

人工智能之机器学习

SVM

主讲人：刘老师(GerryLiu)

课程要求

- 课上课下 “九字” 真言
 - 认真听, **善摘录, 勤思考**
 - **多温故, 乐实践**, 再发散
- 四不原则
 - **不懒散惰性, 不迟到早退**
 - **不请假旷课, 不拖延作业**
- 一点注意事项
 - 违反 “四不原则” , 不推荐就业

课程内容

- 梯度下降法、拉格朗日乘子法、KKT条件回顾
- 感知器模型回顾
- SVM线性可分
- SVM线性不可分
- 核函数
- SMO

梯度下降法

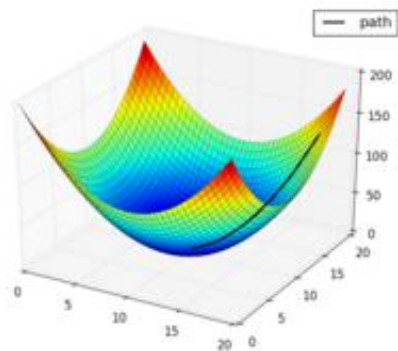
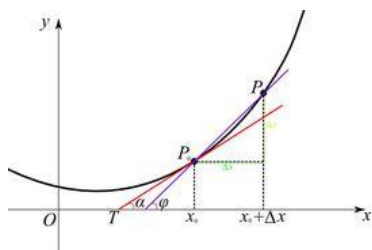
- 梯度下降法(Gradient Descent, GD)常用于求解**无约束**情况下**凸函数(Convex Function)**的**极小值**，是一种**迭代类型**的算法，因为凸函数只有一个极值点，故求解出来的极小值点就是函数的**最小值点**。

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

$$\theta^* = \arg \min_{\theta} J(\theta)$$

梯度下降法

- 导数：一个函数在某一点的导数描述了这个函数在这一点附近的**变化率**，也可以认为是函数在某一点的导数就是该函数所代表的曲线在这一点**的切线斜率**。导数值越大，表示函数在该点处的变化越大。
- 梯度：梯度是一个向量，表示某一函数在该点处的**方向导数**沿着该方向取的最大值，即函数在该点处沿着该方向变化最快，变化率最大(即该梯度向量的模)；当函数为一维函数的时候，梯度其实就是导数。



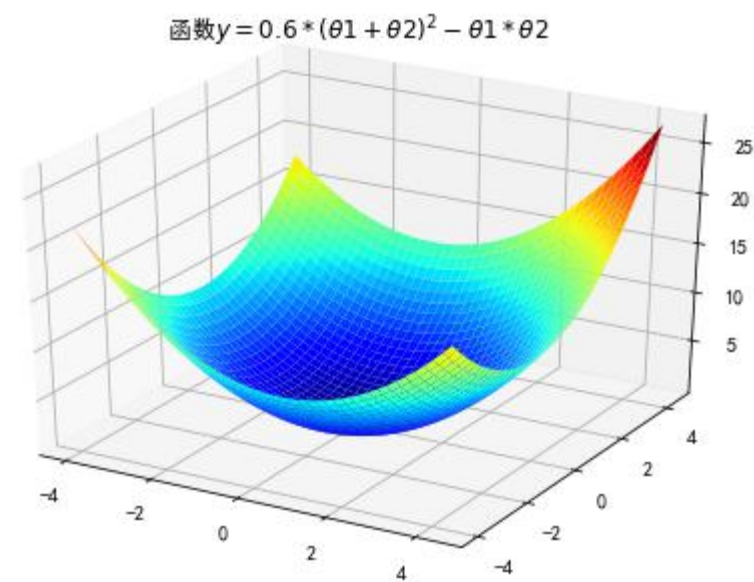
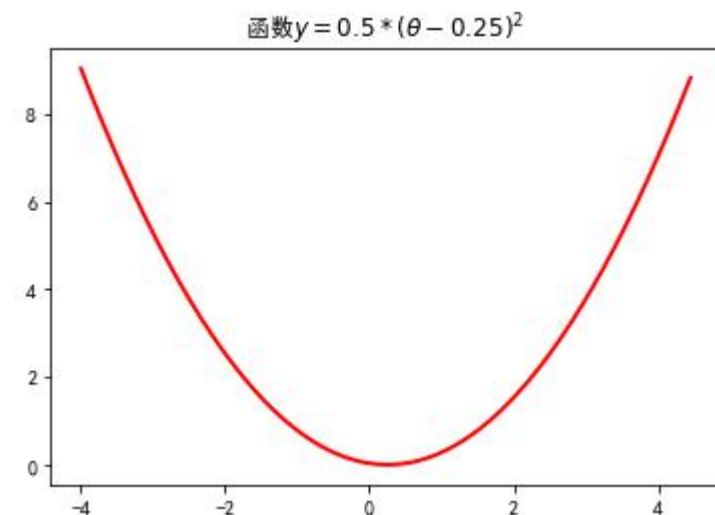
$$y' = \frac{\partial f(x)}{\partial x}$$
$$\nabla f(x_1, x_2) = \left(\frac{\partial f(x_1, x_2)}{\partial x_1}, \frac{\partial f(x_1, x_2)}{\partial x_2} \right)$$

梯度下降法

- 梯度下降法的优化思想是用当前位置负梯度方向作为搜索方向，因为该方向为当前位置的最快下降方向，所以梯度下降法也被称为“最速下降法”。梯度下降法中越接近目标值，变量变化越小。计算公式如下：

$$\theta^{k+1} = \theta^k - \alpha \nabla f(\theta^k)$$

- α 被称为**步长**或者**学习率(learning rate)**，表示自变量 θ 每次迭代变化的大小。
- 收敛条件：当目标函数的函数值变化非常小的时候或者达到最大迭代次数的时候，就结束循环。



梯度下降法

- 由于梯度下降法中负梯度方向作为变量的变化方向，所以有可能导致最终求解的值是局部最优解，所以在使用梯度下降的时候，一般需要进行一些调优策略：
 - 学习率的选择：学习率过大，表示每次迭代更新的时候变化比较大，有可能会跳过最优解；学习率过小，表示每次迭代更新的时候变化比较小，就会导致迭代速度过慢，很长时间都不能结束；
 - 算法初始参数值的选择：初始值不同，最终获得的最小值也有可能不同，因为梯度下降法求解的是局部最优解，所以一般情况下，选择多次不同初始值运行算法，并最终返回损失函数最小情况下的结果值；
 - 标准化：由于样本不同特征的取值范围不同，可能会导致在各个不同参数上迭代速度不同，为了减少特征取值的影响，可以将特征进行标准化操作。

梯度下降法

- 批量梯度下降法(Batch Gradient Descent, BGD): 使用所有样本在当前点的梯度值来对变量参数进行更新操作。

$$\theta^{k+1} = \theta^k - \alpha \sum_{i=1}^m \nabla f_{\theta^k}(x^i)$$

- 随机梯度下降法(Stochastic Gradient Descent, SGD): 在更新变量参数的时候, 选取一个样本的梯度值来更新参数。

$$\theta^{k+1} = \theta^k - \alpha \nabla f_{\theta^k}(x^i)$$

- 小批量梯度下降法(Mini-batch Gradient Descent, MBGD): 集合BGD和SGD的特性, 从原始数据中, 每次选择n个样本来更新参数值, 一般n选择10.

$$\theta^{k+1} = \theta^k - \alpha \sum_{i=t}^{t+n-1} \nabla f_{\theta^k}(x^i)$$

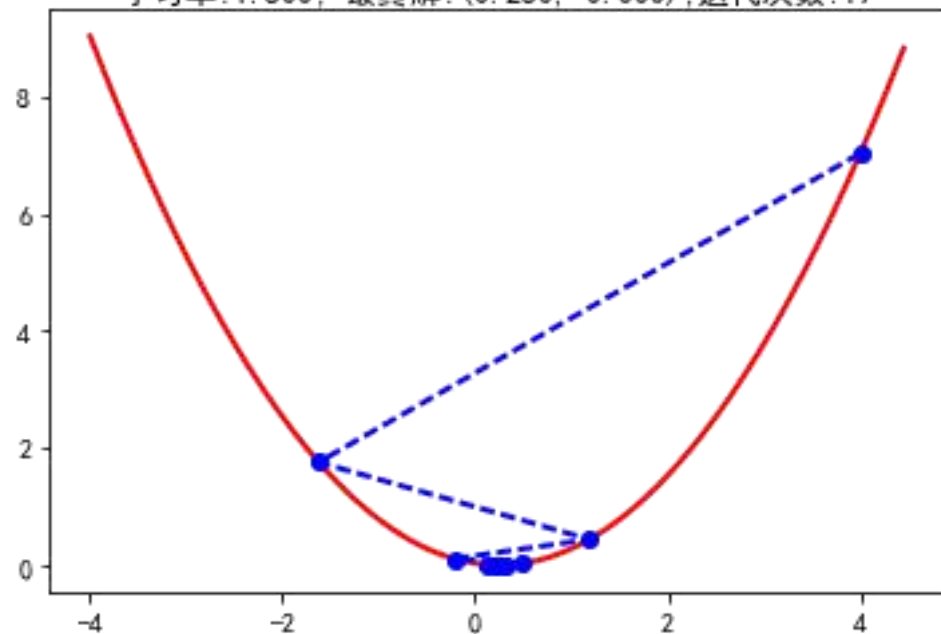
梯度下降法

- BGD、SGD、MBGD的区别：
 - 当样本量为 m 的时候，每次迭代BGD算法中对于参数值更新一次，SGD算法中对于参数值更新 m 次，MBGD算法中对于参数值更新 m/n 次，相对来讲SGD算法的更新速度最快；
 - SGD算法中对于每个样本都需要更新参数值，当样本值不太正常的时候，就有可能导致本次的参数更新会产生相反的影响，也就是说SGD算法的结果并不是完全收敛的，而是在收敛结果处波动的；
 - SGD算法是每个样本都更新一次参数值，所以SGD算法特别适合样本数据量大的情况以及在线机器学习(Online ML)。

梯度下降法案例代码

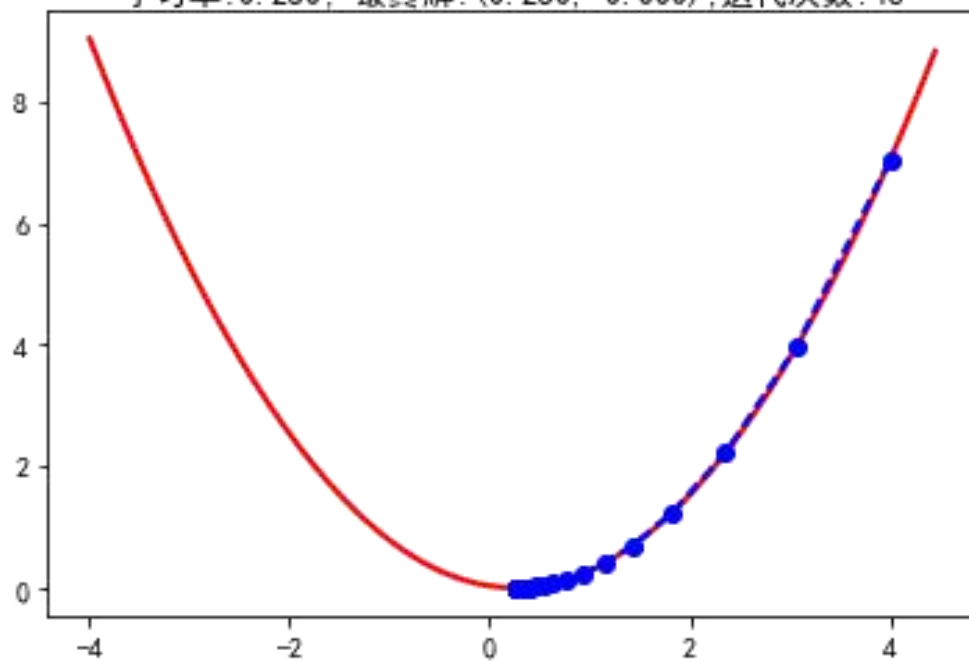
函数 $y = 0.5 * (\theta - 0.25)^2$;

学习率: 1.500; 最终解: (0.250, 0.000); 迭代次数: 19



函数 $y = 0.5 * (\theta - 0.25)^2$;

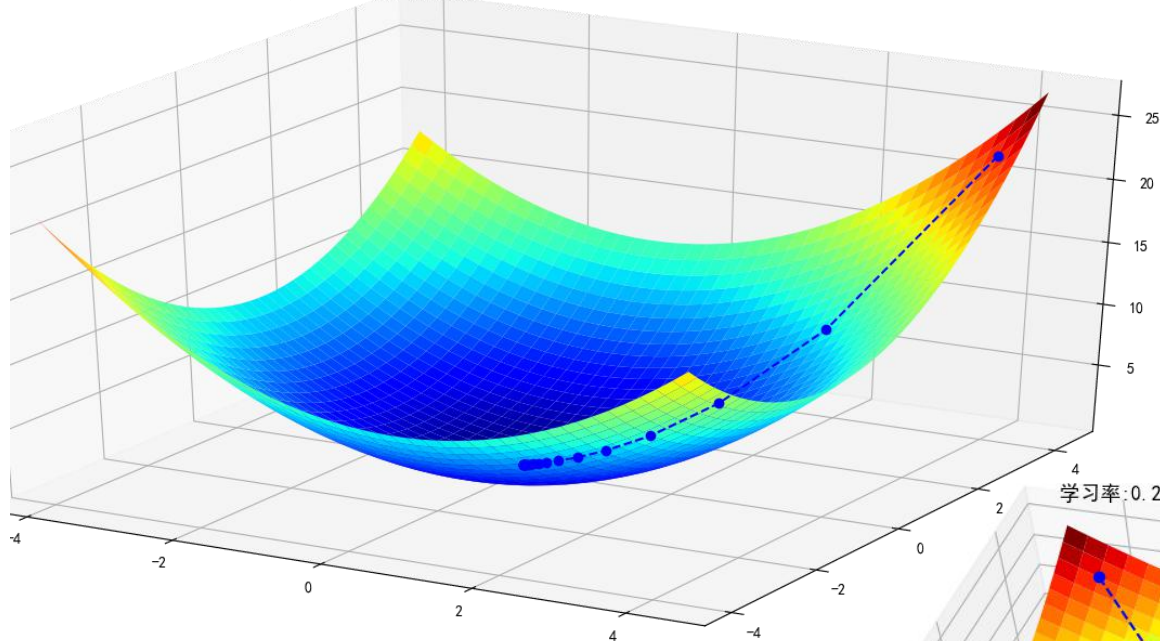
学习率: 0.250; 最终解: (0.250, 0.000); 迭代次数: 43



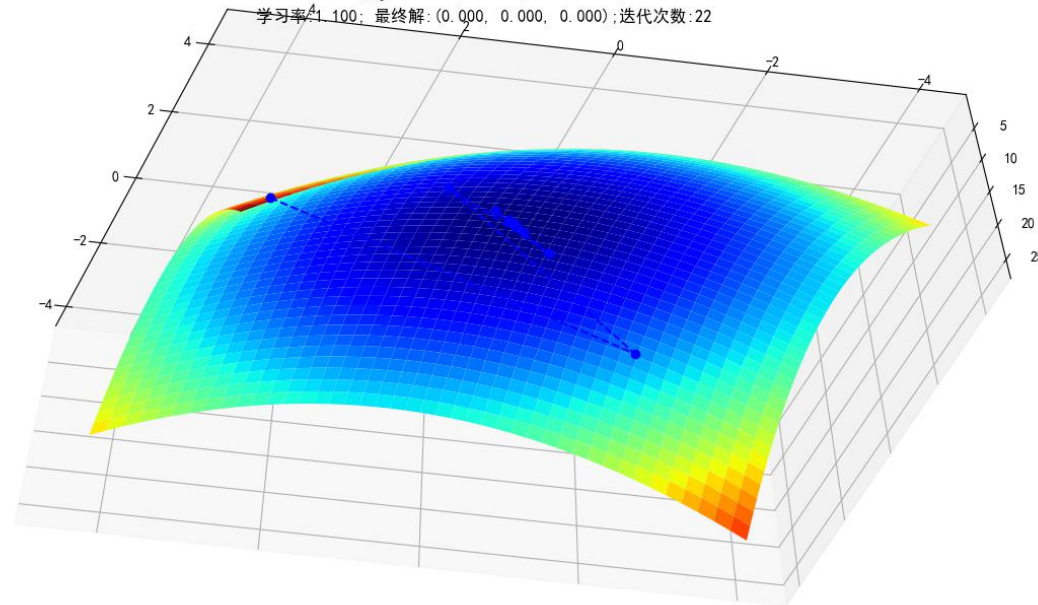
梯度下降法案例代码

函数 $y = 0.6 * (\theta_1 + \theta_2)^2 - \theta_1 * \theta_2$;

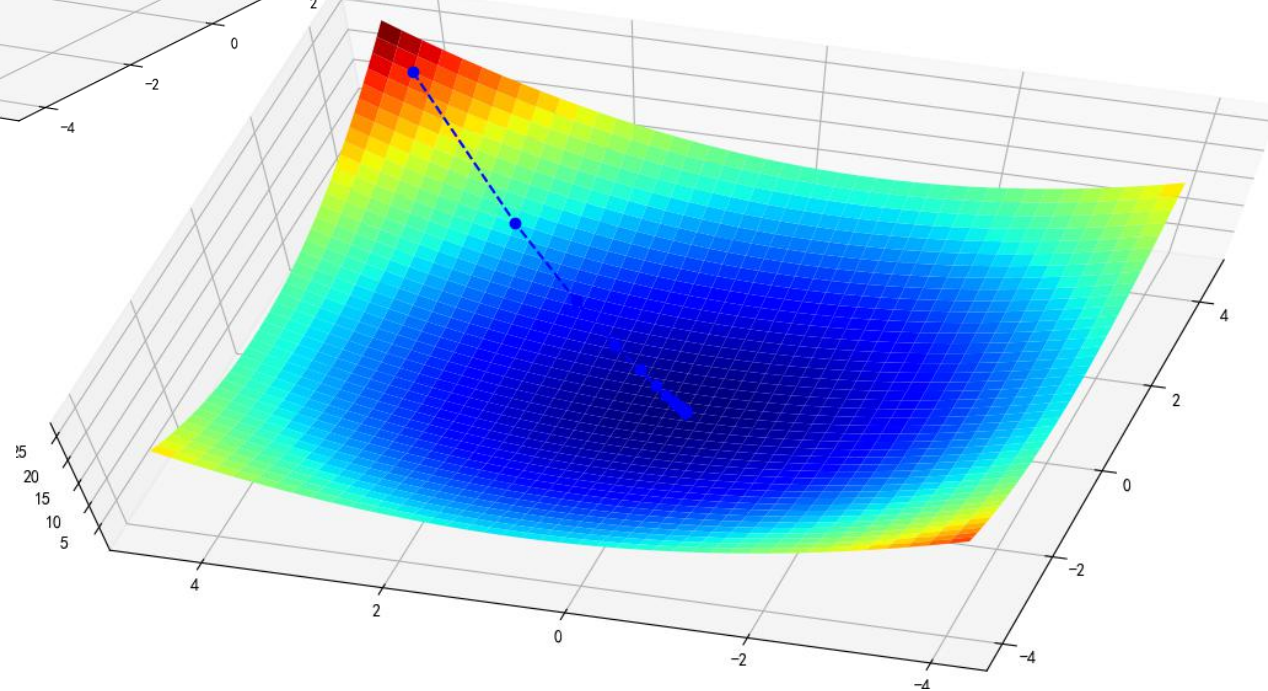
学习率: 0.250; 最终解: (0.000, 0.000, 0.000); 迭代次数: 31



函数 $y = 0.6 * (\theta_1 + \theta_2)^2 - \theta_1 * \theta_2$;
学习率: 1.100; 最终解: (0.000, 0.000, 0.000); 迭代次数: 22



函数 $y = 0.6 * (\theta_1 + \theta_2)^2 - \theta_1 * \theta_2$;
学习率: 0.250; 最终解: (0.000, 0.000, 0.000); 迭代次数: 31



有约束的最优化问题

- 最优化问题一般是指对于某一个函数而言，求解在其指定作用域上的全局最小值问题，一般分为以下三种情况(备注：以下几种方式求出来的解都有可能是局部极小值，只有当函数是凸函数的时候，才可以得到全局最小值)：
 - 无约束问题：求解方式一般求解方式梯度下降法、牛顿法、坐标轴下降法等；
 - 等式约束条件：求解方式一般为拉格朗日乘子法
 - 不等式约束条件：求解方式一般为KKT条件

$$\min_x f(x)$$

$$\min_x f(x)$$

$$\text{subject to : } h_k(x) = 0, k = 1, 2, \dots, p$$

$$\min_x f(x)$$

$$\text{s.t : } h_k(x) = 0, k = 1, 2, \dots, p$$

$$g_j(x) \leq 0, j = 1, 2, \dots, q$$

拉格朗日乘子法

- 拉格朗日乘子法就是当我们的优化函数存在等值约束的情况下的
一种最优化求解方式；其中参数 α 被称为**拉格朗日乘子**，要求 α 不

等于0

$$\min_x f(x)$$

$$s.t : h_i(x) = 0, i = 1, 2, \dots, p$$



$$\min_x f(x) + \sum_{i=1}^p \alpha_i h_i(x); \alpha_i \neq 0$$

拉格朗日乘子法理解

- 假设现在有一个二维的优化问题

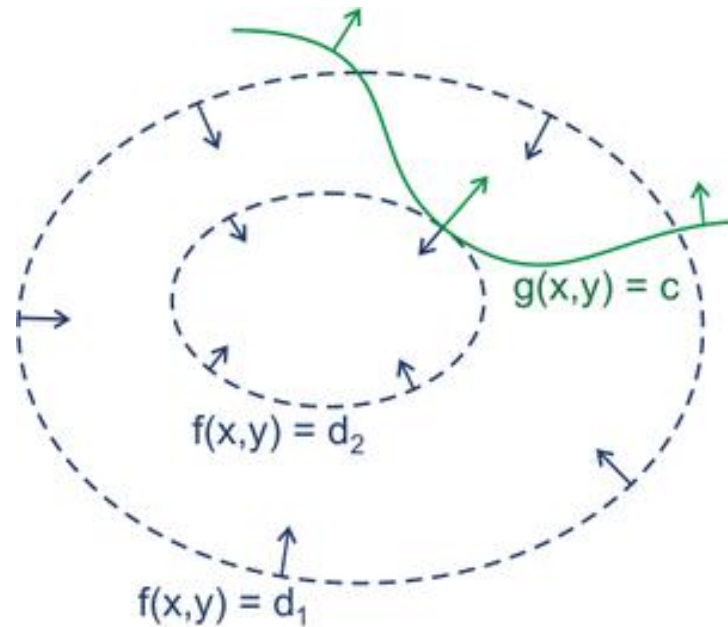
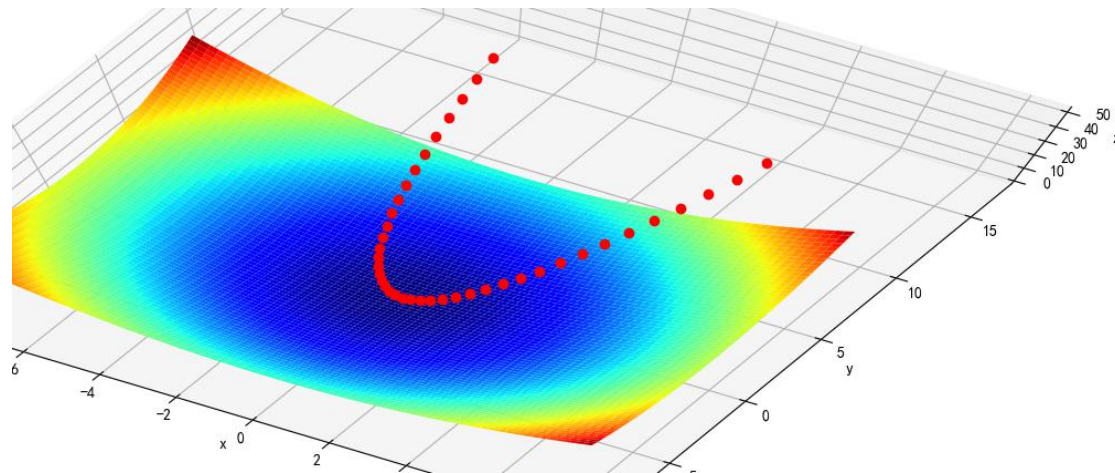
$$\min_{x,y} f(x,y)$$

$$s.t : g(x,y) = c$$

$$L(x,y,\alpha) = f(x,y) + \alpha(g(x,y) - c); \alpha \neq 0$$

$$\begin{cases} \min_{x,y} f(x,y) \\ g(x,y) - c = 0 \end{cases} \Rightarrow \min_{x,y} L(x,y,\alpha)$$

$$\nabla_{x,y,\alpha} L(x,y,\alpha) = 0, \alpha \neq 0$$



对偶问题

- 在优化问题中，目标函数 $f(x)$ 存在多种形式，如果目标函数和约束条件都为变量 x 的线性函数，则称问题为**线性规划**；如果目标函数为二次函数，则称最优化问题为**二次规划**；如果目标函数或者约束条件为非线性函数，则称最优化问题为**非线性优化**。每个线性规划问题都有一个对应的**对偶问题**。对偶问题具有以下几个特性：
 - 1. 对偶问题的对偶是原问题；
 - 2. 无论原始问题是否是凸的，对偶问题都是凸优化问题；
 - 3. 对偶问题可以给出原始问题的一个下界；
 - 4. 当满足一定条件的时候，原始问题和对偶问题的解是完美等价的。

KKT条件

- KKT条件是泛拉格朗日乘子法的一种形式；主要应用在当我们的优化函数存在不等值约束的情况下的一种最优化求解方式；KKT条件即满足不等式约束条件下的条件。

$$\min_x f(x)$$

$$s.t : h_k(x) = 0, k = 1, 2, \dots, p$$

$$g_j(x) \leq 0, j = 1, 2, \dots, q$$

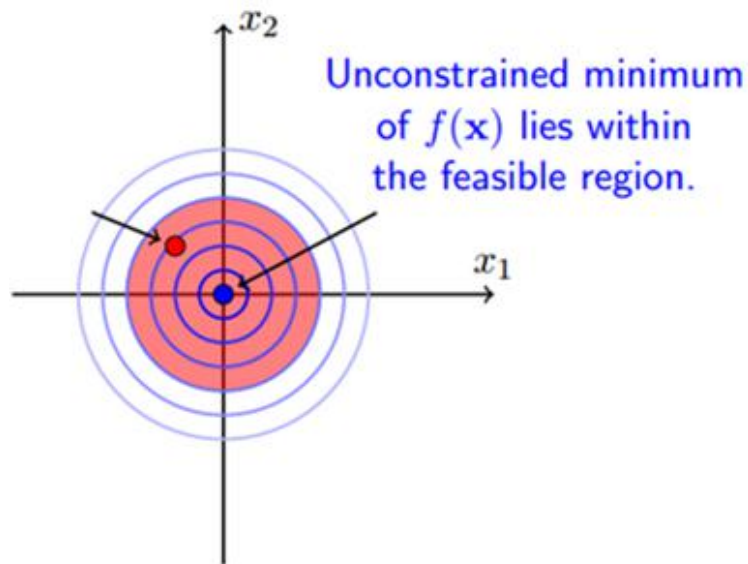
↓

$$L(x, \alpha, \beta) = f(x) + \sum_{i=1}^p \alpha_i h_i(x) + \sum_{i=1}^q \beta_i g_i(x); \alpha_i \neq 0, \beta_i \geq 0$$

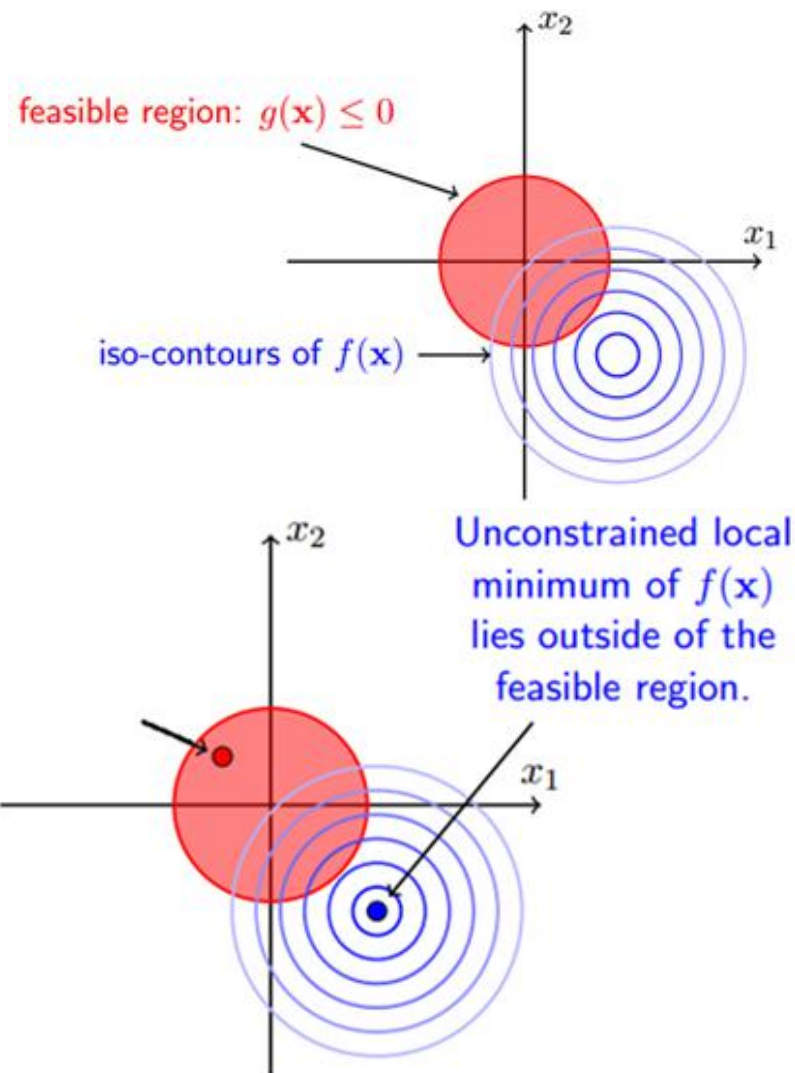
$$\min_x L(x, \alpha, \beta)$$

KKT条件理解

- 可行解必须在约束区域 $g(x)$ 之内，由图可知可行解 x 只能在 $g(x) < 0$ 和 $g(x) = 0$ 的区域取得；
 - 当可行解 x (没有约束情况下的函数 f 的最优解)在 $g(x) < 0$ 的区域中的时候，此时直接极小化 $f(x)$ 即可得到；
 - 当可行解 x 在 $g(x) = 0$ 的区域中的时候，此时直接等价于等式约束问题的求解。



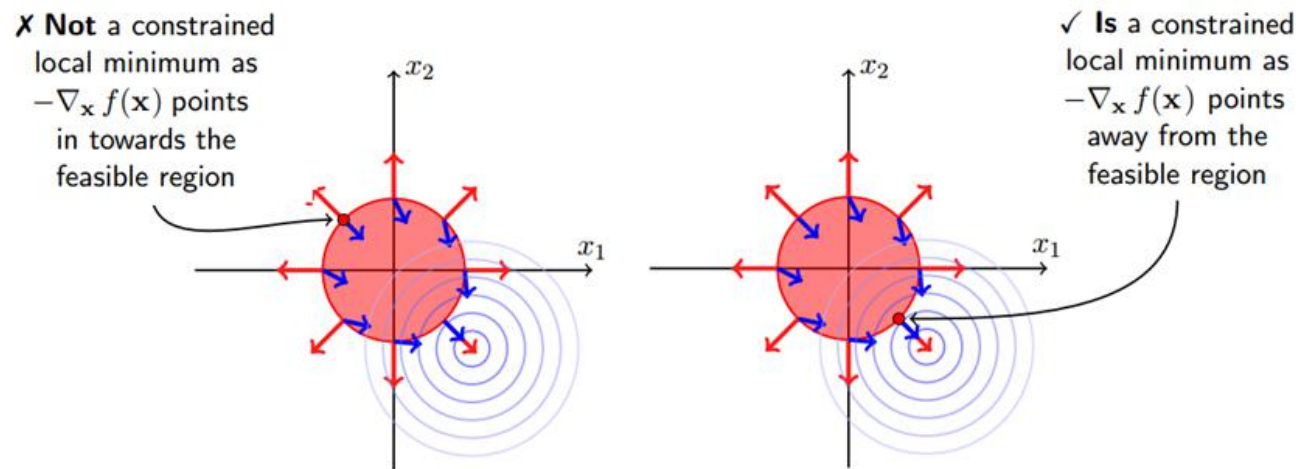
$$\min_x f(x)$$
$$s.t : g(x) \leq 0$$



KKT条件理解

- 当可行解在约束内部区域的时候，令 $\beta=0$ 即可消去约束。
- 对于参数 β 的取值而言，在等值约束中，约束函数和目标函数的梯度只要满足平行即可，而在不等式约束中，若 $\beta \neq 0$ ，则说明可行解在约束区域的边界上，这个时候可行解应该尽可能的靠近无约束情况下的解，所以在约束边界上，目标函数的负梯度方向应该远离约束区域朝无约束区域时的解，此时约束函数的梯度方向与目标函数的负梯度方向应相同；从而可以得出 $\beta > 0$ 。

$$\nabla_x f(x) = \beta * \nabla_x g(x)$$



KKT理解

$$\min_x f(x)$$

$$s.t : g_j(x) \leq 0, j = 1, 2, \dots, q$$

$$L(x, \beta) = f(x) + \sum_{i=1}^q \beta_i g_i(x); \beta_i \geq 0$$

$$\therefore \left. \begin{array}{l} \beta_i \geq 0 \\ g_i(x) \leq 0 \end{array} \right\} \Rightarrow \beta_i g_i(x) \leq 0$$

$$\therefore f(x) = \max_{\beta} L(x, \beta)$$

$$\therefore \min_x f(x) = \min_x \max_{\beta} L(x, \beta)$$

KKT理解

$$\begin{aligned}\max_{\beta} \min_x L(x, \beta) &= \max_{\beta} \left[\min_x f(x) + \min_x \beta g(x) \right] \\ &= \max_{\beta} \min_x f(x) + \max_{\beta} \min_x \beta g(x) = \min_x f(x) + \max_{\beta} \min_x \beta g(x)\end{aligned}$$

$$\left. \begin{array}{l} \beta_i \geq 0 \\ g_i(x) \leq 0 \end{array} \right\} \Rightarrow \min_x \beta_i g_i(x) = \begin{cases} 0, & \beta \text{为0或者} g(x) \text{为0} \\ -\infty, & \beta > 0 \text{且} g(x) < 0 \end{cases}$$

$$\therefore \max_{\beta} \min_x \beta_i g_i(x) = 0$$

$$\therefore \min_x f(x) = \max_{\beta} \min_x L(x, \beta); \quad \beta = 0 \text{或者} g(x) = 0$$

KKT理解

$$\min_x f(x) = \min_x \max_{\beta} L(x, \beta) \quad \min_x f(x) = \max_{\beta} \min_x L(x, \beta)$$

$$\min_x \max_{\beta} L(x, \beta) = \max_{\beta} \min_x L(x, \beta)$$

原问题

对偶问题

- 对偶问题的直观理解：最小的里面的那个最大的要比最大的那个里面的最小的大；从而就可以为原问题引入一个下界

KKT条件总结

- 1. 拉格朗日取得可行解的充要条件;
- 2. 将不等式约束转换后的一个约束, 称为松弛互补条件;
- 3. 初始的约束条件; $1).\nabla_x L(x, \alpha, \beta) = 0$
- 4. 初始的约束条件; $2).\beta_i g_i(x) = 0, i = 1, 2, \dots, q$
- 5. 不等式约束需要满足的条件。 $3).h_i(x) = 0, i = 1, 2, \dots, p$
 $4).g_i(x) \leq 0, i = 1, 2, \dots, q$
 $5).\beta_i \geq 0, i = 1, 2, \dots, q$

高中距离知识回顾

- 点到直线/平面的距离公式:

- 假定点 $p(x_0, y_0)$, 平面方程为 $f(x, y) = Ax + By + C$, 那么点 p 到平面 $f(x)$ 的距离为:

$$dist = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

- 从三维空间扩展到多维空间中, 如果存在一个超平面 $f(X) = \theta X + b$; 那么某一个点 X_0 到这个超平面的距离为:

The diagram shows the formula $dist = \frac{|f(X_0)|}{\|\theta\|_2}$ enclosed in a rounded rectangle. The numerator $|f(X_0)|$ is highlighted with a yellow box. An orange arrow points from a yellow box labeled "函数距离" (Function Distance) to the yellow box. Another orange arrow points from an orange box labeled "几何距离" (Geometric Distance) to the denominator $\|\theta\|_2$.

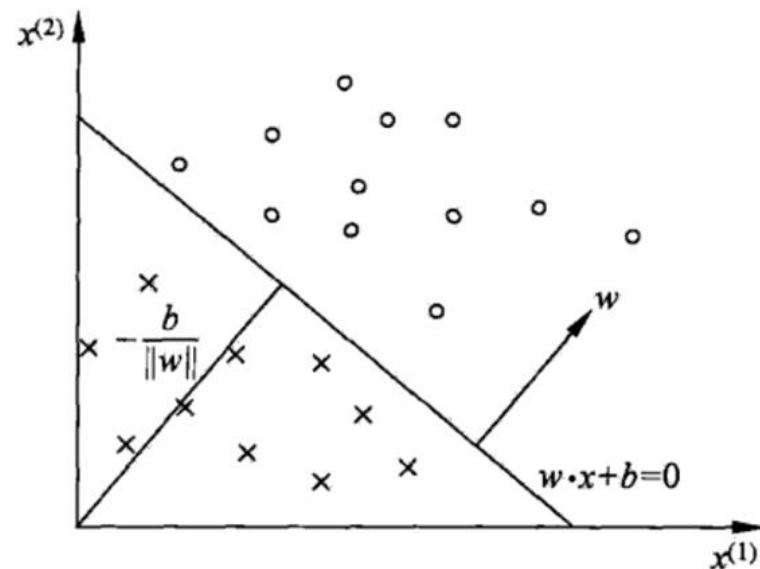
$$dist = \frac{|f(X_0)|}{\|\theta\|_2}$$

- 参考链接:

<https://wenku.baidu.com/view/d26d2ba39e31433239689374.html>

感知器模型

- 感知器算法是最古老的分类算法之一，原理比较简单，不过模型的分类泛化能力比较弱，不过感知器模型是SVM、神经网络、深度学习等算法的基础。
- 感知器的思想很简单：比如北风上有很多的学员，分为男学员和女学员，感知器模型就是试图找到一条直线，能够把所有的男学员和女学员分隔开，如果是高维空间中，感知器模型寻找的就是一个超平面，能够把所有的二元类别分割开。感知器模型的前提是：**数据是线性可分的。**



感知器模型

- 对于m个样本，每个样本n维特征以及一个二元类别输出y，如下：

$$\left\{ \left(x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)} \right), \left(x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)} \right), \dots, \left(x_1^{(m)}, x_2^{(m)}, \dots, x_n^{(m)} \right) \right\}$$

- 目标是找到一个超平面，即：

$$\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n = 0 \Rightarrow \theta \bullet x = 0$$

- 让一个类别的样本满足： $\theta \bullet x > 0$ ； 另外一个类别的满足： $\theta \bullet x < 0$

- 感知器模型为： $\hat{y} = \text{sign}(\theta \bullet x) = \begin{cases} +1, & \theta \bullet x > 0 \\ -1, & \theta \bullet x < 0 \end{cases}$

感知器模型

- 正确分类（预测和实际类别一致）： $y\theta x > 0$ ，错误分类（预测和实际类别不一致）： $y\theta x < 0$ ；所以我们可以定义我们的损害函数为：期望使分类错误的所有样本(k条样本)到超平面的距离之和最小。

$$L = \sum_{i=1}^k \frac{-y^{(i)} \theta \bullet x^{(i)}}{\|\theta\|_2}$$

- 因为此时分子和分母中都包含了 θ 值，当分子扩大N倍的时候，分母也会随之扩大，也就是说分子和分母之间存在倍数关系，所以可以固定分子或者分母为1，然后求另一个即分子或者分母的倒数的最小化作为损失函数，简化后的损失函数为（分母为1）：

$$L = -\sum_{i=1}^k y^{(i)} \theta \bullet x^{(i)}$$

感知器模型

$$L = -\sum_{i=1}^k y^{(i)} \theta \bullet x^{(i)}$$

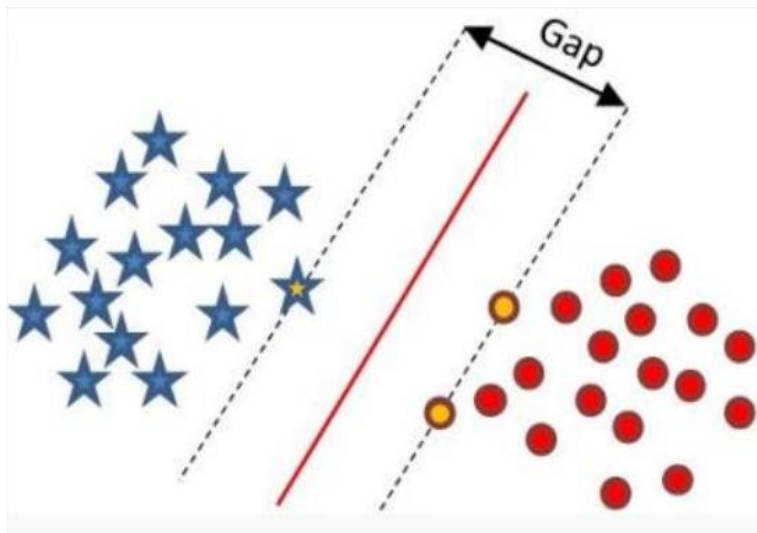
- 直接使用梯度下降法就可以对损失函数求解，不过由于这里的k是分类错误的样本点集合，不是固定的，所以我们不能使用批量梯度下降法(BGD)求解，只能使用随机梯度下降(SGD)或者小批量梯度下降(MBGD)；一般在感知器模型中使用SGD来求解。

$$\frac{\partial L(\theta)}{\partial \theta} = -\sum_{i=1}^k y^{(i)} x^{(i)}$$

$$\theta^{k+1} = \theta^k + \alpha y^{(i)} x^{(i)}$$

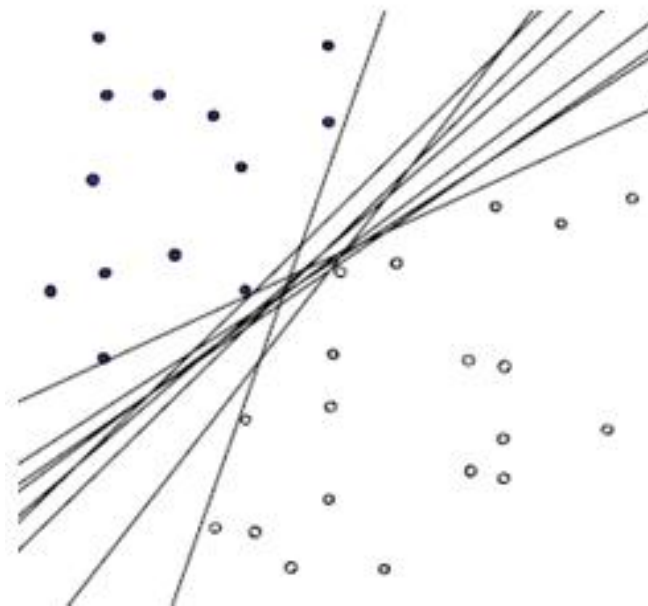
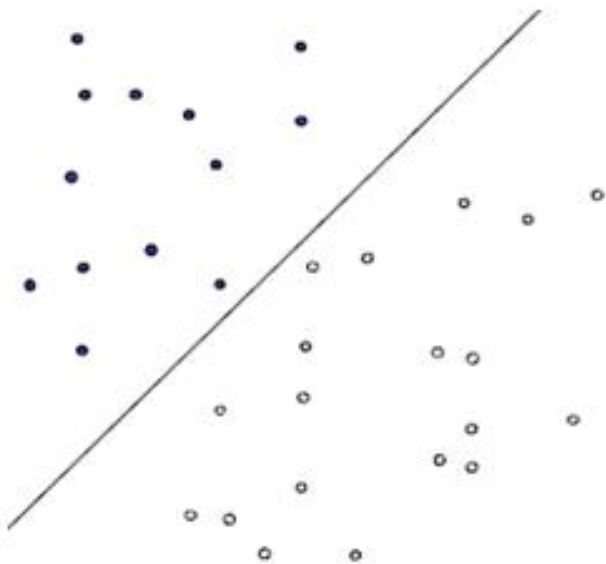
SVM

- 支持向量机(Support Vector Machine, SVM)本身是一个**二元分类算法**，是对感知器算法模型的一种扩展，现在的SVM算法支持**线性分类**和**非线性分类**的分类应用，并且也能够直接将SVM应用于**回归应用**中，同时通过OvR或者OvO的方式我们也将SVM应用在**多元分类**领域中。在不考虑集成学习算法，不考虑特定的数据集的时候，在分类算法中SVM可以说是特别优秀的。



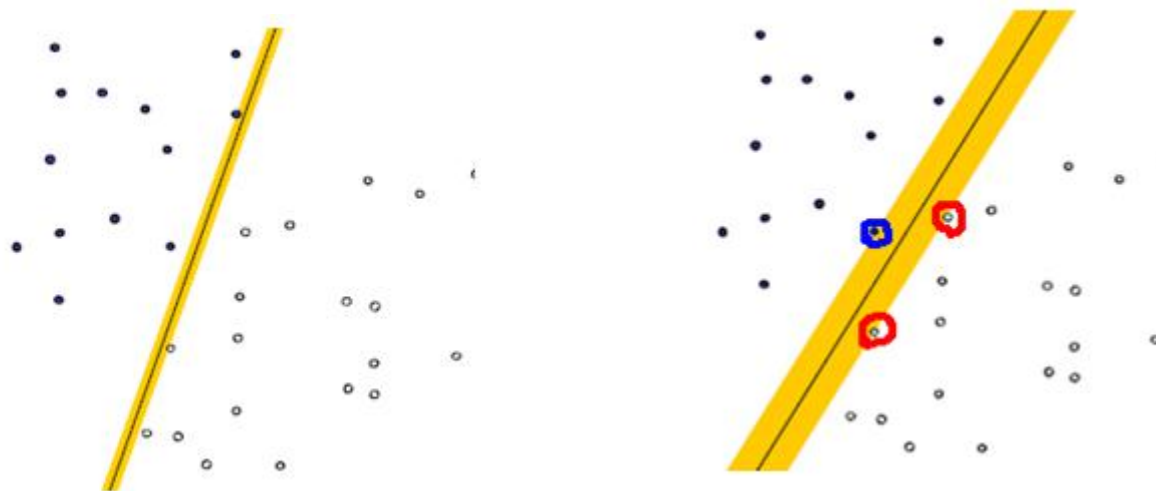
线性可分SVM

- 在感知器模型中，算法是在数据中找出一个划分超平面，让尽可能多的数据分布在这个平面的两侧，从而达到分类的效果，但是在实际数据中这个符合我们要求的超平面是可能存在多个的。



线性可分SVM

- 在感知器模型中，我们可以找到多个可以分类的超平面将数据分开，并且优化时希望所有的点(预测正确的点)都离超平面尽可能的远，但是实际上离超平面足够远的点基本上都是被正确分类的，所以这个是没有意义的；反而比较关心那些离超平面很近的点，这些点比较容易分错。所以说我们只要**让离超平面比较近的点尽可能的远离这个超平面(找一个最大间隔的超平面)**，那么我们的模型分类效果应该就会比较不错喽。SVM其实就是这个思想。



线性可分SVM

- **线性可分(Linearly Separable)**: 在数据集中, 如果可以找出一个超平面, 将两组数据分开, 那么这个数据集叫做线性可分数据。
- **线性不可分(Linear Inseparable)**: 在数据集中, 没法找出一个超平面, 能够将两组数据分开, 那么这个数据集就叫做线性不可分数据。
- **分割超平面(Separating Hyperplane)**: 将数据集分割开来的直线/平面叫做分割超平面。
- **支持向量(Support Vector)**: 离分割超平面最近的那些点叫做支持向量。
- **间隔(Margin)**: 支持向量数据点到分割超平面的距离称为间隔。

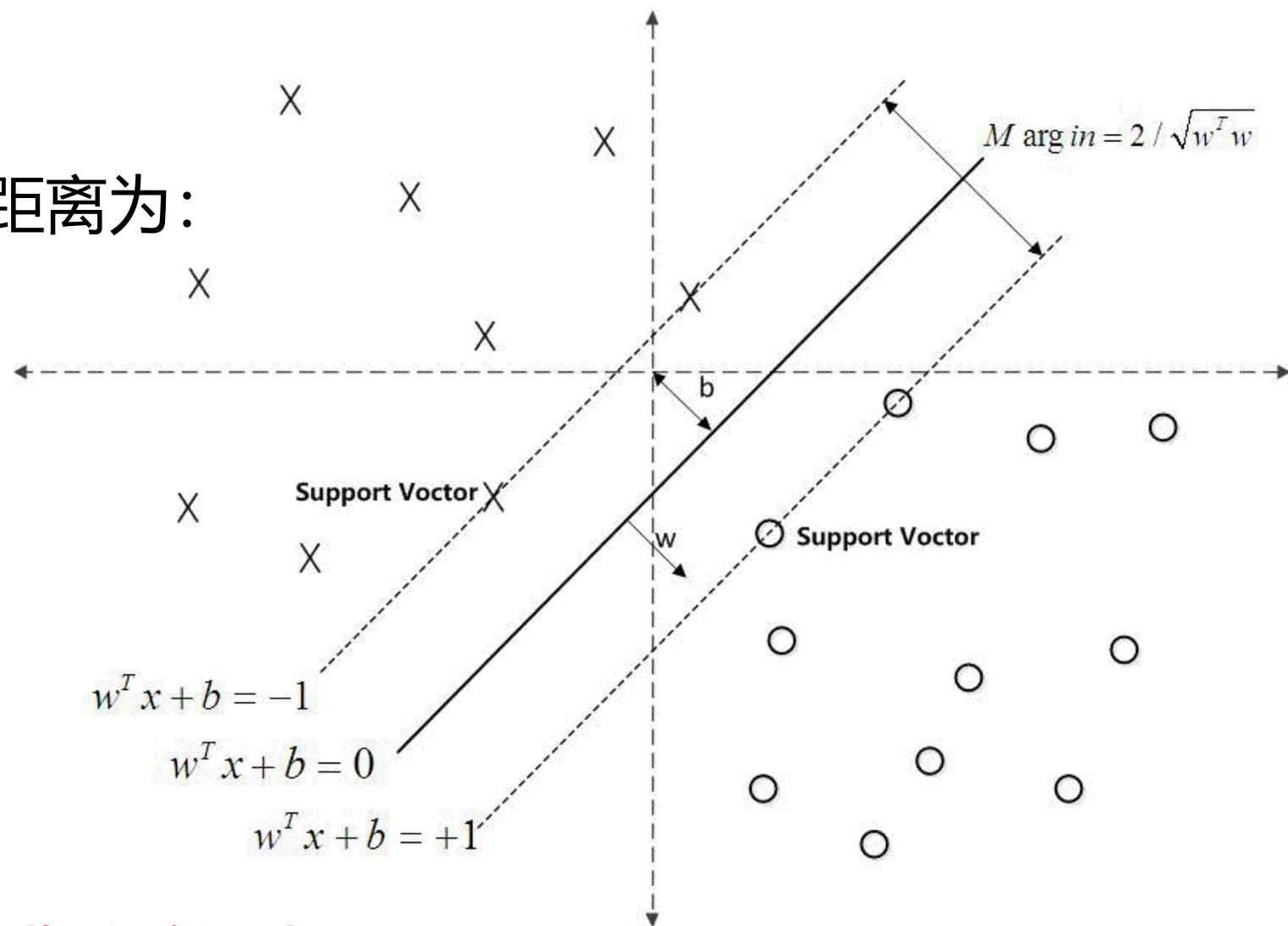
线性可分SVM

- 支持向量到超平面的距离为：

$$\because w^T x + b = \pm 1$$

$$\because y \in \{+1, -1\}$$

$$\therefore \frac{|(w^T x + b)|}{\|w\|_2} = \frac{1}{\|w\|_2}$$



- 备注：在SVM中支持向量到超平面的函数距离一般设置为1

线性可分SVM

- SVM模型是让所有的分类点在各自类别的支持向量远离超平面的一侧，同时要求支持向量尽可能的远离这个超平面(**最大间隔**)，用数学公式表示如下：

$$\begin{aligned} \max_{w,b} \quad & \frac{1}{\|w\|_2} \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, 2, \dots, m \end{aligned}$$

线性可分SVM

$$w^T = (w_1, w_2, \dots, w_n)$$

$$\|w\|_2 = \sqrt{w_1^2 + w_2^2 + \dots + w_n^2}$$

$$\max_{w,b} \frac{1}{\|w\|_2}$$

$$s.t : y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, 2, \dots, m$$

优化问题等价于

$$\min_{w,b} \frac{1}{2} \|w\|_2^2$$

$$s.t : y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, 2, \dots, m$$

线性可分SVM

- SVM原始目标函数/损失函数为：

$$J(w) = \frac{1}{2} \|w\|_2^2$$

$$s.t : y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, 2, \dots, m$$

$$\xrightarrow{\text{优化目标}} w^*, b^* = \min_{w, b} J(w)$$

线性可分SVM

- 将此时的目标函数和约束条件使用KKT条件转换为拉格朗日函数，从而转换为无约束的优化函数。

$$\begin{array}{ll} J(w) = \frac{1}{2} \|w\|_2^2 & \Rightarrow J(w) = \frac{1}{2} \|w\|_2^2 \\ s.t : y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, 2, \dots, m & s.t : 1 - y^{(i)}(w^T x^{(i)} + b) \leq 0, i = 1, 2, \dots, m \end{array}$$

↓

$$L(w, b, \beta) = \frac{1}{2} \|w\|_2^2 + \sum_{i=1}^m \beta_i [1 - y^{(i)}(w^T x^{(i)} + b)], \beta_i \geq 0$$

线性可分SVM

$$L(w, b, \beta) = \frac{1}{2} \|w\|_2^2 + \sum_{i=1}^m \beta_i [1 - y^{(i)}(w^T x^{(i)} + b)], \beta_i \geq 0$$

- 引入拉格朗日乘子后，优化目标变成：

$$\min_{w, b} \max_{\beta \geq 0} L(w, b, \beta)$$

- 根据拉格朗日对偶化特性，将该优化目标转换为等价的对偶问题来求解，从而优化目标变成：

$$\max_{\beta \geq 0} \min_{w, b} L(w, b, \beta)$$

- 所以对于该优化函数而言，可以先求优化函数对于w和b的极小值，然后再求解对于拉格朗日乘子 β 的极大值。

线性可分SVM

$$\max_{\beta \geq 0} \min_{w, b} L(w, b, \beta)$$

$$L(w, b, \beta) = \frac{1}{2} \|w\|_2^2 + \sum_{i=1}^m \beta_i [1 - y^{(i)}(w^T x^{(i)} + b)], \beta_i \geq 0$$

- 首先求让函数L极小化的时候w和b的取值，这个极值可以直接通过对函数L分别求w和b的偏导数得到：

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w - \sum_{i=1}^m \beta_i y^{(i)} x^{(i)} = 0 \Rightarrow w = \sum_{i=1}^m \beta_i y^{(i)} x^{(i)}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow -\sum_{i=1}^m \beta_i y^{(i)} = 0 \Rightarrow \sum_{i=1}^m \beta_i y^{(i)} = 0$$

线性可分SVM

$$L(w, b, \beta) = \frac{1}{2} \|w\|_2^2 + \sum_{i=1}^m \beta_i [1 - y^{(i)}(w^T x^{(i)} + b)], \beta_i \geq 0$$

$$w = \sum_{i=1}^m \beta_i y^{(i)} x^{(i)} \quad \sum_{i=1}^m \beta_i y^{(i)} = 0$$

- 将求解出来的w和b带入优化函数L中，定义优化之后的函数如下：

$$\begin{aligned} \ell(\beta) &= \frac{1}{2} \|w\|_2^2 + \sum_{i=1}^m \beta_i [1 - y^{(i)}(w^T x^{(i)} + b)] \\ &= \frac{1}{2} \|w\|_2^2 - \sum_{i=1}^m \beta_i [y^{(i)}(w^T x^{(i)} + b) - 1] \\ &= \frac{1}{2} w^T w - \sum_{i=1}^m \beta_i y^{(i)} w^T x^{(i)} - \sum_{i=1}^m \beta_i y^{(i)} b + \sum_{i=1}^m \beta_i \\ &= \frac{1}{2} w^T \sum_{i=1}^m \beta_i y^{(i)} x^{(i)} - w^T \sum_{i=1}^m \beta_i y^{(i)} x^{(i)} - b \sum_{i=1}^m \beta_i y^{(i)} + \sum_{i=1}^m \beta_i \\ &= -\frac{1}{2} w^T \sum_{i=1}^m \beta_i y^{(i)} x^{(i)} - b \sum_{i=1}^m \beta_i y^{(i)} + \sum_{i=1}^m \beta_i \\ &= -\frac{1}{2} \left(\sum_{j=1}^m \beta_j y^{(j)} x^{(j)} \right)^T \left(\sum_{i=1}^m \beta_i y^{(i)} x^{(i)} \right) + \sum_{i=1}^m \beta_i \\ &= -\frac{1}{2} \sum_{j=1}^m \beta_j y^{(j)} x^{(j)T} \sum_{i=1}^m \beta_i y^{(i)} x^{(i)} + \sum_{i=1}^m \beta_i \\ &= \sum_{i=1}^m \beta_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} x^{(j)T} x^{(i)} \end{aligned}$$

$$\Rightarrow \ell(\beta) = \sum_{i=1}^m \beta_i - \frac{1}{2} \sum_{i=1, j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)}$$

$$s.t : \sum_{i=1}^m \beta_i y^{(i)} = 0$$

$$\beta_i \geq 0, i = 1, 2, \dots, m$$

线性可分SVM

$$L(w, b, \beta) = \frac{1}{2} \|w\|_2^2 + \sum_{i=1}^m \beta_i [1 - y^{(i)}(w^T x^{(i)} + b)], \beta_i \geq 0$$
$$\ell(\beta) = \sum_{i=1}^m \beta_i - \frac{1}{2} \sum_{i=1, j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)} \quad \max_{\beta \geq 0} \min_{w, b} L(w, b, \beta)$$

- 通过对 w 、 b 极小化后，我们最终得到的优化函数只和 β 有关，所以此时我们可以直接极大化我们的优化函数，得到 β 的值，从而可以最终得到 w 和 b 的值。

$$\begin{array}{ccc} \max_{\beta \geq 0} \ell(\beta) & & \min_{\beta \geq 0} -\ell(\beta) \\ s.t: \sum_{i=1}^m \beta_i y^{(i)} = 0 & \xrightarrow{\text{等价于}} & s.t: \sum_{i=1}^m \beta_i y^{(i)} = 0 \\ & \downarrow & \\ \min_{\beta \geq 0} \frac{1}{2} \sum_{i=1, j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)} - \sum_{i=1}^m \beta_i & & \\ s.t: \sum_{i=1}^m \beta_i y^{(i)} = 0 & & \end{array}$$

- 备注： β 值的求解使用SMO算法，后续课程中介绍。

线性可分SVM

- 假设存在最优解 β^* ；根据 w 、 b 和 β 的关系，可以分别计算出对应的 w 值和 b 值(一般使用所有支持向量的计算均值来作为实际的 b 值)；

$$w^* = \sum_{i=1}^m \beta_i^* y^{(i)} x^{(i)}$$

$$y^s (w^T x^s + b) = y^s \left(\sum_{i=1}^m \beta_i^* y^{(i)} x^{(i)T} x^s + b \right) = 1 \Rightarrow b^* = y^s - \sum_{i=1}^m \beta_i^* y^{(i)} x^{(i)T} x^s$$

- 这里的 (x^s, y^s) 即支持向量，根据KKT条件中的对偶互补条件(松弛条件约束)，支持向量必须满足一下公式：

$$\beta_i (1 - y^{(i)} (w^T x^{(i)} + b)) = 0 \Rightarrow \left\{ (x^{(i)}, y^{(i)}) \mid \beta_i > 0, i = 1, 2, \dots, m \right\}$$

线性可分SVM算法流程

- 输入线性可分的 m 个样本数据 $\{(x^1, y^1), (x^2, y^2), \dots, (x^m, y^m)\}$, 其中 x 为 n 维的特征向量, y 为二元输出, 取值为 $+1$ 或者 -1 ; SVM模型输出为参数 w 、 b 以及分类决策函数。

- 构造约束优化问题;
$$\min_{\beta \geq 0} \frac{1}{2} \sum_{i=1, j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)} - \sum_{i=1}^m \beta_i$$

$$s.t: \sum_{i=1}^m \beta_i y^{(i)} = 0$$

- 使用SMO算法求出上式优化中对应的最优解 β^* ;
- 找出所有的支持向量集合 S ; $S = \{(x^{(i)}, y^{(i)}) | \beta_i > 0, i = 1, 2, \dots, m\}$
- 更新参数 w^* 、 b^* 的值;

$$w^* = \sum_{i=1}^m \beta_i^* y^{(i)} x^{(i)} \quad b^* = \frac{1}{S} \sum_{s=1}^S \left(y^s - \sum_{i=1}^m \beta_i^* y^{(i)} x^{(i)T} x^s \right)$$

- 构建最终的分类器

$$f(x) = \text{sign}(w^* \bullet x + b^*)$$

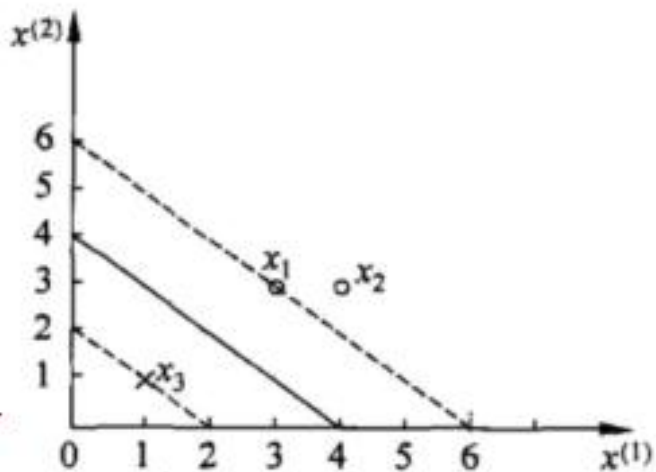
线性可分SVM案例

$$\min \frac{1}{2} \sum_{i=1, j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)} - \sum_{i=1}^m \beta_i$$

$$s.t : \sum_{i=1}^m \beta_i y^{(i)} = 0$$

$$\beta_i \geq 0, i = 1, 2, \dots, m$$

- 给定三个数据点：正例点 $x_1=(3,3), x_2=(4,3)$ ，负例点 $x_3=(1,1)$ ，构造此时的约束优化条件。



$$\frac{1}{2} (18\beta_1^2 + 25\beta_2^2 + 2\beta_3^2 + 42\beta_1\beta_2 - 12\beta_1\beta_3 - 14\beta_2\beta_3) - \beta_1 - \beta_2 - \beta_3$$

$$s.t : \beta_1 + \beta_2 - \beta_3 = 0$$

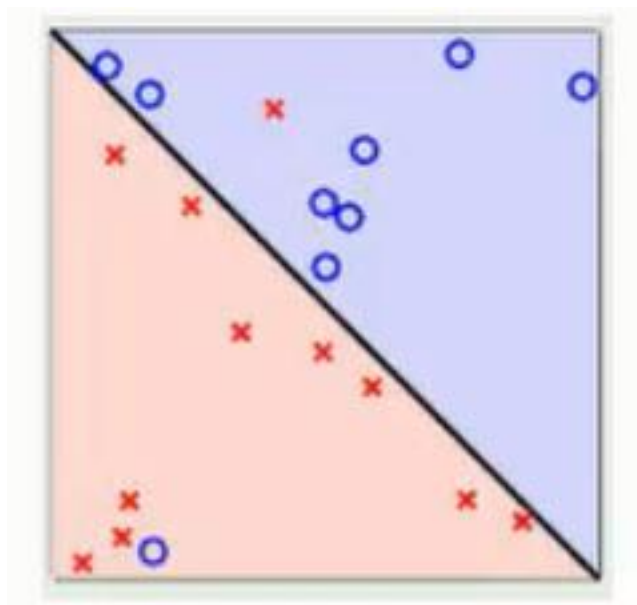
$$\beta_i \geq 0, i = 1, 2, \dots, m$$

线性可分SVM总结

- 1. 要求数据必须是线性可分的;
- 2. 纯线性可分的SVM模型对于异常数据的预测可能会不太准;
- 3. 对于线性可分的数据, 线性SVM分类器的效果非常不错。

SVM的软间隔模型

- 线性可分SVM中要求数据必须是线性可分的，才可以找到分类的超平面，但是有的时候线性数据集中存在少量的异常点，由于这些异常点导致了数据集不能够线性划分；直白来讲就是：正常数据本身是线性可分的，但是由于存在异常点数据，导致数据集不能够线性可分；



SVM的软间隔模型

- 如果线性数据中存在异常点导致没法直接使用SVM线性分割模型的时候，我们可以通过引入软间隔的概念来解决这个问题；
- **硬间隔**：可以认为线性划分SVM中的距离度量就是硬间隔，在线性划分SVM中，要求函数距离一定是大于1的，最大化硬间隔条件为：

$$\min_{w,b} \frac{1}{2} \|w\|_2^2; \quad s.t: y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, 2, \dots, m$$

- **软间隔**：SVM对于训练集中的每个样本都引入一个松弛因子(ξ)，使得函数距离加上松弛因子后的值是大于等于1；这表示相对于硬间隔，对样本到超平面距离的要求放松了。

$$y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, i = 1, 2, \dots, m, \xi_i \geq 0$$

SVM的软间隔模型

$$y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, i = 1, 2, \dots, m, \xi_i \geq 0$$

- 松弛因子(ξ)越大, 表示样本点离超平面越近, 如果松弛因子大于1, 那么表示允许该样本点分错, 所以说加入松弛因子是有成本的, 过大的松弛因子可能会导致模型分类错误, 所以最终的目标函数就转换成为:

$$\begin{aligned} \min_{w, b} \quad & \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, i = 1, 2, \dots, m \\ & \xi_i \geq 0, i = 1, 2, \dots, m \end{aligned}$$

- 备注: 函数中的 $C > 0$ 是惩罚参数, 是一个超参数, 类似L1/L2 norm的参数; C 越大表示对误分类的惩罚越大, 也就是越不允许存在分错的样本; C 越小表示对误分类的惩罚越小, 也就是表示允许更多的分错样本存在; C 值的给定需要调参。

SVM的软间隔模型

$$\begin{aligned} \min_{w, b} \quad & \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & y^{(i)} (w^T x^{(i)} + b) \geq 1 - \xi_i, i = 1, 2, \dots, m \\ & \xi_i \geq 0, i = 1, 2, \dots, m \end{aligned}$$

- 同线性可分SVM，根据KK条件构造软间隔最大化的约束问题对应的拉格朗日函数如下：

$$L(w, b, \xi, \beta, \mu) = \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^m \xi_i + \sum_{i=1}^m \beta_i [1 - \xi_i - y^{(i)} (w^T x^{(i)} + b)] - \sum_{i=1}^m \mu_i \xi_i, \beta_i \geq 0, \mu_i \geq 0$$

- 从而将我们的优化目标函数转换为：

$$\min_{w, b, \xi} \max_{\beta, \mu} L(w, b, \xi, \beta, \mu)$$

- 优化目标同样满足KKT条件，所以使用拉格朗日对偶将优化问题转换为等价的对偶问题：

$$\max_{\beta, \mu} \min_{w, b, \xi} L(w, b, \xi, \beta, \mu)$$

SVM的软间隔模型

- 先求优化函数对于 w 、 b 、 ξ 的极小值，这个可以通过分别对优化函数 L 求 w 、 b 、 ξ 的偏导数得，从而可以得到 w 、 b 、 ξ 关于 β 和 μ 之间的关系。

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w - \sum_{i=1}^m \beta_i y^{(i)} x^{(i)} = 0 \Rightarrow w = \sum_{i=1}^m \beta_i y^{(i)} x^{(i)}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow -\sum_{i=1}^m \beta_i y^{(i)} = 0 \Rightarrow \sum_{i=1}^m \beta_i y^{(i)} = 0$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow C - \beta_i - \mu_i = 0$$

SVM的软间隔模型

- 将 w 、 b 、 ξ 的值带入L函数中，就可以消去优化函数中的 w 、 b 、 ξ ，定义优化之后的函数如下：

$$\begin{aligned}\ell(\beta, \mu) &= \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^m \xi_i + \sum_{i=1}^m \beta_i [1 - \xi_i - y^{(i)}(w^T x^{(i)} + b)] - \sum_{i=1}^m \mu_i \xi_i \\&= \frac{1}{2} \|w\|_2^2 - \sum_{i=1}^m \beta_i [y^{(i)}(w^T x^{(i)} + b) - 1 + \xi_i] + \sum_{i=1}^m (C - \mu_i) \xi_i \\&= \frac{1}{2} \|w\|_2^2 - \sum_{i=1}^m \beta_i [y^{(i)}(w^T x^{(i)} + b) - 1] \\&= \frac{1}{2} w^T w - \sum_{i=1}^m \beta_i y^{(i)} w^T x^{(i)} - \sum_{i=1}^m \beta_i y^{(i)} b + \sum_{i=1}^m \beta_i \\&= -\frac{1}{2} w^T \sum_{i=1}^m \beta_i y^{(i)} x^{(i)} - b \sum_{i=1}^m \beta_i y^{(i)} + \sum_{i=1}^m \beta_i \\&= -\frac{1}{2} \left(\sum_{i=1}^m \beta_i y^{(i)} x^{(i)} \right)^T \left(\sum_{i=1}^m \beta_i y^{(i)} x^{(i)} \right) + \sum_{i=1}^m \beta_i \\&= -\frac{1}{2} \sum_{i=1}^m \beta_i y^{(i)} x^{(i)T} \sum_{i=1}^m \beta_i y^{(i)} x^{(i)} + \sum_{i=1}^m \beta_i \\&= \sum_{i=1}^m \beta_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)}\end{aligned}$$

$$\Rightarrow \ell(\beta) = \sum_{i=1}^m \beta_i - \frac{1}{2} \sum_{i=1, j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)}$$

$$s.t : \sum_{i=1}^m \beta_i y^{(i)} = 0$$

$$C - \beta_i - \mu_i = 0$$

$$\beta_i \geq 0, i = 1, 2, \dots, m$$

$$\mu_i \geq 0, i = 1, 2, \dots, m$$

SVM的软间隔模型

- 最终优化后的目标函数/损失函数和线性可分SVM模型基本一样，除了约束条件不同而已，也就是说也可以使用SMO算法来求解。

$$\begin{aligned} \min & \frac{1}{2} \sum_{i=1, j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)} - \sum_{i=1}^m \beta_i \\ \text{s.t.} & \sum_{i=1}^m \beta_i y^{(i)} = 0 \\ & 0 \leq \beta_i \leq C, i = 1, 2, \dots, m \end{aligned}$$

SVM的软间隔模型

- 在硬间隔最大化的时候，支持向量比较简单，就是离超平面的函数距离为1的样本点就是支持向量。
- 在软间隔中，根据KKT条件中的对偶互补条件: $\beta(1-\xi-y(wx+b))=0$ 和 $\mu(-\xi)=0$ ，以及 $C-\beta-\mu=0$ ；从而有：
 - 当 $0 < \beta_i \leq C$ 的时候，并且 $\xi_i=0$ 的样本点均是支持向量(即所有的 $0 < \beta_i < C$)。即满足 $|wx+b|=1$ 的所有样本均是支持向量
 - 当 $0 < \beta_i < C$ 对应的样本就是支持向量。
 - 备注：软间隔和硬间隔中的支持向量的规则是一样的；

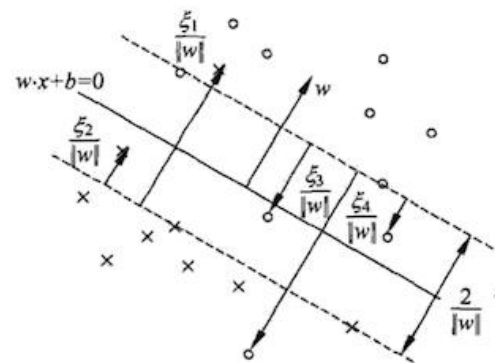


图 7.5 软间隔的支持向量

SVM的软间隔模型算法流程

- 输入线性可分的 m 个样本数据 $\{(x^1, y^1), (x^2, y^2), \dots, (x^m, y^m)\}$, 其中 x 为 n 维的特征向量, y 为二元输出, 取值为 $+1$ 或者 -1 ; SVM模型输出为参数 w 、 b 以及分类决策函数。

- 选择一个惩罚系数 $C > 0$, 构造约束优化问题;
$$\min \frac{1}{2} \sum_{i=1, j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} x^{(i)^T} x^{(j)} - \sum_{i=1}^m \beta_i$$
$$s.t : \sum_{i=1}^m \beta_i y^{(i)} = 0$$
$$0 \leq \beta_i \leq C, i = 1, 2, \dots, m$$

- 使用SMO算法求出上式优化中对应的最优解 β^* ;
- 找出所有的支持向量集合 S ; $S = \{(x^{(i)}, y^{(i)}) | 0 < \beta_i < C, i = 1, 2, \dots, m\}$
- 更新参数 w^* 、 b^* 的值;

$$w^* = \sum_{i=1}^m \beta_i^* y^{(i)} x^{(i)} \quad b^* = \frac{1}{S} \sum_{s=1}^S \left(y^s - \sum_{i=1}^m \beta_i^* y^{(i)} x^{(i)^T} x^s \right)$$

- 构建最终的分类器

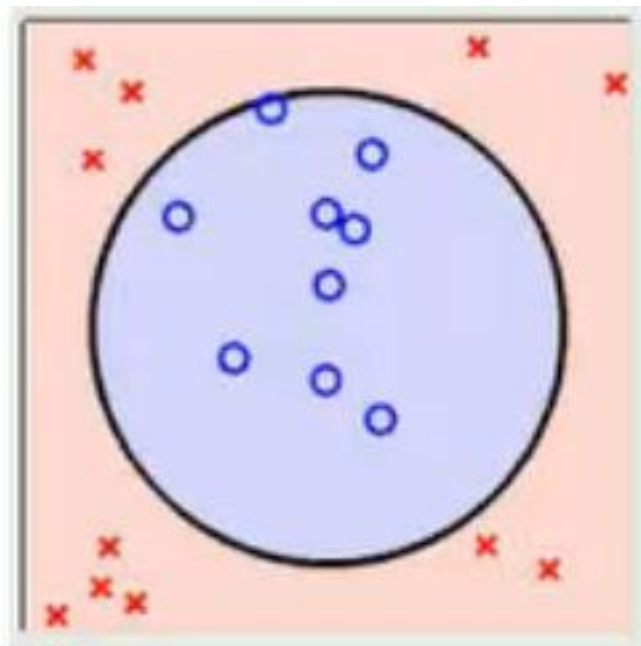
$$f(x) = \text{sign}(w^* \bullet x + b^*)$$

SVM的软间隔模型总结

- 1. 可以解决线性数据中携带异常点的分类模型构建的问题;
- 2. 通过引入惩罚项系数(松弛因子), 可以增加模型的泛化能力, 即鲁棒性;
- 3. 如果给定的惩罚项系数 C 越小, 表示在模型构建的时候, 就允许存在越多的分类错误的样本, 也就表示此时模型的准确率会比较低; 如果惩罚项系数越大, 表示在模型构建的时候, 就越不允许存在分类错误的样本, 也就表示此时模型的准确率会比较高。

非线性可分SVM

- 不管是线性可分SVM还是加入惩罚系数后的软间隔线性可分SVM其实都要求数据本身是线性可分的，对于完全不可以线性可分的数据，这两种算法模型就没法解决这个问题了



多项式回归回顾

- 在线性回归中，我们可以通过多项式扩展将低维度的数据扩展成为高维度的数据，从而可以使用线性回归模型来解决问题。也就是说对于二维空间中不是线性可分的数据，将其映射到高维空间中后，变成了线性可分的数据。

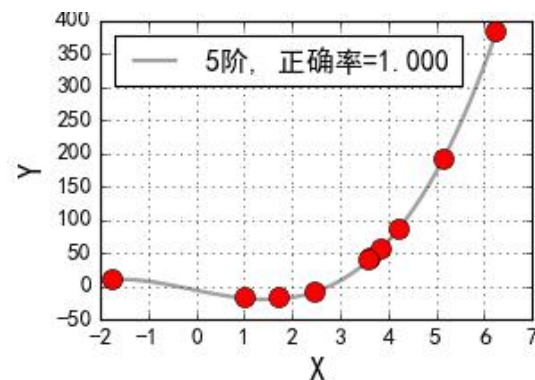
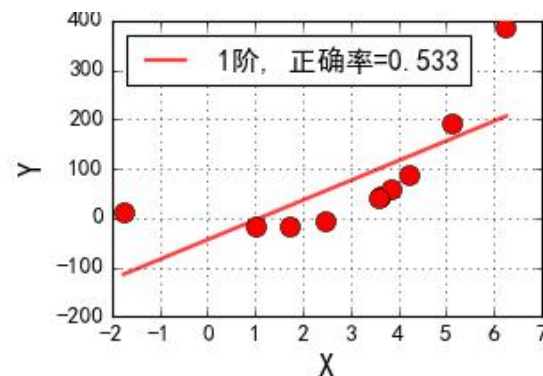
二维线性模型:
$$h_{\theta}(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$$(x_1, x_2) \xrightarrow{\text{多项式扩展}} (x_1, x_2, x_1^2, x_2^2, x_1 x_2)$$

五维线性模型:

$$h_{\theta}(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2$$

$$\xrightarrow{\text{等价}} h_{\theta}(x_1, x_2) = \theta_0 + \theta_1 z_1 + \theta_2 z_2 + \theta_3 z_3 + \theta_4 z_4 + \theta_5 z_5$$



非线性可分SVM

- 结合多项式回归在处理非线性可分数据时候的作用，在SVM的线性不可分的数据上，如果将数据映射到高维空间中，那么数据就会变成线性可分的，从而就可以使用线性可分SVM模型或者软间隔线性可分SVM模型。
- 也就是说，对于线性不可分SVM模型来讲，重点在于低维特征数据到高维特征数据之间的映射。

非线性可分SVM

- 定义一个从低维特征空间到高维特征空间的映射函数 Φ ，非线性可分SVM的优化目标函数：

$$\begin{aligned} \min & \frac{1}{2} \sum_{i=1, j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} \phi(x^{(i)}) \bullet \phi(x^{(j)}) - \sum_{i=1}^m \beta_i \\ \text{s.t.} & \sum_{i=1}^m \beta_i y^{(i)} = 0 \\ & 0 \leq \beta_i \leq C, i = 1, 2, \dots, m \end{aligned}$$

- 可以看到的是，只需要将原来的低维空间中的两个向量的点积转换为高维空间中两个向量的点积即可。

非线性可分SVM

- 这样一来问题就解决了吗？似乎是的：拿到非线性数据，就找一个映射，然后一股脑把原来的数据映射到新空间中，再做线性 SVM 即可。不过事实上没有这么简单！其实刚才的方法稍想一下就会发现有问题：在最初的例子里做了一个二阶多项式的转换，对一个二维空间做映射，选择的新空间是原始空间的所有一阶和二阶的组合，得到了5个维度；如果原始空间是三维，那么我们会得到9维的新空间；如果原始空间是 n 维，那么我们会得到一个 $n(n+3)/2$ 维的新空间；这个数目是呈爆炸性增长的，这给计算带来了非常大的困难，而且如果遇到无穷维的情况，就根本无从计算。

核函数

- 假设函数 Φ 是一个从低维特征空间到高维特征空间的一个映射, 那么如果存在函数 $K(x,z)$, 对于任意的低维特征向量 x 和 z , 都有:

$$K(x, z) = \phi(x) \bullet \phi(z)$$

- 称函数 $K(x,z)$ 为**核函数(kernel function)**;
- 核函数: 在低维空间上的计算值等价于向量做维度扩展后的点乘的结果。

核函数

- 核函数在解决线性不可分问题的时候，采取的方式是：使用低维特征空间上的计算来避免在高维特征空间中向量内积的恐怖计算量；也就是说此时SVM模型可以应用在高维特征空间中数据可线性分割的优点，同时又避免了引入这个高维特征空间恐怖的内积计算量。
- 即：用低维空间中少的内积的计算量来让模型具有高维空间中的线性可分的优点。
- 强调：由于核函数是不存在将数据映射到高维空间，所以SVM模型此时是不存在模型参数 W 的具体值。

核函数

- 不妨还是从最开始的简单例子出发，设两个向量 $x_1 = (\mu_1, \mu_2)^T$ 和 $x_2 = (\eta_1, \eta_2)^T$ ，而即是到前面说的五维空间的映射，因此映射过后的内积为：

$$\phi(x_1) \bullet \phi(x_2) = \mu_1 \eta_1 + \mu_2 \eta_2 + \mu_1^2 \eta_1^2 + \mu_2^2 \eta_2^2 + \mu_1 \mu_2 \eta_1 \eta_2$$

- 而同时我们可以发现有一下公式：

$$(x_1 \bullet x_2 + 1)^2 = 2\mu_1 \eta_1 + 2\mu_2 \eta_2 + \mu_1^2 \eta_1^2 + \mu_2^2 \eta_2^2 + 2\mu_1 \mu_2 \eta_1 \eta_2 + 1$$

- 可以发现两者之间非常相似，所以我们只要乘上一个相关的系数，就可以让这两个式子的值相等，这样不就将五维空间的一个内积转换为两维空间的内积的运算。

核函数

- 现有有两个两维的向量，进行二阶多项式扩展，然后进行内积计算，这个时候映射高维后计算的计算量为：11次乘法+4次加法；采用近似计算的计算量为：3次乘法+2次加法；采用加系数后的近似计算的计算量为：4次乘法+2次加法；

x_1	3	5
x_2	4	2

z_1	3	5	$3*3=9$	$5*5=25$	$3*5=15$
z_2	4	2	$4*4=16$	$2*2=4$	$4*2=8$

$$\phi(x_1) \bullet \phi(x_2) = z_1 \bullet z_2 = 3*4 + 2*5 + 9*16 + 4*25 + 8*15 = 386$$

$$(x_1 \bullet x_2 + 1)^2 = (3*4 + 2*5 + 1)^2 = 23*23 = 529$$

$$(0.8476x_1 \bullet x_2 + 1)^2 = (0.8476*(3*4 + 2*5) + 1)^2 = 18.6472*18.6472 \approx 386.01$$

核函数

- 线性核函数(Linear Kernel):

$$K(x, z) = x \bullet z$$

- 多项式核函数(Polynomial Kernel): 其中 γ 、 r 、 d 属于超参, 需要调参定义;

$$K(x, z) = (\gamma x \bullet z + r)^d$$

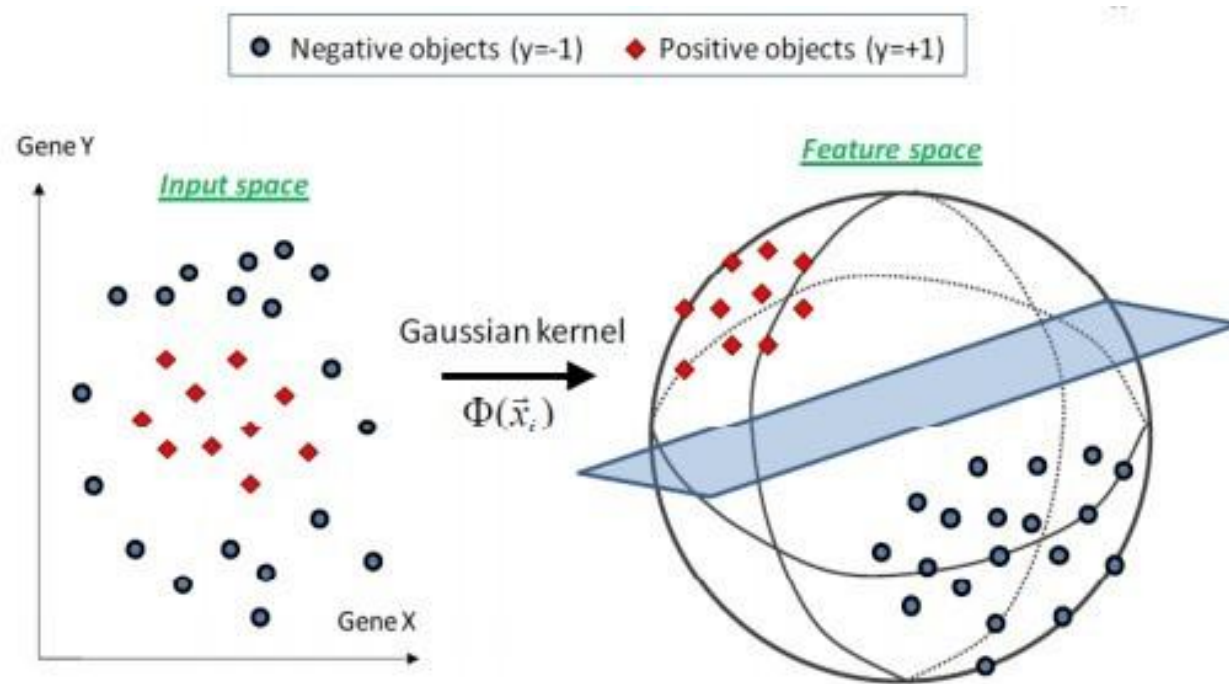
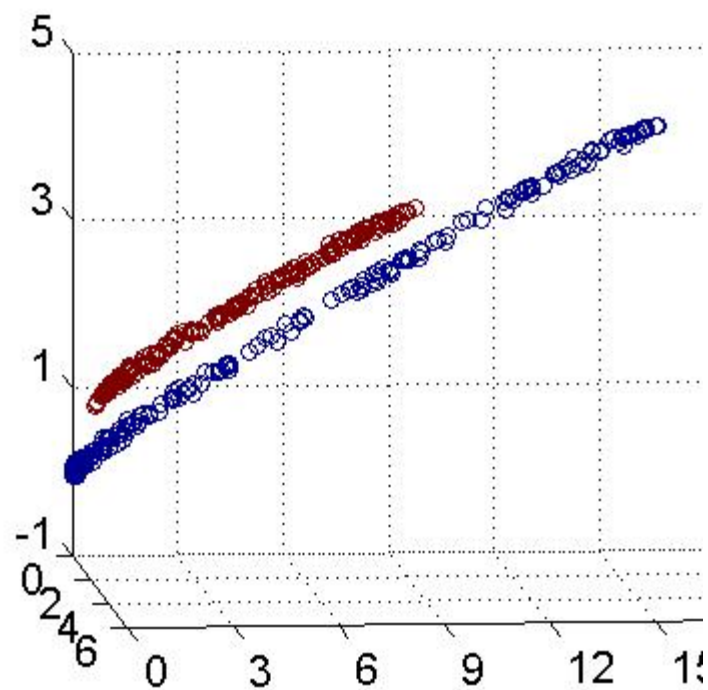
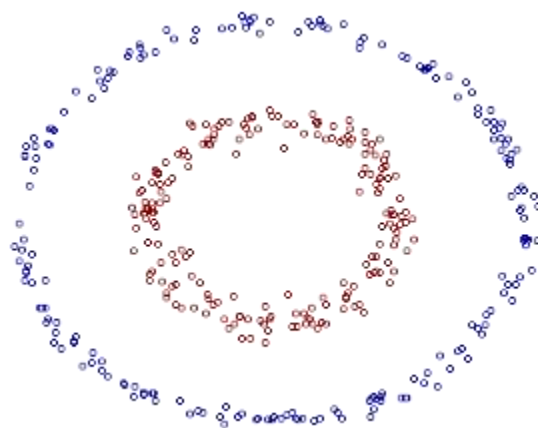
- 高斯核函数(Gaussian Kernel): 其中 γ 属于超参, 要求大于0, 需要调参定义;

$$K(x, z) = e^{-\gamma \|x - z\|_2^2}$$

- Sigmoid核函数(Sigmoid Kernel): 其中 γ 、 r 属于超参, 需要调参定义;

$$K(x, z) = \tanh(\gamma x \bullet z + r)$$

核函数



核函数总结

- 1. 核函数可以自定义；核函数必须是正定核函数，即Gram矩阵是半正定矩阵；
- 2. 核函数的价值在于它的效果相当于将特征进行从低维到高维的转换，但核函数它是在低维空间上的计算，而将实质上的分类效果表现在了高维上，也就如上文所说的避免了直接在高维空间中的复杂计算；
- 3. 通过核函数，可以将非线性可分的数据转换为线性可分数据来进行SVM划分；

$$Gram = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \dots & K(x_1, x_m) \\ K(x_2, x_1) & K(x_2, x_2) & \dots & K(x_2, x_m) \\ \dots & \dots & \dots & \dots \\ K(x_m, x_1) & K(x_m, x_2) & \dots & K(x_m, x_m) \end{bmatrix}$$

高斯核公式证明(扩展)

- 令 $z=x$ ；那么进行多维变换后，应该是同一个向量，从而可以得到以下公式：

$$K(x, z) = e^{-\gamma \|x-z\|_2^2}$$

$$= e^{-\gamma \|x\|_2^2} e^{-\gamma \|z\|_2^2} e^{2\gamma x^T z}$$

$$\stackrel{\text{Taylor}}{=} e^{-\gamma \|x\|_2^2} e^{-\gamma \|z\|_2^2} \left(\sum_{i=0}^{+\infty} \frac{(2\gamma x^T z)^i}{i!} \right)$$

$$= \sum_{i=0}^{+\infty} \left(e^{-\gamma \|x\|_2^2} e^{-\gamma \|z\|_2^2} \frac{(2\gamma x^T z)^i}{i!} \right) \Rightarrow \phi(x) = e^{-\gamma \|x\|_2^2} \cdot \left(1, \sqrt{\frac{(2\gamma)^1}{1!}} \|x\|_2^1, \sqrt{\frac{(2\gamma)^2}{2!}} \|x\|_2^2, \dots \right)$$

$$= \sum_{i=0}^{+\infty} \left(e^{-\gamma \|x\|_2^2} e^{-\gamma \|z\|_2^2} \sqrt{\frac{(2\gamma)^i}{i!}} \sqrt{\frac{(2\gamma)^i}{i!}} \|x\|_2^i \|z\|_2^i \right)$$

$$= \phi(x) \bullet \phi(z)$$

scikit-learn SVM算法库概述

- scikit-learn中SVM的算法库主要分为两类，一类是分类算法，包括：SVC、NuSVC和LinearSVC这三个类，另外一类是回归算法，包括：SVR、NuSVR和LinearSVR这三个类；除此之外，用的比较多的SVM模型还有OneClassSVM类(主要功能是：异常点检测)。
- 详见：<http://scikit-learn.org/0.18/modules/classes.html#module-sklearn.svm>

Estimators

<code>svm.SVC</code> ([C, kernel, degree, gamma, coef0, ...])	C-Support Vector Classification.
<code>svm.LinearSVC</code> ([penalty, loss, dual, tol, C, ...])	Linear Support Vector Classification.
<code>svm.NuSVC</code> ([nu, kernel, degree, gamma, ...])	Nu-Support Vector Classification.
<code>svm.SVR</code> ([kernel, degree, gamma, coef0, tol, ...])	Epsilon-Support Vector Regression.
<code>svm.LinearSVR</code> ([epsilon, tol, C, loss, ...])	Linear Support Vector Regression.
<code>svm.NuSVR</code> ([nu, C, kernel, degree, gamma, ...])	Nu Support Vector Regression.
<code>svm.OneClassSVM</code> ([kernel, degree, gamma, ...])	Unsupervised Outlier Detection.

scikit-learn SVM算法库概述

分类算法

参数	LinearSVC	SVC	NuSVC
C	惩罚项系数，默认为1，一般通过交叉验证给定合适值		不支持
nu	不支持		表示训练集上错误率的上限，默认为0.5，取值范围(0,1]，功能和C类似
kernel	不支持，只能使用线性核函数	给定具体的核函数或者Gram矩阵，参数可选:linear、rbf、poly、sigmoid、precomputed	
penalty	对线性拟合有意义，参数可选l1和l2	不支持	
degree	不支持	多项式核函数中，d的参数值，一般需要通过交叉验证给定	
gamma	不支持	rbf、poly和sigmoid核函数中的参数值 γ ，一般需要通过交叉验证给定	
coef0	不支持	poly和sigmoid核函数中的参数值r，一般需要通过交叉验证给定	

scikit-learn SVM算法库概述

分类算法

参数	LinearSVC	SVC	NuSVC
class_weight	样本类别权重，主要是为了防止训练集中某些类别的样本过多，导致训练出来的决策偏向这些类别。参数可选：权重列表、balanced，如果是balanced的话，表示算法会自己计算权重，样本量少的类别权重会高；如果样本没有比较明显的偏重的话，一般不用管这个参数，默认为None		
decision_function_shape	不支持该参数，使用multi_class代替	指定进行多元分类的时候，进行如何操作：可选ovo和ovr；推荐选择： ovr	
multi_class	指定进行多元分类的时候，进行如何构建操作，可选ovr和crammer_singer；一般情况为ovr；(crammer_singer为ovr的优化，但是实际效果不好)	不支持该参数，使用decision_function_shape代替	
cache_size	不支持	在大样本的时候，缓存会影响模型的构建速度的；当机器内存比较大的时候，可以考虑将该值设置为500或者1000，默认为200(单位为M)	

scitit-learn SVM算法库概述

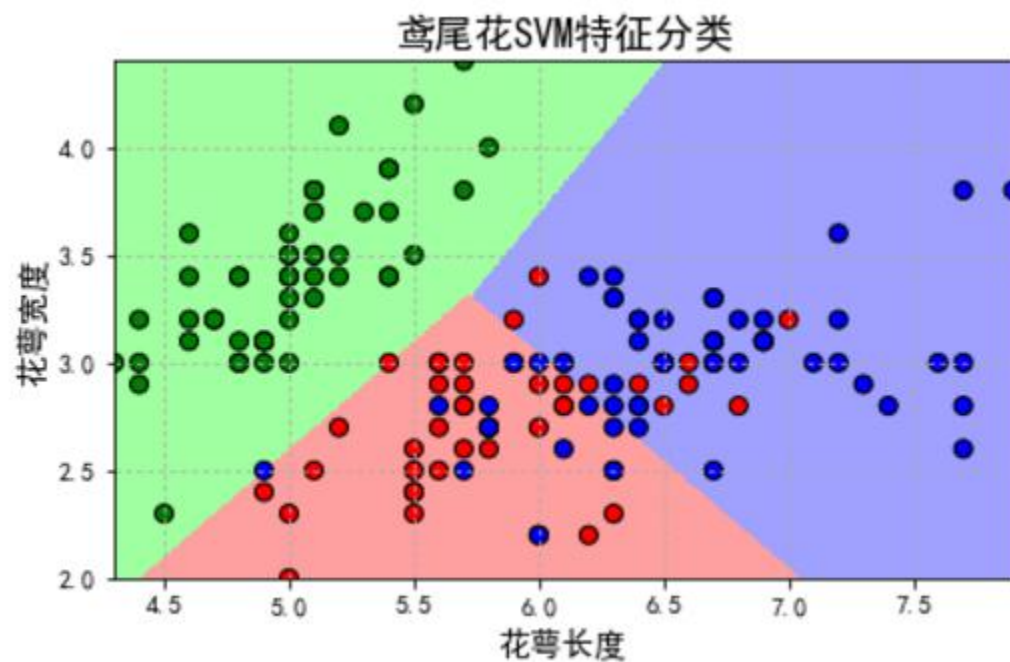
- OneClassSVM是一种异常点检测的算法，算法原理：先使用SVM算法，对正常数据进行模型训练，找出正常数据的数据特性(eg: 数据集中在哪儿?? 数据的边界是什么???)，也就是找出数据中分割面(也就是正常数据的边界)，以边界作为算法找到的支持向量。预测的时候，如果样本点落在支持向量内部，那就认为属于正常样本，否则就认为属于异常样本。

参数	OneClassSVM
kernel	核函数，可选: 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'
nu	控制分类错误的样本百分比上限，或者说控制支持向量个数的下限，默认为0.5，可取值范围为:(0,1]
degree	多项式核函数超参，一般通过交叉验证给定
gamma	rbf、poly、sigmoid核函数超参，一般通过交叉验证给定
coef0	poly、sigmoid核函数超参，一般通过交叉验证给定
cache_size	在大样本的时候，缓存会影响模型的构建速度的；当机器内存比较大的时候，可以考虑将该值设置为500或者1000，默认为200(单位为M)

案例一：鸢尾花数据分类

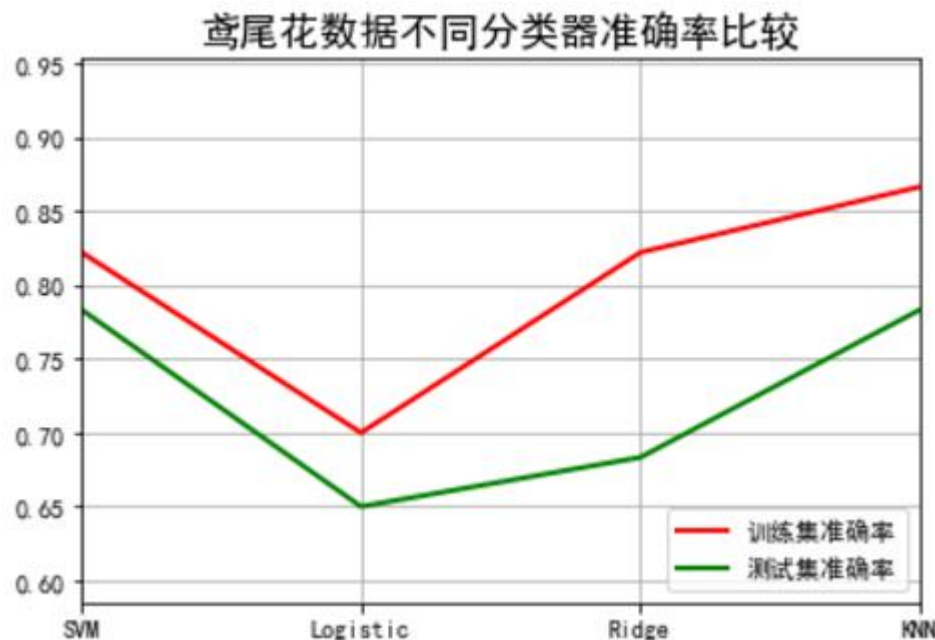
- 使用线性SVM对鸢尾花数据进行分类操作

```
clf = svm.SVC(C=1, kernel='linear')  
clf.fit(x_train, y_train)
```

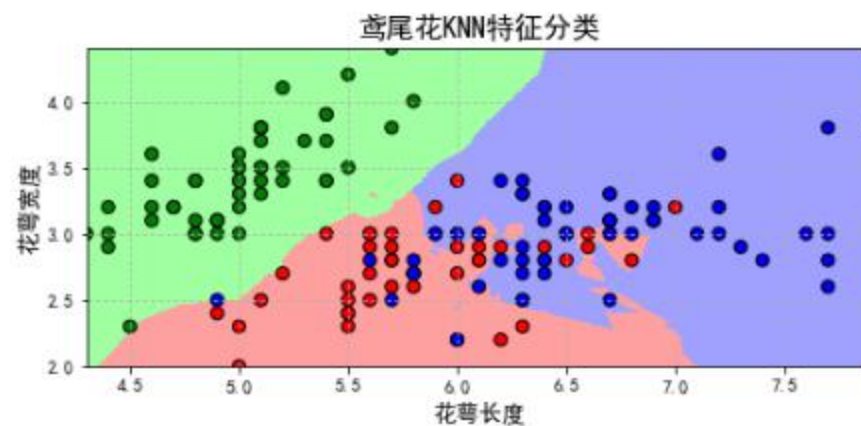
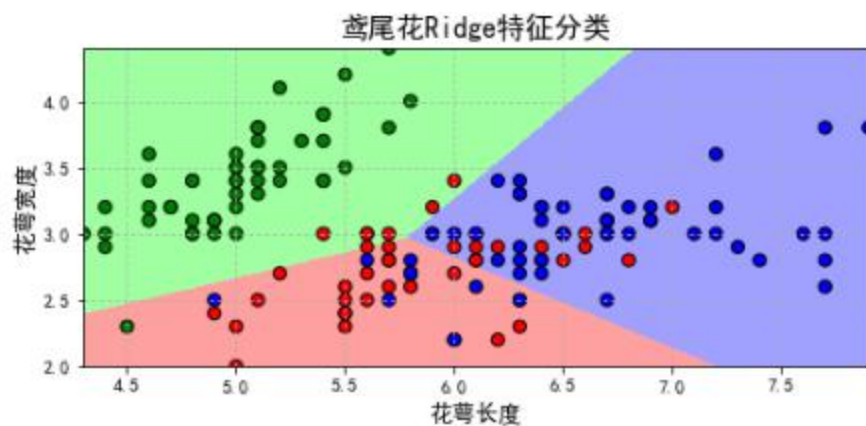
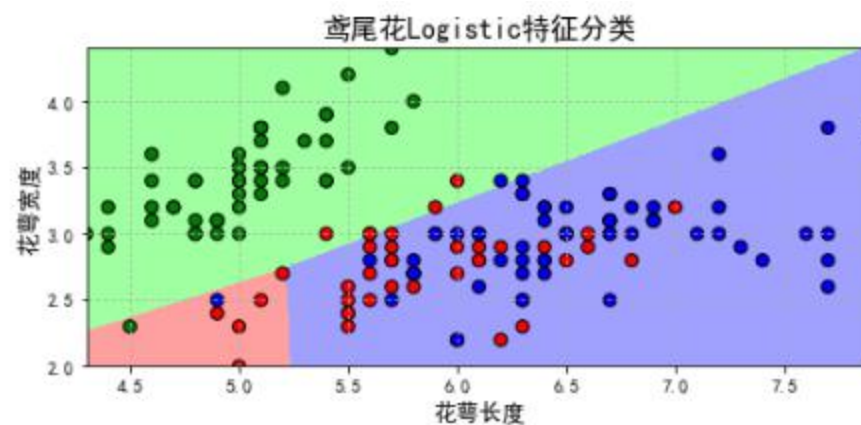
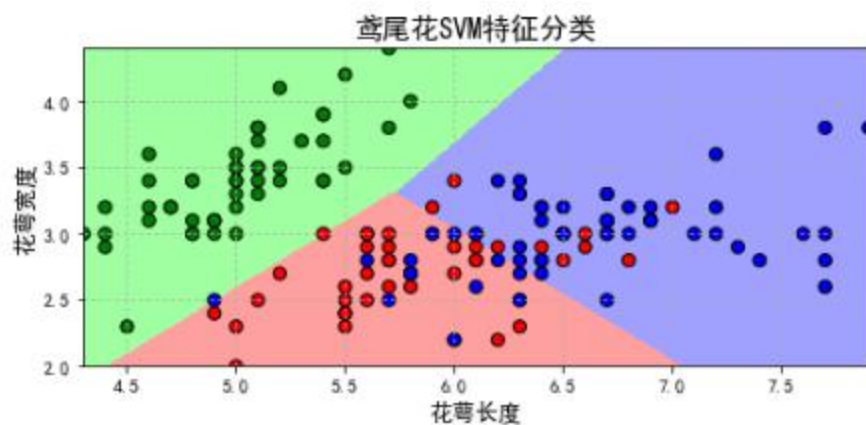


案例二：鸢尾花数据不同分类器效果比较

- 使用线性SVM、 LogisticRegression、 RidgeClassifier、 knn对鸢尾花数据进行分类效果对比



案例二：鸢尾花数据不同分类器效果比较



案例三：不同SVM核函数效果比较

- 使用SVM不同核函数对鸢尾花数据进行分类操作

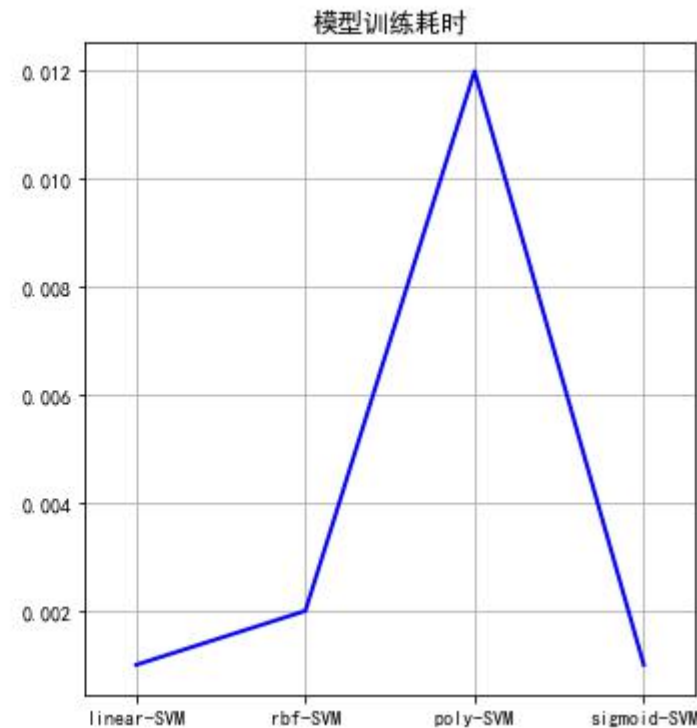
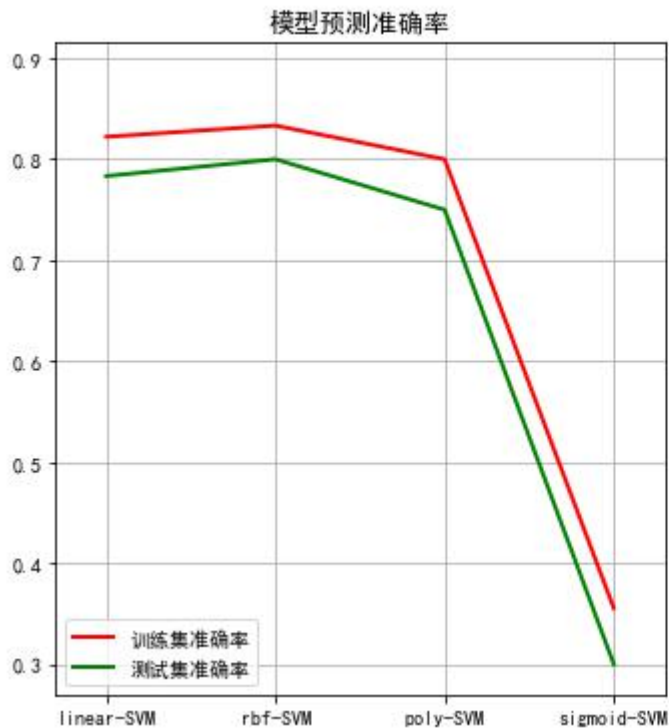
```
svm1 = SVC(C=1, kernel='linear')
```

```
svm2 = SVC(C=1, kernel='rbf')
```

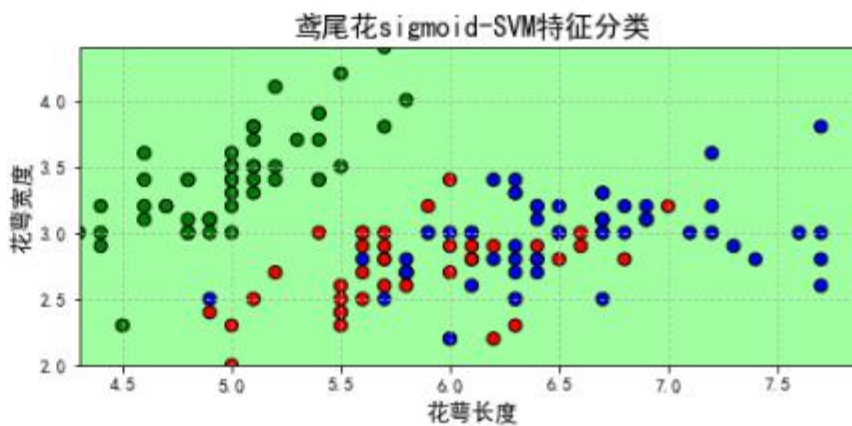
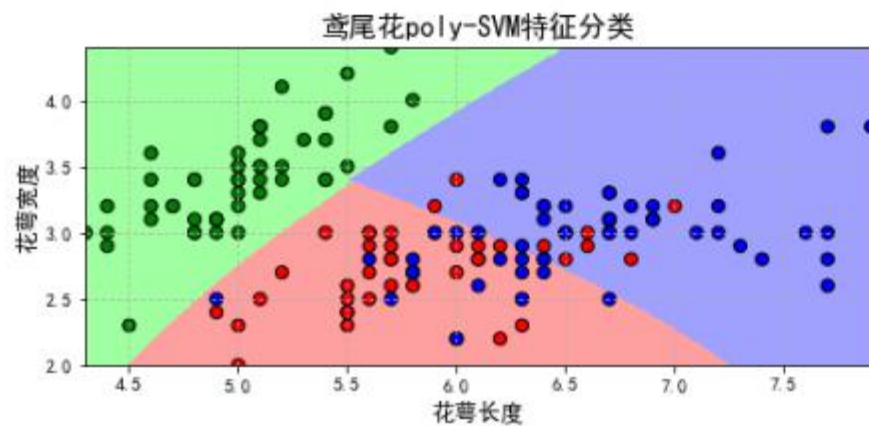
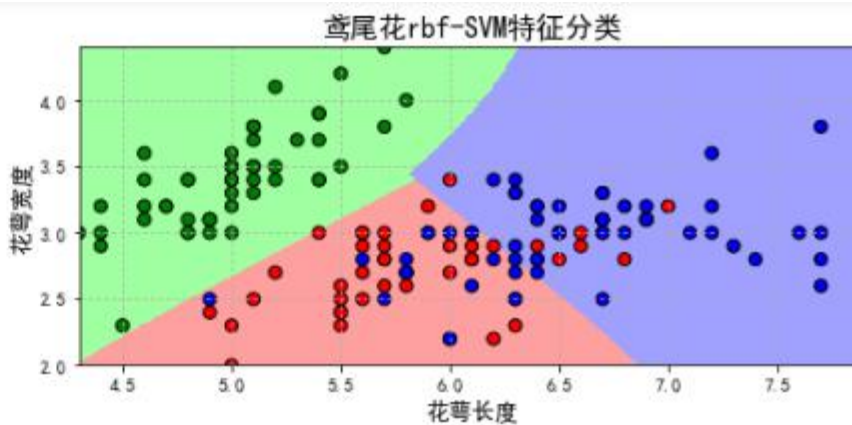
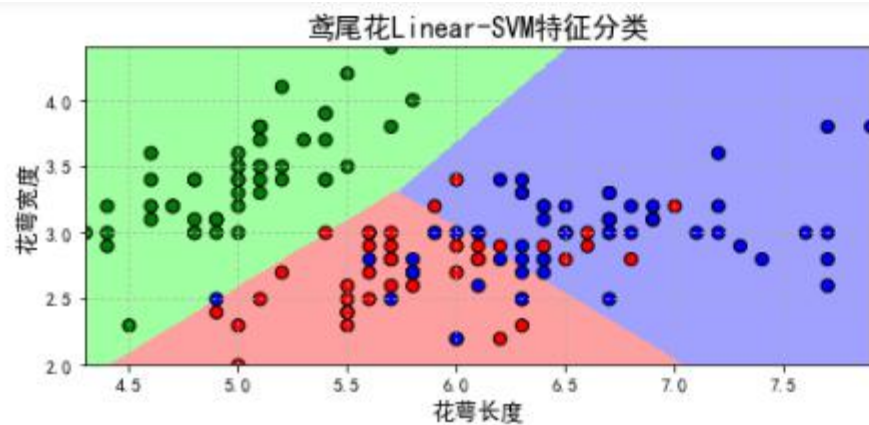
```
svm3 = SVC(C=1, kernel='poly')
```

```
svm4 = SVC(C=1, kernel='sigmoid')
```

鸢尾花数据SVM分类器不同内核函数模型比较



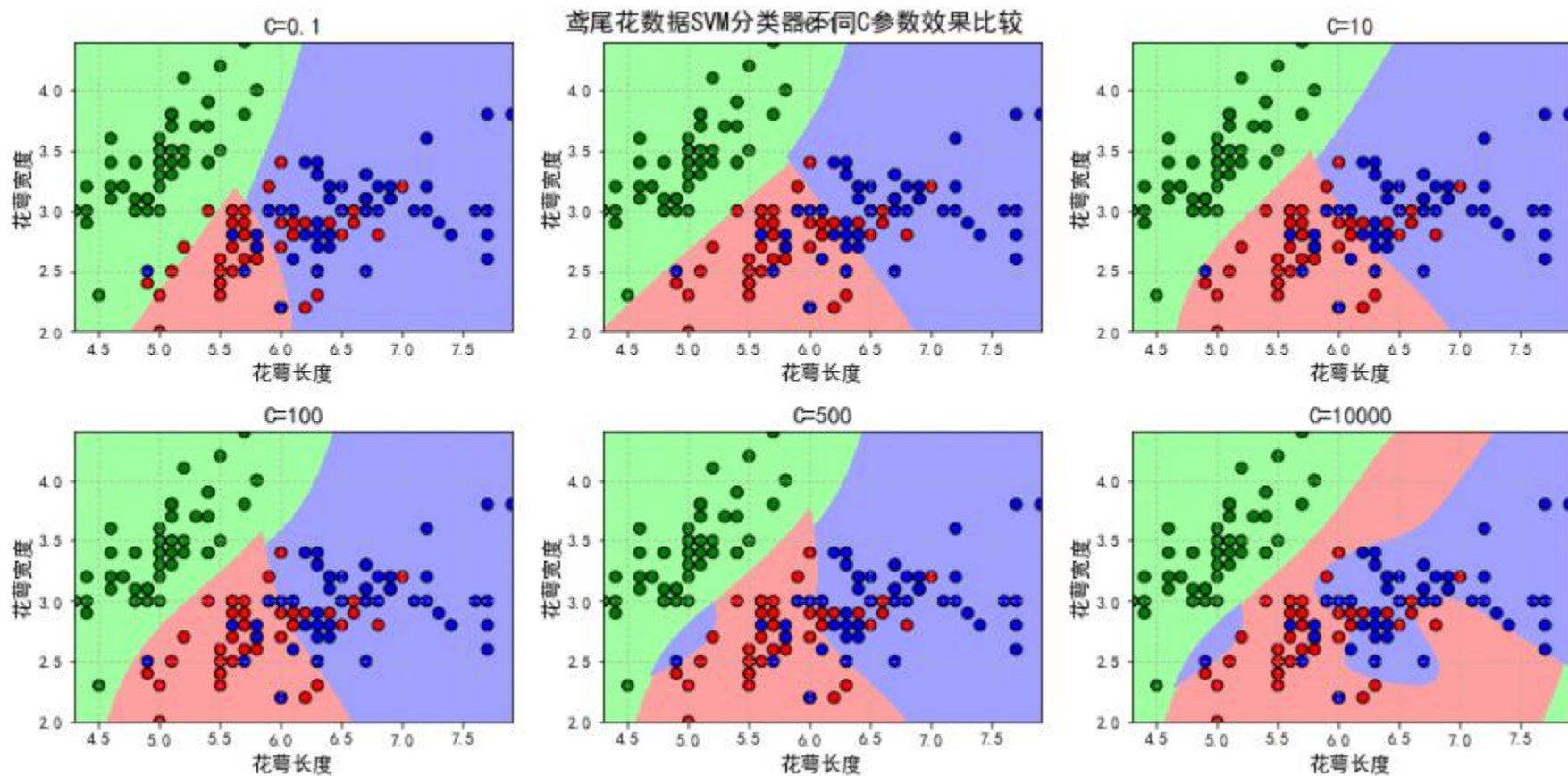
案例三：不同SVM核函数效果比较



案例四：不同SVM惩罚参数C值不同效果比较

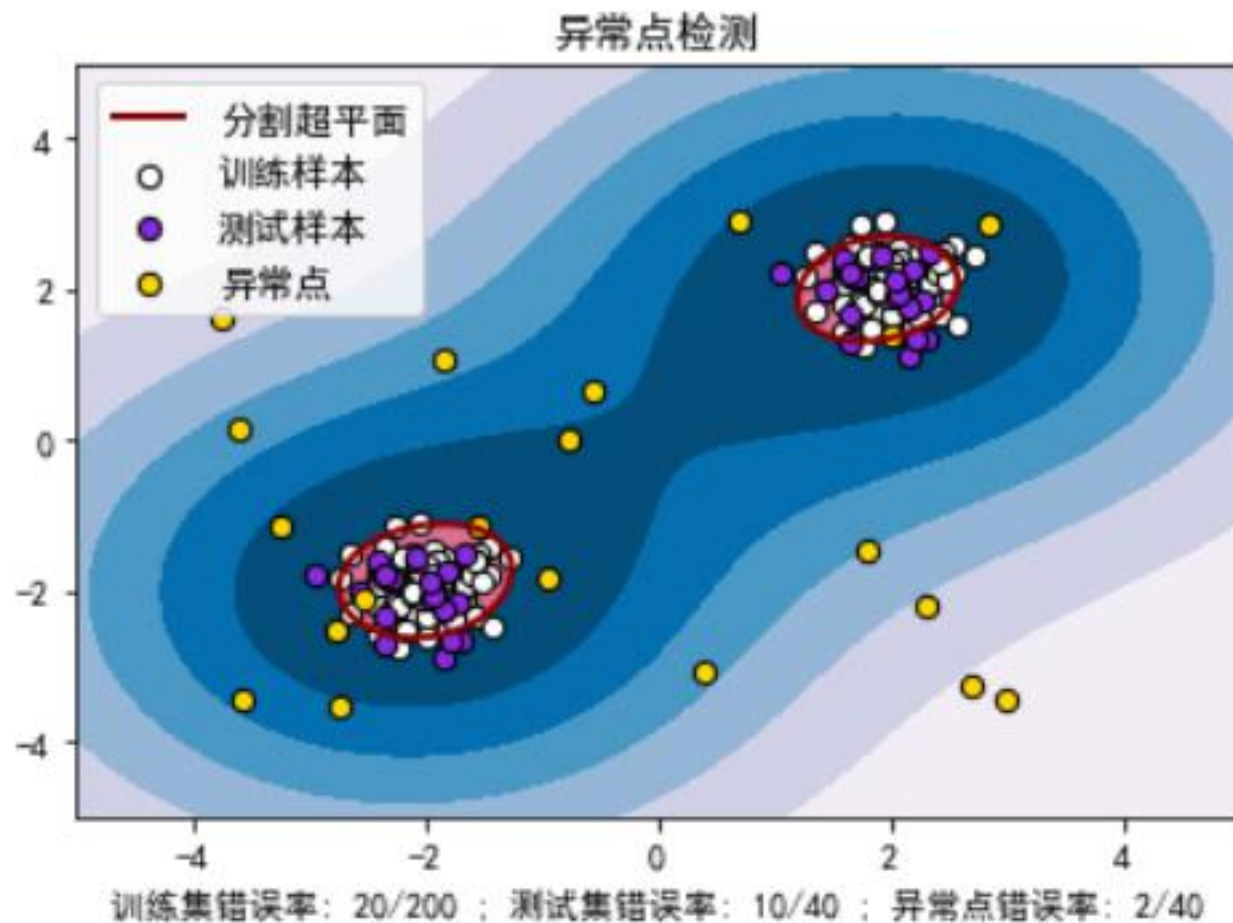
- 使用SVM惩罚参数C值效果对比

```
svm1 = SVC(C=0.1, kernel='rbf')
svm2 = SVC(C=1, kernel='rbf')
svm3 = SVC(C=10, kernel='rbf')
svm4 = SVC(C=100, kernel='rbf')
svm5 = SVC(C=500, kernel='rbf')
svm6 = SVC(C=100000, kernel='rbf')
```



案例五：异常值检测

- 使用OneClassSVM实现异常值检测的功能；



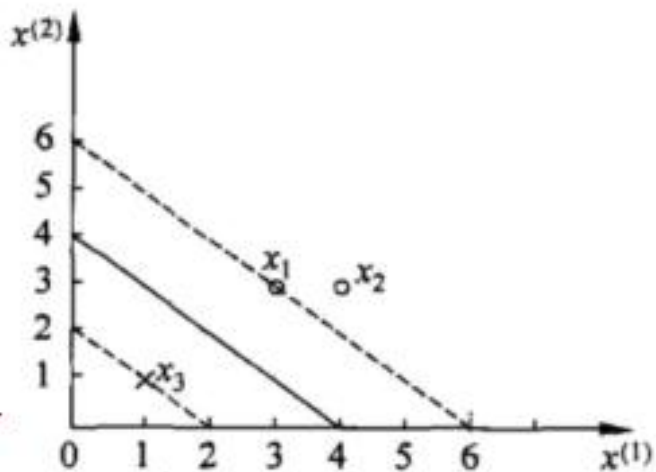
线性可分SVM案例

$$\min \frac{1}{2} \sum_{i=1, j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)} - \sum_{i=1}^m \beta_i$$

$$s.t : \sum_{i=1}^m \beta_i y^{(i)} = 0$$

$$\beta_i \geq 0, i = 1, 2, \dots, m$$

- 给定三个数据点：正例点 $x_1=(3,3), x_2=(4,3)$ ，负例点 $x_3=(1,1)$ ，构造此时的约束优化条件。



$$\frac{1}{2} (18\beta_1^2 + 25\beta_2^2 + 2\beta_3^2 + 42\beta_1\beta_2 - 12\beta_1\beta_3 - 14\beta_2\beta_3) - \beta_1 - \beta_2 - \beta_3$$

$$s.t : \beta_1 + \beta_2 - \beta_3 = 0$$

$$\beta_i \geq 0, i = 1, 2, \dots, m$$

- **序列最小优化算法**(Sequential minimal optimization, **SMO**)是一种用于解决SVM训练过程中所产生的优化问题的算法。于1998年由John Platt发明, 论文详见: [Sequential Minimal Optimization-a Fast Alg for Training SVM.pdf](#)

$$\min_{\beta} \frac{1}{2} \sum_{i=1, j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} K(x^{(i)}, x^{(j)}) - \sum_{i=1}^m \beta_i$$
$$s.t: \sum_{i=1}^m \beta_i y^{(i)} = 0$$
$$0 \leq \beta_i \leq C, i = 1, 2, \dots, m$$

- 假定存在一个 $\beta^*=(\beta_1,\beta_2,\dots,\beta_m)$ 是我们最终的最优解，那么根据KKT条件我们可以计算出 w 和 b 的最优解，如下：

$$w = \sum_{i=1}^m \beta_i y^{(i)} x^{(i)} \quad b = y - \sum_{i=1}^m \beta_i y^{(i)} K(x^{(i)}, x)$$

- 进而我们可以得到最终的分离超平面为：

$$g(x) = wx + b = \sum_{i=1}^m \beta_i y^{(i)} K(x^{(i)}, x) + b$$

SMO

- 拉格朗日乘子法和KKT的对偶互补条件为：

$$\beta_i(1 - \xi_i - y^{(i)}(w^T x^{(i)} + b)) = 0 \quad \mu_i(-\xi_i) = 0$$

- β 、 μ 和C之间的关系为： $C - \beta_i - \mu_i = 0$
- 根据这个对偶互补条件，我们有如下关系式：

$$\beta_i = 0 \Rightarrow \mu_i > 0 \Rightarrow \xi_i = 0 \Rightarrow y^{(i)}(w^T x^{(i)} + b) > 1 \quad \Rightarrow$$

$$0 < \beta_i < C \Rightarrow \mu_i > 0 \Rightarrow \xi_i = 0 \Rightarrow y^{(i)}(w^T x^{(i)} + b) = 1$$

$$\beta_i = C \Rightarrow \mu_i = 0 \Rightarrow \xi_i > 0 \Rightarrow y^{(i)}(w^T x^{(i)} + b) < 1$$
$$y^{(i)}g(x^{(i)}) = \begin{cases} > 1, \{(x^{(i)}, y^{(i)}) | \beta_i = 0\} \\ = 1, \{(x^{(i)}, y^{(i)}) | 0 < \beta_i < C\} \\ < 1, \{(x^{(i)}, y^{(i)}) | \beta_i = C\} \end{cases}$$

- 也就是说我们找出的最优的分割超平面必须满足下列的**目标条件**

(g(x)):

$$y^{(i)} g(x^{(i)}) = \begin{cases} > 1, \{(x^{(i)}, y^{(i)}) | \beta_i = 0\} \\ = 1, \{(x^{(i)}, y^{(i)}) | 0 < \beta_i < C\} \\ < 1, \{(x^{(i)}, y^{(i)}) | \beta_i = C\} \end{cases}$$

- 拉格朗日对偶化要求的两个**限制的初始条件**为:

$$\sum_{i=1}^m \beta_i y^{(i)} = 0$$

$$0 \leq \beta_i \leq C, i = 1, 2, \dots, m$$

- 从而可以得到解决问题的思路如下：
 - 首先，初始化后一个 β 值，让它满足对偶问题的两个**初始限制条件**；
 - 然后不断优化这个 β 值，使得由它确定的分割超平面满足 **$g(x)$ 目标条件**；
而且在优化过程中，始终保证 β 值满足**初始限制条件**。
 - 备注：这个求解过程中，和传统的思路不太一样，不是对目标函数求最小值，而是让 **$g(x)$ 目标条件**尽可能的满足。

- 在这样一个过程中，到底如何优化这个 β 值呢？？？整理可以发现 β 值的优化必须遵循以下两个基本原则：
 - 每次优化的时候，必须同时优化 β 的两个分量；因为如果只优化一个分量的话，新的 β 值就没法满足**初始限制条件**中的**等式约束条件**了。
 - 每次优化的两个分量应该是违反 **$g(x)$ 目标条件**比较多的。也就是说，本来应当是大于等于1的，越是小于1违反 **$g(x)$ 目标条件**就越多。
- 或者换一种思路来理解，因为目标函数中存在 m 个变量，直接优化比较难，利用启发式的方法/EM算法的思想，每次优化的时候，只优化两个变量，将其它的变量看成常数项，这样SMO算法就将一个复杂的优化算法转换为一个比较简单的两变量优化问题了。

SMO

$$\begin{aligned} \min & \frac{1}{2} \sum_{i=1, j=1}^m \beta_i \beta_j y^{(i)} y^{(j)} K(x^{(i)}, x^{(j)}) - \sum_{i=1}^m \beta_i \\ \text{s.t.} & \sum_{i=1}^m \beta_i y^{(i)} = 0 \\ & 0 \leq \beta_i \leq C, i = 1, 2, \dots, m \end{aligned} \quad K_{ij} = K(x^{(i)}, x^{(j)})$$

- 认为 β_1 、 β_2 是变量，其它 β 值是常量，从而将目标函数转换如下
(C是常数项):

$$\begin{aligned} \min_{\beta_1, \beta_2} & \frac{1}{2} K_{11} \beta_1^2 + \frac{1}{2} K_{22} \beta_2^2 + K_{12} y^{(1)} y^{(2)} \beta_1 \beta_2 - \beta_1 - \beta_2 + y^{(1)} \beta_1 \sum_{i=3}^m y^{(i)} \beta_i K_{i1} + y^{(2)} \beta_2 \sum_{i=3}^m y^{(i)} \beta_i K_{i2} + C \\ \text{s.t.} & \beta_1 y^{(1)} + \beta_2 y^{(2)} = - \sum_{i=3}^m \beta_i y^{(i)} = k \\ & 0 \leq \beta_i \leq C, i = 1, 2 \end{aligned}$$

- 由于 $\beta_1 y^{(1)} + \beta_2 y^{(2)} = k$, 并且 $y^2 = 1$, 也就是我们使用 β_2 来表示 β_1 的值:

$$\beta_1 = y^{(1)}(k - \beta_2 y^{(2)})$$

- 将上式带入目标优化函数, 就可以消去 β_1 , 从而只留下仅仅包含 β_2 的式子。 令:
$$v_j = \sum_{i=3}^m y^{(i)} \beta_i K_{ij}$$

$$W(\beta_2) = \frac{1}{2} K_{11} (k - \beta_2 y^{(2)})^2 + \frac{1}{2} K_{22} \beta_2^2 + K_{12} y^{(2)} (k - \beta_2 y^{(2)}) \beta_2 \\ - y^{(1)} (k - \beta_2 y^{(2)}) - \beta_2 + (k - \beta_2 y^{(2)}) v_1 + y^{(2)} \beta_2 v_2$$

SMO 求解 β_2 的值

$$g(x) = \sum_{i=1}^m \beta_i y^{(i)} K(x^{(i)}, x) + b$$

$$v_j = \sum_{i=1}^m y^{(i)} \beta_i K_{ij}$$

$$\frac{\partial W}{\partial \beta_2} = K_{11} \beta_2 - K_{11} k y^{(2)} + K_{22} \beta_2 + K_{12} k y^{(2)} - 2K_{12} \beta_2 + y^{(1)} y^{(2)} - 1 - y^{(2)} v_1 + y^{(2)} v_2$$

$$\xrightarrow{\text{令 } \frac{\partial W}{\partial \beta_2} = 0}$$

$$(K_{11} + K_{22} - 2K_{12}) \beta_2 = y^{(2)} (y^{(2)} - y^{(1)} + kK_{11} - kK_{12} + v_1 - v_2)$$

$$= y^{(2)} \left(y^{(2)} - y^{(1)} + kK_{11} - kK_{12} + \left(g(x_1) - \sum_{i=1}^2 \beta_i y^{(i)} K_{i1} - b \right) - \left(g(x_2) - \sum_{i=1}^2 \beta_i y^{(i)} K_{i2} - b \right) \right)$$

SMO 求解 β_2 的值

将 $k = \beta_1 y^{(1)} + \beta_2 y^{(2)}$
带入上页ppt中式子



$$\begin{aligned} & (K_{11} + K_{22} - 2K_{12})\beta_2 = \\ & = y^{(2)} \left(y^{(2)} - y^{(1)} + \beta_1^{old} y^{(1)} K_{11} + \beta_2^{old} y^{(2)} K_{11} - \beta_1^{old} y^{(1)} K_{12} - \beta_2^{old} y^{(2)} K_{12} \right) \\ & + y^{(2)} \left(g(x_1) - \sum_{i=1}^2 \beta_i^{old} y^{(i)} K_{i1} - b \right) - y^{(2)} \left(g(x_2) - \sum_{i=1}^2 \beta_i^{old} y^{(i)} K_{i2} - b \right) \\ & = y^{(2)} \left((K_{11} - 2K_{12} + K_{22}) \beta_2^{old} y^{(2)} + y^{(2)} - y^{(1)} + g(x^{(1)}) - g(x^{(2)}) \right) \\ & = (K_{11} - 2K_{12} + K_{22}) \beta_2^{old} + y^{(2)} \left[(g(x^{(1)}) - y^{(1)}) - (g(x^{(2)}) - y^{(2)}) \right] \end{aligned}$$

SMO算法

$$\begin{aligned} (K_{11} + K_{22} - 2K_{12})\beta_2 &= \\ &= (K_{11} - 2K_{12} + K_{22})\beta_2^{old} + y^{(2)} \left[(g(x^{(1)}) - y^{(1)}) - (g(x^{(2)}) - y^{(2)}) \right] \end{aligned}$$

$$\xrightarrow{\text{令 } E_i = g(x^{(i)}) - y^{(i)}} \rightarrow$$

$$\beta_2^{new,unt} = \beta_2 = \beta_2^{old} + \frac{y^{(2)}(E_1 - E_2)}{K_{11} + K_{22} - 2K_{12}}$$

$$\beta_1 y^{(1)} + \beta_2 y^{(2)} = - \sum_{i=3}^m \beta_i y^{(i)} = k$$

$$0 \leq \beta_i \leq C, i = 1, 2$$

- 考虑 β_1 和 β_2 的取值限定范围，假定新求出来的 β 值是满足我们的边界限制的，即如下所示： $L \leq \beta_2^{new} \leq H$
- 当 $y_1=y_2$ 的时候， $\beta_1+\beta_2=k$ ；由于 β 的限制条件，我们可以得到：

$$\begin{cases} 0 \leq k - \beta_2^{new} \leq C \\ 0 \leq \beta_2^{new} \leq C \end{cases} \Rightarrow \begin{cases} L = \max(0, \beta_2^{old} + \beta_1^{old} - C) \\ H = \min(C, \beta_2^{old} + \beta_1^{old}) \end{cases}$$

- 当 $y_1 \neq y_2$ 的时候， $\beta_1 - \beta_2 = k$ ；由于 β 的限制条件，我们可以得到：

$$\begin{cases} 0 \leq k + \beta_2^{new} \leq C \\ 0 \leq \beta_2^{new} \leq C \end{cases} \Rightarrow \begin{cases} L = \max(0, \beta_2^{old} - \beta_1^{old}) \\ H = \min(C, C + \beta_2^{old} - \beta_1^{old}) \end{cases}$$

- 结合 β 的取值限制范围以及函数 W 的 β 最优解，我们可以得带迭代

过程中的最优解为：

$$\beta_2^{new} = \begin{cases} H; \beta_2^{new,unt} > H \\ \beta_2^{new,unt}; L \leq \beta_2^{new,unt} \leq H \\ L; \beta_2^{new,unt} < L \end{cases}$$

- 然后根据 β_1 和 β_2 的关系，从而可以得到迭代后的 β_1 的值：

$$\begin{aligned} \beta_1^{old} y^{(1)} + \beta_2^{old} y^{(2)} &= \beta_1^{new} y^{(1)} + \beta_2^{new} y^{(2)} \\ \Rightarrow \beta_1^{new} &= \beta_1^{old} + y^{(1)} y^{(2)} (\beta_2^{old} - \beta_2^{new}) \end{aligned}$$

SMO

- 求解 β 的过程中，相关公式如下：

$$K_{ij} = K(x^{(i)}, x^{(j)}) \quad E_i = g(x^{(i)}) - y^{(i)} = \sum_{j=1}^m \beta_j y^{(j)} K_{ij} + b - y^{(i)}$$

$$\beta_2^{new,unt} = \beta_2^{old} + \frac{y^{(2)}(E_1 - E_2)}{K_{11} + K_{22} - 2K_{12}}$$

$$\beta_2^{new} = \begin{cases} H; \beta_2^{new,unt} > H \\ \beta_2^{new,unt}; L \leq \beta_2^{new,unt} \leq H \\ L; \beta_2^{new,unt} < L \end{cases}$$

$$\beta_1^{new} = \beta_1^{old} + y^{(1)} y^{(2)} (\beta_2^{old} - \beta_2^{new})$$

- 可以发现SMO算法中，是选择两个合适的 β 变量做迭代，其它变量作为常量来进行优化的一个过程，那么这两个变量到底怎么选择呢???
- 每次优化的时候，必须同时优化 β 的两个分量；因为如果只优化一个分量的话，新的 β 值就没法满足**初始限制条件**中的**等式约束条件**了。
- 每次优化的两个分量应该是违反 **$g(x)$ 目标条件**比较多的。也就是说，本来应当是大于等于1的，越是小于1违反 **$g(x)$ 目标条件**就越多。

SMO

第一个 β 变量的选择

- SMO算法在选择第一个 β 变量的时候，需要选择在训练集上违反KKT条件最严重的样本点。一般情况下，先选择 $0 < \beta < C$ 的样本点(即支持向量)，只有当所有的支持向量都满足KKT条件的时候，才会选择其它样本点。因为此时违反KKT条件越严重，在经过一次优化后，会让变量 β 尽可能的发生变化，从而可以以更少的迭代次数让模型达到 **$g(x)$ 目标条件**。

$$y^{(i)}g(x^{(i)}) = \begin{cases} > 1, \{(x^{(i)}, y^{(i)}) \mid \beta_i = 0\} \\ = 1, \{(x^{(i)}, y^{(i)}) \mid 0 < \beta_i < C\} \\ < 1, \{(x^{(i)}, y^{(i)}) \mid \beta_i = C\} \end{cases}$$

- 在选择第一个变量 β_1 后，在选择第二个变量 β_2 的时候，希望能够按照优化后的 β_1 和 β_2 有尽可能多的改变来选择，也就是说让 $|E_1 - E_2|$ 足够的大，当 E_1 为正的时候，选择最小的 E_i 作为 E_2 ；当 E_1 为负的时候，选择最大的 E_i 作为 E_2 。
- 备注：如果选择的第二个变量不能够让目标函数有足够的下降，那么可以通过遍历所有样本点来作为 β_2 ，直到目标函数有足够的下降，如果都没有足够的下降的话，那么直接跳出循环，重新选择 β_1 ；

- 在每次完成两个 β 变量的优化更新之后，需要重新计算阈值b和差值

E_i 。当 $0 < \beta_1^{new} < C$ 时，有：
$$y^{(1)} - \sum_{i=1}^m \beta_i y^{(i)} K_{i1} - b_1 = 0$$

- 化简可得：

$$b_1^{new} = y^{(1)} - \sum_{i=3}^m \beta_i y^{(i)} K_{i1} - \beta_1^{new} y^{(1)} K_{11} - \beta_2^{new} y^{(2)} K_{12}$$

$$\because E_1 = g(x^{(1)}) - y^{(1)} = \sum_{j=3}^m \beta_j y^{(j)} K_{j1} + \beta_1^{old} y^{(1)} K_{11} + \beta_2^{old} y^{(2)} K_{12} + b^{old} - y^{(1)}$$

$$\therefore b_1^{new} = -E_1 - y^{(1)} K_{11} (\beta_1^{new} - \beta_1^{old}) - y^{(2)} K_{12} (\beta_2^{new} - \beta_2^{old}) + b^{old}$$

SMO

计算阈值b和差值E_i

$$b_1^{new} = -E_1 - y^{(1)}K_{11}(\beta_1^{new} - \beta_1^{old}) - y^{(2)}K_{12}(\beta_2^{new} - \beta_2^{old}) + b^{old}$$

- 同样的当 β_2 的取值为: $0 < \beta_2 < C$ 的时候, 我们也可以得到

$$b_2^{new} = -E_2 - y^{(1)}K_{12}(\beta_1^{new} - \beta_1^{old}) - y^{(2)}K_{22}(\beta_2^{new} - \beta_2^{old}) + b^{old}$$

- 最终计算出来的b为:

$$b^{new} = \frac{b_1^{new} + b_2^{new}}{2}$$

- 当更新计算阈值b后，就可以得到差值 E_i 为：

$$E_i = g(x^{(i)}) - y^{(i)} = \sum_{j=3}^m \beta_j y^{(j)} K_{ij} + \beta_1^{new} y^{(1)} K_{11} + \beta_2^{new} y^{(2)} K_{12} + b^{new} - y^{(1)}$$

SMO算法流程总结

- 输入线性可分的m个样本数据 $\{(x^1, y^1), (x^2, y^2), \dots, (x^m, y^m)\}$, 其中x为n维的特征向量, y为二元输出, 取值为+1或者-1; 精度为e。
- 取初值 $\beta^0=0$, $k=0$;
- 选择需要进行更新的两个变量: β_1^k 和 β_2^k , 计算出来新的 $\beta_2^{new, unt}$;

$$\beta_2^{new, unt} = \beta_2^k + \frac{y^{(2)}(E_1 - E_2)}{K_{11} + K_{22} - 2K_{12}}$$

- 按照下列式子求出具体的 β_2^{k+1} ;

$$\beta_2^{k+1} = \begin{cases} H; \beta_2^{new, unt} > H \\ \beta_2^{new, unt}; L \leq \beta_2^{new, unt} \leq H \\ L; \beta_2^{new, unt} < L \end{cases}$$

SMO算法流程总结

- 按照 β_1^k 和 β_2^k 的关系, 求出 β_1^{k+1} 的值:

$$\beta_1^{k+1} = \beta_1^k + y^{(1)}y^{(2)}(\beta_2^k - \beta_2^{k+1})$$

- 按照公式计算 b^{k+1} 和 E_i 的值;

$$b^{new} = \frac{b_1^{new} + b_2^{new}}{2}$$

$$E_i = g(x^{(i)}) - y^{(i)} = \sum_{j=3}^m \beta_j y^{(j)} K_{ij} + \beta_1^{new} y^{(1)} K_{11} + \beta_2^{new} y^{(2)} K_{12} + b^{new} - y^{(1)}$$

- 检查函数 $y^{(i)} * E_i$ 的绝对值是否在精度范围内, 并且求解出来的 β 解满足KKT相关约束条件, 那么此时结束循环, 返回此时的 β 解即可, 否则继续迭代计算 $\beta_2^{new,unt}$ 的值。

SVR

- SVM和决策树一样，可以将模型直接应用到回归问题中；在SVM的分类模型

(SVC)中，目标函数和限制条件如下：
$$\min_{w,b} \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^m \xi_i$$

$$s.t : y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, i = 1, 2, \dots, m$$

$$\xi_i \geq 0, i = 1, 2, \dots, m$$

- 在SVR中，目的是为了尽量拟合一个线性模型 $y=wx+b$ ；从而我们可以定义常量 $\epsilon > 0$ ，对于任意一点 (x,y) ，如果 $|y-wx-b| \leq \epsilon$ ，那么认为没有损失，从而我们可以得到目标函数和限制条件如下：

$$\min_{w,b} \frac{1}{2} \|w\|_2^2$$

$$s.t : |y^{(i)} - w^T x^{(i)} - b| \leq \epsilon, i = 1, 2, \dots, m$$

- 加入松弛因子 $\xi > 0$, 从而我们的目标函数和限制条件变成:

$$\min_{w,b} \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^m (\xi_i^\vee + \xi_i^\wedge)$$

$$s.t : -\varepsilon - \xi_i^\vee \leq y^{(i)} - w^T x^{(i)} - b \leq \varepsilon + \xi_i^\wedge, i = 1, 2, \dots, m$$

$$\xi_i^\wedge \geq 0, \xi_i^\vee \geq 0, i = 1, 2, \dots, m$$

- 构造拉格朗日函数:

$$\begin{aligned} L(w, b, \xi^{\vee}, \xi^{\wedge}, \beta^{\vee}, \beta^{\wedge}, \mu^{\vee}, \mu^{\wedge}) = & \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^m (\xi_i^{\vee} + \xi_i^{\wedge}) + \sum_{i=1}^m \beta_i^{\vee} [-\varepsilon - \xi_i^{\vee} - y^{(i)} + w^T x^{(i)} + b] \\ & + \sum_{i=1}^m \beta_i^{\wedge} [y^{(i)} - w^T x^{(i)} - b - \varepsilon - \xi_i^{\wedge}] - \sum_{i=1}^m \mu_i^{\vee} \xi_i^{\vee} - \sum_{i=1}^m \mu_i^{\wedge} \xi_i^{\wedge} \end{aligned}$$

$$s.t : \beta_i^{\vee} \geq 0, \beta_i^{\wedge} \geq 0, \mu_i^{\vee} \geq 0, \mu_i^{\wedge} \geq 0, i = 1, 2, \dots, m$$

- 拉格朗日函数对偶化

$$\min_{w, b, \xi^{\vee}, \xi^{\wedge}} \max_{\beta^{\wedge}, \beta^{\vee}, \mu^{\wedge}, \mu^{\vee}} L(w, b, \xi^{\vee}, \xi^{\wedge}, \beta^{\vee}, \beta^{\wedge}, \mu^{\vee}, \mu^{\wedge})$$

$$\max_{\beta^{\wedge}, \beta^{\vee}, \mu^{\wedge}, \mu^{\vee}} \min_{w, b, \xi^{\wedge}, \xi^{\vee}} L(w, b, \xi^{\vee}, \xi^{\wedge}, \beta^{\vee}, \beta^{\wedge}, \mu^{\vee}, \mu^{\wedge})$$

- 首先来求优化函数对于 w 、 b 、 ξ 的极小值，通过求导可得：

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^m (\beta_i^{\wedge} - \beta_i^{\vee}) x^{(i)}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^m (\beta_i^{\wedge} - \beta_i^{\vee}) = 0$$

$$\frac{\partial L}{\partial \xi_i^{\vee}} = 0 \Rightarrow C - \beta_i^{\vee} - \mu_i^{\vee} = 0$$

$$\frac{\partial L}{\partial \xi_i^{\wedge}} = 0 \Rightarrow C - \beta_i^{\wedge} - \mu_i^{\wedge} = 0$$

- 将 w 、 b 、 ξ 的值带入函数 L 中，就可以将 L 转换为只包含 β 的函数，从而我们可以得到最终的优化目标函数为(备注：对于 β 的求解照样可以使用SMO算法来求解)：

$$\min_{\beta^{\wedge}, \beta^{\vee}} \frac{1}{2} \sum_{i=1, j=1}^m (\beta_i^{\wedge} - \beta_i^{\vee}) (\beta_j^{\wedge} - \beta_j^{\vee}) K_{ij} + \sum_{i=1}^m (\varepsilon - y^{(i)}) \beta_i^{\wedge} + (\varepsilon + y^{(i)}) \beta_i^{\vee}$$

$$s.t : \sum_{i=1}^m (\beta_i^{\wedge} - \beta_i^{\vee}) = 0$$

$$0 < \beta_i^{\wedge} < C, i = 1, 2, \dots, m$$

$$0 < \beta_i^{\vee} < C, i = 1, 2, \dots, m$$

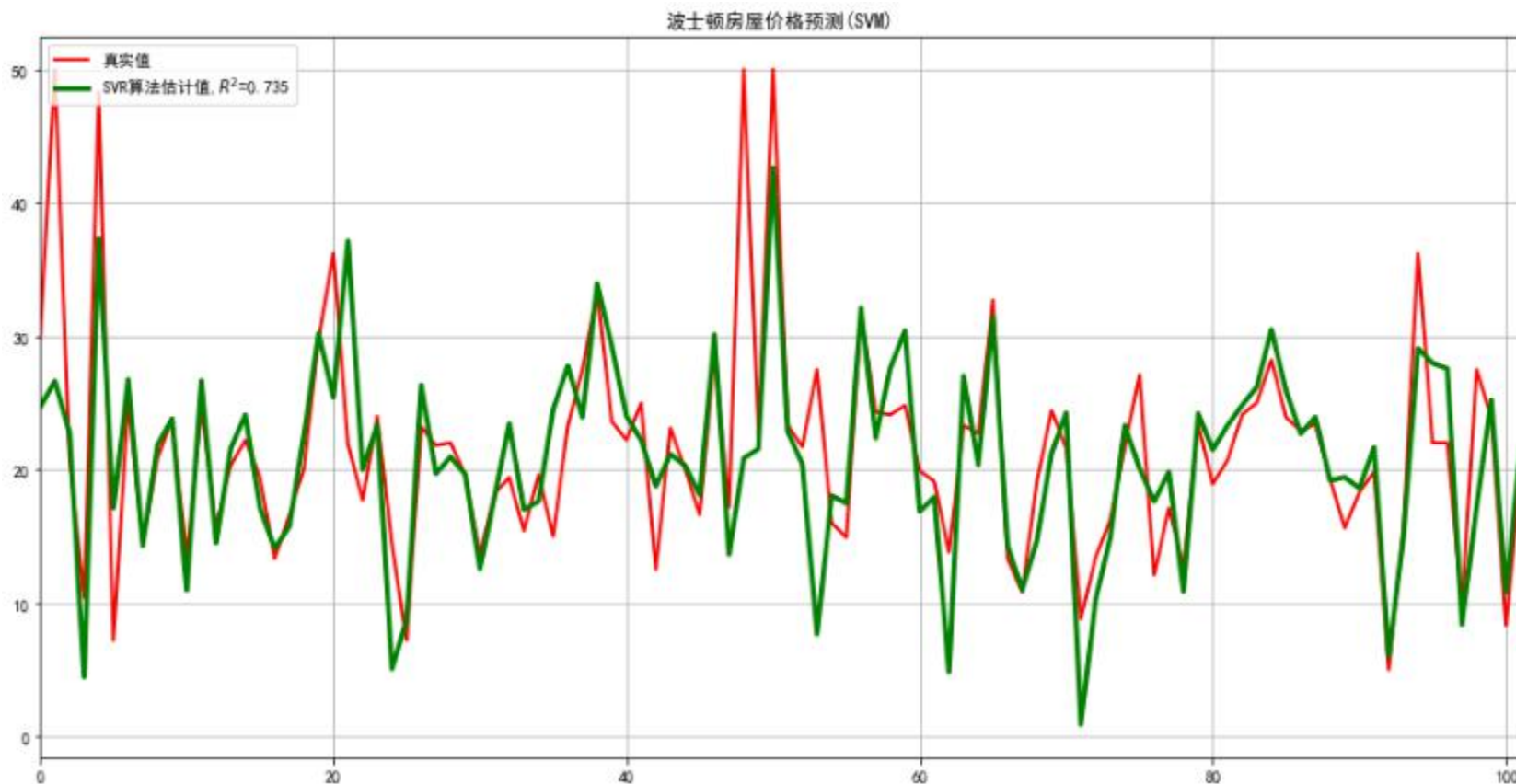
scitit-learn SVM算法库概述

回归算法

参数	LinearSVR	SVR	NuSVR
C	惩罚项系数，默认为1，一般通过交叉验证给定适合值		
nu	不支持，使用epsilon代替		表示训练集上错误率的上限，或者说是支持向量的百分比下限；默认为0.5，取值范围(0,1]，功能和epsilon类似
epsilon	回归模型中引入的误差epsilon值		不支持该，使用nu代替
kernel	不支持，只能使用线性核函数		给定具体的核函数或者Gram矩阵，参数可选:linear、rbf、poly、sigmoid、precomputed
penalty	对线性拟合有意义，参数可选l1和l2		不支持
degree、gamma、coef0	不支持		核函数中的参数值，一般需要通过交叉验证给定；和分类模型一样
loss	可选"epsilon_insensitive"和"squared_epsilon_insensitive"; 第一个表示我们的损坏度量使用绝对值，第二个表示我们的损坏度量使用平方来表示，一般选择epsilon_insensitive即可		不支持

综合案例3：使用SVM预测波士顿房价

- 使用SVM中SVR对波士顿房价预测，熟悉SVM的回归问题应用



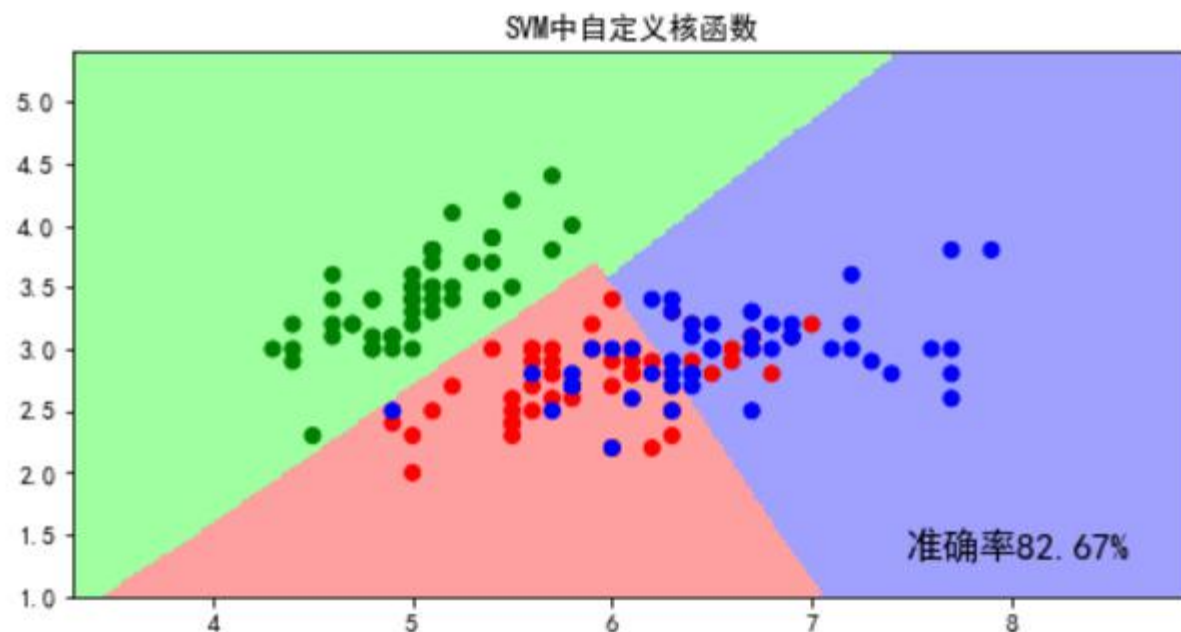
综合案例1：手写数字识别

- 使用SVM对手写数字的识别



综合案例2：自定义SVM内部核函数

- 练习SVM自定义函数的应用



综合案例4：分类算法比较

- 比较逻辑回归、KNN、决策树、随机森林、GBDT、Adaboost、SVM等分类算法的效果，数据集使用sklearn自带的模拟数据进行测试。

