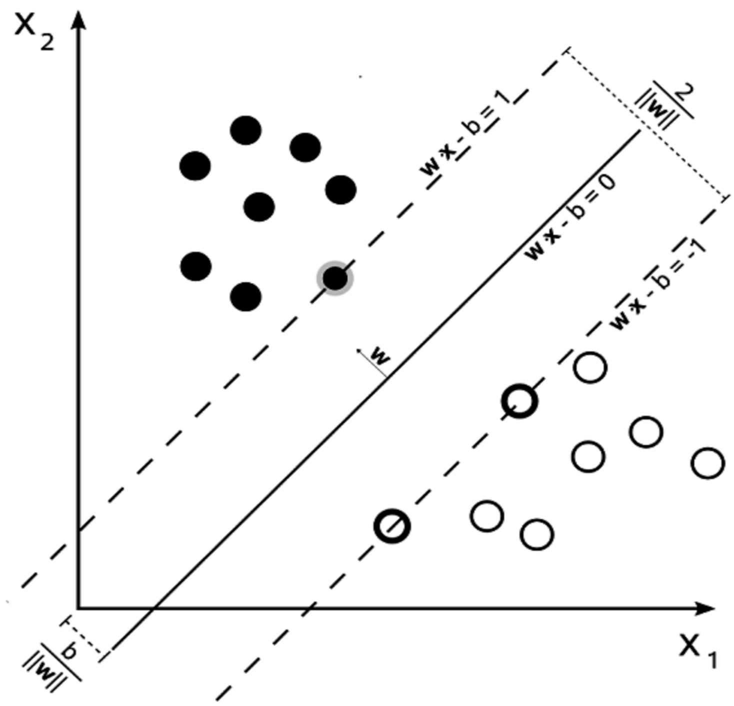
SVM 支持向量机

1 线性svm

1.1 目标函数

二维情况下的分类问题：



考虑多维情况的二分类问题，假设存在一个超平面能够将正负样本划分开，则所有的样本点满足以下条件：

$$\begin{aligned} w^T x_i + b \geq \ \ 1 \quad y_i = +1 \\ w^T x_i + b \leq -1 \quad y_i = -1 \end{aligned}$$

计算两个离超平面最近的正负样本的间隔距离，该样本点称为**支持向量**：

在n维空间中，分割超平面方程：

$$f(x) = w_1 x_1 + w_2 x_2 + w_3 x_3 \ldots w_n x_n + b$$

假设距离分割超平面最近距离的样本点是x_k，计算距离：

$$d = \frac{|w^{(1)} x_k^{(1)} + w^{(2)} x_k^{(2)} + w^{(3)} x_k^{(3)} \ldots . + w^{(n)} x_k^{(n)} + b|}{||\vec{w}||} = \frac{1}{||\vec{w}||}$$

$$\text{间隔距离} \ D = 2 * d = \frac{2}{||\vec{w}||}$$

因为$||\vec{w}|| > 0$ 所以可以 $||\vec{w}||^2$代替，方便计算。

SVM的目的是在能够满足上述不等式的情况下，使得间隔距离最大，使得分类效果最好：

$$\max_{(w,b)} \frac{2}{||\vec{w}||^2}$$
$$s.t. \ y_i(w^T x_i + b) \geq 1 \ , i = 1, 2, 3..n$$

转换成等价问题，得出目标函数：

$$\min_{(w,b)} \frac{1}{2} ||\vec{w}||^2$$

$$s.t. \ y_i(w^T x_i + b) - 1 \geq 0$$

该问题带有不等式约束，可由拉格朗日乘子法求解。

1.2 拉格朗日函数

运用拉格朗日函数将不等式约束引入到目标函数中，方便求解：

$$L(w,b,\alpha)=\frac{1}{2}||\vec{w}||^2+\sum_{i}^{N}\alpha_i(1-y_i(w^Tx_i+b)),\ \ \alpha_i\geq 0$$

$L(w,b,\alpha)$对α,β求极大值，就等价于原目标函数

$$\max_{\alpha} L(w,b,\alpha)=\begin{cases}\frac{1}{2}||\vec{w}||^2,(1-y_i(w^Tx_i+b))<0\\+\infty,\quad\ (1-y_i(w^Tx_i+b))\geq 0\end{cases}$$

$$=\begin{cases}\frac{1}{2}||\vec{w}||^2,(y_i(w^Tx_i+b)-1)\geq 0\\+\infty,\quad\ (y_i(w^Tx_i+b)-1)<0\end{cases}$$

---

1. 如果 $(1-y_i(w^Tx_i+b))<0$ ，那么 $\alpha_i(1-y_i(w^Tx_i+b))$ 一定小于等于0，为了使的$L(w,b,\alpha)$取得极大值，α_i 一定等于0，这种情况下$\max L(w,b,\alpha)=\frac{1}{2}||\vec{w}||^2$ ，且满足不等式约束。
2. 如果 $(1-y_i(w^Tx_i+b))\geq 0$ ，那么α_i可以取任意无限大的值使得$L(w,b,\alpha)$趋于无限大

再对上式求极小，可得：

$$\min_{w,b}\max_{\alpha} L(w,b,\alpha)=\min_{w,b}\begin{cases}\frac{1}{2}||\vec{w}||^2,(y_i(w^Tx_i+b)-1)\geq 0\\+\infty,\quad\ (y_i(w^Tx_i+b)-1)<0\end{cases}$$

$$=\min_{w,b}\ \frac{1}{2}||\vec{w}||^2,\quad (y_i(w^Tx_i+b)-1)\geq 0\quad (\text{原目标问题})$$

这样，就将带有不等式约束条件的优化问题，转换成了对函数$L(x,w,\alpha)$先求极大再求极小的问题，即目标问题转换成：

$$\text{目标问题等价于}:\ \min_{w,b}\max_{\alpha} L(w,b,\alpha)=\min_{w,b}\max_{\alpha}\left(\ \frac{1}{2}||\vec{w}||^2+\sum_{i}^{N}\alpha_i(1-y_i(w^Tx_i+b))\ \right),\ \ \alpha_i\geq 0$$

1.3 对偶问题和KKT条件

什么是对偶问题，通俗描述，就是将复杂问题转换成简单问题求解的方法

要想求解极值问题：

$$\min_{w,b}\max_{\alpha} L(w,b,\alpha)=\min_{w,b}\max_{\alpha}\left(\ \frac{1}{2}||\vec{w}||^2+\sum_{i}^{N}\alpha_i(1-y_i(w^Tx_i+b))\ \right),\ \ \alpha_i\geq 0\quad -----(1)$$

我们可以将问题转换成这样的形式求解，能够简化问题的求解过程(后续会说明SVM为什么使用对偶)：

$$\max_{\alpha}\min_{w,b} L(w,b,\alpha)=\max_{\alpha}\min_{w,b}\left(\ \frac{1}{2}||\vec{w}||^2+\sum_{i}^{N}\alpha_i(1-y_i(w^Tx_i+b))\ \right),\ \ \alpha_i\geq 0\quad -----(2)$$

假设(1)式的解为w^*,b^*,α^* ，式(2)的解为$w^{*d},b^{*d},\alpha^{*d}$ ，需证明

$$L(w^*,b^*,\alpha^*)=L(w^{*d},b^{*d},\alpha^{*d})$$

拉格朗日对偶性定理：

带不等式约束和等式约束的优化问题

$$\min_x f(x)$$
$$s.t. c_i(x) \le 0, h_j(x) = 0$$

其拉格朗日函数：

$$L(x,\alpha,\beta) = f(x) + \sum_i^m \alpha_i c_i(x) + \sum_j^n \beta_j h_j(x)$$

假设，原问题 $\max_{\alpha,\beta} \min_x L(x,\alpha,\beta)$ 最优解为x^*, α^*, β^*，对应的最优值为p^*

对偶问题 $\min_x \max_{\alpha,\beta} L(x,\alpha,\beta)$ 最优解为$x^{*d}, \alpha^{*d}, \beta^{*d}$，对应的最优值为$d^*$

定理1：

$$d^* = L(x^*,\alpha^*,\beta^*) \le L(x^{*d},\alpha^{*d},\beta^{*d}) = p^*$$

定理2:

如果 1. $f(x), c_i(x)$ 为凸函数 2. $h_j(x)$ 为仿射函数 3. $c_i(x)$ 严格可行

那么：

$$p^* = d^* \quad \text{的充要条件} \quad <==> \quad KKT\text{条件：} \begin{cases} \frac{d}{dx}L(x^*,\alpha^*,\beta^*) = 0 \\ \alpha_i^* c_i^* = 0 \quad (\text{松弛互补条件}) \\ c_i^*(x) \le 0 \\ \alpha_i^* \ge 0 \\ h_j^*(x) = 0 \end{cases}$$

在原问题中，$\frac{1}{2}||\vec{w}||^2, (y_i(w^Tx_i + b) - 1)$ 均为凸函数，根据上述定理2可知，只要满足KKT条件，就可以用求解对偶问题来代替求解原问题。

综上，原问题等价为：

$$\max_\alpha \min_{w,b} L(w,b,\alpha) = \max_\alpha \min_{w,b} (\quad \frac{1}{2}||\vec{w}||^2 + \sum_i^N \alpha_i (1 - y_i(w^Tx_i + b)) \quad)$$
$$\text{解必须满足：} \begin{cases} \alpha_i (1 - y_i(w^Tx_i + b)) = 0 \\ (y_i(w^Tx_i + b) - 1) \ge 0 \\ \alpha_i \ge 0 \end{cases}$$

1.4 求解过程

目标函数：

$$L(w,b,\alpha)= \quad \frac{1}{2}||\vec{w}||^2+\sum_i^N \alpha_i(1-y_i(w^T x_i+b))$$

先求解：

$$\min_{w,b} L(w,b,\alpha)$$

分别对 w,b 求导,并令其为0：

$$w-\sum_i^N \alpha_i y_i x_i=0$$
$$\sum_i^N \alpha_i y_i=0$$

将结果带入$L(w,b,\alpha)$：

$$L(w,b,\alpha)=\frac{1}{2}(\sum_i^N \alpha_i y_i x_i)^T\cdot(\sum_i^N \alpha_i y_i x_i)-\sum_i^N \alpha_i y_i(\,(\sum_j^N \alpha_i y_i x_i)^T\cdot x_i+b\,)+\sum_i^N \alpha_i$$
$$=-\frac{1}{2}\sum_i^N\sum_j^N \alpha_i\alpha_j y_i y_j x_i^T x_j+\sum_i^N \alpha_i$$
$$s.t.\ \ \sum_i^N \alpha_i y_i=0,\alpha_i\geq 0$$

现在参数只剩下α，α是一个N维的向量，要求：

$$\max_{\alpha} L(\alpha)$$
$$s.t.\ \ \sum_i^N \alpha_i y_i=0,\alpha_i\geq 0$$

最后的参数用SMO求解，文章最后会介绍。

用SMO求解不但要用到这个最终表达式，还是用到对偶问题解成立的KKT条件：

$$\begin{cases}\alpha_i(1-y_i(w^T x_i+b))=0\\(y_i(w^T x_i+b)-1)\geq 0\\\alpha_i\geq 0\end{cases}$$

2 非线性可分样本和核函数

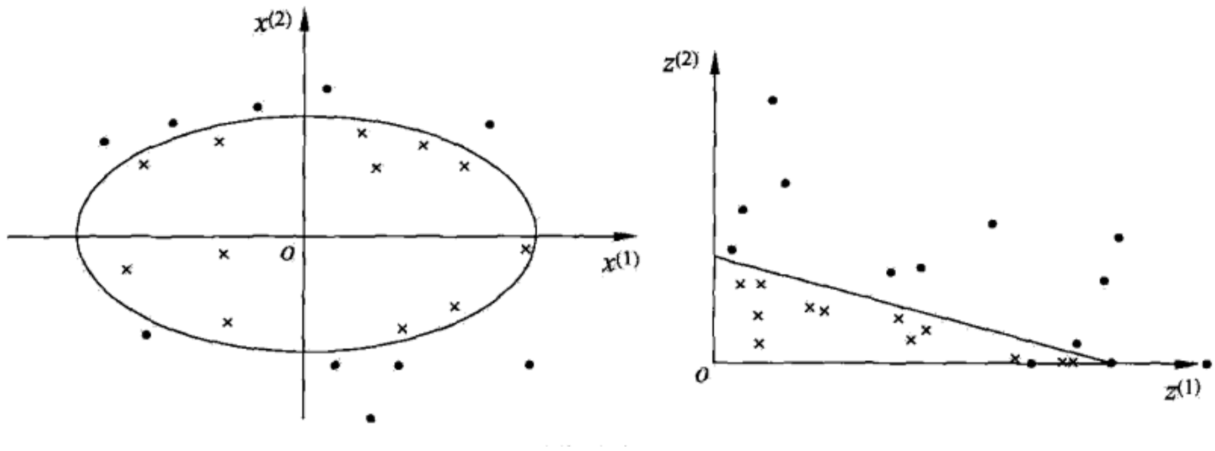
接上，最后的结果：

$$L(w,b,\alpha) = -\frac{1}{2}\sum_i^N\sum_j^N \alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_i^N \alpha_i$$

$$s.t.\ \sum_i^N \alpha_i y_i = 0, \alpha_i \ge 0$$

该结果来源于最初的假设，就是样本线性可分，我们用一个简单地线性超平面来分割样本集：

$$f(x) = w_1x_1 + w_2x_2 + w_3x_3 \ldots w_nx_n + b$$



假设二维的情况，用一条直线去分割二维空间,如右图：

$$f(x) = w_1x_1 + w_2x_2 + b$$

如果已知的样本点是非线性可分，如左图，那么最好的分割曲线应该是椭圆状曲线：

$$f(x) = w_1x_1^2 + w_2x_2^2 + b$$

设 $\phi(x)$ 为映射函数

$$\text{线性可分下：}\phi_1(x) = x$$

$$\text{椭圆情况下的映射：}\phi_2(x) = x^2$$

定义函数：

$$K_1(x_1,x_2) = \phi_1(x_1)\cdot\phi_1(x_2) = x_1\cdot x_2$$

$$K_2(x_1,x_2) = \phi_2(x_1)\cdot\phi_2(x_2) = x_1^2\cdot x_2^2$$

根据上述推导，线性可分情况的解：

$$L(w,b,\alpha) = -\frac{1}{2}\sum_i^N\sum_j^N \alpha_i\alpha_j y_i y_j K_1(x_i,x_j) + \sum_i^N \alpha_i$$

$$s.t.\ \sum_i^N \alpha_i y_i = 0, \alpha_i \ge 0$$

同理可得出线性不可分，椭圆曲线分割线情况下的解：

$$L(w,b,\alpha)=-\frac{1}{2}\sum_{i}^{N}\sum_{j}^{N}\alpha_i\alpha_j y_i y_j K_2(x_i,x_j)+\sum_{i}^{N}\alpha_i$$

$$s.t.\ \sum_{i}^{N}\alpha_i y_i=0,\alpha_i\geq 0$$

形如这种，能够将样本点映射到特定空间 的函数 $K(x_i,x_j)$ 称为核函数

常用核函数

$$\text{线性核：}\ K\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)=\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j$$

$$\text{多项式核：}\ K\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)=\left(\xi+\gamma\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j\right)^d$$

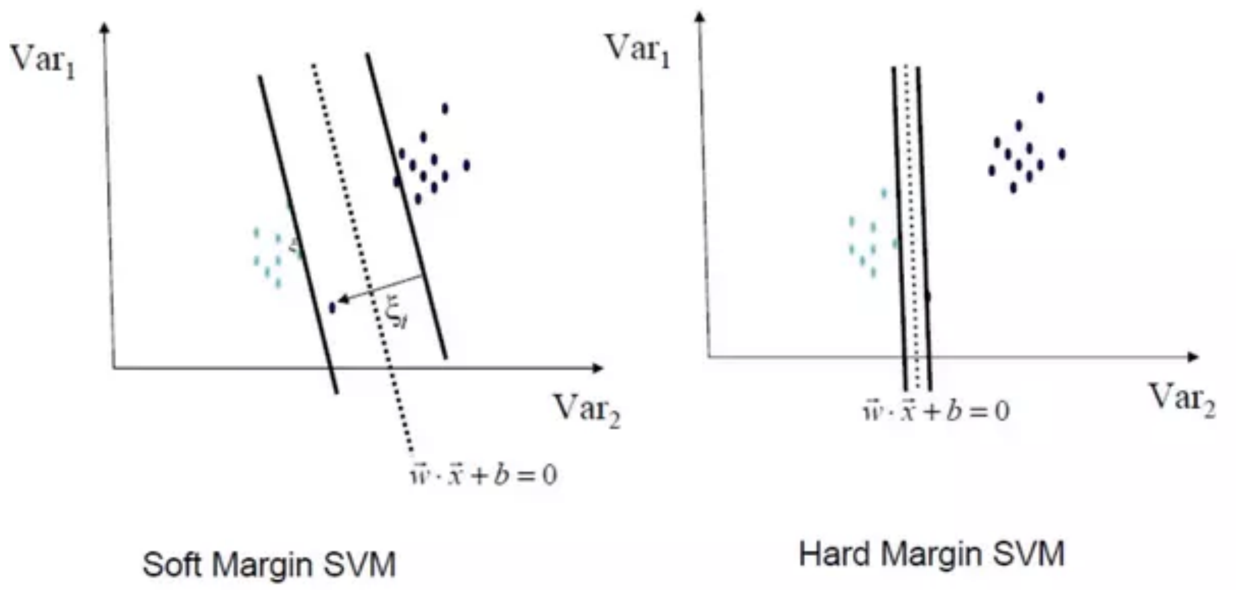
$$\text{高斯核：}K\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)==exp\left(-\gamma\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2\right)$$

$$\text{拉普拉斯核：}K\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)=\exp\left(-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|}{\sigma}\right)$$

$$sigmoid\text{核：}K\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)=\tanh\left(\gamma\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j+\xi\right)$$

3 软间隔SVM

有些数据集虽然大体上是线性可分的，但会有一些点，可能是噪点或是异常点，使得求出严格的分割平面较为困难，软间隔就是为了解决这样的问题，通过放松条件，允许每个点都可以有**一定程度的越界**，引入参数 ξ_i 来表示每个样本点的越界程度，该参数称为**松弛变量**，最后通过最小化该参数使所有点的越界程度最小，即达到最好的分类效果，也容忍了一定程度的噪点问题 。



分类条件：

$$
\begin{aligned}
w^T x_i + b \geq \ \ 1 - \xi_i \quad y_i = +1 \\
w^T x_i + b \geq \xi_i - 1 \quad y_i = -1
\end{aligned}
$$

优化目标：

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

$$
\text{s.t.} \quad y_i\left(w \cdot x_i + b\right) \geqslant 1 - \xi_i, \quad \xi_i \geqslant 0
$$

C称为惩罚因子，C值得大小表征对越界的容忍程度，即惩罚力度的大小。对松弛参数的理解，可以类比在其他学习算法中的正则项，通过引入正则项来防止过拟合。如上左图，如果用硬间隔SVM，中间那个异常点将会起到支持向量的作用，那么正负样本的间隔将会非常小，这可以类比成一种过拟合。

推导过程与不带松弛参数的线性SVM类似：

写出拉格朗日方程：

$$
L(w,b,\xi,\alpha,\mu) \equiv \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left(y_i\left(w \cdot x_i + b\right) - 1 + \xi_i\right) - \sum_{i=1}^{N}\mu_i\xi_i
$$
$$
s.t. \alpha_i \geq 0, \mu_i \geq 0
$$

原问题：

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

转换成对偶问题：

$$
\max_{\alpha,\mu}\min_{w,b,\xi} L(w,b,\xi,\alpha,\mu) = \min_{w,b,\xi}\max_{\alpha,\mu} L(w,b,\xi,\alpha,\mu)
$$
$$
s.t. \quad KKT
$$

转换成对偶问题后就可以先对w, b, ξ 求导求极小值：

$$
\begin{aligned}
\nabla_w L(w,b,\xi,\alpha,\mu) = w - \sum_{i=1}^{N}\alpha_i y_i x_i = 0 \\
\nabla_b L(w,b,\xi,\alpha,\mu) = -\sum_{i=1}^{N}\alpha_i y_i = 0 \\
\nabla_{\xi_i} L(w,b,\xi,\alpha,\mu) = C - \alpha_i - \mu_i = 0
\end{aligned}
$$

回带到原函数，可以发现，正好消去了ξ, w, b

最后，又得出以下熟悉的形式

$$\min_{\alpha} L(w,b,\xi,\alpha,\mu)=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(x_i^T x_j\right)+\sum_{i=1}^{N}\alpha_i$$
$$\begin{array}{ll}\text{s.t.} & \sum_{i=1}^{N}\alpha_i y_i=0\\ & 0\leqslant\alpha_i\leqslant C\end{array}$$

与之前的推导结果唯一不同的是 α_i 有了上界的限制， 该限制来自于 上述对 ξ 求导的结果 $C-\alpha_i-\mu_i=0$

最后，相应的KKT条件：

$$\begin{cases}\left(y_i\left(w\cdot x_i+b\right)-1+\xi_i\right)\geqslant 0\\ \alpha_i\left(y_i\left(w\cdot x_i+b\right)-1+\xi_i\right)=0\\ \alpha_i\geqslant 0\\ \mu_i\geqslant 0\\ \xi_i\geqslant 0\end{cases}$$

4 SMO算法

SMO算法SVM的最后一步，是用于计算SVM的最后参数α的一种优化算法。

以最一般化的SVM来做SMO的计算，带软间隔，带核函数的情况：

$$\text{分割面}:f(x_i)=w\phi(x_i)+b=\sum_{j}^{N}\alpha_j y_j K(x_i,x_j)+b\qquad ---(1)$$
$$(w=\sum_{j}^{N}\alpha_j y_j\phi(x_j))$$

$$\text{优化目标}:\ \min_{\alpha} L(w,b,\xi,\alpha,\mu)=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K\left(x_i,x_j\right)+\sum_{i=1}^{N}\alpha_i$$
$$\begin{array}{ll}\text{s.t.} & \sum_{i=1}^{N}\alpha_i y_i=0\\ & 0\leqslant\alpha_i\leqslant C\end{array}$$

$$KKT\text{条件}:\begin{cases}\left(y_i\left(w\cdot x_i+b\right)-1+\xi_i\right)\geqslant 0\\ \alpha_i\left(y_i\left(w\cdot x_i+b\right)-1+\xi_i\right)=0\\ \alpha_i\geqslant 0\\ \mu_i\geqslant 0\\ \xi_i\geqslant 0\end{cases}$$

计算步骤：

每次迭代只选择N个α_i中的两个α_i 做优化，最终达到所有α收敛的效果

指定一个分类精度ε，来控制收敛的精度

1. 随机初始化所有α，b

2. 根据$f(x_i)$ 计算误差

$$E_i = f(x_i) - y_i \quad ---(2)$$

3. 循环选择一个最不满足KKT条件的 α_v ，如下选择规则：

$$\begin{gathered}\alpha_v < C \quad and \quad E_v < -\varepsilon \\ or \\ 0 < \alpha_v \quad and \quad E_v > \varepsilon\end{gathered} \qquad ---(3)$$

如果找不到，退出循环，优化结束

有些算法用随机选择的方法选择第一α_v 这样的做法收敛较慢

4. 根据选择出来的α_v 循环寻找α_w 使得 $|E_v - E_w|$ 最大

5. 计算L,H

$$if \;\; y_v == y_w$$

$$L = \max\left(0, \alpha_w + \alpha_v - C\right), \quad H = \min\left(C, \alpha_w + \alpha_v\right)$$

$$else$$

$$L = \max\left(0, \alpha_w - \alpha_v\right), \quad H = \min\left(C, C + \alpha_w - \alpha_v\right)$$

6. 更新α_w 的值

$$\alpha_w^{'} = \alpha_w^{\text{old}} + \frac{y_w\left(E_v - E_w\right)}{\eta}$$

$$\eta = K(x_v, x_v) + K(x_w, x_w) - 2K(x_v, x_w) \quad (\eta \geq 0) \quad ---(4)$$

$$\alpha_w^{\text{new}} = \begin{cases} H, & \alpha_w^{'} > H \\ \alpha_w^{'} & L \leqslant \alpha_w^{'} \leqslant H \\ L, & \alpha_w^{'} < L \end{cases} \quad ---(5)$$

7. 更新α_v的值

$$\alpha_v^{\text{new}} = \alpha_v^{\text{old}} + y_1 y_2 \left(\alpha_w^{\text{old}} - \alpha_w^{\text{new}}\right) \quad ---(6)$$

8. 计算b

$$b_v^{\text{new}} = -E_v - y_v K(x_v, x_v)\left(\alpha_v^{\text{new}} - \alpha_v^{\text{old}}\right) - y_w K(x_w, x_v)\left(\alpha_w^{\text{new}} - \alpha_w^{\text{old}}\right) + b^{\text{old}}$$

$$b_w^{\text{new}} = -E_w - y_v K(x_v, x_w)\left(\alpha_v^{\text{new}} - \alpha_v^{\text{old}}\right) - y_w K(x_w, x_w)\left(\alpha_w^{\text{new}} - \alpha_w^{\text{old}}\right) + b^{\text{old}}$$

$$if \quad 0 < \alpha_v^{new} < C:$$

$$b^{new} = b_v^{new}$$

$$else \;\; if \quad 0 < \alpha_w^{new} < C:$$

$$b^{new} = b_w^{new}$$

$$else:$$

$$b^{new} = \frac{1}{2}(b_v^{new} + b_w^{new})$$

9. 跳回2

```
def SVM(X, y, C, e, maxEpoch):

    #N是样本数，D是样本维度
    N,D = shape(X)

    #初始化alpha,b
    alphas = np.zeros(N,1)
    b = 0;

    for i in range(maxEpoch):

        for v in range(N):  #在数据集上遍历每一个alpha

            #(1)式
            #如果需要加核函数 ： fx_v = (alphas*y).T*K(x,x[v]) + b
            fx_v = (alphas*y).T*(X.dot(X[v].T)) + b

            #(2)式
            E_v  = fx_v-y[v]

            #根据(3)式规则选择alpha_v
            if ( ( y[v]*E_v < -e ) and ( alphas[v]<C ) ) or \
               ( ( y[v]*E_v >  e ) and ( alphas[v]>0 ) ):

                #根据第四点选择alpha_w
                w = -1
                E_w = -1
                tempMax = -1
                for j in range(N):
                    if j==v:
                        continue
                    fx_j = (alphas*y).T*(X.dot(X[j].T)) + b
                    E_j  = fx_j-y[j]
                    if abs(E_v-E_j) > tempMax:
```

```
            E_w = E_j
            w = j


    #根据第五点计算L和H
    if(y[v]!=y[w]):
        L=max(0, alphas[w]-alphas[v])
        H=min(C, C+alphas[w]-alphas[v])
    else:
        L=max(0, alphas[w]+alphas[v]-C)
        H=min(C, alphas[w]+alphas[v])
    if L==H:
        print('L==H')
        continue



    #根据式4计算eta
    eta= X[i]*X[i].T + X[j]*X[j].T - 2.0*X[v]*X[w].T
    if eta<0:
        print('eta<0')
        continue


    #根据式5计算更新alpha_w
    alpha_w_old = alphas[w]
    alphas[w]+=y[w]*(E_v-E_w)/eta  #调整alphas[j]
    if alphas[w]>H:
        alphas[w]=H
    if alphas[w]<L:
        alphas[w]=L

    #如果变化很小，后面就不做了，直接下一轮更新
    if(abs(alphas[w]-alpha_w_old)<0.00001):
        print('w not moving enough')
        continue


    #根据式6计算更新alpha_v
    alpha_v_old=alphas[v]
    alphas[v]+=y[w]*y[v]*(alpha_w_old-alphas[w])  #调整alphas[i]

    #根据第八点计算b
    b_v = b-E_v-\
    y[v]*(alphas[v]-alpha_v_old)*X[v]*X[v].T-\
    y[w]*(alphas[w]-alpha_w_old)*X[w]*X[v].T

    b_w = b-E_w-\
    y[v]*(alphas[v]-alpha_v_old)*X[v]*X[w].T-\
    y[w]*(alphas[w]-alpha_w_old)*X[w]*X[w].T

    if(0<alphas[i]) and (C>alphas[i]):
```

```
                b=b_v
            elif(0<alphas[j]) and (C>alphas[j]):
                b=b_w
            else:
                b=(b_v+b_w)/2.0


    return b, alphas
```

5 SVM面试常考题

5.1 为什么要转化成对偶问题求解SVM

原问题：

$$\min_{w,b}\max_{\alpha} L(w,b,\alpha) = \min_{w,b}\max_{\alpha}\ (\quad \frac{1}{2}||\vec{w}||^2 + \sum_i^N \alpha_i(1 - y_i(w^T x_i + b))\quad),\quad \alpha_i \geq 0 \qquad ------(1)$$

对偶问题：

$$\max_{\alpha}\min_{w,b} L(w,b,\alpha) = \max_{\alpha}\min_{w,b}\ (\quad \frac{1}{2}||\vec{w}||^2 + \sum_i^N \alpha_i(1 - y_i(w^T x_i + b))\quad),\quad \alpha_i \geq 0 \qquad ------(2)$$

- 求解问题的复杂度不同，原问题最外层的是$\min_{w,b}$， 最后求解问题的复杂度和w的维度相关，即，和样本的维度相关。而对偶问题的外层是max_α， 最后的求解问题与α的维度相关，即，和样本数量相关。传统机器学习最初是针对高纬度低样本数的数据进行设计的(人工智能课老师)，所以svm对于高维空间中较稀疏的样本表现较好。
- 能够运用核函数 (主要原因)，$L(w,b,\alpha)$只有先对w,b求导才能导出最后带有 $(x_i \cdot x_j)$ 的形式，才有机会运用核函数来应对非线性情况
- 引入了KKT条件，简化了约束条件

5.2 核函数的选择问题

吴恩达的选择：

- 特征数量很大， 跟样本差不多时， 用线性核
- 特征数量比较小， 样本数量很大， 需手动添加特征，后用线性核
- 特征数量比较小， 样本数量一般， 用高斯核

从调参复杂度选择

线性核： 无

多项式核： 幂次d, 系数γ, 常数项ξ

高斯核： 系数γ

sigmoid核：系数γ,常数项ξ

5.3 sklearn中SVM的参数使用

```
class sklearn.svm.SVC(
            C=1.0,
            kernel='rbf',
            degree=3,
            gamma='auto',
            coef0=0.0,
            shrinking=True,
            probability=False,
            tol=0.001,
            cache_size=200,
            class_weight=None,
            verbose=False,
            max_iter=-1,
            decision_function_shape='ovr',
            random_state=None)


主要需要调整的参数:
C:  对应上文的C，软间隔中的惩罚因子
kernel: 核函数 ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ ，默认rbf
degree: kernel为多项式核的时候有效，调整的是多项式核的幂次
gamma:  当kernel为‘rbf’, ‘poly’或‘sigmoid’时生效，对应的核函数中的gama值
coef0:  当kernel为‘poly’,‘sigmoid’时有效
tol  :  训练精度，对应上面SMO的精度
max_iter: 对应上述的maxEpoch
```

5.4 样本类别不均衡情况是否影响SVM分类效果

会有影响，原因：

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

假设负样本的数量远大于正样本的数量，那么负样本越界的点一定大于正样本越界的点，算法最小化 $C\sum_i^N \xi_i$ 会使得超平面往正样本移动，来减少负样本越界的点，从而减小 $C\sum_i^N \xi_i$。

如何解决：

1. 对正样本和负样本设置不同的C，来保证平衡
2. 欠采样

5.5 SVM多分类问题

一对一法：

任意两类样本之间设计一个SVM，最终有k(k-1)/2个分类器，投票决定

一对多法：

把某个类别的样本归为一类，其余类别归为另一类，做一个SVM，这样，k个类别就有k个SVM，样本经过k个SVM分别计算，以函数值最大的那个类别为最终类别