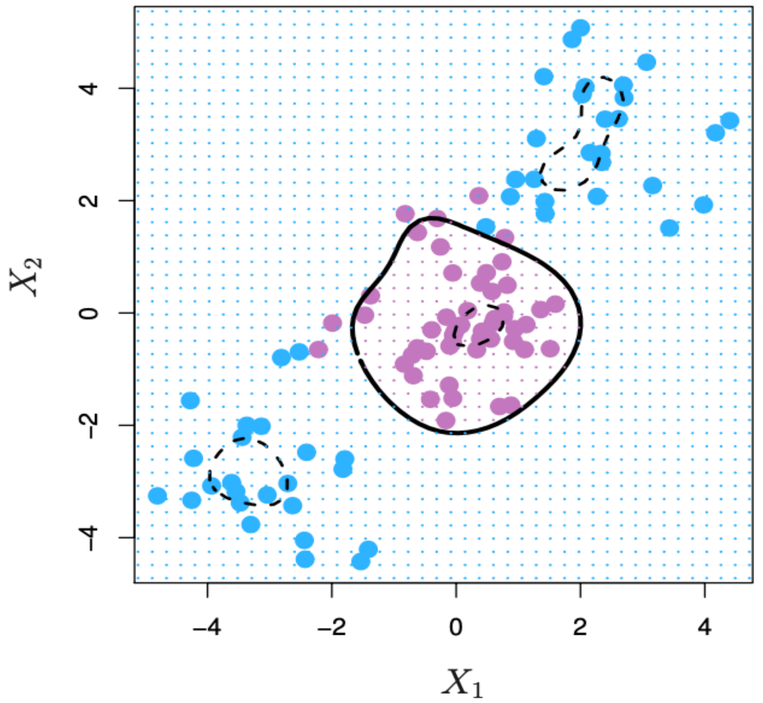
Support Vector Machines

支持向量机

开始之前

- 考虑两分类问题
- 之前所学的分类算法大多寻找线性的决策边界
- 当然，我们也可以将特征映射到多项式得到非线性决策边界。
- 这么做有很多困难，尤其是特征维度很高时
- 支持向量机允许我们用另一种方法去构造极为复杂的非线性决策边界
- 虽然有时效果不如神经网络，但仍旧大受欢迎：turn-key algorithm，没有太多参数



Outline

- 超平面
- 函数间隔与几何间隔
- 最优间隔分类器（线性可分，线性决策边界）
- 支持向量分类器（线性不可分，线性决策边界）
- 基于核函数的支持向量机（线性不可分，非线性决策边界）
- 基于R的实现

超平面(Hyperplane)

- 在 d 维空间中，超平面是 $d-1$ 维的平坦仿射子空间
- 例如，在二维空间中，超平面是平坦的一维子空间——线；在三维空间中，超平面是平坦的二维子空间——面
- 超平面的数学定义：$\beta_0 + \beta_1 X_1 + \cdots + \beta_d X_d = 0$
- 如果d维空间中的点 $\boldsymbol{x} = (x_1, \ldots, x_d)^T$ 满足上式，则这个点在超平面上
- 向量 $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_d)$ 叫做法向量(normal vector)，其指向与超平面正交的方向

超平面

- 如果$\beta_0 + \beta_1 X_1 + \cdots + \beta_d X_d > 0$，这告诉我们点$\boldsymbol{X}$在超平面的一边；否则，点$\boldsymbol{X}$在超平面的另一边
- 可以理解为：超平面将 d 维空间切割成了两半，因此可以非常自然地用于分类任务
- 可以画出法向量感受一下

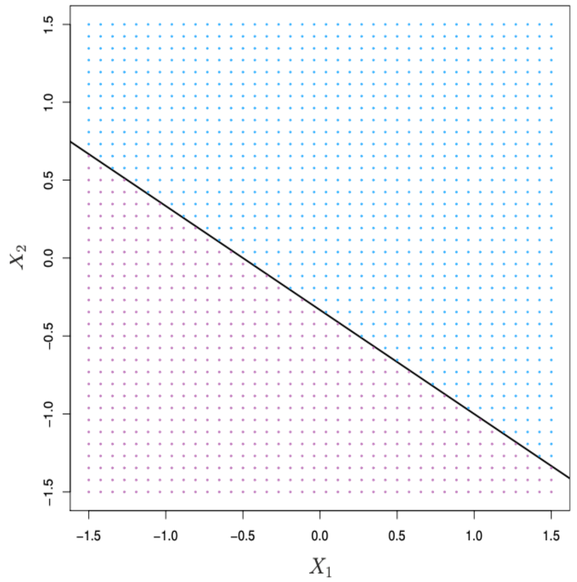


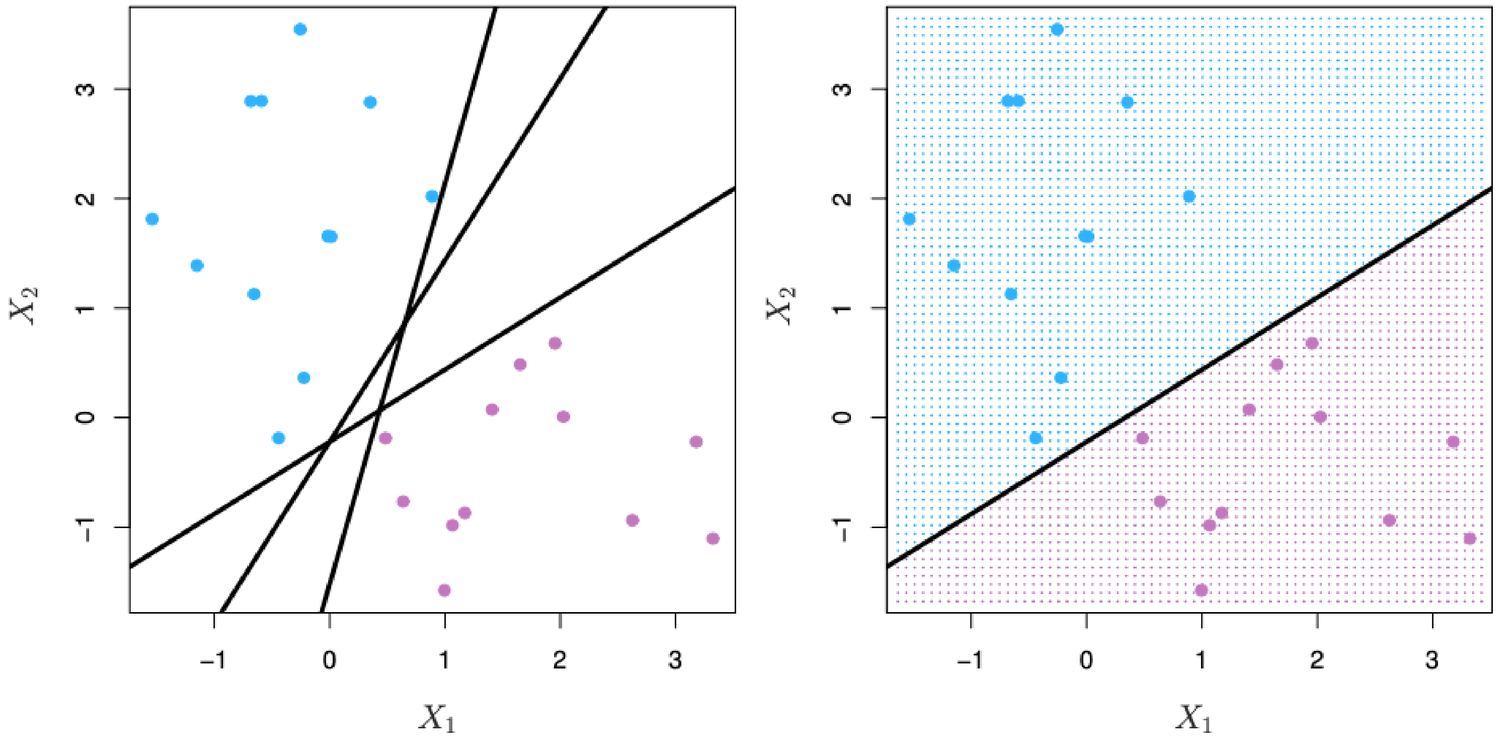
FIGURE 9.1. *The hyperplane* $1 + 2X_1 + 3X_2 = 0$ *is shown. The blue region is the set of points for which* $1 + 2X_1 + 3X_2 > 0$*, and the purple region is the set of points for which* $1 + 2X_1 + 3X_2 < 0$.

分离超平面(Separating Hyperplanes)

- 为了简便，我们令$b = \beta_0, \boldsymbol{w} = (\beta_1, \dots, \beta_d)^T$，超平面可表示为$\boldsymbol{w^T X} + b = 0$
- 训练样本：$\{\boldsymbol{x}^{(i)}, y^{(i)}\},\ i = 1, \dots, n,\ \ y^{(i)} \in \{1, -1\}$

- **分离超平面**是指可以将两类样本完全分离的超平面$\boldsymbol{w^T X} + b = 0$：
  - 当$y^{(i)} = 1,\quad \boldsymbol{w^T x}^{(i)} + b > 0$
  - 当$y^{(i)} = -1,\quad \boldsymbol{w^T x}^{(i)} + b < 0$
  - $y^{(i)}\left(\boldsymbol{w^T x}^{(i)} + b\right) > 0, i = 1, \dots, n$
- 分类函数：$f(\boldsymbol{x}) = g\left(\boldsymbol{w^T x} + b\right)$，$g(z) = \begin{cases} 1, & z \geq 0 \\ -1, & z < 0 \end{cases}$
- 这里的$f(\boldsymbol{x})$不再与$P(y = 1|\boldsymbol{x})$建立联系

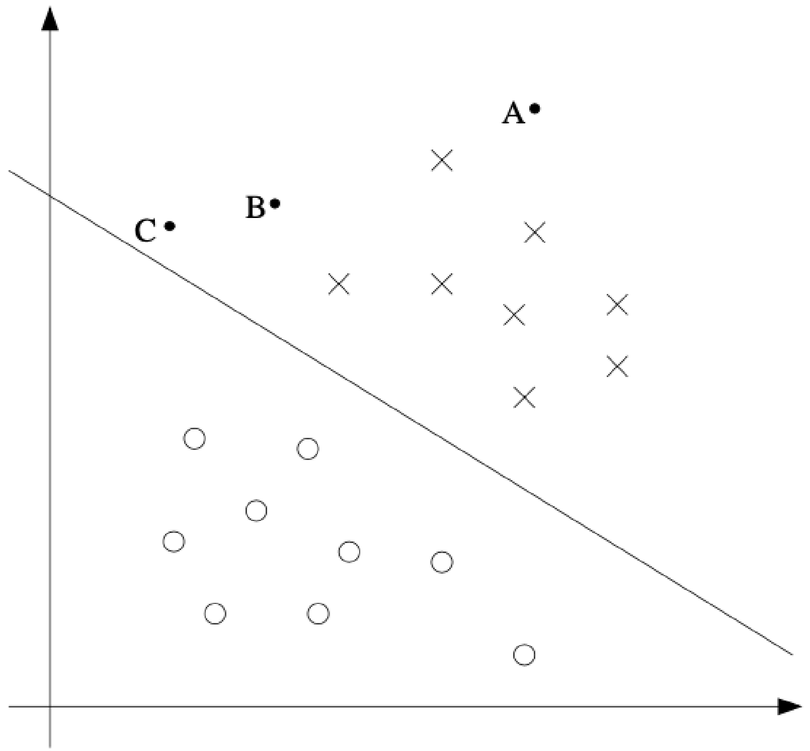
分离超平面



- 右图：分离超平面预测
- 左图：分离超平面若存在，一定不唯一。大家觉得哪个超平面最好呢？
- 随之而来的问题是：怎样才是好的分离超平面？

分离超平面

- 点A离分离超平面很远，如果我们要对其y值进行预测，应该会非常有信心
- 点C离分离超平面很近，一个小的改变都可能导致$\hat{y}$的变化，信心不足
- 点B在两者之间
- 这告诉我们：如果一个点离分离超平面越远，我们对其分类就越有信心
- 理想情况是：找到一个分离超平面，使我们对训练数据作出完全正确，且信心十足的预测（使训练数据离超平面尽可能远）
- 接下来，我们引入**函数间隔**和**几何间隔**的概念，来阐述这一思想



函数间隔(Functional Margin)

- 由 $(\boldsymbol{w}, b)$ 定义的超平面关于单个训练样本的函数间隔为：

$$\hat{\gamma}^{(i)} = y^{(i)}\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})} + b\right)$$

  - 当$\hat{\gamma}^{(i)} > 0$，说明我们对于样本 i 的预测是正确的
  - 当$y^{(i)} = 1$：　　我们想要$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})} + b \gg 0$，即函数间隔越大越好
  - 当$y^{(i)} = -1$：我们想要$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})} + b \ll 0$，即函数间隔越大越好
  - 换句话说：　　我们想要$\hat{\gamma}^{(i)} = y^{(i)}\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})} + b\right) \gg 0$

- 因此，函数间隔刻画了**分类的自信和准确程度**：函数间隔越大，说明分类越有信心

函数间隔(Functional Margin)

- 由$(\boldsymbol{w}, b)$定义的超平面关于整个训练数据集的函数间隔为：

$$\hat{\gamma} = \min_{i=1,\ldots,n} \hat{\gamma}^{(i)}$$

- **解释：**分离超平面在表现最差的样本上的函数间隔是多少

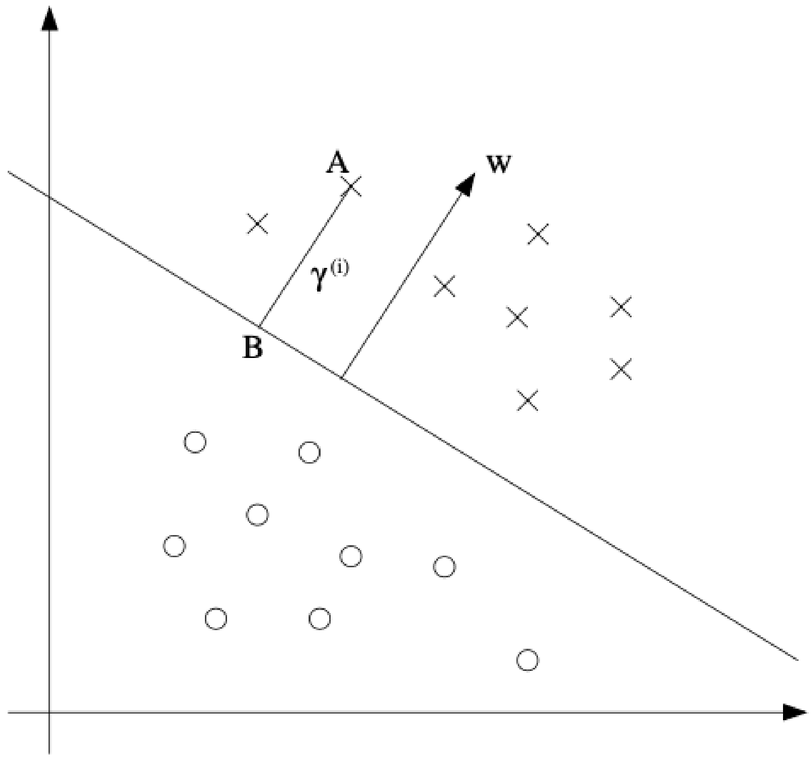
- 问题：将$(\boldsymbol{w}, b)$变为$(2\boldsymbol{w}, 2b)$，不影响分类结果，但会影响函数间隔的大小
- 解决方案：参数标准化。将$(\boldsymbol{w}, b)$变为$(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}, \frac{b}{\|\boldsymbol{w}\|})$，并考虑$(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}, \frac{b}{\|\boldsymbol{w}\|})$的函数间隔。

几何间隔(Geometric Margin)

- 法向量$\boldsymbol{w}$与分离超平面$\boldsymbol{w^T X} + b = 0$呈90度正交
- 右图点A代表一个$y^{(i)} = 1$的训练样本$\boldsymbol{x^{(i)}}$。其到分离超平面的距离 $\gamma^{(i)}$ 由线段AB表示
- $\boldsymbol{w}/\|\boldsymbol{w}\|$是一个单位长度的向量，和$\boldsymbol{w}$方向相同
- 点B则是$\boldsymbol{x^{(i)}} - \gamma^{(i)} \cdot \boldsymbol{w}/\|\boldsymbol{w}\|$，这个点在超平面上，则：

$$\boldsymbol{w^T}\left(\boldsymbol{x^{(i)}} - \gamma^{(i)} \cdot \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right) + b = 0$$

- $\gamma^{(i)} = \frac{\boldsymbol{w^T x^{(i)}} + b}{\|\boldsymbol{w}\|} = \left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right)^{\boldsymbol{T}} \boldsymbol{x^{(i)}} + \frac{b}{\|\boldsymbol{w}\|}$
- **思考：如果点A在分离超平面左侧呢？结合两侧结果，如何统一定义几何间隔？**



函数间隔与几何间隔

- 超平面关于某样本的**几何间隔**：

$$\gamma^{(i)} = y^{(i)}\left(\left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right)^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})} + \frac{b}{\|\boldsymbol{w}\|}\right)$$

- $\gamma^{(i)} = \frac{\hat{\gamma}^{(i)}}{\|\boldsymbol{w}\|}$。 如果$\|\boldsymbol{w}\| = 1$，函数间隔等于几何间隔

- 几何间隔不受参数缩放的影响。因此，我们就可以任意设置缩放参数来约束。例如： $\|\boldsymbol{w}\| = 1$。 这在后面SVM的提出中非常有用

- 由$(\boldsymbol{w}, b)$定义的分离超平面关于整个训练数据集的几何间隔为：

$$\gamma = \min_{i=1,\ldots,n} \gamma^{(i)} = \frac{\hat{\gamma}}{\|\boldsymbol{w}\|}$$

最优间隔分类器

最优间隔分类器(Optimal Margin Classifier)

- 给定一个训练数据集（线性可分），我们想要找到一个离训练数据最远的分离超平面，即找到一个有着最大间隔的分离超平面
- 思想：若分离超平面离训练数据集较远，则也离测试数据集较远，从而能够对测试数据进行准确、自信的分类预测

$$\max_{\gamma,w,b} \ \gamma$$

$$\text{s.t.} \ \frac{y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}+b\right)}{\|\boldsymbol{w}\|} \geq \gamma, \qquad i=1,\dots,n$$

- 解释：最大化几何间隔γ，在所有训练数据的几何间隔都大于等于γ的约束下。**最大化最坏情况下的几何间隔**
- 非凸，还需要进一步转换。首先将几何间隔转化为函数间隔：

$$\max_{\hat{\gamma},w,b} \ \frac{\hat{\gamma}}{\|\boldsymbol{w}\|}$$

$$\text{s.t.} \ y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}+b\right) \geq \hat{\gamma}, \qquad i=1,\dots,n$$

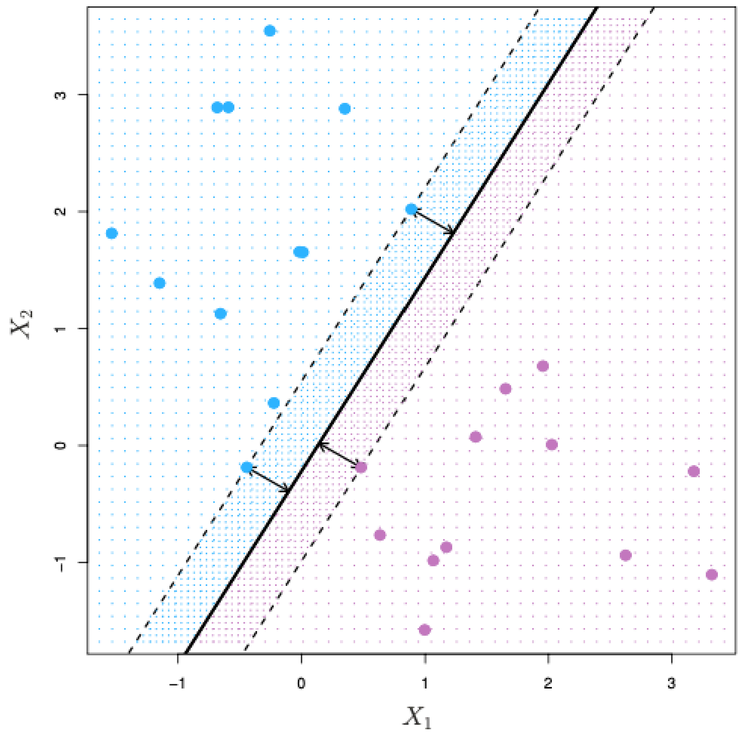
最优间隔分类器

- 前文讲过，我们可以对$(\boldsymbol{w}, b)$设置任意倍数的缩放，而不改变分离超平面
- 不妨设置缩放参数使得函数间隔$\hat{\gamma} = 1$:

$$\max_{w,b} \frac{1}{\|\boldsymbol{w}\|}$$
$$\text{s.t. } y^{(i)}\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})} + b\right) \geq 1, \qquad i = 1, \ldots, n$$

$$\min_{w,b} \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{s.t. } y^{(i)}\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})} + b\right) \geq 1, \qquad i = 1, \ldots, n$$

- 看似仅与$\boldsymbol{w}$有关，但事实上b通过不等式约束影响着$\boldsymbol{w}$的取值，进而影响分离超平面



最优间隔分类器

$$\min_{w,b} \ \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{s.t.} \ \ y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) \geq 1, \qquad i = 1, \ldots, n$$

- 这是一个凸二次规划（Quadratic Programming）问题，能直接利用现成的优化计算包求解，得到最优间隔分类器
- 尽管如此，我们要介绍基于拉格朗日对偶的求解方法，得到带约束最优化问题的对偶形式
- 这对我们后续引入核函数，并进行超高维空间上的高效计算非常有帮助
- 对偶形式也会帮助我们提出一个比凸二次规划更加有效的优化算法

讨论

- 我们一直在一个合理的特征维度下推导最优间隔分类器。例如：$\boldsymbol{x}^{(i)} \in \mathbb{R}^{10}$ or $\mathbb{R}^{100}$
- 为了得到SVM，我们将进一步假设 $\boldsymbol{w}$ 可以通过训练样本的线性组合来表示：

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \quad \boldsymbol{x}^{(i)}$$

- 如此一来，即使 $\boldsymbol{x}$ 的维度超级高，我们也可以推导出可以高效运行的算法
- 事实上，这并不完全是一个假设：接下来我们会看到，最优的$\boldsymbol{w}$就是可以表示成 $\boldsymbol{x}$ 的线性组合
- 在正式的推导之前，我们来看两个intuition，为什么这是一个合理的假设

Intuition #1

- $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})} + b = \sum_{j=1}^{n} \alpha_j \left(\boldsymbol{x}^{(\boldsymbol{j})}\right)^{\boldsymbol{T}} \boldsymbol{x}^{(\boldsymbol{i})} + b$

Intuition #2

- 如果α_i 中存在0?

拉格朗日对偶

拉格朗日乘子法

- 考虑如下带等式约束的最优化问题：

$$\min_{w} \ f(w)$$
$$\text{s.t.} \ \ h_i(w) = 0, \qquad i = 1, \dots, l$$

- 定义拉格朗日函数：$\mathcal{L}(w, \beta) = f(w) + \sum_{i=1}^{l} \beta_i h_i(w)$，　$\beta_i \in R$ 称为拉格朗日乘子
- 通过求解 $\frac{\partial \mathcal{L}(w,\beta)}{\partial w} = 0, \frac{\partial \mathcal{L}(w,\beta)}{\partial \beta_i} = 0$ 得到 $(w, \beta_1, \dots, \beta_l)$ 的值
- 进一步考虑不等式约束

拉格朗日乘子法—原始(Primal)问题

- 考虑如下最优化问题：

$$\begin{aligned} \min_{w} \ & f(w) \\ \text{s.t.} \ & g_i(w) \le 0, \qquad i = 1, \dots, k \\ & h_i(w) = 0, \qquad i = 1, \dots, l \end{aligned}$$

- 定义拉格朗日函数：

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w), \qquad \alpha_i \ge 0, \beta_i \in R$$

- 给定w，考虑$\theta_P(w) = \max_{\alpha,\beta:\alpha_i \ge 0} \mathcal{L}(w, \alpha, \beta) = \max_{\alpha,\beta:\alpha_i \ge 0} f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$
  - 如果所有约束全部满足，$\theta_P(w) = f(w)$
  - 如果约束中至少有一个不满足，$\theta_P(w) = \infty$

拉格朗日乘子法—原始(Primal)问题

- 因此，如果我们考虑最小化问题：

$$\min_{w} \theta_P(w) = \min_{w} \max_{\alpha,\beta:\alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

就可以发现，这个问题和最初的最优化问题等价，从而有着相同的解

- 我们将最优的目标值记为 $p^* = \min_{w} \max_{\alpha,\beta:\alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$

- 如果交换min与max的顺序，会发生什么？

拉格朗日乘子法—对偶(Dual)问题

- $\theta_D(\alpha,\beta)=\min_w \mathcal{L}(w,\alpha,\beta)$
- 定义对偶最优化问题：

$$\max_{\alpha,\beta:\alpha_i\geq 0}\theta_D(\alpha,\beta)=\max_{\alpha,\beta:\alpha_i\geq 0}\min_w \mathcal{L}(w,\alpha,\beta)$$

- 与原始问题相比，仅仅是min与max交换了顺序！
- 我们将最优的目标值记为 $d^*=\max_{\alpha,\beta:\alpha_i\geq 0}\min_w \mathcal{L}(w,\alpha,\beta)$
- d^*与 p^*的关系如何？

原始与对偶

- 弱对偶性：

$$d^* = \max_{\alpha,\beta:\alpha_i \geq 0} \min_{w} \mathcal{L}(w,\alpha,\beta) \leq \min_{w} \max_{\alpha,\beta:\alpha_i \geq 0} \mathcal{L}(w,\alpha,\beta) = p^*$$

- 证明：

原始与对偶

- 在特定条件下，我们有强对偶性（$d^* = p^*$），因此可以用对偶问题来替代原始问题：
  - f与g_i都是凸函数
  - h_i为带截距的线性函数
  - 存在w，使得$g_i(w) < 0$对所有i都成立

- 可以验证，最优间隔分类器满足这些条件

Karush-Kuhn-Tucker (KKT) 条件

- 在上述条件下，必然存在参数解w^*, α^*, β^*（w^* 是原始问题的解； α^*, β^* 是对偶问题的解），使得：

$$d^* = p^* = \mathcal{L}(w^*, \alpha^*, \beta^*)$$

- 此外， w^*, α^*, β^*满足KKT条件：

$$\begin{aligned}
\frac{\partial}{\partial w_i}\mathcal{L}(w^*, \alpha^*, \beta^*) &= 0, \quad i = 1, \ldots, d \\
\frac{\partial}{\partial \beta_i}\mathcal{L}(w^*, \alpha^*, \beta^*) &= 0, \quad i = 1, \ldots, l \\
\alpha_i^* g_i(w^*) &= 0, \quad i = 1, \ldots, k \\
g_i(w^*) &\leq 0, \quad i = 1, \ldots, k \\
\alpha^* &\geq 0, \quad i = 1, \ldots, k
\end{aligned}$$

- 反过来，如果参数满足KKT条件，他们也是原始与对偶最优化问题的解
- 重点关注第三个条件：若$\alpha_i^* > 0$，则$g_i(w^*) = 0$；若$g_i(w^*) < 0$，则$\alpha_i^* = 0$
- 这是解释SVM只有极少数**支持向量**的关键

最优间隔分类器的对偶形式

- 最优间隔分类器与其对偶形式的关系，以及原始变量与对偶变量的关系，是推导SVM的关键

- 原始问题：

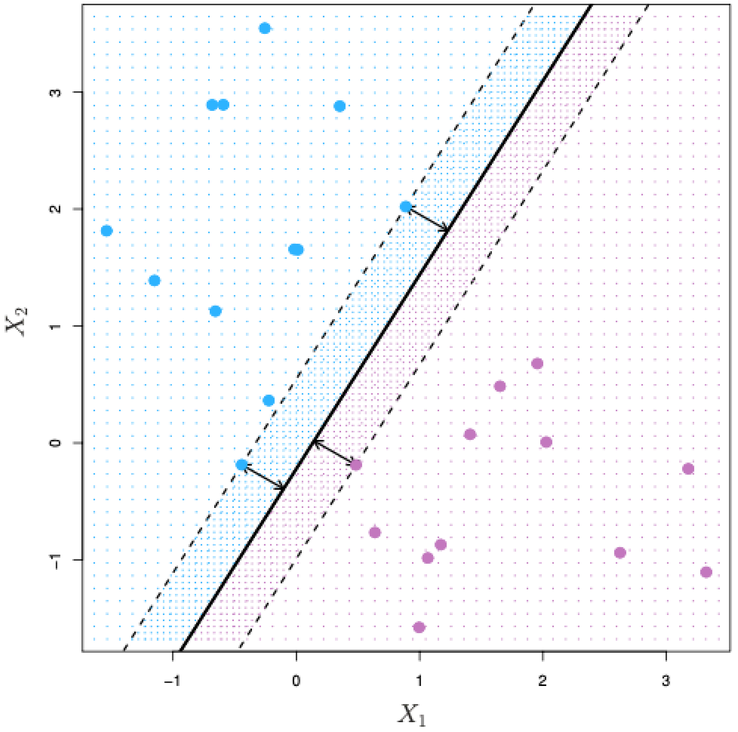
$$\min_{w,b} \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{s.t.} \quad y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}+b\right) \geq 1, \quad i = 1, \dots, n$$

- 将不等式约束写作：$g_i(\boldsymbol{w}) = -y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}+b\right)+1 \leq 0, \ i = 1, \dots, n$

- 通过KKT条件，只有当$g_i(\boldsymbol{w}) = 0$，即函数间隔等于1，才有$\alpha_i > 0$；否则 $\alpha_i = 0$

支持向量(Support Vectors)

- 考虑图中黑色实线所示的最优间隔分类器
- 有着最小间隔（函数间隔等于1）的三个点离分离超平面最近，且距离相同
- 只有这三个点的$\alpha_i > 0$，其余点的α_i都等于0。这三个点被称作**支持向量**
- 支持向量的个数远远小于训练数据总数
- 有趣的是，最优间隔分类器只取决于这些支持向量



最优间隔分类器的对偶形式

- 在正式的推导之前，我们先做一个前瞻
- 最重要的想法之一是：尝试将算法写作$\langle \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \rangle$的形式
- 这将会是我们后面引入**核函数**的关键

最优间隔分类器的对偶形式

- 构造拉格朗日函数：

$$\mathcal{L}(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y^{(i)} \left(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b \right) - 1 \right]$$

- 首先固定α，求$\theta_D(\alpha) = \min_{\boldsymbol{w},b} \mathcal{L}(\boldsymbol{w}, b, \alpha)$
- $\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, b, \alpha) = \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i \, y^{(i)} \boldsymbol{x}^{(i)} = 0 \;\Rightarrow \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \, y^{(i)} \boldsymbol{x}^{(i)}$
- $\frac{\partial}{\partial b} \mathcal{L}(\boldsymbol{w}, b, \alpha) = \sum_{i=1}^{n} \alpha_i \, y^{(i)} = 0$

$$\mathcal{L}(\boldsymbol{w}, b, \alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} \alpha_i \alpha_j \left\langle \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \right\rangle$$

最优间隔分类器的对偶形式

- 最终的对偶问题：

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \rangle$$

$$\text{s.t.} \quad \alpha_i \geq 0, \qquad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i \, y^{(i)} = 0$$

- 可以验证满足使$d^* = p^*$ 和KKT成立的条件，对偶问题的最优解与原始问题的最优解相同
- 利用SMO（sequential minimal optimization）算法求解上式得到α_i^*，从而得到w^*与b^*：
  - $w^* = \sum_{i=1}^{n} \alpha_i^* \, y^{(i)} \boldsymbol{x}^{(i)}$
  - $b^* = -\dfrac{\max\limits_{i:y^{(i)}=-1} w^{*T} \boldsymbol{x}^{(i)} + \min\limits_{i:y^{(i)}=1} w^{*T} \boldsymbol{x}^{(i)}}{2}$

坐标上升(Coordinate Ascent)

- 考虑不带约束的最优化问题：

$$\max_{\alpha} W(\alpha_1, \ldots, \alpha_n)$$

- 与梯度上升算法不同，我们考虑一个新的算法——坐标上升：

Loop until convergence: {

For $i = 1, \ldots, n$, {

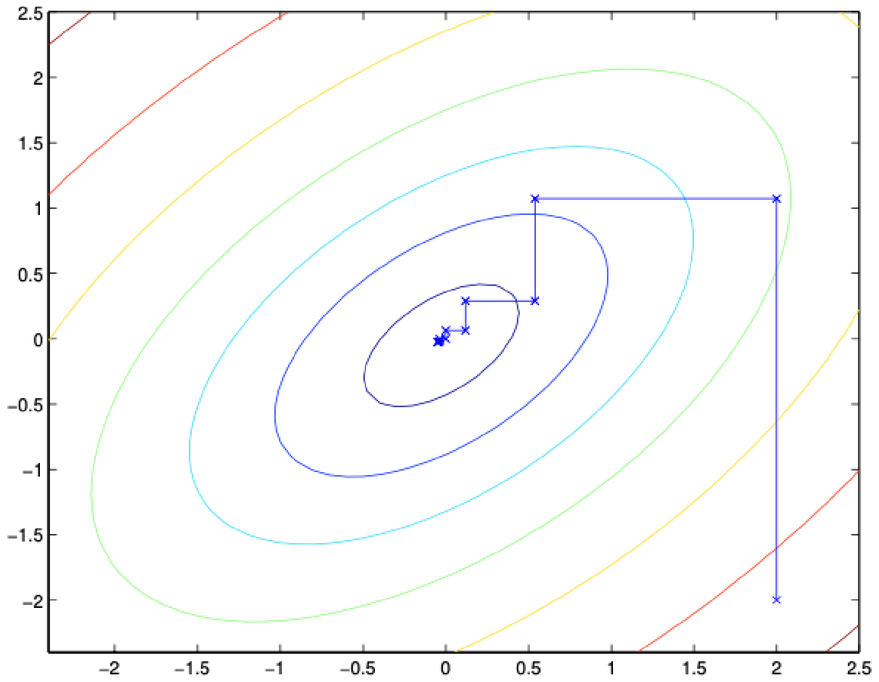
$\alpha_i := \arg\max_{\hat{\alpha}_i} W(\alpha_1, \ldots, \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, \ldots, \alpha_n)$.

}

}

- 在每次循环，我们固定除了α_i以外的所有变量，仅关于α_i优化W
- 一个更加精确的做法是：在每次选择更新的变量时，我们选择会使W变化最大的变量

坐标上升(Coordinate Ascent)



SMO (Sequential Minimal Optimization)

- 回到最优间隔分类器：

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \rangle$$
$$\text{s.t.} \quad \alpha_i \geq 0, \ i = 1, \ldots, n$$
$$\sum_{i=1}^{n} \alpha_i \, y^{(i)} = 0$$

- 假设有一组α_i满足约束。如果我们想要利用坐标上升去更新其中一个，会怎么样？
- $\alpha_1 y^{(1)} = -\sum_{i=2}^{n} \alpha_i \, y^{(i)}$
- $\alpha_1 = -y^{(1)} \sum_{i=2}^{n} \alpha_i \, y^{(i)}$
- α_1完全由$\alpha_2, \ldots, \alpha_n$决定。在不破坏约束的条件下，我们无法做到给定$\alpha_2, \ldots, \alpha_n$去更新$\alpha_1$
- 同时更新两个呢？

SMO (Sequential Minimal Optimization)

- 重复直到收敛 {

1. 选择一对要更新的α_i和α_j
2. 关于α_i和α_j最优化目标函数，保持其他α取值不变

}

预测

- 现在有一个新的样本$\boldsymbol{x}$，我们想要对其进行预测
- 计算${\boldsymbol{w}^*}^T\boldsymbol{x}+b^*$，如果大于0则预测为1，否则预测为-1

$$\begin{aligned}{\boldsymbol{w}^*}^T\boldsymbol{x}+b^* &= \left(\sum_{i=1}^{n}\alpha_i^* y^{(i)}\boldsymbol{x}^{(i)}\right)^T\boldsymbol{x}+b^* \\ &= \sum_{i=1}^{n}\alpha_i^* y^{(i)}\left\langle\boldsymbol{x}^{(i)},\boldsymbol{x}\right\rangle+b^*\end{aligned}$$

- 除了支持向量，其余训练数据的$\alpha_i^*=0$。因此，只需要计算$\boldsymbol{x}$与支持向量的内积
- 训练完成后，大部分的训练样本都无需保留，**最终模型仅与支持向量有关**

为什么要对偶？

- 对偶问题将原始问题中的不等式约束转为了对偶问题中的等式约束，对偶问题往往更加容易求解。
- 可以很自然的引入核函数（拉格朗日表达式里面有内积，而核函数也是通过内积进行映射）。
- 改变了问题的复杂度。由求特征向量$\boldsymbol{w}$转化为求拉格朗日乘子α，在原始问题下，求解的复杂度与样本的维度有关，即$\boldsymbol{w}$的维度。在对偶问题下，只与样本数量有关。
- 求解更高效，因为只用求解拉格朗日乘子α，而α只有支持向量才为非0，其他全为0。

接下来的问题

- 计算样本$\boldsymbol{x}$之间的内积，计算量还是可能很大，有没有什么方法减小计算量呢？
- 若训练数据集不是线性可分的怎么办？
- 如果出现少量异常点怎么办？

- 对于前两个问题，我们将引入kernel核函数方法，将维度进行转换。
- 对于第三个问题，我们将引入惩罚项，效果类似于正则化，允许一定程度上的错误分类，从hard margin转为soft margin。

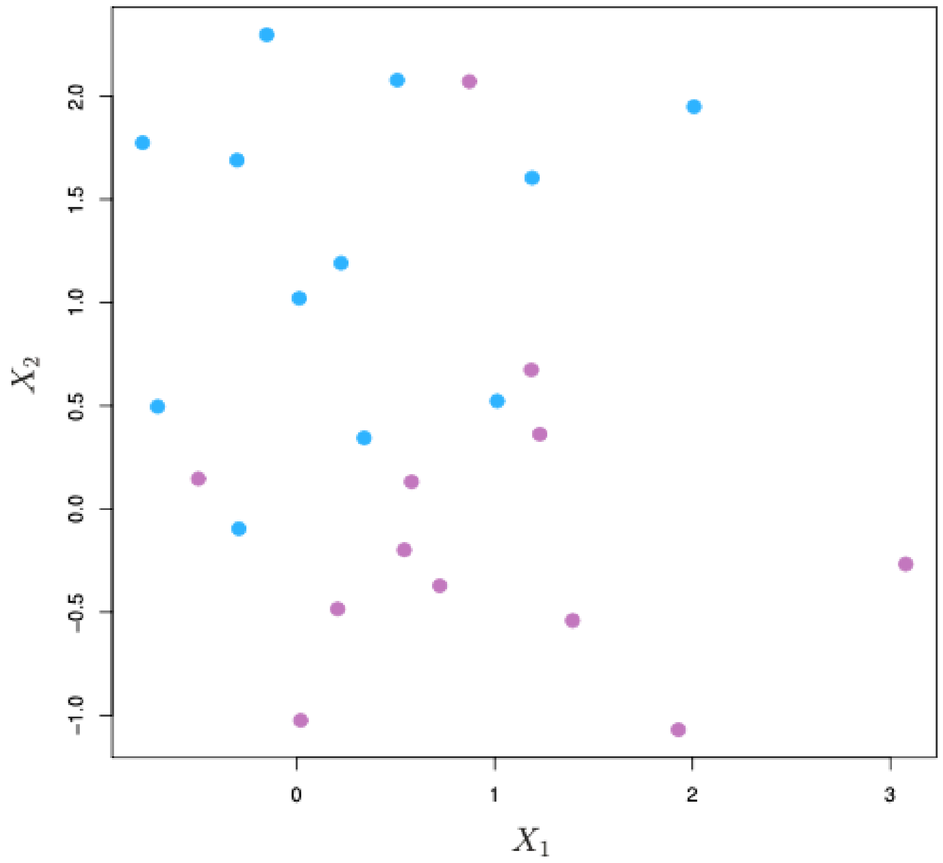
本章总结

- 超平面、分离超平面
- 函数间隔、几何间隔
- 最优间隔分类器
- 拉格朗日对偶
- 最优间隔分类器的对偶形式

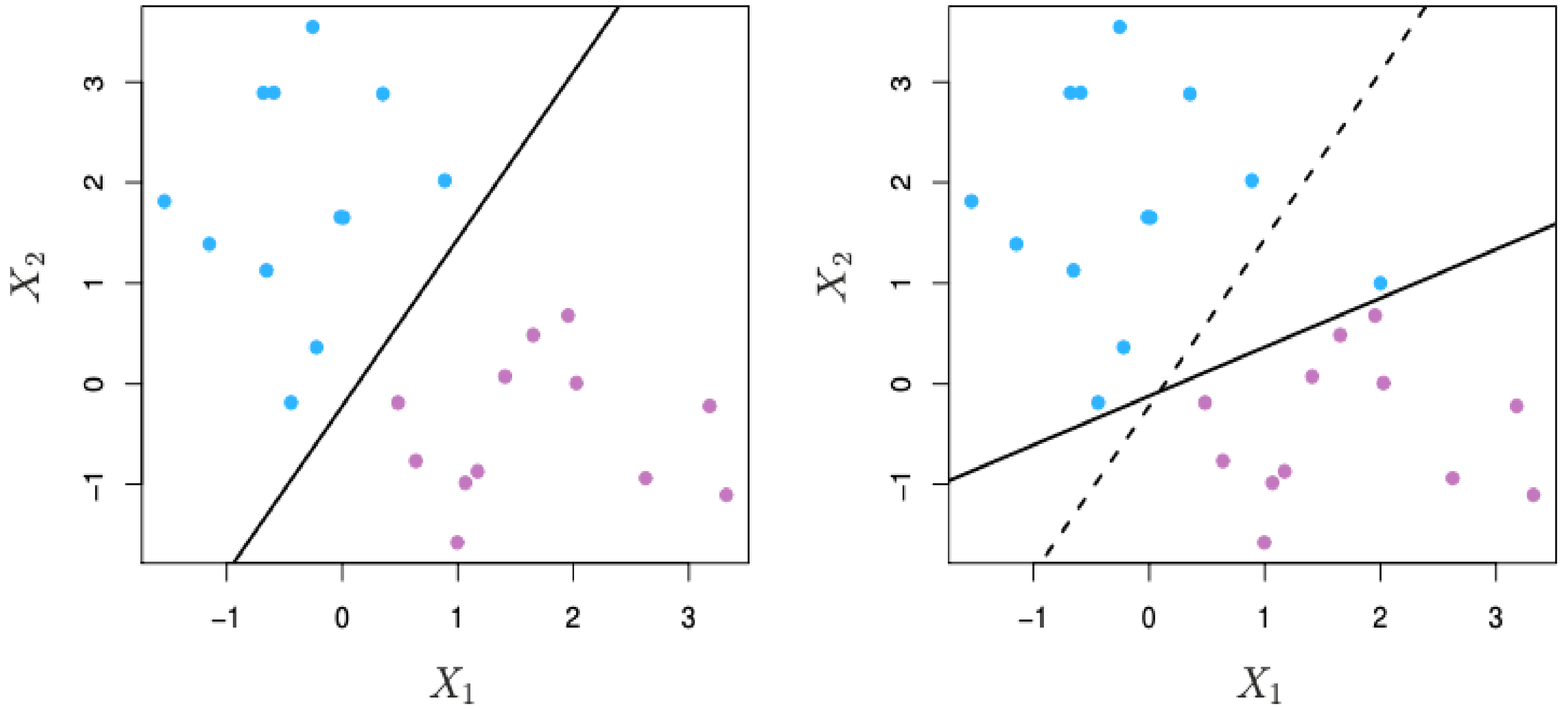
支持向量分类器

Motivation

- 目前为止，我们假设线性可分，实际中往往并不如此，从而导致最优间隔分类器的失效
- 利用$\phi(x)$将数据映射到更高维的空间可以增加数据可分离的概率，但并不能保证总是如此
- 另外，在某些情况下最优间隔分类器并不是最佳选择，例如存在异常值(Outlier)



异常值

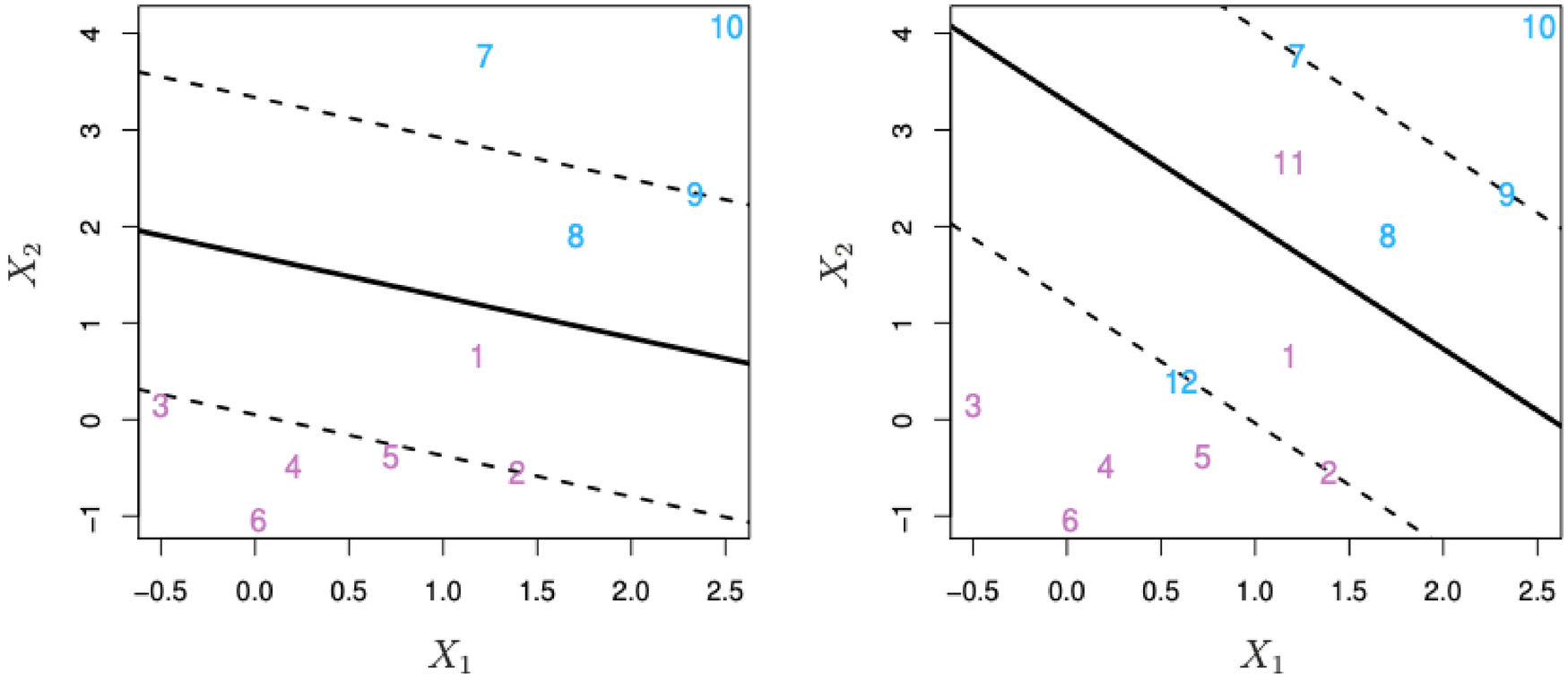


- 一个异常训练数据的增加可能会导致最优间隔分类器发生剧烈变化
- 最优间隔分类器效果并不好：间隔很小，影响预测信心
- 最优间隔分类器对单个观测过于敏感：可能存在过拟合问题

支持向量分类器

- 为此，我们想要分类器的分类超平面**不会强行刻意地**完美区分两类，出于以下两点：
  - 对于单个训练样本有更强的稳健性，减小方差
  - 对于**大部分**训练样本有更好的分类效果
- 即：错分一小部分样本，从而实现对大部分样本更好的预测
- 支持向量分类器：不再执着于寻找一个对所有训练数据都正确分类且信心十足的分离超平面，而是允许一部分训练数据在间隔的错误一侧，甚至是超平面的错误一侧
- 又名软间隔(soft margin)分类器

支持向量分类器——软间隔



- 左图：1和8跑到了间隔的错误一侧
- 右图：　1和8跑到了间隔的错误一侧；11和12跑到了超平面的错误一侧

支持向量分类器

$$\min_{w,b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t. } y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) \geq 1 - \xi_i, \qquad i = 1, \dots, n$$

$$\xi_i \geq 0, \qquad i = 1, \dots, n$$

- ξ_i 称为松弛变量（slack variables），允许部分观测的函数间隔小于1，从而在间隔或超平面的错误一侧
- 如果一个观测的函数间隔小于1，即为$1 - \xi_i$（ $\xi_i > 0$），我们需要付出代价，这个代价随着ξ_i 的增大而增大。因此，最小化的目标函数增加$C\xi_i$
- $C \geq 0$控制了两个目标之间的相对权重：使$\|\boldsymbol{w}\|^2$变小；保证大部分观测函数间隔大于等于1

支持向量分类器

$$\min_{w,b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t. } y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) \geq 1 - \xi_i, \qquad i = 1, \dots, n$$

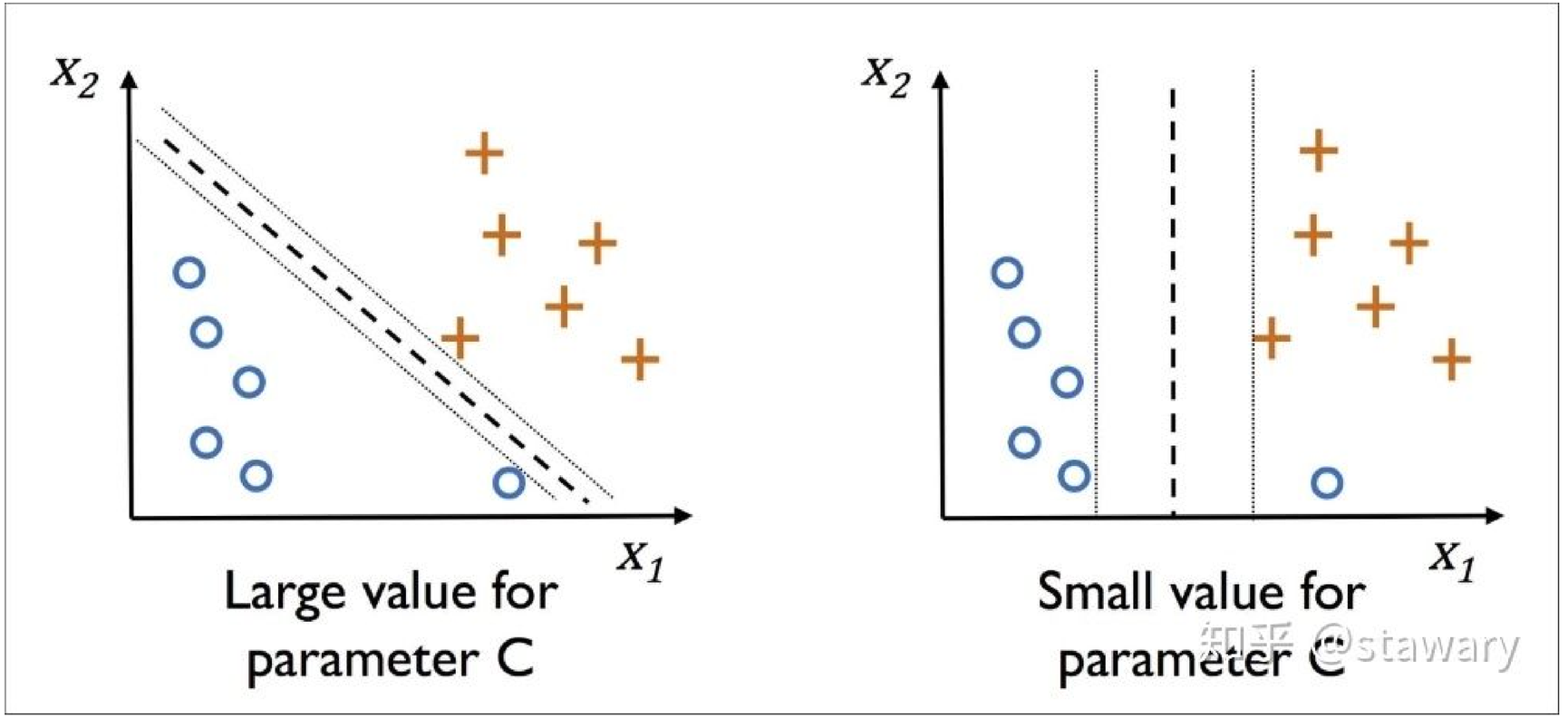
$$\xi_i \geq 0, \qquad i = 1, \dots, n$$

- 上面的问题看似复杂，但是它做的事情其实很简单：
- 首先，松弛变量ξ_i告诉了我们第i个训练数据相对于间隔和超平面的位置：
  - 如果$\xi_i = 0$，第i个训练数据在间隔的正确一侧
  - 如果$\xi_i > 0$，第i个训练数据在间隔的错误一侧
  - 如果$\xi_i > 1$，第i个训练数据在超平面的错误一侧

超参数C

- 如果C非常大，$\xi_i > 0$的代价非常高，对于在间隔错误一侧的观测容忍度很低，会使大部分的ξ_i等于0，间隔变窄，使得支持向量分类器近似等价于最优间隔分类器
- 如果C非常小，$\xi_i > 0$的代价非常低，对于在间隔错误一侧的观测容忍度很高，会使更多的ξ_i大于0，间隔变宽，使得支持向量分类器变得更加soft
- C 作为超参数，需要通过交叉验证来选择
- 从这个角度讲，C控制了偏差方差的平衡
  - 如果C非常大，分类器对数据拟合的更严格，低偏差，高方差
  - 如果C非常小，分类器对数据拟合的更宽松，高偏差，低方差

C


x_2
x_1
Large value for parameter C
x_2
x_1
Small value for parameter C

支持向量分类器——对偶问题

$$\min_{w,b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t. } y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) \geq 1 - \xi_i, \qquad i = 1, \dots, n$$

$$\xi_i \geq 0, \qquad i = 1, \dots, n$$

- 练习：构造拉格朗日函数，利用拉格朗日乘子法推导对偶问题。

对偶问题的推导

对偶问题

- 构造拉格朗日函数：

$$\mathcal{L}(w,b,\xi,\alpha,r)=\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{n}\xi_i-\sum_{i=1}^{n}\alpha_i\left[y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}+b\right)-1+\xi_i\right]-\sum_{i=1}^{n}r_i\xi_i$$

其中α_i 与r_i为非负拉格朗日乘子

- 拉格朗日对偶：$\min_{w,b,\xi}\ \max_{\alpha,r:\alpha_i\geq 0,r_i\geq 0}\mathcal{L}(w,b,\xi,\alpha,r)\rightarrow\max_{\alpha,r:\alpha_i\geq 0,r_i\geq 0}\ \min_{w,b,\xi}\mathcal{L}(w,b,\xi,\alpha,r)$

- 最终的对偶问题：

$$\max_{\alpha}\ \sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}y^{(i)}y^{(j)}\alpha_i\alpha_j\left\langle\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}\right\rangle$$
$$\text{s.t.}\ \ 0\leq\alpha_i\leq C,i=1,\dots,n$$
$$\sum_{i=1}^{n}\alpha_i\,y^{(i)}=0$$

- 仍可通过SMO算法求解

讨论

- 从最终的对偶问题形式来看，与最优间隔分类器唯一的区别就是对于α_i的约束：从$0 \leq \alpha_i$变成了$0 \leq \alpha_i \leq C$
- $w^* = \sum_{i=1}^{n} \alpha_i^* y^{(i)} \boldsymbol{x}^{(i)}$ 不变
- 此外，KKT条件也发生了变化：

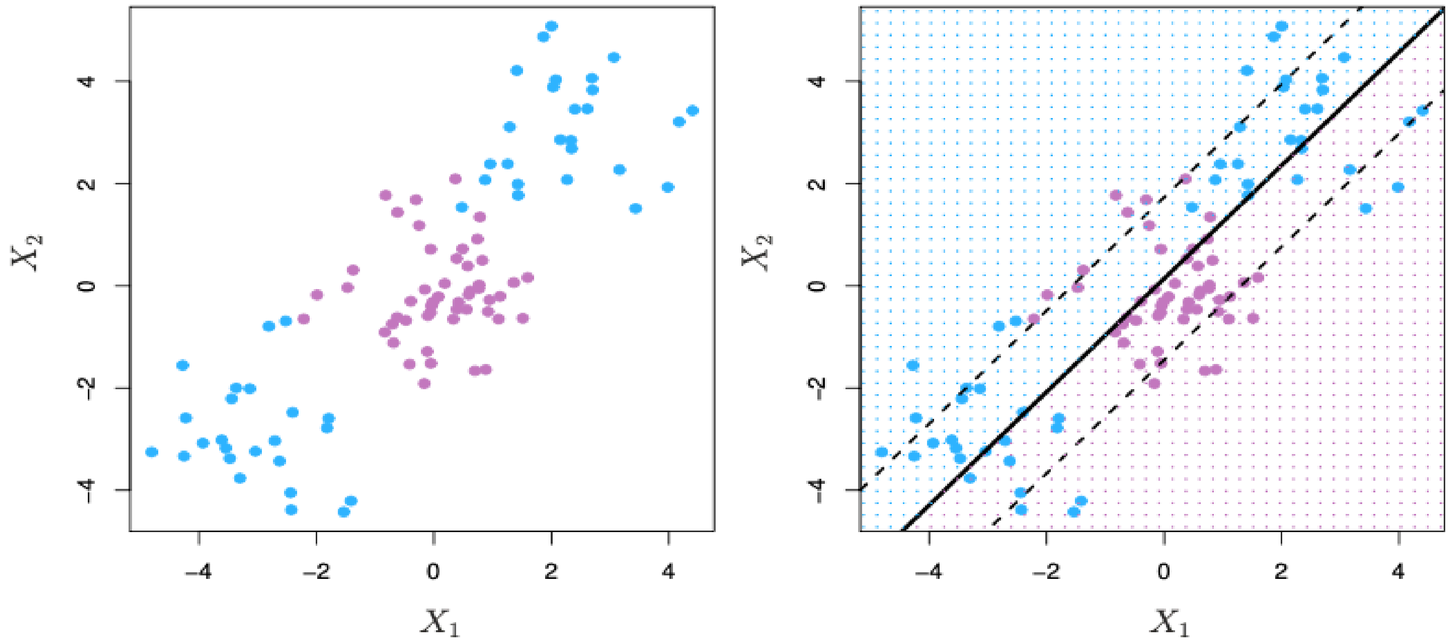
$$\begin{aligned} \alpha_i = 0 \quad &\Rightarrow \quad y^{(i)}(w^T x^{(i)} + b) \geq 1 \\ \alpha_i = C \quad &\Rightarrow \quad y^{(i)}(w^T x^{(i)} + b) \leq 1 \\ 0 < \alpha_i < C \quad &\Rightarrow \quad y^{(i)}(w^T x^{(i)} + b) = 1. \end{aligned}$$

- 也就是说，只有在间隔上的和在间隔错误一边的观测会影响超平面，而在间隔正确一边的观测完全不影响支持向量分类器
- 此时支持向量的定义发生了变化

核方法 Kernel Trick

Motivation

- 软间隔解决的是允许模型忽略某些少数异常点来进行划分的问题。
- 即在绝大部分样本都能被正确分类的情况下，某几个样本无法被超平面分开，我们不认为应该考虑它来重新修改模型，而是认为给出的样本是被错误分类的。
- 如果大部分样本都线性不可分怎么办？软间隔也无能为力



Motivation

- 在线性回归和逻辑回归的学习中，我们碰到过类似的情况。例如当响应变量和特征之间的关系不是线性时，线性回归的表现会变差
- 在这种情况下，我们考虑基于特征的函数来放大特征空间，比如二次项三次项等
- 在支持向量分类器中，我们仍然可以采取类似的方法。例如，将d个特征$X_1, \ldots, X_d$放大到$2d$个特征$X_1, X_1^2, X_2, X_2^2, \ldots, X_d, X_d^2$，在新的特征空间上构造线性决策边界
- 可以证明，同一个样本数据在越高维的空间中越有可能线性可分
- 回到原始的d维特征空间视角来看，得到的决策边界是非线性的
- 有太多的方法来放大特征空间，这会带来海量的特征，计算难度失控

- **核函数**，简单来说就是将低维空间的样本**高效地**映射到高维空间

Kernel Trick

- 第一步：基于$\langle x^{(i)}, x^{(j)} \rangle$（或$\langle x, z \rangle$）写出完整算法
- 第二步：构造低维到高维特征空间的映射：$x \to \phi(x)$
- 第三步：找到一种方法计算$K(x, z) = \phi(x)^T \phi(z)$
- 第四步：将算法中的$\langle x, z \rangle$替换为$K(x, z)$

- 优点：在ϕ的维度非常高或无穷维时仍可高效运行，无需显示计算$\phi(x)$

例1

- $\boldsymbol{x} = (x_1, x_2, x_3)^T \in \mathbb{R}^d$
- 注意：为方便展示， $d = 3$，实际中可能维数很高
- $\phi(\boldsymbol{x}) = \left(x_i x_j\right)^T \in \mathbb{R}^{d^2}$
- $\phi(\boldsymbol{z}) = \left(z_i z_j\right)^T \in \mathbb{R}^{d^2}$
- 直接计算$\phi(\boldsymbol{x})$或$\phi(\boldsymbol{x})^T \phi(\boldsymbol{z})$需要的时间为$O(d^2)$。 Too expensive!

例1

- 解决方案：$K(\boldsymbol{x},\boldsymbol{z}) = \langle \boldsymbol{x},\boldsymbol{z} \rangle^2 = \phi(\boldsymbol{x})^T \phi(\boldsymbol{z})$
- 计算时间大幅下降到了$O(d)$
- 证明：

例2

- $K(\boldsymbol{x},\boldsymbol{z}) = (\langle \boldsymbol{x},\boldsymbol{z}\rangle + c)^2 = \phi(\boldsymbol{x})^T\phi(\boldsymbol{z})$

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \\ \sqrt{2c}x_1 \\ \sqrt{2c}x_2 \\ \sqrt{2c}x_3 \\ c \end{bmatrix}$$

- c控制了一阶项与二阶项的相对权重
- 更一般的，$K(\boldsymbol{x},\boldsymbol{z}) = (\langle \boldsymbol{x},\boldsymbol{z}\rangle + c)^k$ 对应了到$\binom{d+k}{k}$维特征空间的映射，包含了全部至多k阶的单项式
- 不管k如何增长，利用核函数$K(\boldsymbol{x},\boldsymbol{z})$的计算量都是$O(d)$

支持向量机Support Vector Machine (SVM)

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} \alpha_i \alpha_j K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right)$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i \, y^{(i)} = 0$$

- 本质：软间隔分类器+核函数，在**超高维**的特征空间里**高效地**运行软间隔分类器
- 通过引入核函数，我们可以在超高维的特征空间里运行SVM，而将计算复杂度控制在只随特征维度线性增长
- 这样做的好处是什么？我们来看一段视频
- 在高维的特征空间中的线性分类超平面，回到原始特征空间中，就是高度非线性的决策边界

另一个视角：Kernel作为相似性度量

- 直觉上来说，如果$\boldsymbol{x}$和$\boldsymbol{z}$很相似，那么$K(\boldsymbol{x},\boldsymbol{z}) = \phi(\boldsymbol{x})^T\phi(\boldsymbol{z})$会很大；如果$\boldsymbol{x}$和$\boldsymbol{z}$很不相似，那么$K(\boldsymbol{x},\boldsymbol{z}) = \phi(\boldsymbol{x})^T\phi(\boldsymbol{z})$会很小
- 因此，我们可以将$K(\boldsymbol{x},\boldsymbol{z})$看做$\boldsymbol{x}$和$\boldsymbol{z}$、 $\phi(\boldsymbol{x})$和$\phi(\boldsymbol{z})$的相似性度量
- 据此，我们可以定义很多核函数。例如高斯核：

$$K(\boldsymbol{x},\boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{z}\|^2}{2\sigma^2}\right)$$

当$\boldsymbol{x}$和$\boldsymbol{z}$接近时趋近于1，当$\boldsymbol{x}$和$\boldsymbol{z}$疏远时趋近于0

- 问题来了：它可以作为一个合法的核函数使用吗？即，能找到与之对应的$\phi(\cdot)$使得$K(\boldsymbol{x},\boldsymbol{z}) = \phi(\boldsymbol{x})^T\phi(\boldsymbol{z})$成立吗？

核函数合法性

- 假设$K(\cdot,\cdot)$是一个合法的核函数，我们来看下其满足什么特征
- $\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(n)}\}$表示$n$个观测，令$\boldsymbol{K}\in\mathbb{R}^{n\times n}$表示kernel矩阵，其元素为：

$$K_{ij}=K\left(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}\right)$$

- 首先，$K_{ij}=\phi\left(\boldsymbol{x}^{(i)}\right)^{T}\phi\left(\boldsymbol{x}^{(j)}\right)=\phi\left(\boldsymbol{x}^{(j)}\right)^{T}\phi\left(\boldsymbol{x}^{(i)}\right)=K_{ji}$，说明$\boldsymbol{K}$是对称矩阵
- 其次，$\boldsymbol{K}$是半正定矩阵：对任意向量$\boldsymbol{z}$，我们有$\boldsymbol{z}^{\boldsymbol{T}}\boldsymbol{K}\boldsymbol{z}\geq 0$

核函数合法性

- $\boldsymbol{K}$是半正定矩阵：对任意向量$\boldsymbol{z}$，我们有$\boldsymbol{z}^T\boldsymbol{K}\boldsymbol{z} \geq 0$
- 证明：

核函数合法性

- 可以证明，这也是一个充分条件：当$\boldsymbol{K}$是一个对称的半正定矩阵时， K是一个合法的核函数，即存在$\phi(\cdot)$使得$K(\boldsymbol{x},\boldsymbol{z}) = \phi(\boldsymbol{x})^T\phi(\boldsymbol{z})$成立

> **Theorem (Mercer).** Let $K : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ be given. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{x^{(1)}, \ldots, x^{(n)}\}$, $(n < \infty)$, the corresponding kernel matrix is symmetric positive semi-definite.

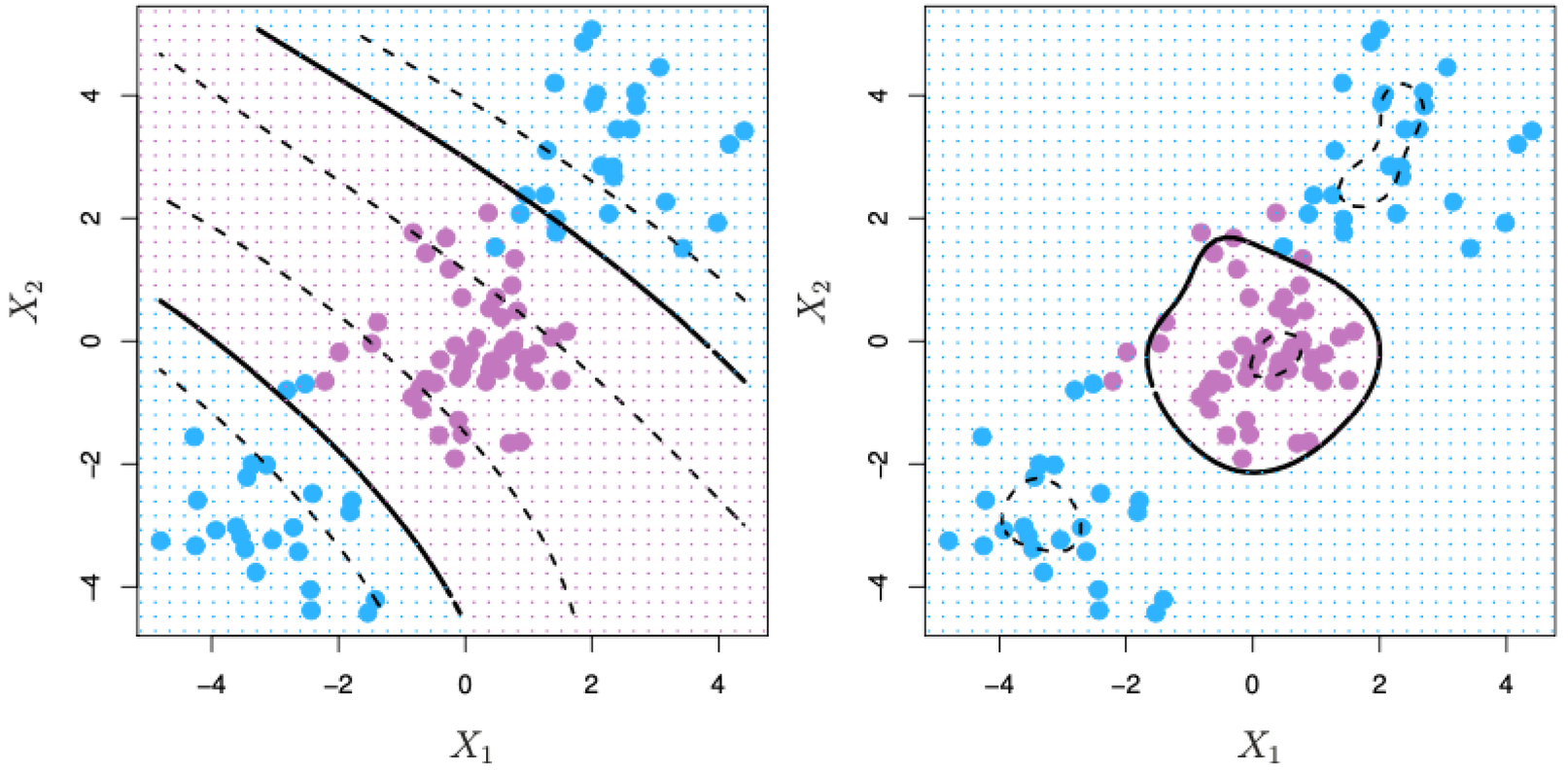
- 可以证明多项式核、高斯核都是合法的核函数

常用核函数

- 线性核： $K(\boldsymbol{x},\boldsymbol{z}) = \boldsymbol{x}^T\boldsymbol{z}, \quad \phi(\boldsymbol{x}) = \boldsymbol{x}$
- 高斯核： $K(\boldsymbol{x},\boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{z}\|^2}{2\sigma^2}\right)$
- 多项式核： $K(\boldsymbol{x},\boldsymbol{z}) = (\langle \boldsymbol{x},\boldsymbol{z}\rangle + c)^k$

SVM示例

- 多项式核与高斯核



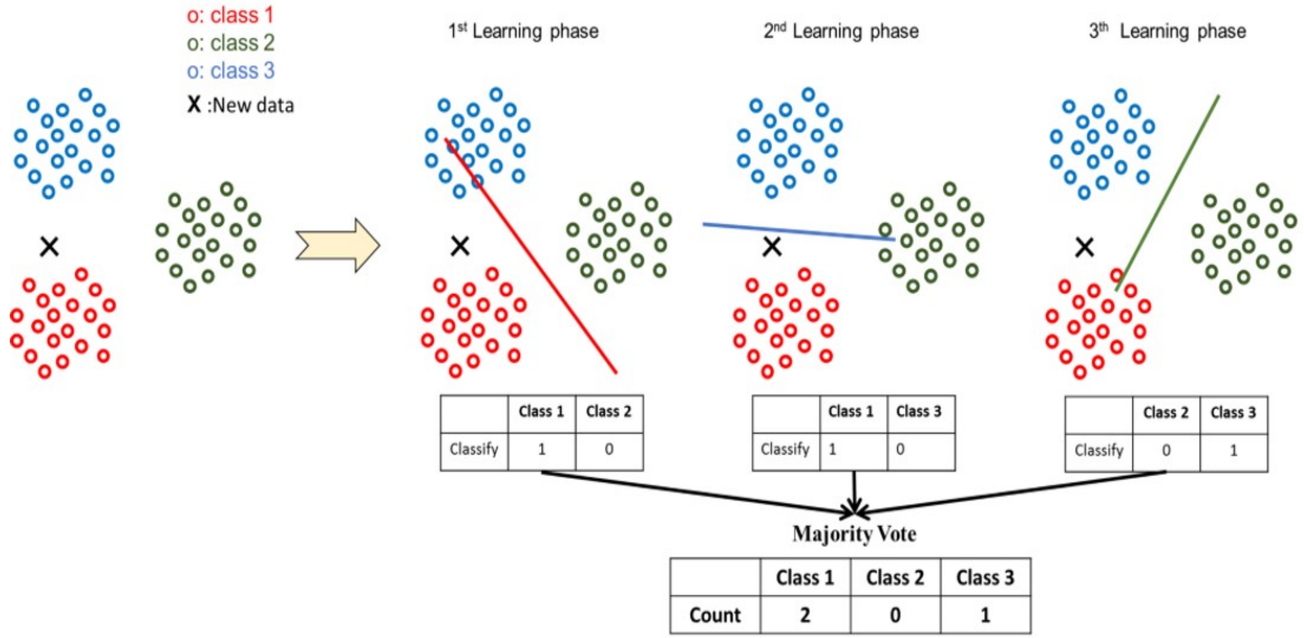
讨论

- 核函数在SVM的加持下大放异彩
- 事实上，只要可以表示成$\langle \boldsymbol{x}, \boldsymbol{z} \rangle$形式的算法，都可以引入核函数
- 我们目前学习过的线性回归，逻辑回归，GLM都可以结合核函数，只是相较于SVM应用少很多

- 目前为止，我们关于SVM的讨论都是在binary classification，我们怎么才能延伸到多分类问题下呢？
- 尽管有很多关于多分类SVM推广的讨论，最著名的方法是one-versus-one以及one-versus-all方法

多分类SVM： One-Versus-One

- 假设我们想要利用SVM进行分类，并且有$K > 2$类
- OVO方法构造$\binom{K}{2}$个SVM，每一个SVM比较一对类别。例如，其中一个SVM用来比较第k类与第k'类并构造两分类SVM
- 对于一个测试数据，我们利用每一个SVM分类器对其进行分类，并统计其被分到每一类中的次数
- 最终的分类结果即为次数最多的分类



本章小节

- 支持向量分类器（软间隔）
- 核函数
- 支持向量机

基于ADMM算法求解支持向量机

基于支持向量机原始问题的优化 I

- 给定训练样本 $\mathcal{Z}^{\mathrm{n}} = \{(y_i, x_i)\}_{i=1}^{n}$， SVM 标准正则化形式如下

$$\min_{f} \quad \frac{\lambda}{2} \| f \|^2 + \frac{1}{n} \sum_{i=1}^{n} max(0, 1 - y_i f(x_i))$$

- 考虑线性模型: $f(x_i) = \boldsymbol{\beta}^T x_i$
- 上述优化问题变为

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} max(0, 1 - y_i \boldsymbol{\beta}^T x_i) + \frac{\lambda}{2} \| \boldsymbol{\beta} \|_2^2$$

基于支持向量机原始问题的优化 I

- 考虑如下拆分

$$\min_{\boldsymbol{\beta}} \quad \frac{1}{n}\sum_{i=1}^{\mathrm{n}} a_i + \frac{\lambda}{2} \| \boldsymbol{\beta} \|_2^2$$
$$s.t. \quad 1 - y_i \boldsymbol{\beta}^T x_i = a_i - \xi_i, a_i \geq 0, \xi_i \geq 0,$$

- 上述问题的增广拉格朗日函数 为

$$\mathcal{L}(\boldsymbol{\beta}, a, \boldsymbol{\xi}, \boldsymbol{\mu}) = \frac{1}{n}\sum_{i=1}^{n} a_i + \frac{\lambda}{2} \| \boldsymbol{\beta} \|_2^2 + \frac{\rho}{2}\sum_{i=1}^{n} (1 - y_i \boldsymbol{\beta}^T x_i - a_i + \xi_i + \mu_i)^2$$

以及约束条件$\xi_i, a_i \geq 0.$

基于支持向量机原始问题的优化 I

- 给定初值 $\boldsymbol{\beta}^0, a^0, \boldsymbol{\xi}^0, \mu^0$

- 假设当前的迭代为第 $t \geq 0$ 步，那么具体的优化迭代步骤如下：

$$\boldsymbol{\beta}^{t+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{\lambda}{2} \| \boldsymbol{\beta} \|_2^2 + \frac{\rho}{2} \sum_{i=1}^{n} (1 - y_i \boldsymbol{\beta}^{\boldsymbol{T}} x_i - a_i^t + \xi_i^t + \mu_i^t)^2 \tag{10}$$

$$(a_i^{t+1}, \xi_i^{t+1}) = \underset{\xi_i, a_i \geq 0}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} a_i + \frac{\rho}{2} \sum_{i=1}^{n} (1 - y_i x_i{}^T \boldsymbol{\beta}^{\boldsymbol{t+1}} - a_i + \xi_i + \mu_i^t)^2 \tag{11}$$

$$\mu_i^{t+1} = \mu_i^t + 1 - y_i x_i^T \boldsymbol{\beta}^{t+1} - a_i^{t+1} + \xi_i^{t+1} \tag{12}$$

- 针对

基于支持向量机原始问题的优化 II

- 给定初值 $\boldsymbol{\beta}^0, a^0, \boldsymbol{\xi}^0, \mu^0$，假设当前的迭代为第 $t \geq 0$ 步，那么具体的优化迭代步骤如下：

$$\boldsymbol{\beta}^{t+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{\lambda}{2} \| \boldsymbol{\beta} \|_2^2 + \frac{\rho}{2} \sum_{i=1}^{n} (1 - y_i \boldsymbol{\beta}^{\boldsymbol{T}} x_i - a_i^t + \xi_i^t + \mu_i^t)^2$$

$$a_i^{t+1} = max\{1 - y_i x_i^T \boldsymbol{\beta}^{t+1} + \mu_i^t - 1/(n\rho), 0\}$$

$$\xi_i^{t+1} = max\{-(1 - y_i x_i^T \boldsymbol{\beta}^{\boldsymbol{t+1}}) - \mu_i^t, 0\}$$

$$\mu_i^{t+1} = \mu_i^t + 1 - y_i x_i^T \boldsymbol{\beta}^{t+1} - \alpha_i^{t+1} + \xi_i^{t+1}$$

重复迭代上述步骤，直到达到指定收敛误差。

基于支持向量机的人脸识别

Outline

- 我们使用基于高斯核的非线性支持向量机，对sklearn中的有关人脸的fetch_lfw_people数据进行分析
- 所考虑为62×47像素点的图片，因而采用主成分(PCA)方法进行降维；再用所得到的主成分与标签拟合支持向量机
- 采用pipeline方法

引入需要使用的Python库

```
###### 从sklearn数据集中引入人脸数据 ##########################
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns; sns.set()
from sklearn.datasets import fetch_lfw_people
from sklearn.svm import SVC
from sklearn.decomposition import PCA as RandomizedPCA
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
```

算法流程

- 从数据集中抽取最少有100张照片的对象组成训练数据集所考虑为62×47像素点的图片，因而采用主成分(PCA)方法进行降维；再用所得到的主成分与标签拟合支持向量机

```
##### 我们从数据集中抽取最少有100张的人脸数据 ##################
faces = fetch_lfw_people(min_faces_per_person=100)
print(faces.target_names)
print(faces.images.shape)
################################################################
```

- 经过筛选过的数据集如下：其包含5个人的不同照片，共有1140张，每张照片为62×47像素

```
['Colin Powell' 'Donald Rumsfeld' 'George W Bush' 'Gerhard Schroeder'
 'Tony Blair']
(1140, 62, 47)
```

算法流程

- 从筛选出的数据集抽取15张图片，并将他们绘制出来

```
############### 画出抽取的人脸图像 ##############################
fig, ax = plt.subplots(3, 5)

for i, axi in enumerate(ax.flat):
    axi.imshow(faces.images[i], cmap='bone')
    #使用axi.imshow画图
    #cmap: 表示颜色图谱，可选择bone， hot，autumn等
    axi.set(xticks=[], yticks=[],
            xlabel=faces.target_names[faces.target[i]])
    #设置所画子图的横纵坐标及标签
    #xticks=[] 表示关闭x轴的坐标及刻度
#################################################################
```

算法流程


George W Bush
Gerhard Schroeder
Donald Rumsfeld
Tony Blair
Donald Rumsfeld
Colin Powell
George W Bush
Colin Powell
George W Bush
Donald Rumsfeld
Gerhard Schroeder
Colin Powell
George W Bush
George W Bush
Tony Blair

算法流程

- 这里所考虑的图片为62×47像素点的图片，因为需要考虑的参数的个数为2914个。因此我们考虑采用PCA进行降维
- 使用PCA提取出前150个主成分，定义相关支持向量机模型

```
35
36  # 使用PCA提取出数据的前150个主成分
37  pca = RandomizedPCA(n_components=150, whiten=True, random_state=1)
38  # 定义C支持向量机
39  svc = SVC(kernel='rbf', class_weight='balanced')
40  #使用pipeline定义一个管道
41  model = make_pipeline(pca, svc)
```

- 我们事先划分百分之七十五的数据作为训练集，剩余数据作为测试集

```
##############################################################
################### 将数据分成训练与测试集合 ###################
Xtrain, Xtest, ytrain, ytest = train_test_split(faces.data, faces.target,
                                                test_size=0.25,random_state=1)
```

算法流程

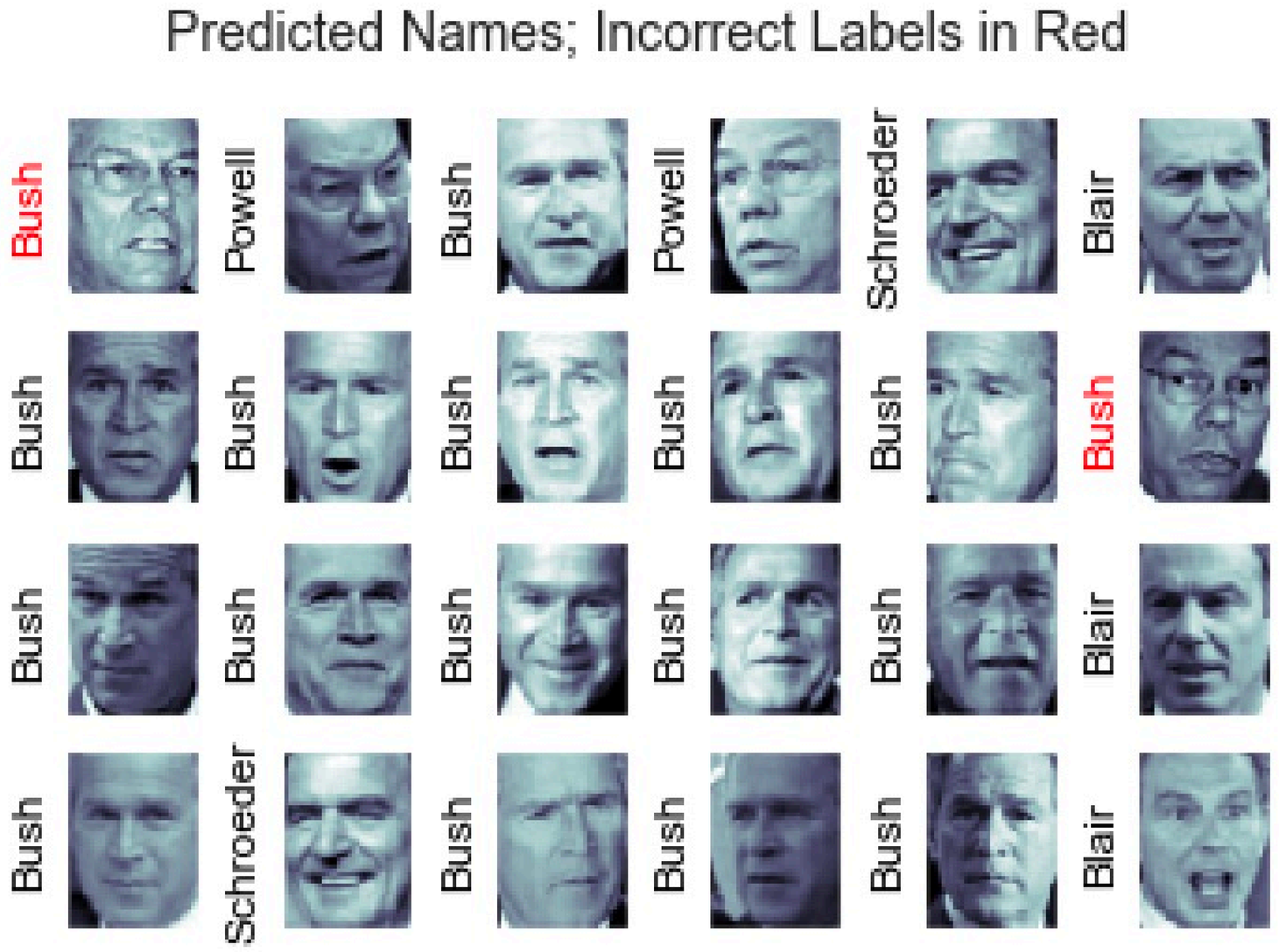
- 设置相关带调参数的取值，并使用交叉验证选取最优值

```
################## 设定需要调的参数 ############################
param_grid = {'svc__C': [1, 5, 10, 50],
              'svc__gamma': [0.0001, 0.0005, 0.001, 0.005]}
grid = GridSearchCV(model, param_grid)
################## 拟合测试集合，并用最优的模型拟合训练集合 ###############
grid.fit(Xtrain, ytrain)
print(grid.best_params_)
model = grid.best_estimator_
```

- 拟合模型，将得到的模型作用在测试集上并画出测试集上的部分结果

```
################## 在测试结合中进行预测 ############################
yfit = model.predict(Xtest)
################## 作出测试集合的预测图 ######################
fig, ax = plt.subplots(4, 6)
for i, axi in enumerate(ax.flat):
    axi.imshow(Xtest[i].reshape(62, 47), cmap='bone')
    axi.set(xticks=[], yticks=[])
    axi.set_ylabel(faces.target_names[yfit[i]].split()[-1],
                   color='black' if yfit[i] == ytest[i] else 'red')
fig.suptitle('Predicted Names; Incorrect Labels in Red', size=14);
```

预测结果

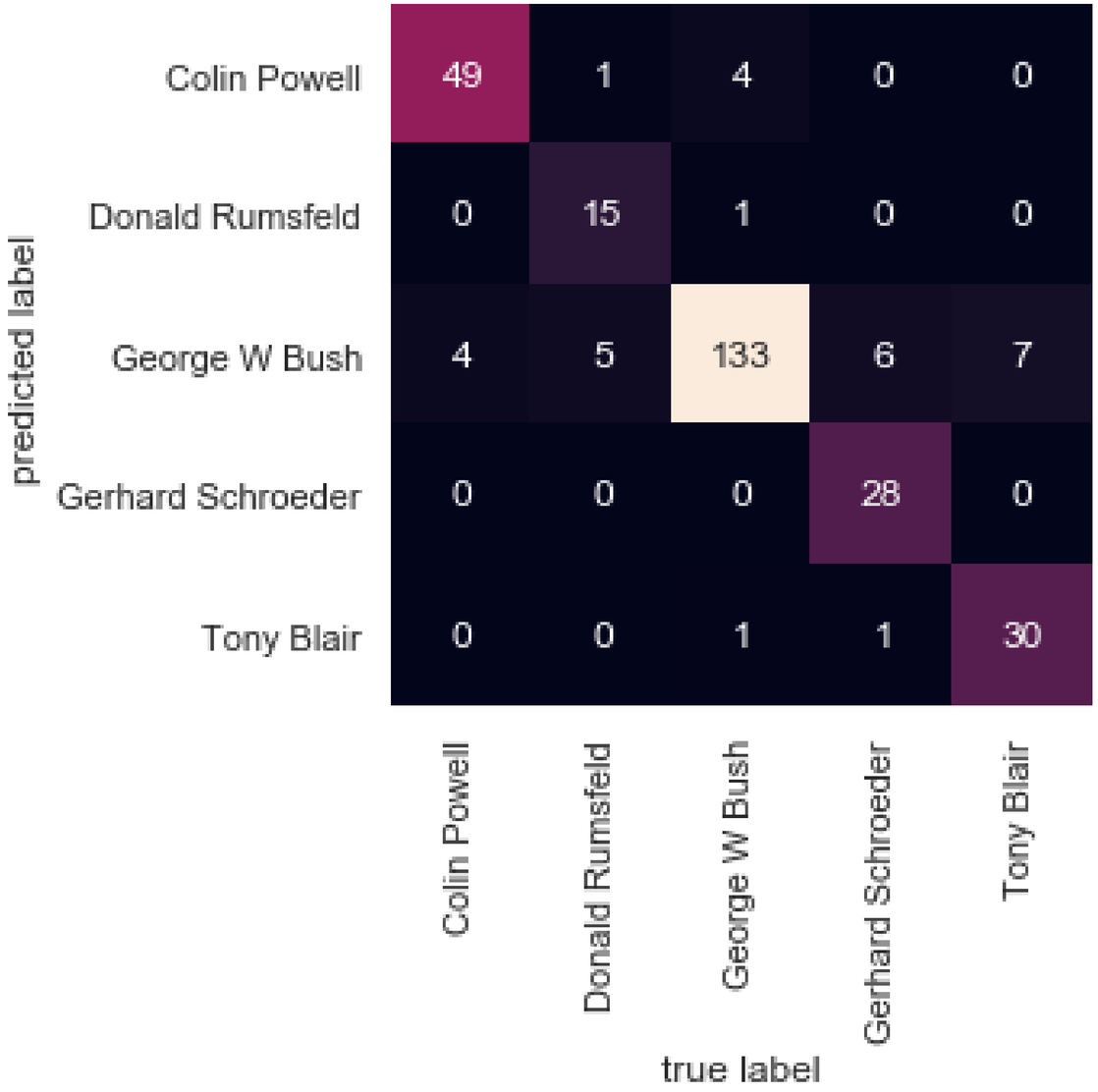

Predicted Names; Incorrect Labels in Red
Bush
Powell
Bush
Powell
Schroeder
Blair
Bush
Bush
Bush
Bush
Bush
Bush
Bush
Bush
Bush
Bush
Bush
Blair
Bush
Schroeder
Bush
Bush
Bush
Bush
Blair

分析相关结果

```
################### 打印出分类的评价标准 ##############################
print(classification_report(ytest, yfit,
                            target_names=faces.target_names))

####################### 作出混淆矩阵 ###################################
mat = confusion_matrix(ytest, yfit)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=faces.target_names,
            yticklabels=faces.target_names)
plt.xlabel('true label')
plt.ylabel('predicted label');
```

```
{'svc__C': 5, 'svc__gamma': 0.005}
                   precision    recall  f1-score   support

     Colin Powell       0.91      0.92      0.92        53
  Donald Rumsfeld       0.94      0.71      0.81        21
    George W Bush       0.86      0.96      0.90       139
Gerhard Schroeder       1.00      0.80      0.89        35
      Tony Blair        0.94      0.81      0.87        37

         accuracy                           0.89       285
        macro avg       0.93      0.84      0.88       285
     weighted avg       0.90      0.89      0.89       285
```

分析相关结果


predicted label
Colin Powell
Donald Rumsfeld
George W Bush
Gerhard Schroeder
Tony Blair
49 1 4 0 0
0 15 1 0 0
4 5 133 6 7
0 0 0 28 0
0 0 1 1 30
Colin Powell
Donald Rumsfeld
George W Bush
Gerhard Schroeder
Tony Blair
true label