分类：SVM（硬间隔、软间隔、梯度下降算法）

覃雄派

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 监督学习算法分类

任务

模型函数

	(二值)分类 Y为类别集合	回归 Y为实数集合	结构预测/排序 Y为结构类型/排列
线性函数 **f(x)=⟨w, x⟩**	Perceptron, SVM	Ridge regression SVR	Structured SVM, Ranking SVM
树	Decision tree	Regression tree	LambdaMART
神经网络	神经网络	神经网络	RankNet
叠加函数 **f(x)=$\sum_t h_t$(x)** **(additive model)**	AdaBoost	Boosted Regression Tree	RankBoost

分类：SVM（硬间隔、软间隔、梯度下降算法）

提纲

- SVM简介
- SVM问题的分类
- 硬间隔
    - 向量点积与距离
    - 硬间隔问题建模
- 软间隔
    - 软间隔问题建模
- 梯度下降求解
- 代码实例

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 支持向量机简介
- 支持向量机(Support Vector Machine, SVM)技术，自从1995年正式发表以来，在中小规模数据的分类任务上，取得了很好的效果
  - SVM算法有完整的理论证明，在20世纪末的10年、以及21世纪初的10年，在深度学习取得重大突破以前，完胜传统神经网络，在文本、图像等数据处理中得到广泛应用

1992-1994 SVMs (Vapnik et al)

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 支持向量机简介
- SVM的历史

ARTICLE FREE ACCESS

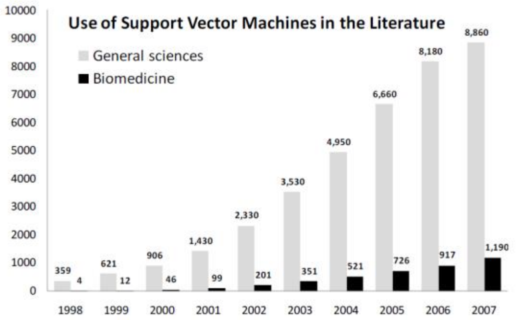
A training algorithm for optimal margin classifiers

Authors: Bernhard E. Boser, Isabelle M. Guyon, Vladimir N. Vapnik Authors Info & Affiliations

Publication: COLT '92: Proceedings of the fifth annual workshop on Computational learning theory • July 1992 • Pages 144–152 • https://doi.org/10.1145/130385.130401

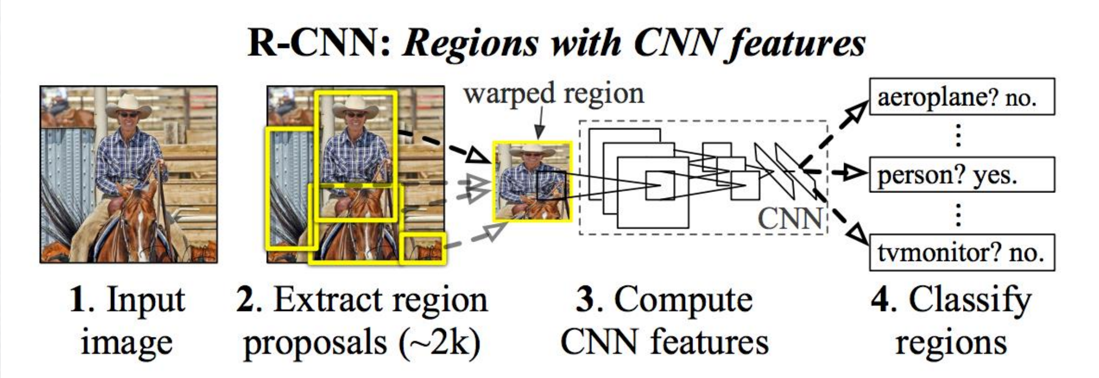
History of SVMs and usage in the literature

- Support vector machine classifiers have a long history of development starting from the 1960's.
- The most important milestone for development of modern SVMs is the 1992 paper by Boser, Guyon, and Vapnik ("*A training algorithm for optimal margin classifiers*")



分类：SVM（硬间隔、软间隔、梯度下降算法）

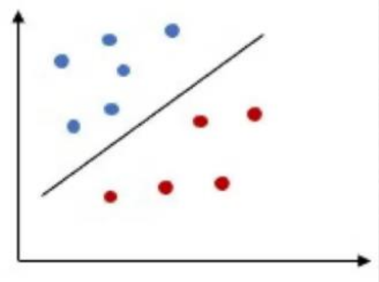
- 支持向量机简介
  - 即便在深度学习取得重大突破，大行其道的时候，SVM算法仍然被广泛使用
  - 有时候SVM算法和深度学习模型可以配合使用，比如在目标检测算法R-CNN中，SVM用于对Regional Proposal做出有物体/无物体的分类判定

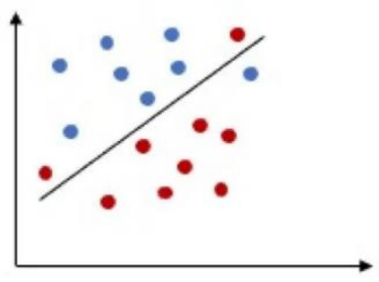


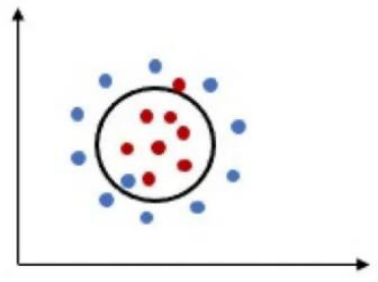
分类：SVM（硬间隔、软间隔、梯度下降算法）

分类：SVM（硬间隔、软间隔、梯度下降算法）

- SVM问题，可以分为如下几种情况：
  - (1) 线性可分情况下的线性分类器(硬间隔Hard Margin)
  - (2) 线性不可分情况下(基本上是线性可分，但是有一些离群值)的线性分类器(软间隔Soft Margin)
  - (3) 线性不可分情况下(不是线性可分的)的非线性分类器(需要使用核(Kernel)函数)







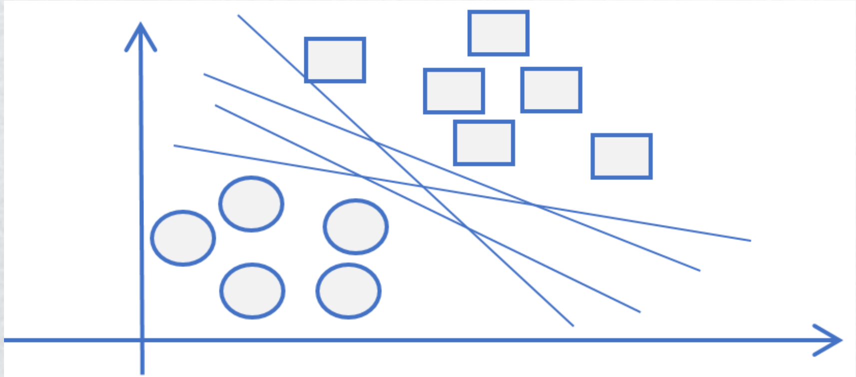
本讲

下一讲

分类：SVM（硬间隔、软间隔、梯度下降算法）

分类：SVM（硬间隔、软间隔、梯度下降算法）

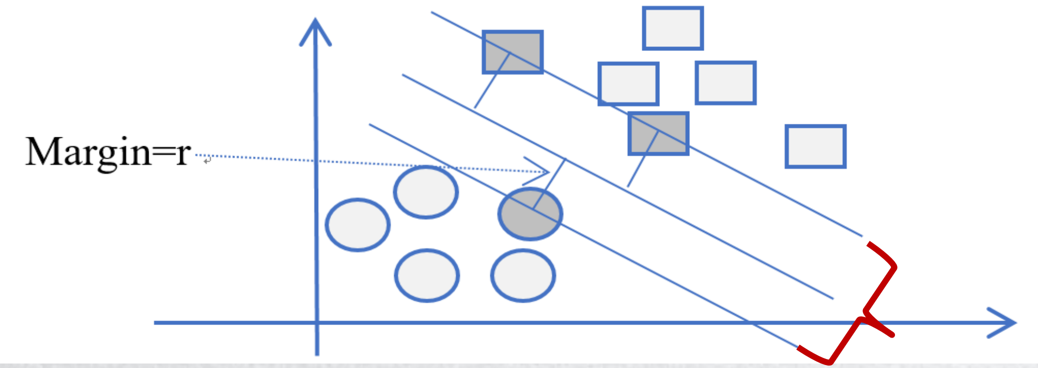
- 最大间隔
- 这里以二维平面上的数据点为例，如果存在两类数据
  - 我们想要找到一个分类边界(决策边界，即一条直线)
  - 将这两类数据给分开



  - 理论上讲，这样的决策边界有无数种选择

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 最大间隔
- 这里以二维平面上的数据点为例，如果存在两类数据
  - 我们想要找到一个分类边界(决策边界，即一条直线)
  - 将这两类数据给分开



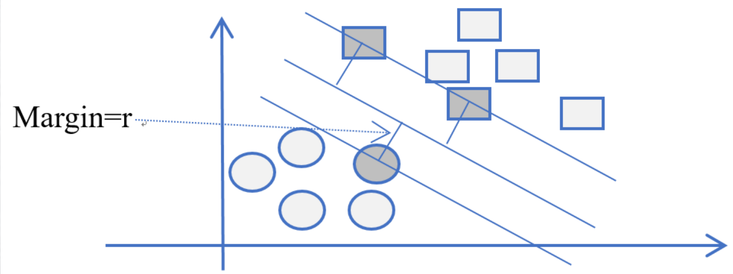
边距是从分类面到最近的训练样本的距离

  - 哪一条分割线最好？
  - 优化目标：间隔最大化

SVM算法认为，靠近决策边界的正样本点和负样本点到决策边界(直线)的距离最大的时候，这样的分类边界(直线)是最好的，如图所示

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 最大间隔
- 这里以二维平面上的数据点为例，如果存在两类数据
  - 我们想要找到一个分类边界(决策边界，即一条直线)，将这两类数据给分开
  - 哪一条分割线最好?
  - 优化目标：间隔最大化



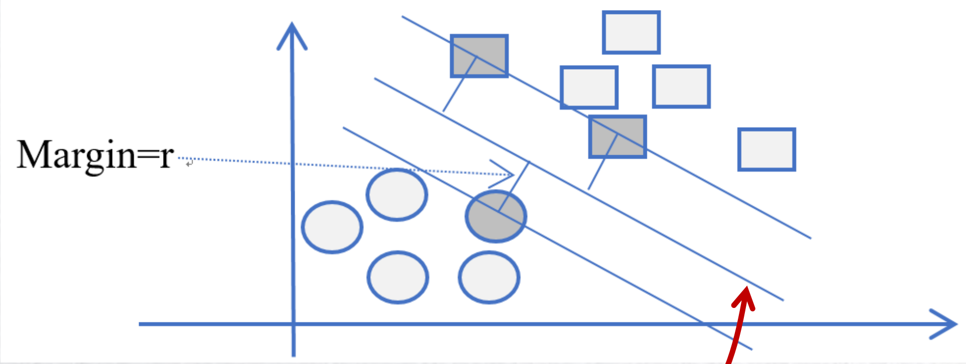
- 最好分割线的直觉：这个分隔线怎么算最好，最直觉的思想就是"不偏不倚"，与两类样本点的边界保持相等的最远的距离
- 更进一步的，是距离两类数据点的前沿最远，所以定义最接近边界的数据点定义为支持向量

  - 中间那条直线就是符合间隔最大化的分类边界(直线)
    - 它由2个正样本点(正方形、深色)、以及1个负样本点(圆形、深色)唯一确定，其它数据点不会影响该分类边界的确定
  - 这三个数据点称为支持向量

位于分类边距上的数据点

分类：SVM（硬间隔、软间隔、梯度下降算法）

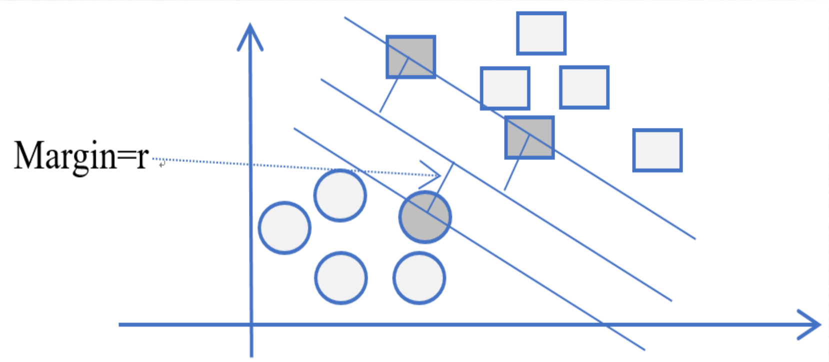
- 最大间隔
- 这里以二维平面上的数据点为例，如果存在两类数据
  - 我们想要找到一个分类边界(决策边界，即一条直线)，将这两类数据给分开
  - 哪一条分割线最好?
  - 优化目标：间隔最大化



  - 对于二维平面，决策边界是一条直线
  - 对于3维空间，决策边界是一个平面
  - 一般地，对于更高维空间，决策边界是一个超平面

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 最大间隔
- 这里以二维平面上的数据点为例，如果存在两类数据
  - 我们想要找到一个分类边界(决策边界，即一条直线)，将这两类数据给分开
  - 哪一条分割线最好?
  - 优化目标：间隔最大化

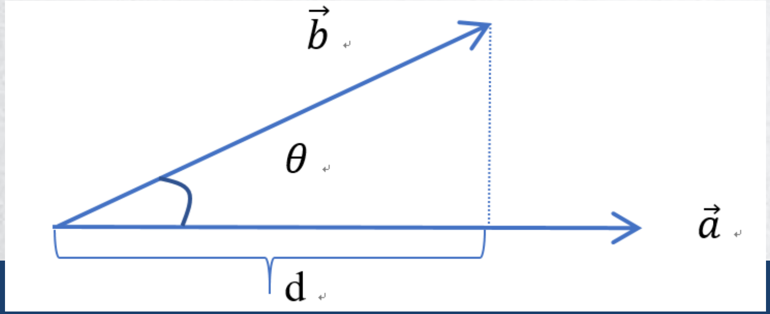


- 直觉上，比其它分类界面更不容易出错，分类面的微小扰动不会导致错误发生
- 理论上，VC理论证明最大边距分类器具有较好的泛化能力(generalization ability)
- 实际应用：SVM在很多应用上取得了很好的效果

分类：SVM（硬间隔、软间隔、梯度下降算法）

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 向量点积、夹角余弦、距离
- 假设有向量$\vec{a}$和向量$\vec{b}$，向量$\vec{b}$在向量$\vec{a}$上的投影的模，即距离d，可以计算如下：
  - 向量的内积为 $\vec{a}.\vec{b}$=||a||×||b||×cos θ
  - 于是夹角θ的余弦为cos $\theta = \frac{\vec{a}.\vec{b}}{||a||\times||b||}$
  - 距离d可以表示为d=||b||cos θ=||b||×$\frac{\vec{a}.\vec{b}}{||a||\times||b||} = \frac{\vec{a}.\vec{b}}{||a||}$

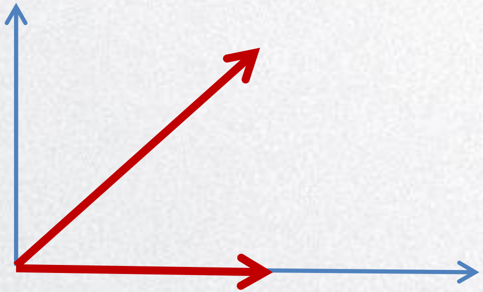


分类：SVM（硬间隔、软间隔、梯度下降算法）

- 向量点积、夹角余弦、距离
- 假设有向量$\vec{a}$和向量$\vec{b}$，向量$\vec{b}$在向量$\vec{a}$上的投影的模，即距离d，可以计算如下：
  - 向量的内积为 $\vec{a}.\vec{b}$=||a||×||b||×cos θ
  - 于是夹角θ的余弦为cos $\theta = \frac{\vec{a}.\vec{b}}{||a|| \times ||b||}$
  - 距离d可以表示为d=||b||cos θ=||b||×$\frac{\vec{a}.\vec{b}}{||a|| \times ||b||} = \frac{\vec{a}.\vec{b}}{||a||}$

课堂练习：
1，计算向量a和b的夹角余弦
2，计算向量a在向量b上的投影

a	<1,0>
b	<1,1>



分类：SVM（硬间隔、软间隔、梯度下降算法）

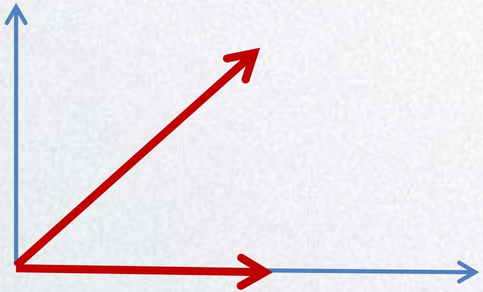
- 向量点积、夹角余弦、距离
- 假设有向量$\vec{a}$和向量$\vec{b}$，向量$\vec{b}$在向量$\vec{a}$上的投影的模，即距离d，可以计算如下：
  - 向量的内积为 $\vec{a}.\vec{b}$=||a||×||b||×cos θ
  - 于是夹角θ的余弦为cos $\theta = \frac{\vec{a}.\vec{b}}{||a||\times||b||}$
  - 距离d可以表示为d=||b||cos θ=||b||×$\frac{\vec{a}.\vec{b}}{||a||\times||b||} = \frac{\vec{a}.\vec{b}}{||a||}$

课堂练习：
1，计算向量a和b的夹角余弦
2，计算向量a在向量b上的投影

$$\cos\theta = \frac{1\times1+0\times1}{\sqrt{1}\times\sqrt{2}} = \frac{1}{\sqrt{2}} = \frac{\sqrt{2}}{2}$$

$$d = \frac{\vec{a}.\vec{b}}{||a||} = \frac{1\times1+0\times1}{\sqrt{1}} = 1$$

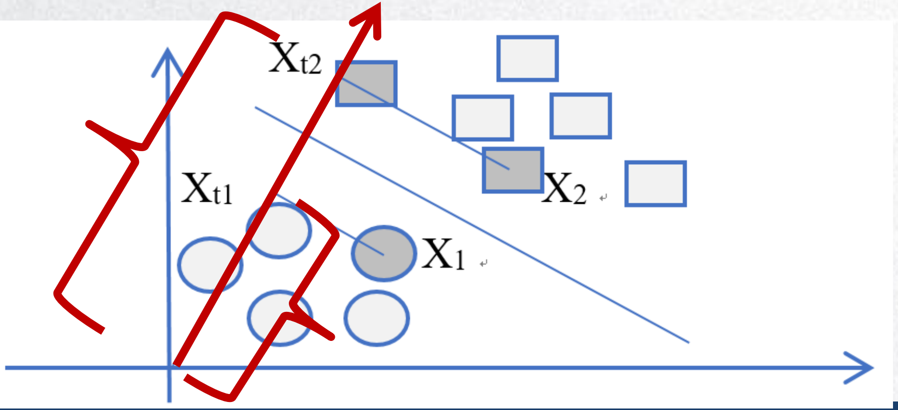
a	<1,0>
b	<1,1>



分类：SVM（硬间隔、软间隔、梯度下降算法）

- 向量点积、夹角余弦、距离
  - 决策边界左侧的数据点在法向量上的投影，原点到其距离为D_1；决策边界右侧数据点在法向量的投影，原点到其距离为D_2
    - 则有$D_1<D_2$；这成为支持向量机实现分类预测的依据
  - 在SVM的特征空间中，假设$\vec{w}$就是决策边界的法向量，现在已经对$\vec{w}$进行单位化，那么它的模为1
  - x1,x2等在法向量上的投影
  - 有$d1=\vec{w}.\vec{x_1}$
  - $d2=\vec{w}.\vec{x_2}$

参考$d=\dfrac{\vec{a}.\vec{b}}{||a||}$

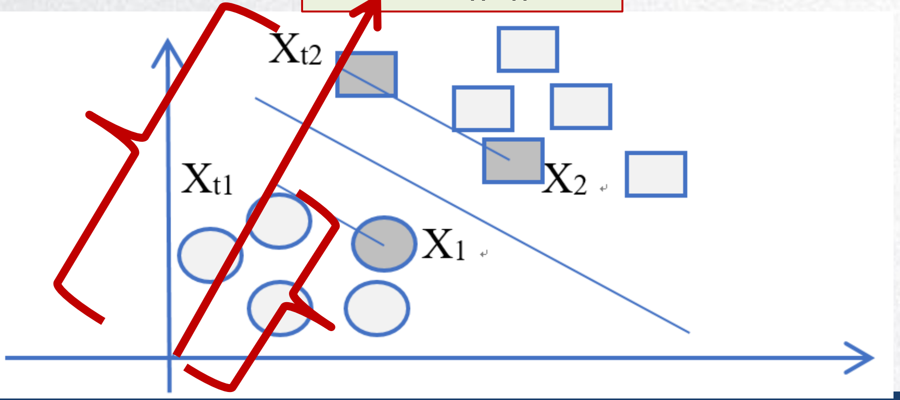


分类：SVM（硬间隔、软间隔、梯度下降算法）

- 向量点积、夹角余弦、距离
  - 决策边界左侧的数据点在法向量上的投影，原点到其距离为D_1；决策边界右侧数据点在法向量的投影，原点到其距离为D_2
    - 则有$D_1<D_2$；这成为支持向量机实现分类预测的依据
  - 在SVM的特征空间中，假设$\vec{w}$就是决策边界的法向量，现在已经对$\vec{w}$进行单位化，那么它的模为1
  - x1,x2等在法向量上的投影
  - 有d1=$\vec{w}.\vec{x_1}$
  - d2=$\vec{w}.\vec{x_2}$

寻找一个偏置量b
使得正例数据： $w^T.x +b>=+1$
负例数据； $w^T.x +b<=-1$
分类超平面线性方程为$w^T.x +b = 0$

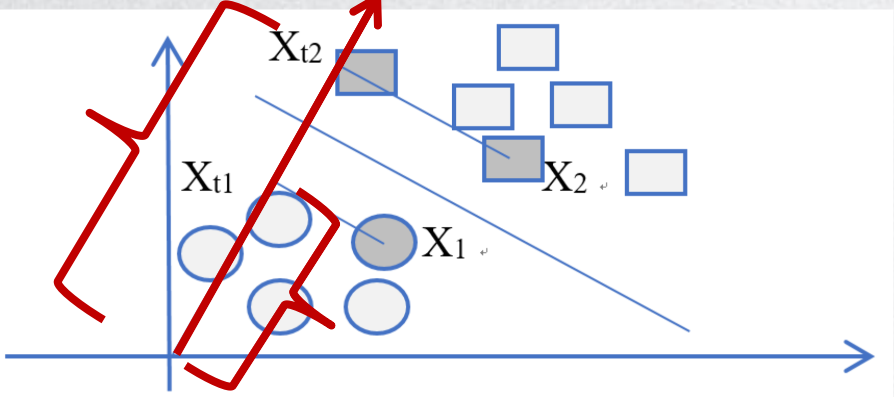
参考d=$\frac{\vec{a}.\vec{b}}{||a||}$



分类：SVM（硬间隔、软间隔、梯度下降算法）

- 向量点积、夹角余弦、距离
  - 决策边界左侧的数据点在法向量上的投影，原点到其距离为D_1；决策边界右侧数据点在法向量的投影，原点到其距离为D_2
    - 则有$D_1<D_2$；这成为支持向量机实现分类预测的依据

参考$d=\frac{\vec{a}.\vec{b}}{||a||}$



寻找一个偏置量b
使得正例数据：　$w^T.x +b>=+1$
负例数据；$w^T.x +b<=-1$
分类超平面线性方程为$w^T.x +b = 0$

1.假设正例数据：　$w^T.x +b>=9$
负例数据；$w^T.x +b<=4$
9和4的中间点为(9+4)/2=6.5

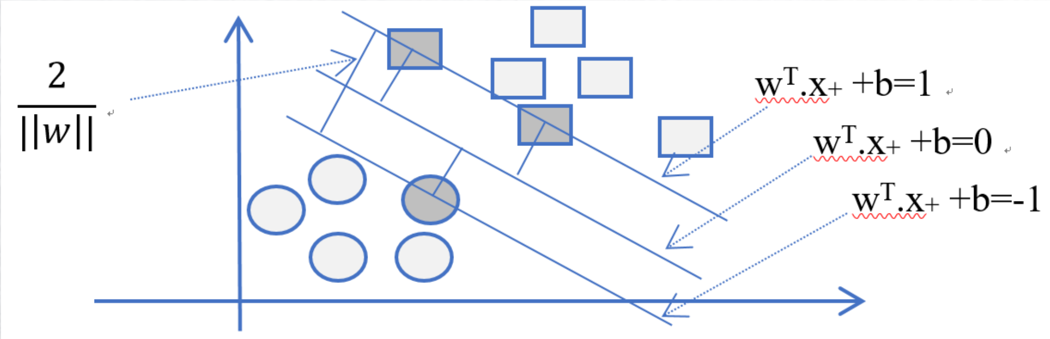
2.对于正例数据，$(w^T.x +b-6.5)/2.5$，经过偏移和缩放，得到新的wnew, bnew，即可符合该要求

3.对于负例数据，$(w^T.x +b-6.5)/2.5$，经过偏移和缩放，得到新的wnew, bnew，即可符合该要求

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 向量点积、夹角余弦、距离

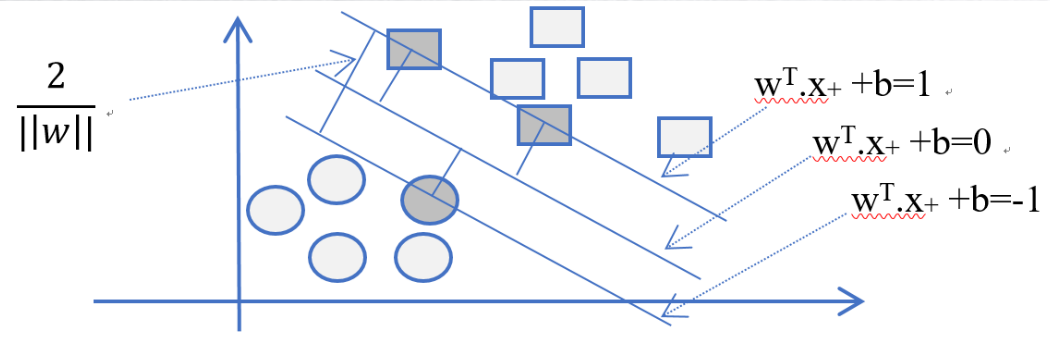
寻找一个偏置量b
使得正例数据： $w^T.x + b >= +1$
负例数据； $w^T.x + b <= -1$
分类超平面线性方程为 $w^T.x + b = 0$



分类：SVM（硬间隔、软间隔、梯度下降算法）

分类：SVM（硬间隔、软间隔、梯度下降算法）

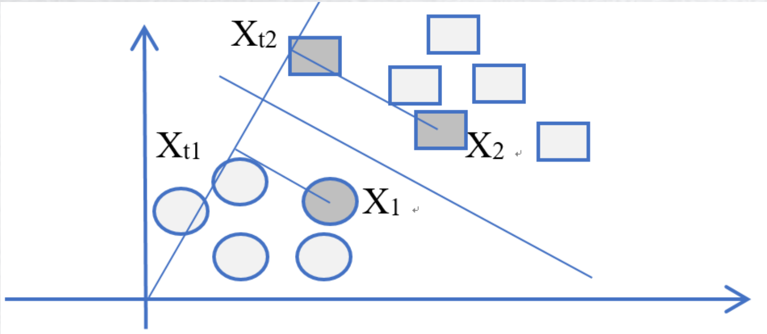
- 硬间隔
  - SVM要训练的参数有w和b，对于所有样本点，SVM期望
    - $w^T.x_+ + b >= 1$
    - $w^T.x_+ + b <= -1$
  - 选用+1和-1是为了计算方便
    - 无论具体间隔(margin)是多少
    - 可以通过缩放w和b，满足上述两个式子



分类：SVM（硬间隔、软间隔、梯度下降算法）

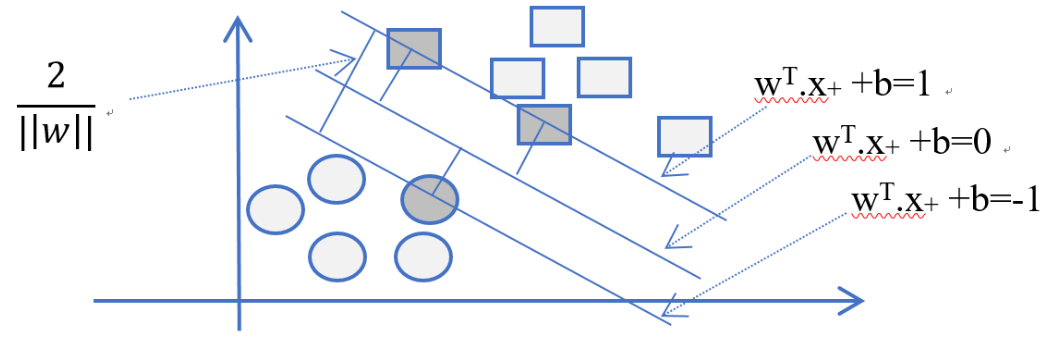
- 硬间隔
  - 决策边界两侧的直线的距离(最大间隔)为2/(||w||)，具体计算过程如下
  - 假设X2为正例数据点，X1为负例数据点，最大间隔就是X2-X1在决策边界(直线)法向量上的投影
  - 于是有$d=\frac{w^T.(x_+-x_-)}{||w||}=\frac{w^T.(x_+-x_-)}{||w||}=\frac{1-b-(-1-b)}{||w||}=\frac{2}{||w||}$

$||w|| = \sqrt{w^T.w}$
W自己做点积，开根方



分类：SVM（硬间隔、软间隔、梯度下降算法）

- 硬间隔
- 这就是SVM优化的目标，也就是max(d)= $\frac{2}{||w||}$
  - 约束条件进一步处理如下
  - $w^T.x_+ +b >= 1$ → $y_+(w^T.x_+ +b) >= 1$
  - $w^T.x_+ +b <= -1$ → $y_-(w^T.x_- +b) >= 1$
  - 于是，不管是正例还是负例，都有$y_i(w^T.x_i +b) >= 1$



分类：SVM（硬间隔、软间隔、梯度下降算法）

- 硬间隔
  - 问题建模

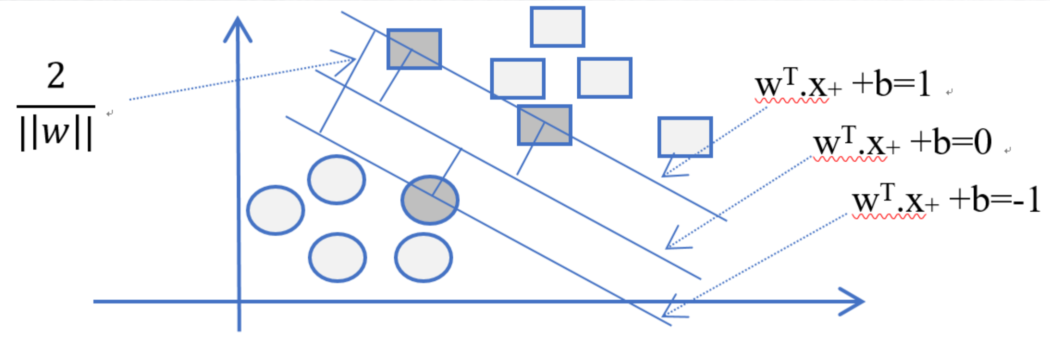
$$\max\left(\frac{2}{||w||}\right)$$

s.t. $y_i(w^T.x_i + b) >= 1$，$i=1,2,\ldots,n$。

或者，转换为最小化问题，建模如下：

$$\min\left(\frac{1}{2}||w||\right)$$

s.t. $y_i(w^T.x_i + b) >= 1$，$i=1,2,\ldots,n$。



优化目标：

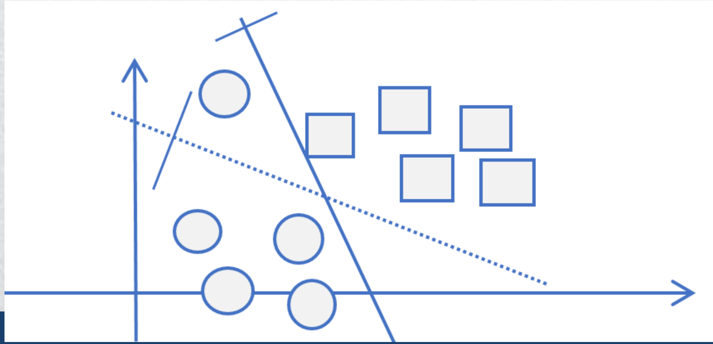
$$\max\left(\frac{2}{||w||}\right)$$
$$\Leftrightarrow min\frac{1}{2}(||w||)$$
$$\Leftrightarrow \min\frac{1}{2}\sqrt{\langle \mathrm{w}, \mathrm{w}\rangle}$$
$$\Leftrightarrow min\frac{1}{2}\langle \mathrm{w}, \mathrm{w}\rangle$$
$$\Leftrightarrow min||\mathrm{w}||^2$$

1. 点积可以写成 $\langle \mathrm{w}, \mathrm{w}\rangle$ 或者 $w^T\mathrm{w}$
2. 上述等价关系后文用到

分类：SVM（硬间隔、软间隔、梯度下降算法）

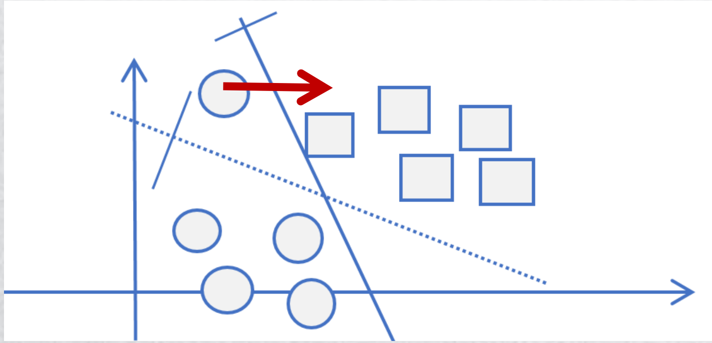
分类：SVM（硬间隔、软间隔、梯度下降算法）

- 软间隔
  - 下图中，实线所表示的决策边界，分开了正负样本，但是最大分类间隔很小
  - 而虚线所表示的决策边界，并没有完全分开正负样本，有分类错误，但是分类间隔较大
    - 这给我们一个折中选择，在允许少量错误的情况下，让分类间隔最大化
    - 也就是实例中实线的分类决策边界，未必见得比虚线的分类决策边界更好
    - 所谓的完美不是我们追求的，有点错误、但是更加鲁棒可能更好



分类：SVM（硬间隔、软间隔、梯度下降算法）

- 软间隔
  - 左上角的圆，再向右侧挪动一点点
    - 那么本问题，就无法实现线性可分，就是找不到一根直线把两类数据点分开。
  - 为此，SVM引入松弛变量，对最大分类间隔和错误分类进行折中

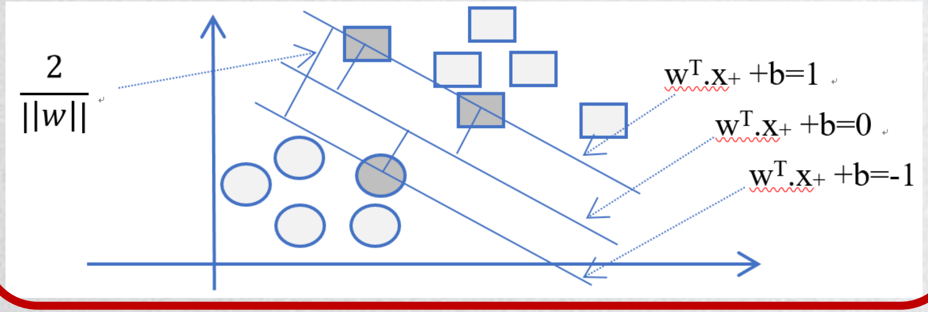


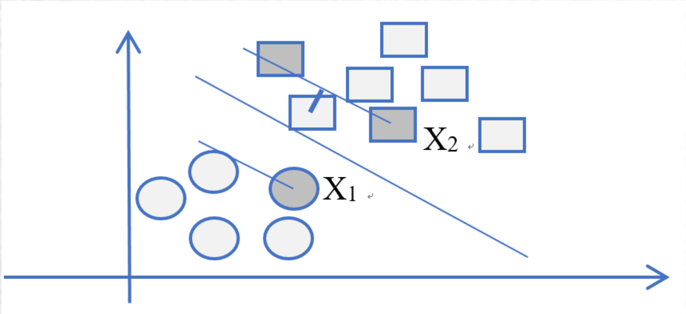
- 不存在一个分类面使得训练数据能够被完美分开
- 解决方案：容忍错误的发生
  - $\min_{w,b} |w|^2 +$ 错误惩罚,
  - 错误惩罚：出错的数据点与其正确位置的距离
  - 边距不再是硬性限制条件(软边距)

  (请见下文)

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 软间隔
  - 引入松弛变量$\xi \geq 0$。原有的约束条件变为：
  - $y_i\left(w^T.x_i + b\right) \geq 1 - \xi_i$
- 松弛变量的理解





当||w||为1，即w为单位向量，那么最大间隔为$\frac{2}{||w||}$为2；

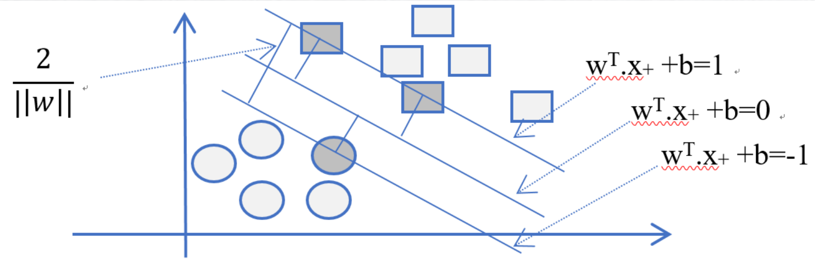
(1)决策边界方程$w^T.x +b=0$

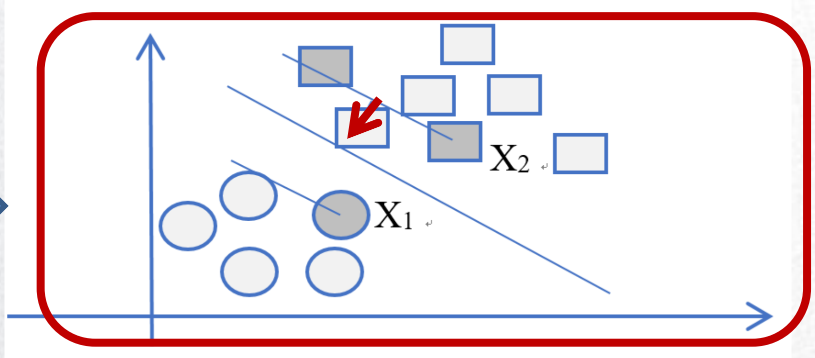
(2)把正例支持向量代入决策边界方程$y_+=w^T.x_+ +b=1$，$y_+(w^T.x_+ +b)=1$，可以理解为支持向量到决策边界的距离为1

(3)把负例支持向量代入决策边界方程$y_-=w^T.x_- +b=-1$，$y_-(w^T.x_- +b) = 1$，可以理解为支持向量到决策边界的距离为1

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 软间隔
  - 引入松弛变量$\xi \geq 0$。原有的约束条件变为：
  - $y_i\left(w^T.x_i + b\right) \geq 1 - \xi_i$
- 松弛变量的理解





引入松弛变量之后，有$y_i\left(w^T.x_i + b\right) \geq 1 - \zeta_i$

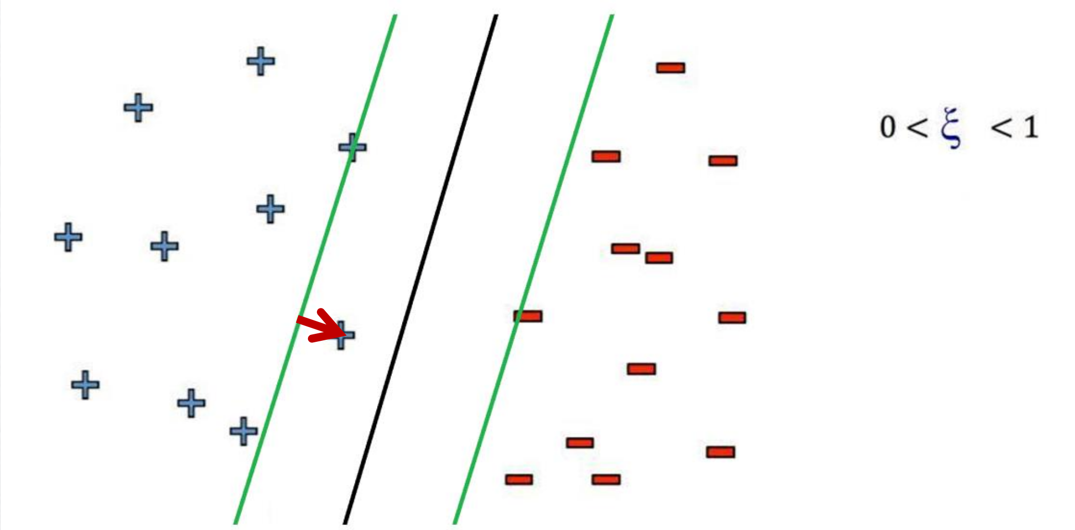
(1)在右图中，有一个不安分的正例x+，由于$y_+\left(w^T.x_+ + b\right) = 1 - \zeta_+$

(2)对照右图，我们看到，$y_+\left(w^T.x_+ + b\right) = 1 - \zeta_+$表达的意思是，该正例到决策边界的距离为$1 - \zeta_+$，由于$\zeta_+ > 0$，这个距离不是1了，比1要小

(3) ζ_+的意思是：其它支持向量$y_+\left(w^T.x_+ + b\right) = 1$所在的直线，为大量正例的最前沿；现在这个不安分的正例，离开前沿，突进了多少

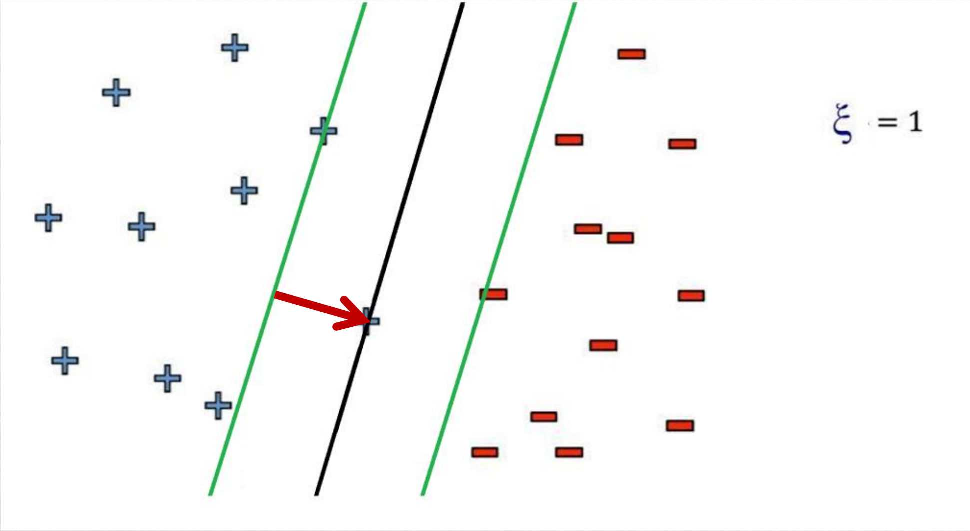
分类：SVM（硬间隔、软间隔、梯度下降算法）

- 软间隔
  - 引入松弛变量$\xi \geq 0$。原有的约束条件变为：
  - $y_i(w^T.x_i + b) \geq 1 - \xi_i$
- 松弛变量的理解
  - 数据点在超平面
  - 和本类间隔面之间



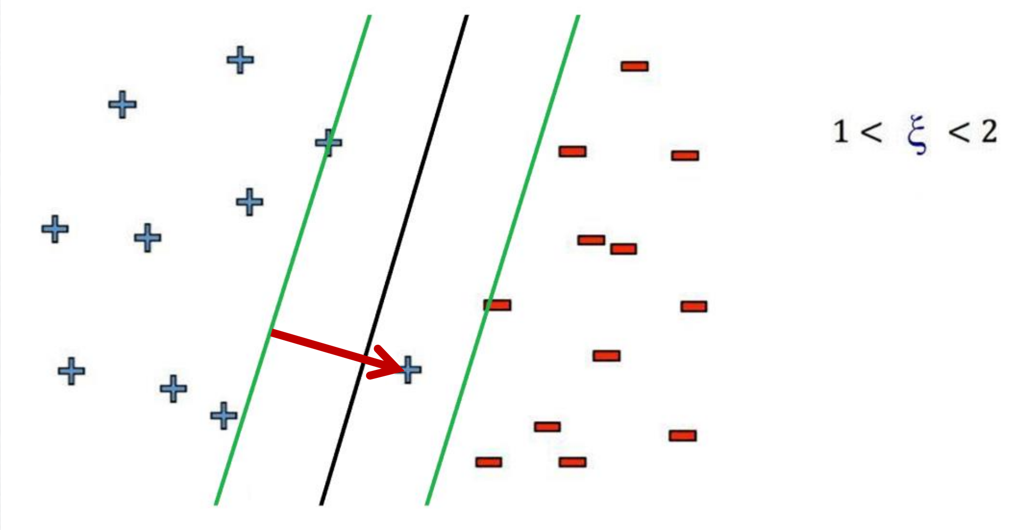
分类：SVM（硬间隔、软间隔、梯度下降算法）

- 软间隔
  - 引入松弛变量$\xi \geq 0$。原有的约束条件变为：
  - $y_i\left(w^T.x_i + b\right) \geq 1 - \xi_i$
- 松弛变量的理解
  - 数据点在超平面上



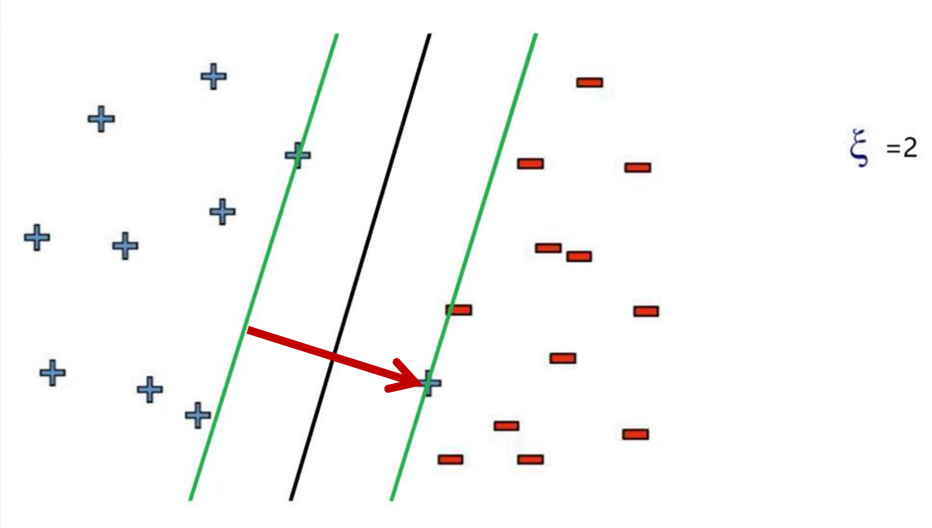
分类：SVM（硬间隔、软间隔、梯度下降算法）

- 软间隔
  - 引入松弛变量$\xi \geq 0$。原有的约束条件变为：
  - $y_i(w^T.x_i + b) \geq 1 - \xi_i$
- 松弛变量的理解
  - 数据点在超平面
  - 与另一侧间隔面之间



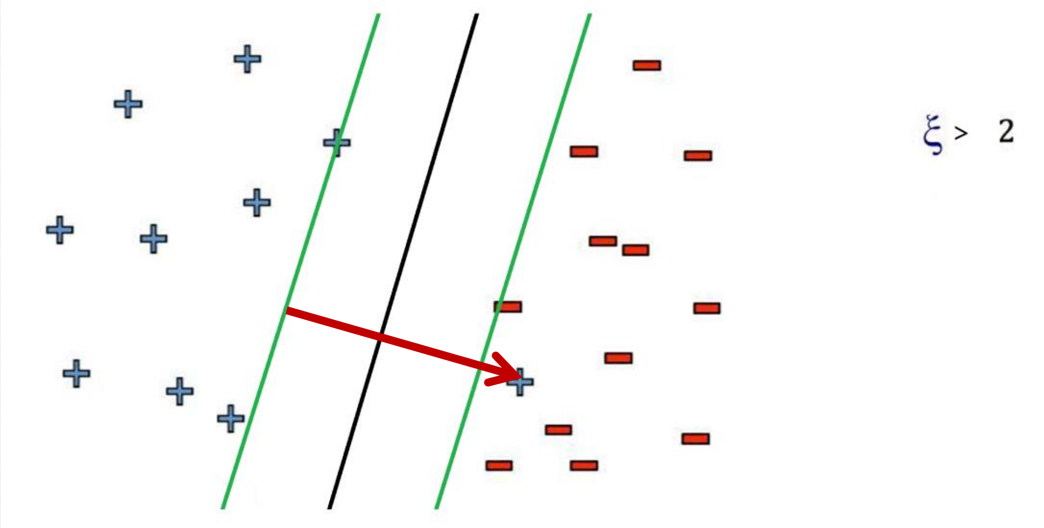
分类：SVM（硬间隔、软间隔、梯度下降算法）

- 软间隔
  - 引入松弛变量$\xi \geq 0$。原有的约束条件变为：
  - $y_i\left(w^T.x_i + b\right) \geq 1 - \xi_i$
- 松弛变量的理解
  - 数据点在
  - 另一侧间隔面上



分类：SVM（硬间隔、软间隔、梯度下降算法）

- 软间隔
  - 引入松弛变量$\xi \geq 0$。原有的约束条件变为：
  - $y_i\left(w^T.x_i + b\right) \geq 1 - \xi_i$
- 松弛变量的理解
  - 数据点
  - 越过另一侧间隔面



分类：SVM（硬间隔、软间隔、梯度下降算法）

- 软间隔
  - 新的目标函数和约束条件
  - $\min(\frac{1}{2}||w|| + C\sum_{i=1}^{n}\xi_i)$
  - C参数称为惩罚因子
    - 惩罚因子的引入
    - 是数学中非常重要的一种处理方法

- 当C的值比较大，那么很小的ξ，都会使得目标函数的值变大
  - 而我们的目标是最小化，所以意味着不能容忍很小的松弛变量(容忍很少的分类错误)
  - 容易过拟合
  - 极端情况是C变得无穷大，软间隔问题，就会退化成一个硬间隔问题
- 当C的值比较小，那么较大的ξ，目标函数都不会受到太大影响
  - 也就是容忍很大的松弛变量(容忍更多的分类错误)
  - 对样本的拟合性下降
  - 可能泛化能力较好

分类：SVM（硬间隔、软间隔、梯度下降算法）

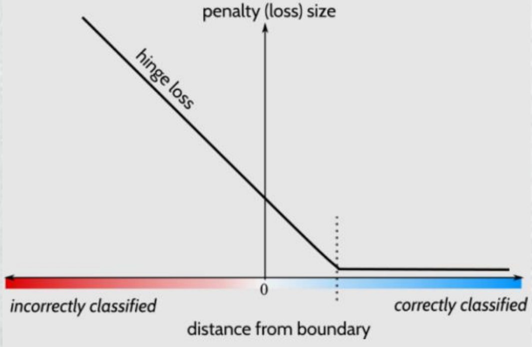
- 软间隔
  - 新的目标函数加上约束条件，有
  - $\min(\frac{1}{2}||w|| + C\sum_{i=1}^{n}\xi_i)$
  - s.t. $y_i(w^T.x_i + b) \geq 1 - \xi_i$, i=1,2,...,n
  - $\xi_i \geq 0$, i=1,2,...,n

分类：SVM（硬间隔、软间隔、梯度下降算法）

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 硬间隔、软间隔：用梯度下降法求解软间隔问题
  - 约束条件$y_i\left(w^T x_i + b\right) \geq 1 - \xi_i$
  - 可以写成$y_i f(x_i) \geq 1 - \xi_i$(注$\xi_i \geq 0$)
  - 即$\xi_i \geq 1 - y_i f(x_i)$
  - 等价于$\xi_i = \max(0,1 - y_i f(x_i))$

问题建模：
min($\frac{1}{2}||w|| + C\sum_{i=1}^{n}\xi_i$)
s.t. $y_i\left(w^T . x_i + b\right) \geq 1 - \xi_i$,
i=1,2,…,n
$\xi_i \geq 0$, i=1,2,…,n



较链hinge损失函数的由来

$\xi_i = \max(0,1 - y_i f(x_i))$的图像(hinge loss)

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 硬间隔、软间隔：用梯度下降法求解软间隔问题
- 我们可以构造SVM的优化目标函数为：
  - $w^* = \min_{w \in R^d} ||w||^2 + C\sum_{i=1}^{N} \max(0, 1 - y_i f(x_i))$
- 这个函数可以改写成另外一种形式，即我们要求解：
  - $w^* = \arg\min_{w} \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - y_i w^T x_i) + \frac{\lambda}{2}||w||^2$

优化$||w||^2$和优化$||w||$是一样的

引入1/2，为了求导数的时候消掉

可以证明，当$\lambda = \frac{2}{C}$时，最小化第2个问题与第1个问题等价

https://u.cs.biu.ac.il/~keshetj/teaching/aml2016/sgd_optimization.pdf

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 硬间隔、软间隔：用梯度下降法求解软间隔问题
- 我们可以构造SVM的优化目标函数为：
  - $w^* = \min_{w \in R^d} ||w||^2 + C\sum_{i=1}^{N} \max(0, 1 - y_i f(x_i))$
- 这个函数可以改写成另外一种形式，即我们要求解：
  - $w^* = \arg\min_{w} \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - y_i w^T x_i) + \frac{\lambda}{2}||w||^2$
- 计算梯度

分两部分

$$\frac{\partial}{\partial w}[\max(0, 1 - y_i w^T x_i)] = \begin{cases} -y_i x_i, & if\ 1 - y_i w^T x_i \geq 0 \\ 0, & if\ 1 - y_i w^T x_i < 0 \end{cases}$$

$$\frac{\partial}{\partial w}\left[\frac{\lambda}{2}||w||^2\right] = \lambda w$$

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 硬间隔、软间隔：用梯度下降法求解软间隔问题
- 我们可以构造SVM的优化目标函数为：
  - $w^* = \min_{w \in R^d} \quad ||w||^2 + C\sum_{i=1}^{N} \max(0, 1 - y_i f(x_i))$
- 这个函数可以改写成另外一种形式，即我们要求解：
  - $w^* = \arg\min_{w} \quad \frac{1}{N}\sum_{i=1}^{N} \max\left(0, 1 - y_i w^T x_i\right) + \frac{\lambda}{2}||w||^2$
- 计算梯度

$$\frac{\partial}{\partial w}[\max\left(0, 1 - y_i w^T x_i\right)] = \begin{cases} -y_i x_i, & if\ 1 - y_i w^T x_i \geq 0 \\ 0, & if\ 1 - y_i w^T x_i < 0 \end{cases}$$

$$\frac{\partial}{\partial w}\left[\frac{\lambda}{2}||w||^2\right] = \lambda w$$

于是就有 $\frac{\partial}{\partial w} Loss = \begin{cases} -yx + \lambda w, & if\ 1 - y w^T x \geq 0 \\ 0 + \lambda w, & if\ 1 - y w^T x < 0 \end{cases}$

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 硬间隔、软间隔：用梯度下降法求解软间隔问题

SVM的梯度下降算法

输入：训练集S={(x_1,y_1), … , (x_N,y_N)}，参数λ, η_0

输出：w_t

算法流程：

1，初始化w=0

2，for t=1, …, T

(1)从样本集随机均匀选择(x_i, y_i)

(2)设置$\eta_t = \eta_0/\sqrt{t}$(注意，随着t不断变大，学习率逐渐变小)

(3) $if\ 1 - y_i w^T x_i \geq 0$

$$w_t = w_{t-1} - \eta_t(-y_i x_i + \lambda w_{t-1})$$

else

$$w_t = w_{t-1} - \eta_t(\lambda w_{t-1})$$

$$\frac{\partial}{\partial w} Loss = \begin{cases} -yx + \lambda w, & if\ 1 - yw^T x \geq 0 \\ 0 + \lambda w, & if\ 1 - yw^T x < 0 \end{cases}$$

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 硬间隔、软间隔：用梯度下降法求解软间隔问题

SVM的梯度下降算法

输入：训练集S={(x_1,y_1), … , (x_N,y_N)}，参数λ, η_0

输出：w_t

算法流程：

1，初始化w=0

2，for t=1, …, T

(1)从样本集随机均匀选择(x_i, y_i)

(2)设置$\eta_t = \eta_0/\sqrt{t}$(注意，随着t不断变大，学习率逐渐变小)

(3) $if\ 1 - y_i w^T x_i \geq 0$

$$w_t = w_{t-1} - \eta_t(-y_i x_i + \lambda w_{t-1})$$

else

$$w_t = w_{t-1} - \eta_t(\lambda w_{t-1})$$

$$\frac{\partial}{\partial w} Loss = \begin{cases} -yx + \lambda w, & if\ 1 - yw^T x \geq 0 \\ 0 + \lambda w, & if\ 1 - yw^T x < 0 \end{cases}$$

在该算法中假设x_i是d维的，那么在上述算法中对x_i进行了变形$x_i = (x_{i1}, x_{i2}, \dots, x_{id}) \Rightarrow (x_{i1}, x_{i2}, \dots, x_{id}, 1)$，同时对 w 进行了变形，$w = (w_1, w_2, \dots, w_d) \Rightarrow (w_1, w_2, \dots, w_d, b)$，以便对w的各个分量和b偏置量统一进行处理

分类：SVM（硬间隔、软间隔、梯度下降算法）

- 硬间隔、软间隔：用梯度下降法求解软间隔问题
  - 最后得到y=$f(x) = w^T x$
- 对于新样本z
  - 可以按照如下式子进行预测，即$\hat{y} = \text{sign}(w^T x)$

关于参数C
- 可以通过交叉验证或者验证集合确定C的值
- C值越大，越可能出现过拟合
- C值越小，越允许错误存在

分类：SVM（硬间隔、软间隔、梯度下降算法）

分类：SVM（硬间隔、软间隔、梯度下降算法）

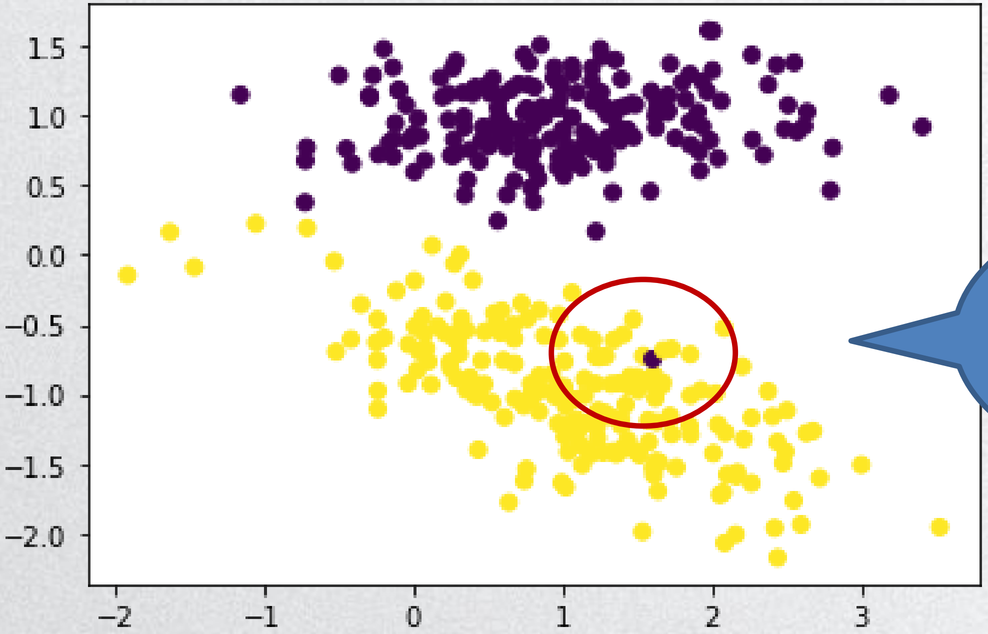
- SVM硬间隔、软间隔实践

名称	类型	大小	修改日期
01SVM_hard_margin_soft_margin_SGD.ipynb	IPYNB 文件	296 KB	2021/10/24 15:14

https://github.com/coding-blocks-archives/machine-learning-online-2018/blob/master/12.%20Support%20Vector%20Machines/SVM.ipynb

分类：SVM（硬间隔、软间隔、梯度下降算法）

- SVM硬间隔、软间隔实践
  - 人工数据集



基本线性可分，
但是有离群值
→软间隔

分类：SVM（硬间隔、软间隔、梯度下降算法）

- SVM硬间隔、软间隔实践
  - SVM类代码

In [102]:
```
class SVM:
    """SVM Class, Author : Prateek Narang"""
    def __init__(self,C=1.0):
        self.C = C
        self.W = 0
        self.b = 0

    def hingeLoss(self,W,b,X,Y):
        loss  = 0.0

        loss += .5*np.dot(W,W.T)

        m = X.shape[0]

        for i in range(m):
            ti = Y[i]*(np.dot(W,X[i].T)+b)
            loss += self.C *max(0,(1-ti))

        return loss[0][0]

    def fit(self,X,Y,batch_size=100,learning_rate=0.001,maxItr=300):

        no_of_features = X.shape[1]
        no_of_samples = X.shape[0]

        n = learning_rate
        c = self.C
```

分类：SVM（硬间隔、软间隔、梯度下降算法）

- SVM硬间隔、软间隔实践
  - SVM类代码（梯度下降）

1.在这里，w向量和b偏置量分开处理
2.采用C，不使用λ
3.可以针对如下优化问题，求解梯度

$$w^* = \min_{w \in R^d} \frac{1}{2}||w||^2 + C\sum_{i=1}^{N} \max(0, 1 - y_i f(x_i))$$

式中$f(x_i) = w^t x_i + b$

$$\frac{\partial}{\partial w}Loss = \begin{cases} w - Cyx, & if\ 1 - yw^Tx \geq 0 \\ w + 0, & if\ 1 - yw^Tx < 0 \end{cases}$$

$$\frac{\partial}{\partial w}Loss = \begin{cases} 0 - Cy, & if\ 1 - yw^Tx \geq 0 \\ 0 + 0, & if\ 1 - yw^Tx < 0 \end{cases}$$

```
for i in range(maxItr):
    #Training Loop

    l = self.hingeLoss(W,bias,X,Y)
    losses.append(l)
    ids = np.arange(no_of_samples)
    np.random.shuffle(ids)

    #Batch Gradient Descent(Paper) with random shuffling
    for batch_start in range(0,no_of_samples,batch_size):
        #Assume 0 gradient for the batch
        gradw = 0
        gradb = 0

        #Iterate over all examples in the mini batch
        for j in range(batch_start,batch_start+batch_size):
            if j<no_of_samples:
                i = ids[j]
                ti =  Y[i]*(np.dot(W,X[i].T)+bias)

                if ti>1:
                    gradw += 0
                    gradb += 0
                else:
                    gradw += c*Y[i]*X[i]
                    gradb += c*Y[i]

        #Gradient for the batch is ready! Update W,B
        W = W - n*W + n*gradw
        bias = bias + n*gradb
```

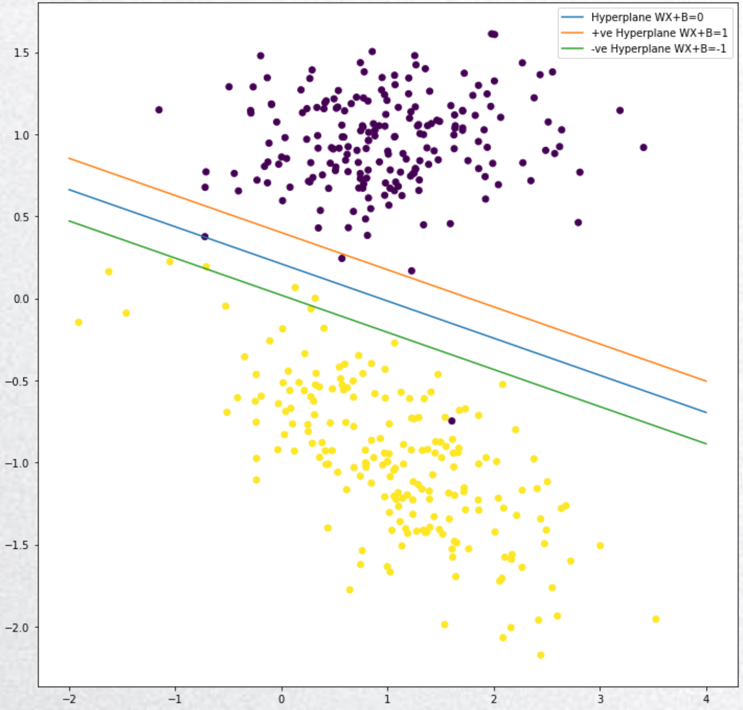
分类：SVM（硬间隔、软间隔、梯度下降算法）

- SVM硬间隔、软间隔实践
  - 比较小的C

```
In  [26]:  mySVM = SVM(C=100)
           W,b,losses = mySVM.fit(X,Y,maxItr=100)
           print(losses[0])
           print(losses[-1])

           plotHyperplane(W[0,0],W[0,1],B)

           40000.0
           596.01861434863
```



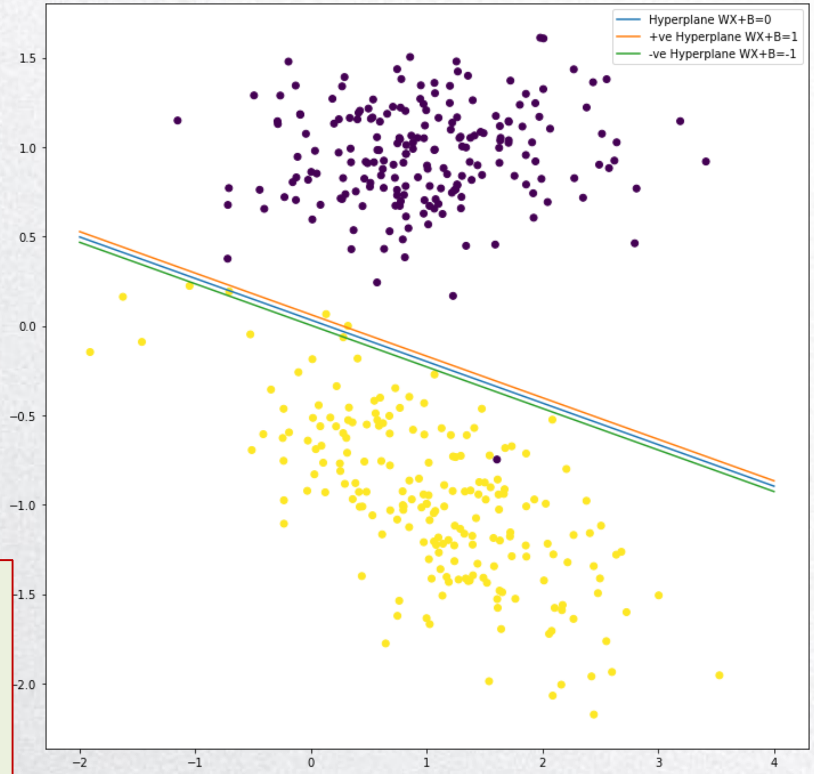
关于参数C
- 可以通过交叉验证或者验证集合确定C的值
- C值越大，越可能出现过拟合
- C值越小，越允许错误存在

分类：SVM（硬间隔、软间隔、梯度下降算法）

- SVM硬间隔、软间隔实践
  - 比较大的C

```
In [28]:  mySVM = SVM(C=1000)
          W,b,losses = mySVM.fit(X,Y,maxItr=100)
          print(losses[0])
          print(losses[-1])

          400000.0
          15880.409133379333
```



关于参数C

- 可以通过交叉验证或者验证集合确定C的值
- C值越大，越可能出现过拟合
- C值越小，越允许错误存在

扩展

分类：SVM（硬间隔、软间隔、梯度下降算法）

情形	问题建模	原问题	对偶问题
线性可分	硬间隔	√	SMO算法
线性可分有少量错误	软间隔	√	SMO算法
非线性可分	核技巧	√	SMO算法

这里分两节讲述

这里未讲

扩展

分类：SVM（硬间隔、软间隔、梯度下降算法）

希腊字母发音对照表

小写

α	β	γ	δ	ε	ζ
Alpha	Beta	Gamma	Delta	Epsilon	Zeta
ν	ξ	ο	π	ρ	σ
Nu	Xi	Omicron	Pi	Rho	Sigma
η	θ	ι	κ	λ	μ
Eta	Theta	Iota	Kappa	Lambada	Mu
τ	υ	φ	χ	ψ	ω
Tau	Upsilon	Phi	Chi	Psi	Omega