2, 3, 2, 4, 6, 3, 6, 9)$ ， $f(y) = (16, 20, 24, 20, 25, 36, 24, 30, 36)$ ，

此时 $\langle f(x), f(y) \rangle = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$

似乎还能计算，但是如果将维数扩大到一个非常大数时候，计算起来可不是一丁点问题了。

但是发现， $K(x, y) = (\langle x, y \rangle)^2$

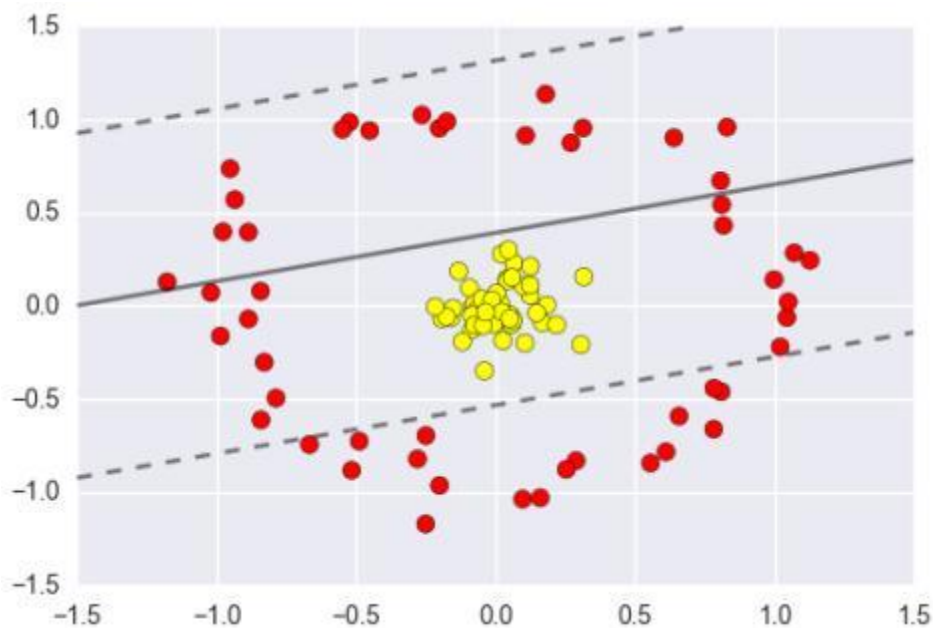
$K(x, y) = (4 + 10 + 18)^2 = 32^2 = 1024$

两者相等， $K(x, y) = (\langle x, y \rangle)^2 = \langle f(x), f(y) \rangle$ ，但是 $K(x, y)$ 计算起来却比 $\langle f(x), f(y) \rangle$ 简单的多，也就是说只要用 $K(x, y)$ 来计算，，效果和 $\langle f(x), f(y) \rangle$ 是一样的，但是计算效率却大幅度提高了，如： $K(x, y)$ 是 $O(n)$ ，而 $\langle f(x), f(y) \rangle$ 是 $O(n^2)$ 。所以使用核函数的好处就是，可以在一个低维空间去完成高维度（或者无限维度）样本内积的计算，比如 $K(x, y) = (4 + 10 + 18)^2$ 的3D空间对比 $\langle f(x), f(y) \rangle = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324$ 的9D空间。

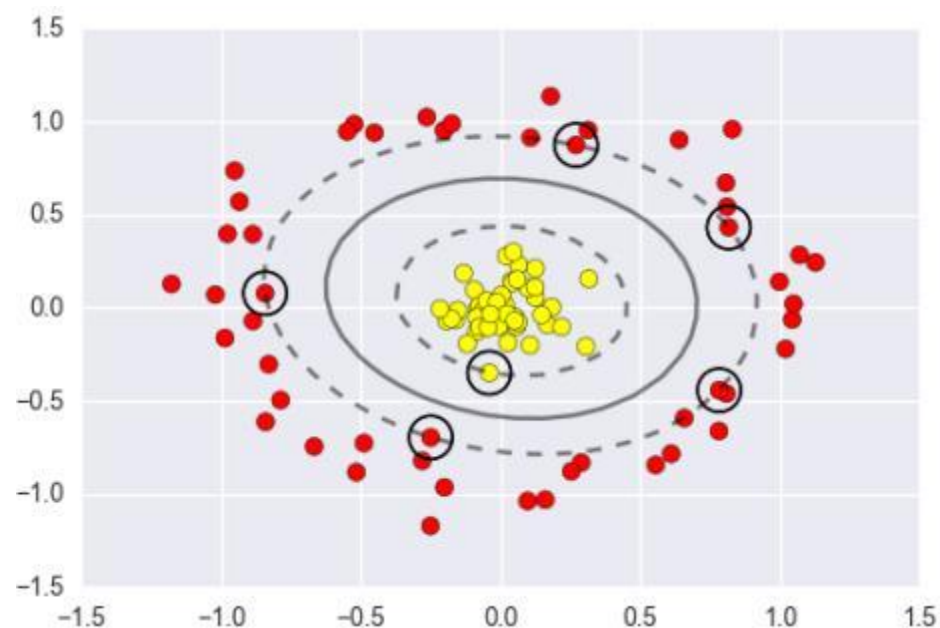
支持向量机

✓ Support Vector Machine

📎 高斯核函数: $K(X, Y) = \exp\left\{-\frac{\|X - Y\|^2}{2\sigma^2}\right\}$



线性核函数



高斯和函数

什么样的函数能作为核函数？

一个函数 $K(x, x')$ 能作为核函数的条件是它必须满足 **Mercer 定理**，即该函数需要满足以下条件：

对称性： $K(x, x') = K(x', x)$ 。

正定性： 对任意数据集 $\{x_1, x_2, \dots, x_n\}$ ，核矩阵 K 定义为：

$$K_{ij} = K(x_i, x_j)$$

该核矩阵 K 必须是半正定的（所有特征值 ≥ 0 ）。

在实际使用中，常用的核函数通常已经被证明满足这些条件。

核函数名称	公式	超参数	适用场景与特点
线性核	$K(x, x') = x \cdot x'$	无	<ul style="list-style-type: none">- 数据是线性可分的- 高维稀疏数据（如文本分类）- 计算效率高
多项式核	$K(x, x') = (\gamma \cdot x \cdot x' + r)^d$	γ, r, d	<ul style="list-style-type: none">- 数据特征具有多项式关系- 阶数 d 决定复杂度- γ 控制缩放, r 是偏置项
RBF 核（高斯核）	$K(x, x') = \exp\left(-\frac{\ x-x'\ ^2}{2\sigma^2}\right)$	σ 或 γ	<ul style="list-style-type: none">- 默认选择, 适合大多数非线性问题- 对高维数据效果较好- γ 控制样本影响范围
拉普拉斯核	$K(x, x') = \exp(-\gamma\ x - x'\)$	γ	<ul style="list-style-type: none">- 类似 RBF 核, 但基于绝对距离- 对稀疏数据敏感
Sigmoid 核	$K(x, x') = \tanh(\gamma \cdot x \cdot x' + r)$	γ, r	<ul style="list-style-type: none">- 类似于神经网络的激活函数- 不总是满足 Mercer 定理
自定义核函数	根据问题定义, 如 $K(x, x') = \text{similarity}(x, x')$	根据具体定义	<ul style="list-style-type: none">- 针对特殊问题（如字符串相似性、图数据）- 必须确保满足 Mercer 定理

导入必要的库

```
import numpy as np
```

```
from sklearn.svm import SVC
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.datasets import make_blobs
```

```
from sklearn.model_selection import train_test_split
```

1. 生成示例数据

使用 make_blobs 创建一个二分类数据集

```
X, y = make_blobs(n_samples=100, centers=2, cluster_std=1.0, random_state=42)
```

数据标签调整为 -1 和 1 (符合 SVM 的理论)

```
y = np.where(y == 0, -1, 1)
```

将数据分为训练集和测试集

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# 2. 定义支持向量机模型
# 使用 SVC 类定义 SVM，选择线性核函数
# 主要超参数说明：
# C：正则化参数（控制松弛变量的权重，值越大，惩罚越强，间隔可能越小）
# kernel：核函数类型，'linear' 表示线性核
# tol：优化器的容忍度，默认值为 1e-3
svm_model = SVC(C=1.0, kernel='linear', tol=1e-3)

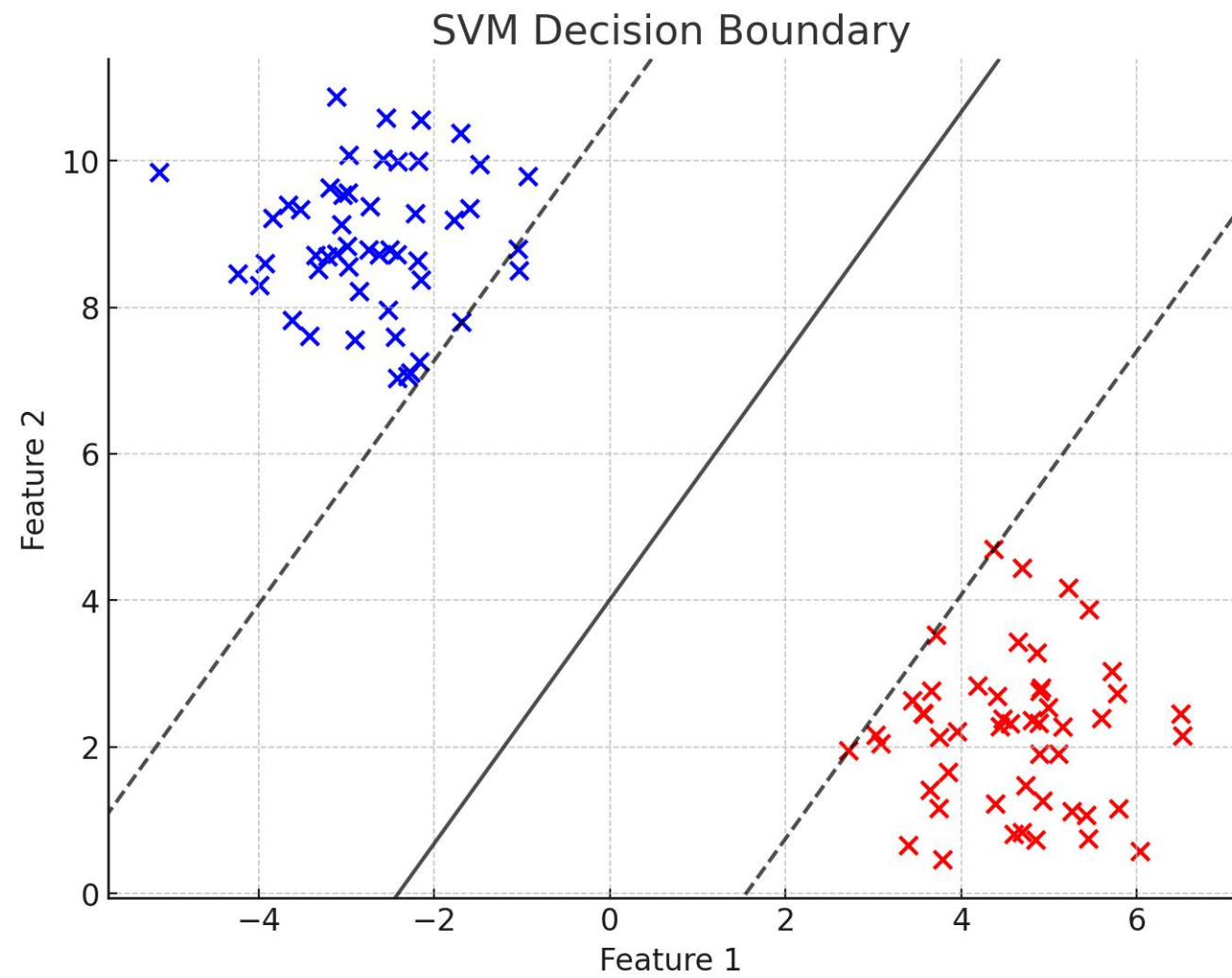
# 3. 训练 SVM 模型
svm_model.fit(X_train, y_train)

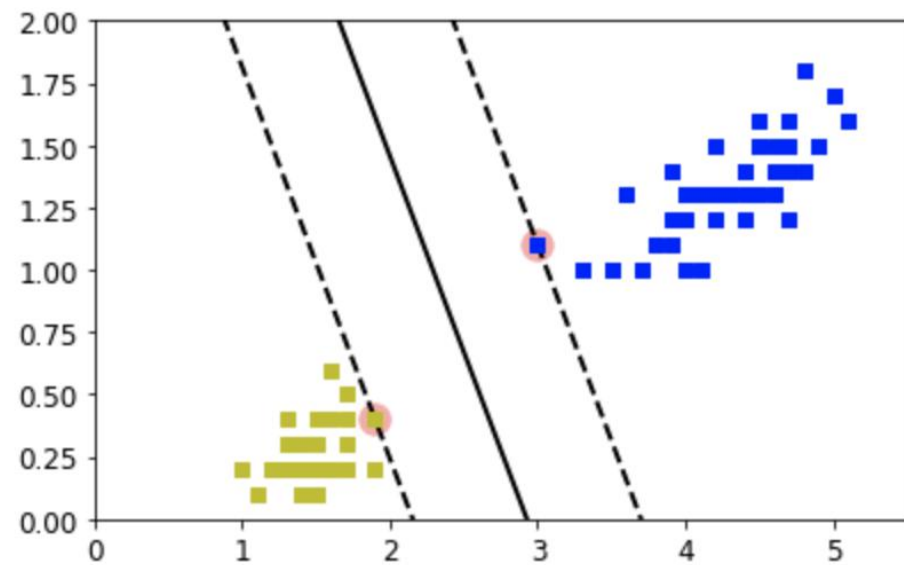
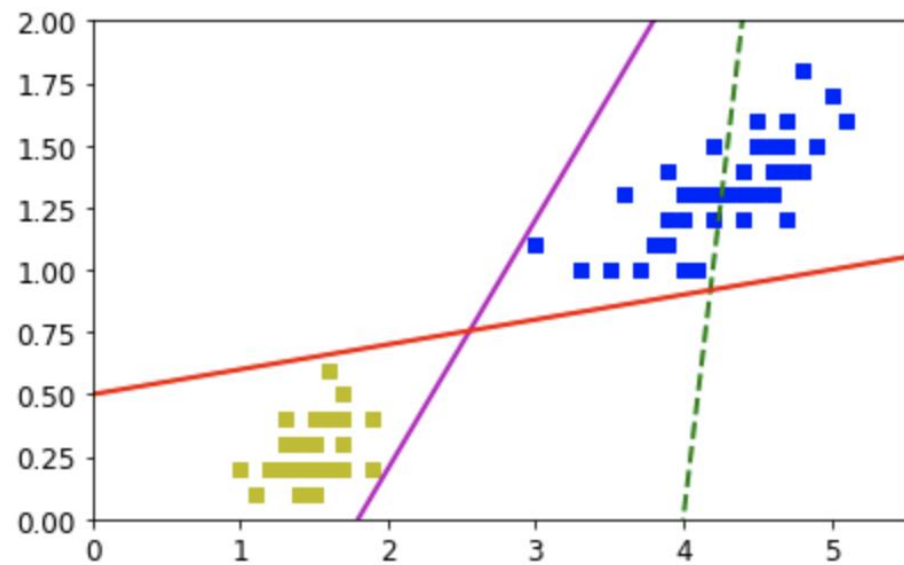
# 4. 查看训练结果
# 支持向量
support_vectors = svm_model.support_vectors_
print("支持向量：\n", support_vectors)
# 支持向量的索引
print("支持向量索引：\n", svm_model.support_)
# 每个类的支持向量个数
print("每个类的支持向量数量：\n", svm_model.n_support_)

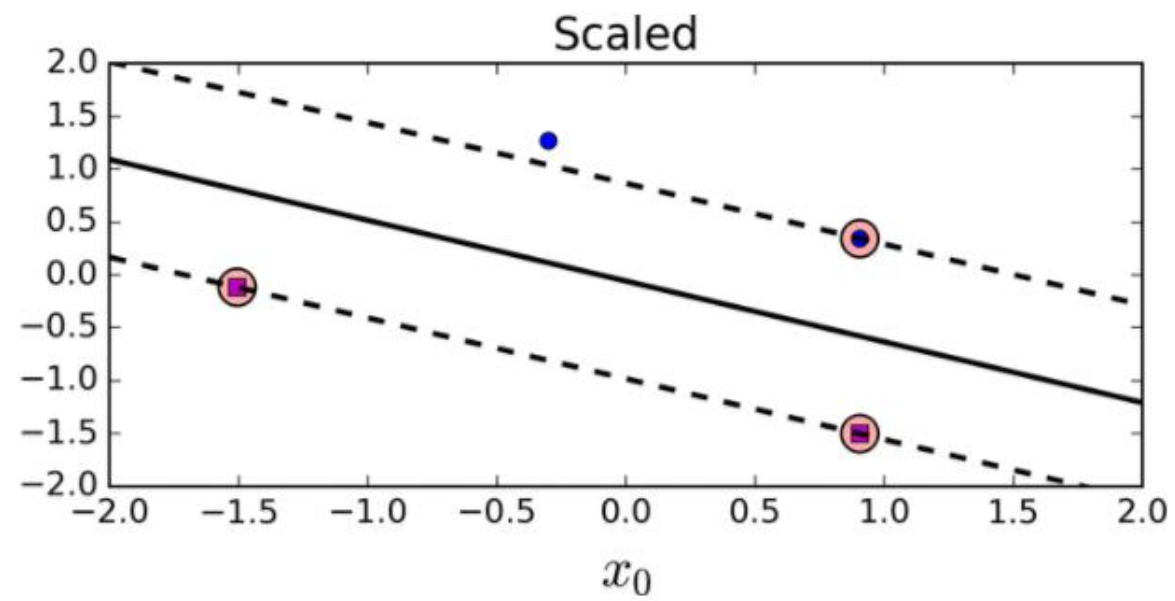
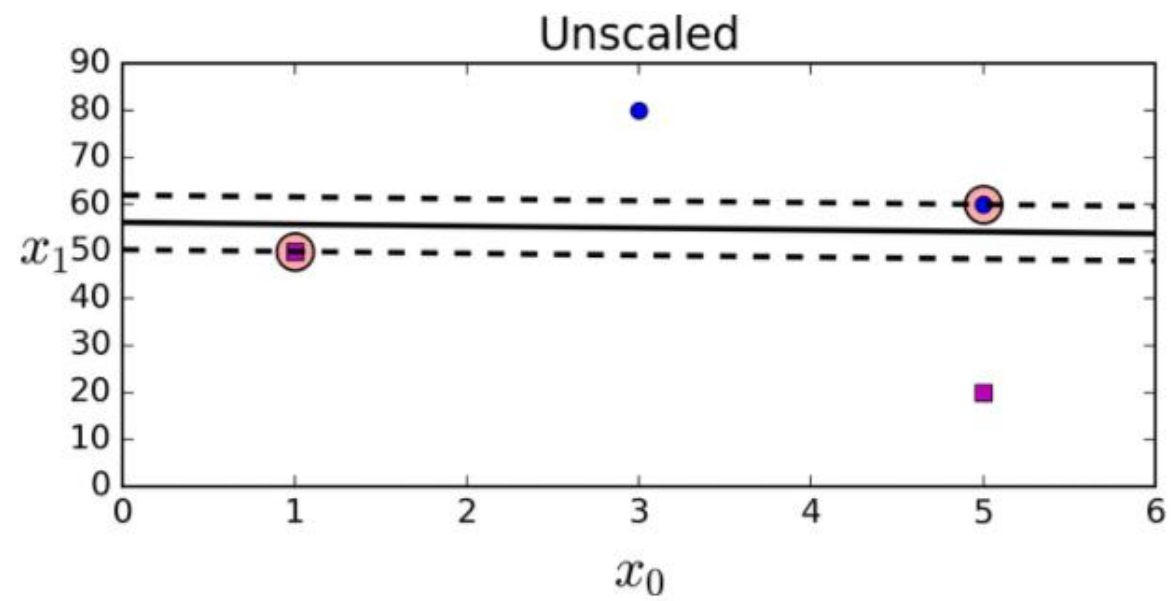
# 5. 测试模型
# 计算测试集的准确率
accuracy = svm_model.score(X_test, y_test)
print("测试集准确率：", accuracy)
```

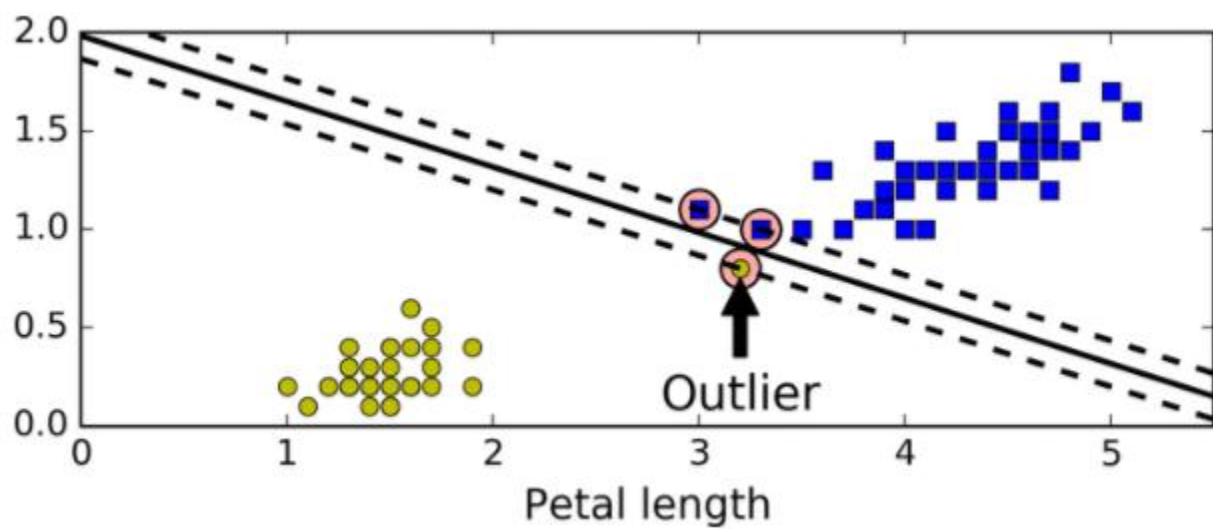
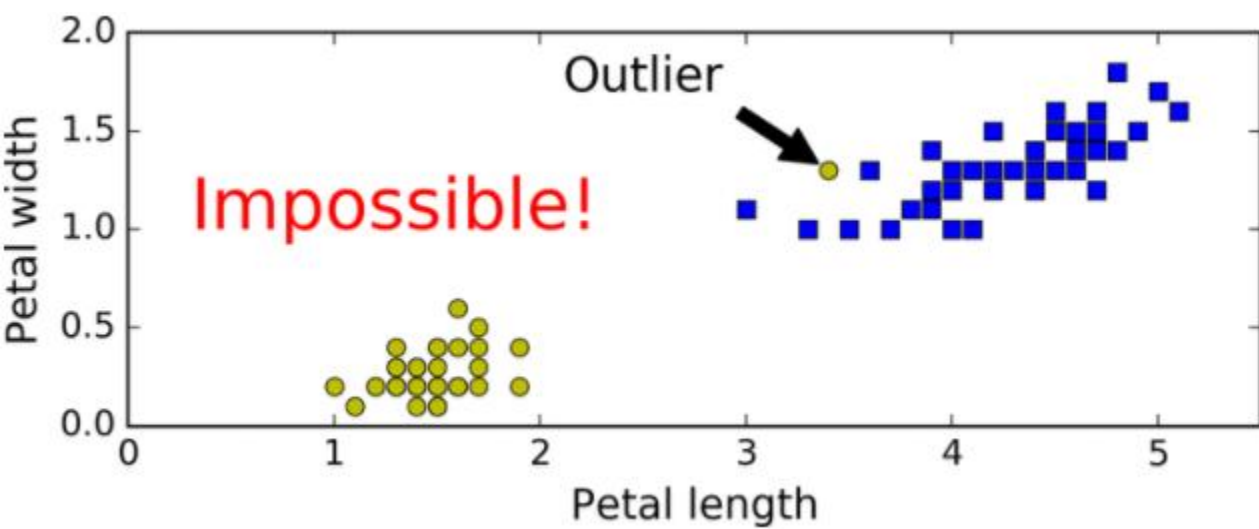
6. 可视化决策边界

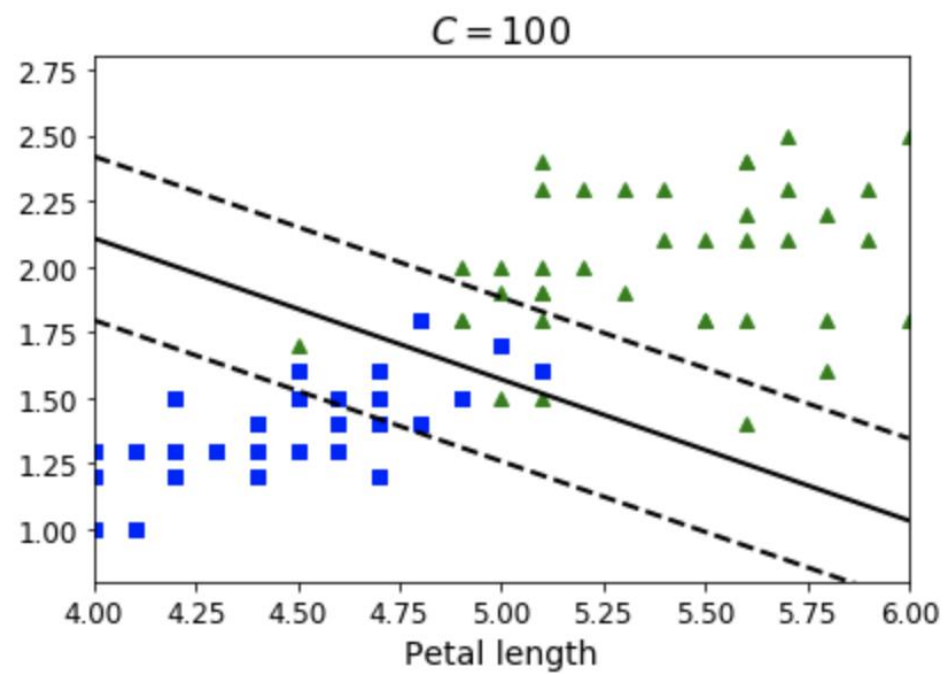
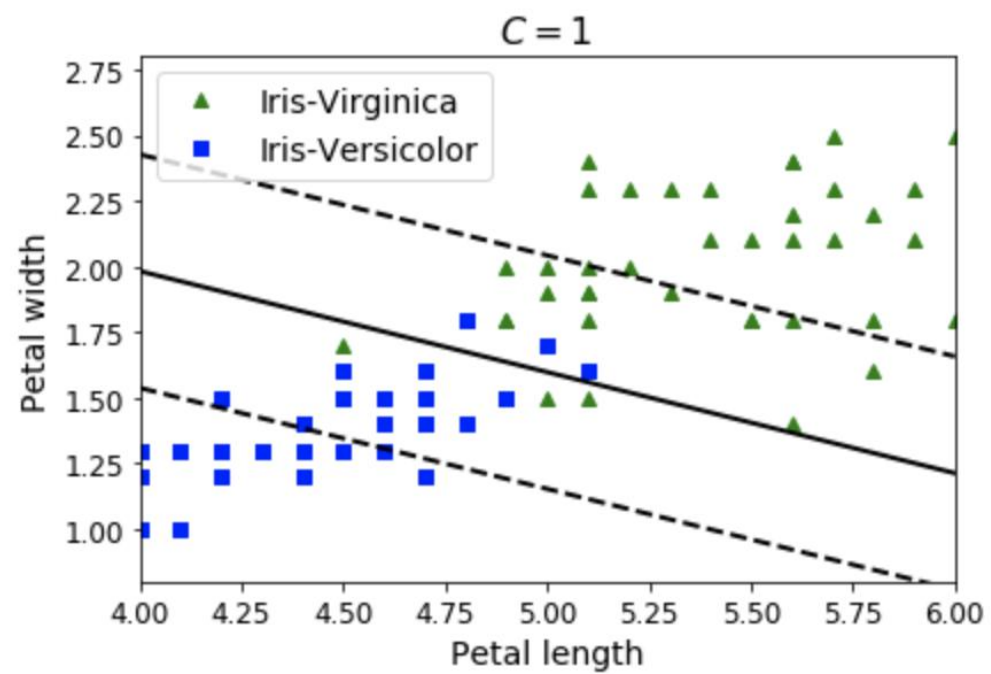
```
def plot_svm_decision_boundary(X, y, model):  
    """绘制数据点和 SVM 的决策边界"""  
    plt.figure(figsize=(8, 6))  
    # 绘制数据点  
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap='bwr', edgecolors='k', s=50)  
    # 绘制决策边界  
    ax = plt.gca()  
    xlim = ax.get_xlim()  
    ylim = ax.get_ylim()  
  
    # 创建网格以评估模型  
    xx, yy = np.meshgrid(np.linspace(xlim[0], xlim[1], 200),  
                          np.linspace(ylim[0], ylim[1], 200))  
    Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])  
    Z = Z.reshape(xx.shape)  
  
    # 绘制决策边界和间隔  
    ax.contour(xx, yy, Z, colors='k', levels=[-1, 0, 1], alpha=0.7,  
              linestyle=['--', '-', '--'])  
    # 绘制支持向量  
    ax.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1],  
              s=100, linewidth=1, facecolors='none', edgecolors='k')  
    plt.title("SVM Decision Boundary")  
    plt.xlabel("Feature 1")  
    plt.ylabel("Feature 2")  
    plt.show()  
  
# 调用绘图函数  
plot_svm_decision_boundary(X, y, svm_model)
```

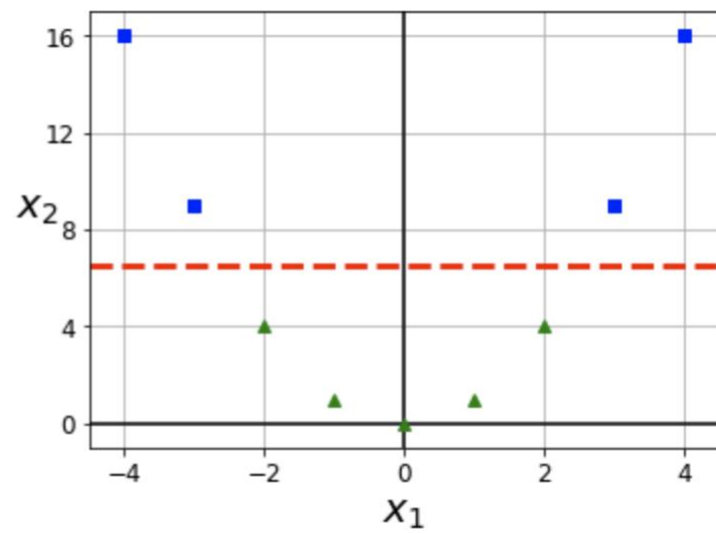
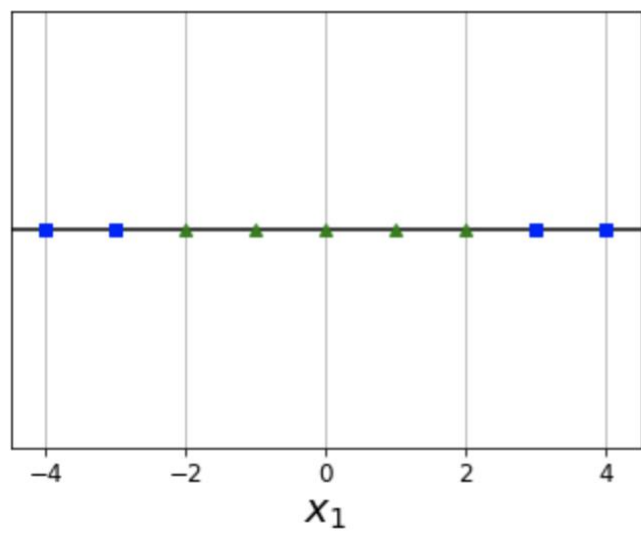


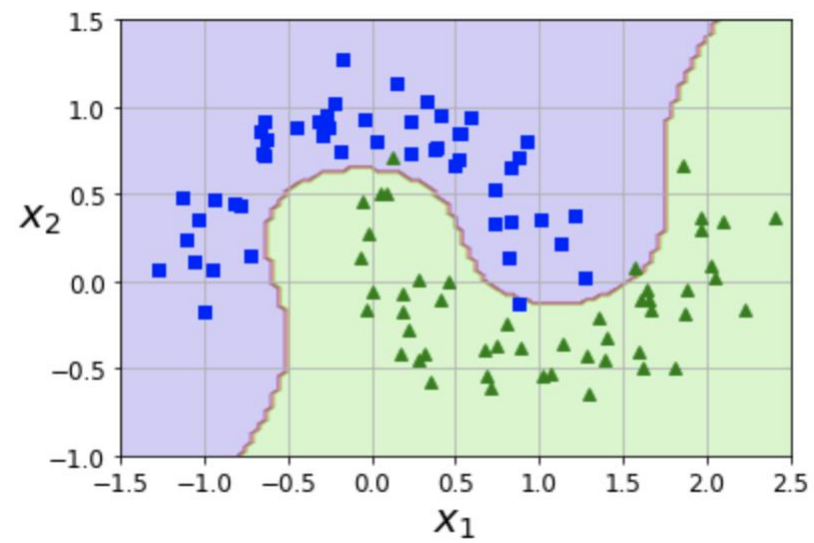
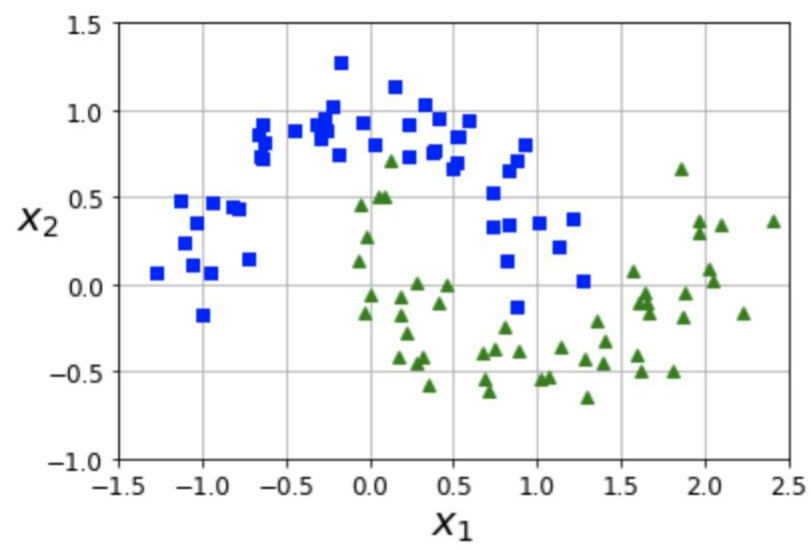












特性	支持向量机 (SVM)	深度学习 (Deep Learning)
理论基础	完备的统计学习理论	理论解释不完善，更多依赖实验
对数据量的需求	适合小数据集	需要大规模数据才能表现出色
计算复杂度	训练复杂度高，适合中小型数据集	对计算资源需求高，依赖 GPU/TPU
特征工程	需要手工设计特征	能够自动学习特征
非线性能力	通过核函数处理非线性问题	深层网络能够处理更复杂的非线性问题
对高维数据的处理	表现优秀，适合稀疏高维数据	可处理，但效率可能较低
泛化能力	泛化能力强，鲁棒性较好	在数据分布变化时表现不稳定
扩展性	二分类扩展为多分类较复杂	天然适用于多分类问题
解释性	可解释性较强	缺乏透明性，难以解释
适用场景	小规模数据、文本分类、稀疏数据	图像、语音、自然语言处理等复杂任务

给定以下 2 维数据点:

正类: $(1, 1), (2, 2)$ (标签 $y = 1$)

负类: $(2, 0), (0, -1)$ (标签 $y = -1$)

目标: 找到一个超平面 $w \cdot x + b = 0$, 使得分类边界最大化。