Support Vector Machine

Fish

2019 年 6 月 30 日

Content

第一章 perception

1.1 concept

1. 感知机是二类分类的线性分类模型, 其输入空间为实例的特征向量, 输出为实例的类别, 取 +1 和-1 二值.

2. 感知机对应于输入控件（(特征空间) 中将实例划分为正负两类的分离超平面, 是一种线性分类模型, 属于判别模型.

3. 感知机学习旨在求出将训练数据进行线性划分的分离超平面.

4. 导入基于误分类的损失函数, 利用梯度下降法对损失函数进行极小化, 求得感知机模型.

5. $f(x) = sign(w \cdot x + b)$

1.2 linearly separable dataset

给定数据集 $T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, 其中，$x_i \in \chi = \mathrm{R}^n$, $y_i \in \gamma = \{+1, -1\}$, $i = 1, 2, ..., N$.

如果存在某个超平面 S: $w \cdot x + b = 0$ 将数据集的正负实例点完全正确划分到超平面的两侧：

对所有 $y_i = +1$ 的实例 i, 有 $w \cdot x_i + b > 0$

对所有 $y_i = -1$ 的实例 i, 有 $w \cdot x_i + b < 0$

则数据集 T 为线性数据可分数据集, 否则称数据集 T 线性不可分

1.3 loss function

假设训练数据集是线性可分的, 为了找出可将训练集正负实例点完全正确分开的分离超平面, 即确定参数 w, b, 需要确定一个学习策略, 即定义 (经验) 损失函数并将损失函数极小化

两种选择：

1. 选择误分类点的总数, 但这样的损失函数不是参数 w, b 的连续可导函数, 不易优化

2. 选择误分类点到超平面 S 的总距离

   定义输入空间 R^n 中任一点 x_0 到超平面 S 的距离：

$$\frac{1}{\parallel w \parallel}|w \cdot x_0 + b| \tag{1.1}$$

这里 $\parallel w \parallel$ 是 w 的 L_2 范数.

对于误分类的数据 (x_i, y_i) 来说：

$$-y_i(w \cdot w_i + b) > 0 \tag{1.2}$$

成立.

因此, 误分类点 x_i 到超平面 S 的距离是

$$-\frac{1}{\parallel w \parallel} y_i(w \cdot x_i + b) \tag{1.3}$$

这样, 假设超平面 S 的误分类点集合为 M, 则所有误分类点到超平面 S 的总距离为

$$-\frac{1}{\parallel w \parallel} \sum_{x_i \in M} y_i(w \cdot x_i + b) \tag{1.4}$$

不考虑 $\frac{1}{\|w\|}$, 得到感知机 $sign(w \cdot x + b)$ 在训练集 T 的损失函数定义为

$$L(w, b) = -\sum_{x_i \in M} y_i(w \cdot x_i + b) \tag{1.5}$$

1.4 original form of perception algorithm

关键: 感知机算法是误分类驱动的, 当误分类点位于分离超平面的错误一侧时, 调整 w, b 的值, 使分离超平面向该误分类点的一侧移动, 以减少该误分类点与超平面间的距离, 直至超平面越过该误分类点使其被正确分类.

输入: 训练数据集 $T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, 其中，$x_i \in \chi = \mathrm{R}^n$, $y_i \in \gamma = \{+1, -1\}$, $i = 1, 2, ..., N$; 学习率 η $(0 < \eta \leq 1)$.

输出: w, b; 感知机模型 $f(x) = sign(w \cdot x + b)$.

1. 选取初值 w_0, b_0

2. 在训练集中选取数据 (x_i, y_i)

3. 如果 $y_i(w_i + b) \leq 0$

   (a) 损失函极小化 $\min\limits_{w,b} L(w,b) = -\sum\limits_{x_i \in M} y_i(w \cdot x_i + b)$

   (b) 梯度

$$\nabla_w L(w,b) = -\sum_{x_i \in M} y_i x_i \tag{1.6}$$

$$\nabla_b L(w,b) = -\sum_{x_i \in M} y_i \tag{1.7}$$

   (c) 随机选取一个误分类点 (x_i, y_i), 更新参数 w, b

$$w \leftarrow w + \eta y_i x_i \tag{1.8}$$

$$b \leftarrow b + \eta y_i \tag{1.9}$$

4. 转至 (2), 直至训练集中没有误分类点。

1.5 dual form of perception algorithm

基本思路: 将 w 和 b 表示为实例 x_i 和标记 y_i 的线性组合的形式, 通过求解其系数而求得 w 和 b.

输入：训练数据集 $T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, 其中，$x_i \in \chi = \mathrm{R}^n$, $y_i \in \gamma = \{+1, -1\}$, $i = 1, 2, ..., N$; 学习率 η $(0 < \eta \leq 1)$.

输出: α, b; 感知机模型 $f(x) = sign\left(\sum\limits_{j=1}^{N} \alpha_j y_j x_j \cdot x + b\right)$

其中, $\alpha = (\alpha_1, \alpha_2, ..., \alpha_N)^T$, 注意这里是一个向量, 所以后面更新时, α 直接定义步长更新

1. $\alpha \leftarrow 0, b \leftarrow 0$

2. 在训练集中选取数据 (x_i, y_i)

3. 如果 $y_i\left(\sum\limits_{j=1}^{N} \alpha_j y_j x_j \cdot x_i + b\right)$

   (a) 损失函极小化 $\min\limits_{w,b} L(w,b) = \min\limits_{\alpha,b} L(\alpha,b) = -\sum\limits_{x_i \in M} y_i(\alpha_j y_j x_j \cdot x_i + b)$

   (b) 梯度

$$\nabla_b L(w,b) = -\sum_{x_i \in M} y_i \tag{1.10}$$

(c) 随机选取一个误分类点 (x_i, y_i), 更新参数 w, b

$$\alpha \leftarrow \alpha + \eta \tag{1.11}$$

$$b \leftarrow b + \eta y_i \tag{1.12}$$

4. 转至 (2) 直到没有误分类数据.

注意: 对偶形式中训练实例仅以内积的形式出现.

为了方便, 预先将训练集中实例间的内积计算出来并以矩阵的形式存储，即 Gram 矩阵

$$G = [x_i \cdot x_j] \tag{1.13}$$

第二章 Convex function

2.1 Convex collection and Convex function

2.1.1 Convex function

$y = x^2$ 是凸函数, 函数图像上位于 $y = x^2$ 上方的区域构成凸集.

1. 凸函数图像的上方区域, 一定是凸集
2. 一个函数图像的上方区域为凸集, 则该函数是凸函数

2.1.2 Convex collection

1. Afiine set 仿射集
   (a) define: 通过集合 C 中任意两个不同点的直线仍然在集合 C 内, 则称集合 C 为仿射集.

$$\forall x_1, x_2 \in C, \forall \theta \in R, \text{ 则 } x = \theta \cdot x_1 + (1-\theta) \cdot x_2 \in C \tag{2.1}$$

   (b) 仿射集的例子: 直线、平面、超平面
   n 维空间的 $n-1$ 维仿射集为 $n-1$ 维超平面
2. Convex 凸集
   两种表述 (可以思考其内涵一样吗) :
   (a) 集合 C 内任意两点间的线段均在集合 C 内, 则称集合 C 为凸集.

$$\forall x_1, x_2 \in C, \forall \theta \in [0,1], \text{ 则 } x = \theta \cdot x_1 + (1-\theta) \cdot x_2 \in C \tag{2.2}$$

   (b) k 个点的版本:

$$\forall x_1, x_2, ..., x_k \in C, \theta_i \in [0,1] \text{且} \sum_{i=1}^{k} \theta_i x_i \in C \tag{2.3}$$

3. 因为仿射集的条件比凸集的条件强, 所以, 仿射集必然是凸集

4. judgement of convex collection

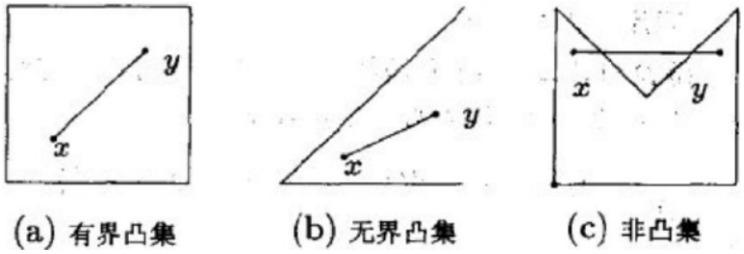


Fig 2.1: convex collection

2.2 affine transformation

2.2.1 concept

函数 $f(x) = Ax + b$ 的形式, 称函数是放射的: 即线性函数加常数的形式

2.2.2 theory

1. 仿射变换 $f(x) = Ax + b$, $A \in \mathrm{R}^{m \times n}$, $b \in \mathrm{R}^m$

   (a) 伸缩、平移、投影

2. 若 f 是仿射变换, $f : \mathrm{R}^n \to \mathrm{R}^m \quad f(S) = \{f(x) | x \in S\}$

   (a) 若 S 为凸集, 则 $f(x)$ 为凸集;

   (b) 若 $f(S)$ 为凸集, 则 S 为凸集.

3. 两个凸集的和为凸集

$$S_1 + S_2 = \{x + y | x \in S_1, y \in S_2\} \tag{2.4}$$

4. 两个凸集的笛卡尔积 (直积) 为凸集

$$S_1 \times S_2 = \{(x_1, x_2) | x \in S_1, x_2 \in S_2\} \tag{2.5}$$

5. 两个集合的部分和为凸集 (分配率)

$$S = \{(x, y_1 + y_2) | (x, y_1) \in S_1, (x, y_2) \in S_2\} \tag{2.6}$$

2.3 perspective collineation

2.3.1 concept

透视函数对向量进行伸缩 (规范化), 使得最后一维的分量为 1 并舍弃之.

$$P : \mathrm{R}^{n+1} \to \mathrm{R}^n, \ P(z,t) = z/t \tag{2.7}$$

2.3.2 direct significance

小孔成像

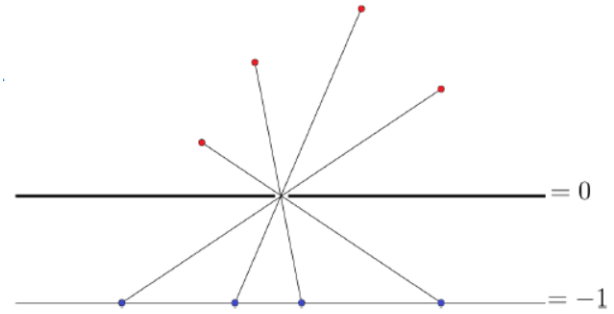


Fig 2.2: significance of perspective collineation

2.3.3 summary

凸集的透视变换仍然是凸集

2.4 segmentation plane

2.4.1 concept

设 C 和 D 为两不相交的凸集, 则存在超平面 P, P 可以将 C 和 D 分离.

$$\forall x \in C, a^T x \leq b \text{ 且 } \forall x \in D, a^T x \geq b \tag{2.8}$$

2.4.2 segmentation plane

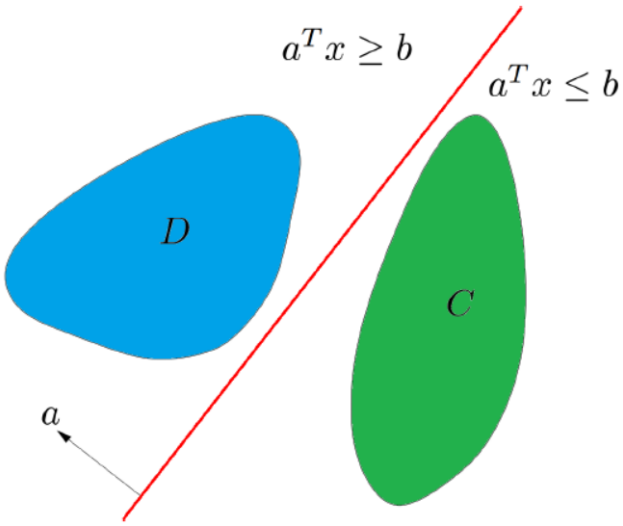


Fig 2.3: segmentation plane

2.4.3 the structure of sefmentation plane

1. 两个集合的距离, 定义为两个集合间元素的最短距离
2. 做集合 C 和集合 D 最短线段的垂直平分线
3. picture

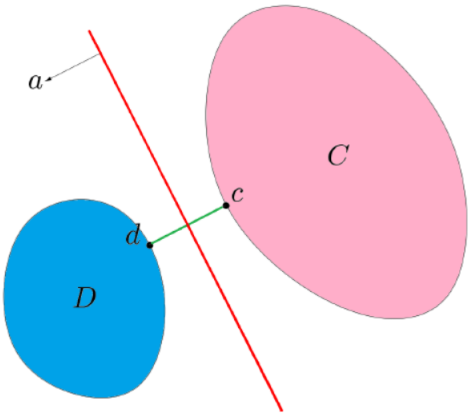


Fig 2.4: the distance of segmentation plane

2.4.4 support hyperplane

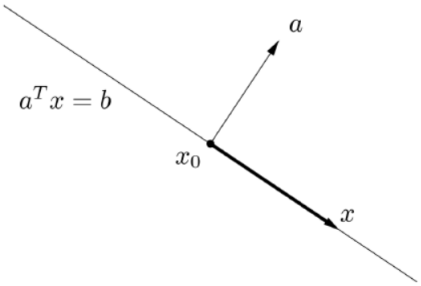


Fig 2.5: hyperplane

1. 设集合 C , x_0 为 C 边界上的点. 若存在 $a \neq 0$, 满足对任意 $x \in C$, 都有 $a^T x \leq a^T x_0$ 成立, 则称超平面 $\{x | a^T x = a^T x_0\}$ 为集合 C 在点 x_0 处的支撑超平面.

2. 凸集边界上任意一点, 均存在支撑超平面

3. 反之, 若在一个闭的非中空 (内部点不为空) 集合, 在边界上的任意一点存在支撑超平面, 则该集合为凸集

2.4.5 question

1. 如何定义两个集合的“最优”分割超平面?

   (a) 找到集合“边界”上的若干点, 以这些点为“基础”计算超平面的方向; 以两个集合边界上的这些点的平均作为超平面的“截距”

   (b) 支持向量: support vector

   (c) optimal segmentation plane

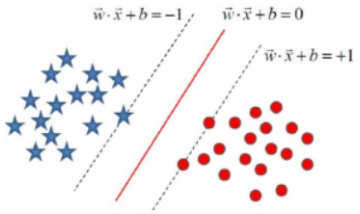


Fig 2.6: optimal segmentation plane

2. 若两个集合有部分相交, 如何定义超平面, 使得两个集合“尽量”分开?

   (a) 注: 上述“集合”不一定是凸集, 可能是由若干离散点组成. 若一组集合为 $(x, 1)$, 另一组集合为 $(x, 2)$, 则为机器学习中的分类问题.

2.5 property of convex function

2.5.1 concept

若函数 $fx(x)$ 的定义域 domf 为凸集, 且满足

$$\forall x, y \in \text{dom } f,\ 0 \leq \theta \leq 1,\ \text{有} \tag{2.9}$$

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y) \tag{2.10}$$

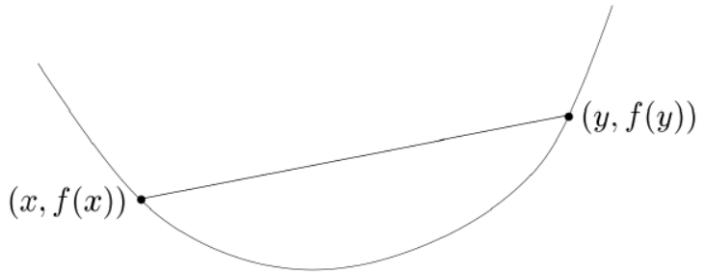


Fig 2.7: convex function

2.5.2 first-order derivative of convex function

若 $f(x)$ 一阶可微, 则函数 f 为凸函数当前仅当 f 的定义域 domf 为凸集, 且

$$\forall x, y \in \text{dom } f,\ f(y) \geq f(x) + \nabla f(x)^T (y - x) \tag{2.11}$$

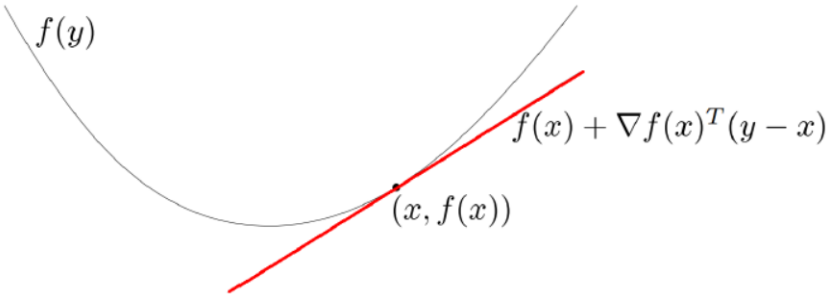


Fig 2.8: first-order derivative of convex function

对于凸函数, 其一阶 Taylor 近似本质上是该函数的全局下估计

2.5.3 second-order derivative of convex function

1. 若函数 f 二阶可微, 则函数 f 为凸函数当前仅当 dom 为凸集, 且

$$\nabla^2 f(x) \succ= 0 \tag{2.12}$$

2. 若 f 是一元函数, 上式表示二阶导大于等于 0

3. 若 f 是多元函数, 上式表示二阶导 Hessian 矩阵半正定

2.6 examples of convex function

- 指数函数 $f(x) = e^{ax}$
- 幂函数 $f(x) = x^a, x \in R^+, a \geq 1$ 或 $a \leq 0$
- 负对数函数 $f(x) = -\ln x$
- 负熵函数 $f(x) = x \ln x$
- 范数函数 $f(\vec{x}) = \| x \|$
- 最大值函数 $f(\vec{x}) = \max(x_1, x_2, ..., x_n)$
- 指数线性函数 $f(\vec{x}) = \log(e^{x_1} + e^{x_2} + ... + e^{x_n})$

第三章 Convex Optimization

3.1 optimization problem

1. common format

$$\begin{aligned} \text{minimize } & f_0(x), x \in \mathrm{R}^n \\ \text{subject to } & g_i(x) \le 0,\ i = 1, ..., m \\ & h_j(x) = 0,\ j = 1, ..., p \\ \text{优化变量} \quad & x \in \boldsymbol{R}^n \\ \text{不等式约束} \quad & g_i(x) \le 0 \\ \text{等式约束} \quad & h_j(x) = 0 \\ \text{无约束优化} \quad & m = p = 0 \end{aligned} \tag{3.1}$$

2. domain of optimal problem

$$D = \bigcap_{i=0}^{m} \mathrm{dom} g_i \cap \bigcup_{j=1}^{p} \mathrm{dom} h_j \tag{3.2}$$

3. feasible dot (solution)

   - $x \in D$, 满足式 3.1 中约束条件

4. feasible domain (feasible collection)

   - 所有可行点的集合

5. optimization value

$$p^* = \inf\{f_0(x) | g_i(x) \le 0, i = 1, ..., m, h_j(x) = 0, j = 1, ..., p\} \tag{3.3}$$

6. optimization solution

$$p^* = f_0(x^*) \tag{3.4}$$

3.2 convex optimization

3.2.1 basic form

$$\begin{aligned} \text{minimize} \quad & f_0(x), x \in \mathrm{R}^n \\ \text{subject to} \quad & g_i(x) \leq 0, i = 1, ..., m \\ & h_j(x) = 0, j = 1, ..., p \end{aligned} \tag{3.5}$$

1. among, $g_i(x)$ 为凸函数, $h_j(x)$ 为仿射函数
2. important property of convex optimization problem
   (a) 凸优化问题的可行域为凸集
   (b) 凸优化问题的局部最优解即为 全局最优解

3.2.2 Lagrange 乘子法

在支持向量机模型 (SVM) 的推导中一步很关键的就是利用 Lagrange 对偶性将原问题转化为对偶问题.

1. theoty:
   一般的求极值问题, 求导等于 0. 但是如果不但要求极值, 还要求一个满足一定约束条件的极值, 那么此时就可以构造 Lagrange 函数, 其实就是 把约束项添加到原函数上, 然后对构造的新函数求导
2. example picture

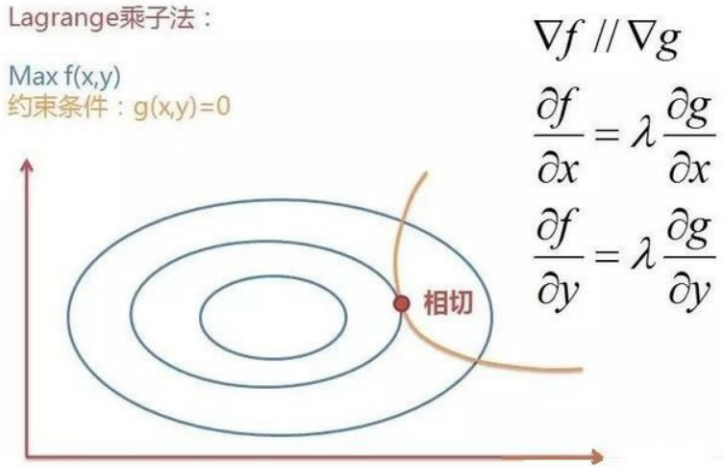


Fig 3.1: Lagrange multiplier method

3. analysis:
   对于一个要求极值的函数 $f(x,y)$, 图上的蓝圈就是这个函数的等高图, 就是说 $f(x,y)=c_1,c_2,...,c_n$ 分别代表不同的数值 (每个值代表一圈，等高图), 我要找到一组 (x,y), 使它的 c_i 值越大越好, 但是这点必须满足约束条件 $g(x,y)$ (在黄线上)

4. conclusion:
   就是说 $f(x,y)$ 和 $g(x,y)$ 相切, 或者说它们的梯度 ∇f 和 ∇g 平行, 因此它们的梯度 (偏导) 成倍关系; 那我们就假设为 λ 倍, 然后把约束条件加到原函数后再对它求导, 其实就等于满足了图上的式子了.

3.2.3 Lagrange 函数的极小极大最优值 $\min\limits_{x}\max\limits_{\lambda,\nu:\lambda_i\geq 0} L(x,\lambda,\nu)$ 和原问题的最优值 p^*

1. 一般优化问题的 Lagrange 乘子法

$$\begin{aligned} \text{minimize} \quad & f_0(x), x\in \mathrm{R}^n \\ \text{subject to} \quad & g_i(x)\leq 0, i=1,...,m \\ & h_j(x)=0, j=1,...,p \end{aligned} \tag{3.6}$$

2. Lagrange 函数

$$L(x,\lambda,\nu)=f_0(x)+\sum_{i=1}^{m}\lambda_i g_i(x)+\sum_{j=1}^{p}\nu_j h_j(x) \tag{3.7}$$

   对固定的 x, Lagrange 函数 $L(x,\lambda,\nu)$ 为关于 λ 和 ν 的仿射函数, 这里, $x=(x^1,x^2,...,x^n)^T\in \mathrm{R}^n, \lambda,\nu$ 是 Lagrange 乘子, $\lambda\geq 0$

3. Lagrange 函数的极小极大最优值 $\min\limits_{x}\max\limits_{\lambda,\nu:\lambda_i\geq 0} L(x,\lambda,\nu)$ 和原问题的最优值 p^* 之间的关系

   考虑 x 的函数:

$$f(x)=\max_{\lambda,\nu:\lambda\geq 0} L(x,\lambda,\nu) \tag{3.8}$$

   即原问题 $f(x)$ 与 Lagrange 函数在乘子变量最大化时的值是等价的

4. 证明上式: 假设给定某个 x，如果 x 违反原始问题的约束条件, 即存在某个 i 使得 $g_i(x)>0$ 或者存在某个 j 使得 $h_j(x)\neq 0$, 那么就有

$$\phi_P(x)=\max_{\lambda,\nu:\lambda\geq 0}\left[f(x)+\sum_{i=1}^{m}\lambda_i g_i(x)+\sum_{i=1}^{p}\nu_i h_i(x)\right]=+\infty \tag{3.9}$$

   这里, 下标 P 表示原始问题

5. 因为若某个 i 使约束 $g_i(x) > 0$, 则可令 $\lambda_i \to +\infty$, 若某个 j 使 $h_j(x) \neq 0$, 则可令 ν_i 使 $\nu_j h_j(x) \to +\infty$, 而将其余各 λ_i, ν_j 均取为 0

   如果 x 满足约束条件, 则由式 3.7 和式 3.8 可知, $\phi_P(x) = f(x)$

6. summary:

$$\phi_P(x) = \begin{cases} f(x), & x \text{ 满足原始问题约束} \\ +\infty, & \text{其他} \end{cases} \tag{3.10}$$

   所以如果考虑极小化问题

$$\min_x \phi_P(x) = \min_x \max_{\lambda,\nu:\lambda\geq 0} L(x, \lambda, \nu) \tag{3.11}$$

   它是与原始最优化问题 3.1 等价的, 即它们有相同的解

   此时将原始最优化问题表示为广义 Lagrange 函数的极小极大问题.

   为了方便, 定义原始问题的最优值

$$p^* = \min_x \phi_P(x) = \min f(x) \tag{3.12}$$

   称为原始问题的值

3.2.4 Lagrange 对偶函数 (dual function)

1. Lagrange 对偶函数, 注意此处 下式中 x 需不需要改成 $\widetilde{x}$

$$\Gamma(\lambda, \nu) = \inf_{x\in D} L(x, \lambda, \nu) = \inf_{x\in D}(f_0(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)) \tag{3.13}$$

2. 若 $\widetilde{x} \in \mathbb{D}$ 为主问题 3.1 可行域中的点, 则对任意 ν 和 $\lambda \succeq 0$ 都有

$$\sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{n} \nu_j h_j(x) \leq 0 \tag{3.14}$$

   进而有

$$\Gamma(\lambda, \nu) = \inf_{x\in D} L(x, \lambda, \nu) \leq \inf L(\widetilde{x}, \lambda, \nu) \leq f(\widetilde{x}) \tag{3.15}$$

3. 若没有下确界, 定义 主问题下界:

$$\Gamma(\lambda, \nu) = -\infty \tag{3.16}$$

4. 根据定义, 显然有: 对 $\forall \lambda > 0$, $\forall \nu$, 若原优化问题有最优值 p^*, 则

$$\Gamma(\lambda, \nu) \leq p^* \tag{3.17}$$

即对偶函数给出了主问题的最优值的下界, 显然, 这个下界取决于 λ 和 ν 的值. 于是, 一个很自然的问题是: 基于对偶函数能获得的最好下界是什么?

5. 上述引出了 对偶优化问题

$$\max \Gamma(\lambda, \nu) \quad \text{s.t. } \lambda \succeq 0 \tag{3.18}$$

式 3.18 就是主问题 3.1 的对偶问题, 其中 λ 和 ν 称为“对偶变量”(dual variable)

6. 进一步: 无论主问题的 3.1 的凸性如何, 对偶问题 Lagrange 对偶函数 3.18 始终是凹函数

7. 假设式 3.18 的最优值为 $d^* = \max\limits_{\lambda,\nu} \Gamma(\lambda, \nu)$, 也即主问题 3.1 中最优值 p^* 的下界

8. 显然有 $d^* = \max\limits_{\lambda,\nu} \Gamma(\lambda, \nu) \leq p^*$, 这称为“弱对偶性”(weak duality) 成立;

若 $d^* = \max\limits_{\lambda,\nu} \Gamma(\lambda, \nu) = p^*$, 则称为“强对偶性”(strong duality) 成立, 此时由对偶问题能获得主问题的最优下界.

9. 对于一般优化问题, 强对偶性通常不成立. 但是, 若主问题为凸优化问题, 如式 3.1 中 $f(x)$ 和 $g_j(x)$ 均为凸函数, $h_i(x)$ 为仿射函数, 且其可行域中至少有一点使不等式约束严格成立, 此时强对偶性成立. 注意: 在强对偶性成立时, 将 Lagrange 函数对原变量求导, 并令导数等于 0, 即可得到原变量与对偶变量的数值关系.

10. 上述分析, 若满足强对偶性的 KKT 条件, 此时:

$$\begin{aligned} d^* &= \max_{\lambda,\nu:\lambda_i \geq 0} \min_x L(x, \lambda, \nu) = \min_x \max_{\lambda,\nu:\lambda_i \geq 0} L(x, \lambda, \nu) \\ &= p^* = \min f(x) \quad \{x \text{ subject to 约束条件}\} \end{aligned} \tag{3.19}$$

11. prove:
若原始问题和对偶问题都有最优值, 则 $d^* = \max\limits_{\lambda,\nu:\lambda_i \geq 0} \min\limits_x L(x, \lambda, \nu) \leq \min\limits_x \max\limits_{\lambda,\nu:\lambda_i \geq 0} L(x, \lambda, \nu) = p^* = \min f(x) \quad \{x \text{ subject to 约束条件}\}$

process:

对任意的 λ, ν 和 x, 有

$$L(x, \lambda, \nu) \leq \max_{\lambda,\nu:\lambda\geq 0} L(x, \lambda, \nu) \tag{3.20}$$

$$\therefore \quad \Gamma_D(\lambda, \nu) = \min_x L(x, \lambda, \nu) \leq L(x, \lambda, \nu) \leq \max_{\lambda,\nu:\lambda\geq 0} L(x, \lambda, \nu) = \phi_P(x) \tag{3.21}$$

‘ 即:

$$\Gamma_D(\lambda, \nu) \leq \phi_P(x) \tag{3.22}$$

由于原始问题和对偶问题均有最优值

$$\therefore \quad \max_{\lambda,\nu:\lambda\geq 0} \Gamma_D(\lambda, \nu) \leq \min_x \phi_P(x) \tag{3.23}$$

即:

$$\begin{aligned} d^* &= \max_{\lambda,\nu:\lambda_i\geq 0} \min_x L(x, \lambda, \nu) \leq \min_x \max_{\lambda,\nu:\lambda_i\geq 0} L(x, \lambda, \nu) \\ &= p^* = \min f(x) \quad \{x \text{ subject to 约束条件}\} \end{aligned} \tag{3.24}$$

12. KKT 条件:

$$g_i(x^*) \leq 0, \quad i = 1, 2, .., k \tag{3.25}$$

$$\lambda_i^* \geq 0, \quad i = 1, 2, ..., k \tag{3.26}$$

$$\lambda_i^* g_i(x^*) = 0, \quad i = 1, 2, ..., k \tag{3.27}$$

$$h_j(x^*) = 0, \quad j = 1, 2, ..., l \tag{3.28}$$

$$\nabla L(x^*, \lambda^*, \nu^*) = \nabla f(x^*) + \sum_{i=1}^{k} \lambda_i^* \nabla g_i(x^*) + \sum_{j=1}^{l} \nu_i^* \nabla h_j(x^*) = 0 \tag{3.29}$$

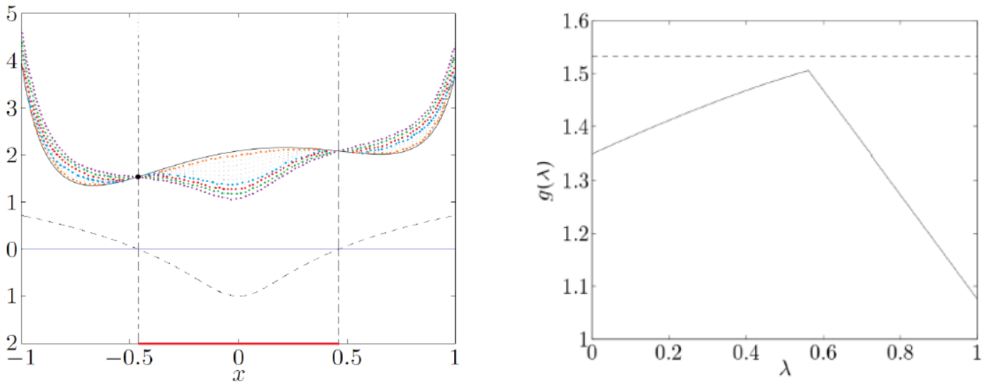
上式中 3.29 等价于

$$\begin{cases} \nabla_x L(x^*, \lambda^*, \nu^*) = 0 \\ \nabla_\lambda L(x^*, \lambda^*, \nu^*) = 0 \\ \nabla_\nu L(x^*, \lambda^*, \nu^*) = 0 \end{cases} \tag{3.30}$$

特别指出, 式 3.27 称为 KKT 的对偶互补条件. 由此条件可知: 若 $\lambda_i^* > 0$, 则 $g_i(x^*) = 0$

上式中, x^* 是原始变量的解, λ^*, ν^* 是对偶变量的解, 并且 $p^* = d^* = L(x^*, \lambda^*, \nu^*)$

13. 左侧是原函数, 右侧为对偶函数



(a) Original function (b) Dual function

Fig 3.2: dual problem

SVM 里的 w,b 看做是 Lagrange 函数里的变量 x

第四章 SVM

4.1 linear separable support vector machine and hard margin maximization

4.1.1 question?

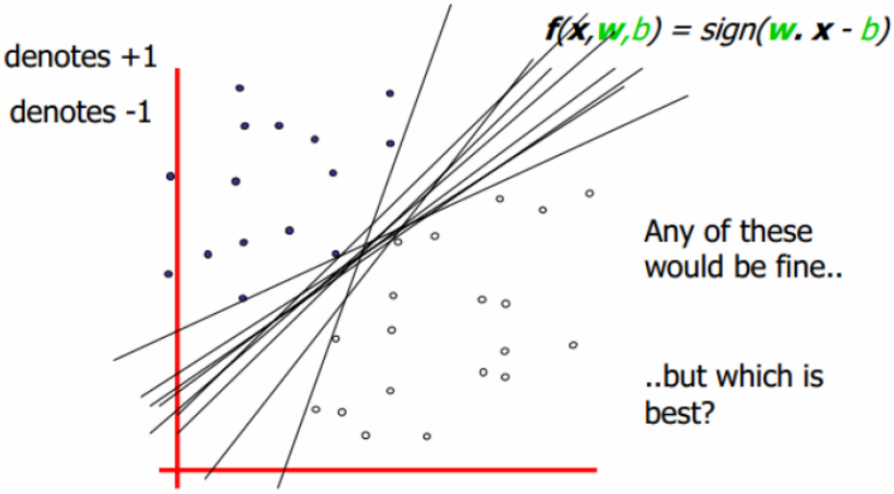


Fig 4.1: linear classifier

4.1.2 input data

1. 假设给定有一个特征空间上的训练数据集 $T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ 其中, $x_i \in \mathrm{R}^n, y_i \in \{+1, -1\}, i = 1, 2, ..., N$
2. x_i 为第 i 个实例 (若 $n > 1$, x_i 为向量)
3. y_i 为 x_i 的类标记
   (a) 当 $y = +1$ 时, 称 x_i 为正例
   (b) 当 $y = -1$ 时, 称 x_i 为负例
4. (x_i, y_i) 称为样本点

4.1.3 linear separable support vector machine

1. 给定线性可分训练数据集, 通过 间隔最大化 得到的分离超平面为

$$y(x) = w^T\phi(x) + b \tag{4.1}$$

   相应的分类决策函数

$$f(x) = sign(w^T\phi(x) + b) \tag{4.2}$$

   该决策函数称为线性可分支持向量机

2. $\phi(x)$ 是某个确定的特征空间转换函数, 它的作用是将 x 映射到 (更高的) 维度

   (a) 最简单直接的: $\phi(x) = x$

3. 求解分离超平面问题可以等价为求解相应的 凸二次规划问题

4.1.4 list symbol

1. 分割平面: $y(x) = w^T\phi(x) + b$
2. 训练集: $x_1, x_2, ..., x_n$
3. 目标值: $y_1, y_2, ..., y_n, \quad y_i \in \{-1, +1\}$
4. 新数据的分类: $sign(y(x))$

4.1.5 derivation target function

1. 根据题设 $y(x) = w^T\phi(x) + b$
2. 有:

$$\begin{cases} y(x_i) > 0 \Leftrightarrow y_i = +1 \\ y(x_i) < 0 \Leftrightarrow y_i = -1 \end{cases} \Rightarrow y_i \cdot y(x_i) > 0 \tag{4.3}$$

3. functional margin and geometric margin

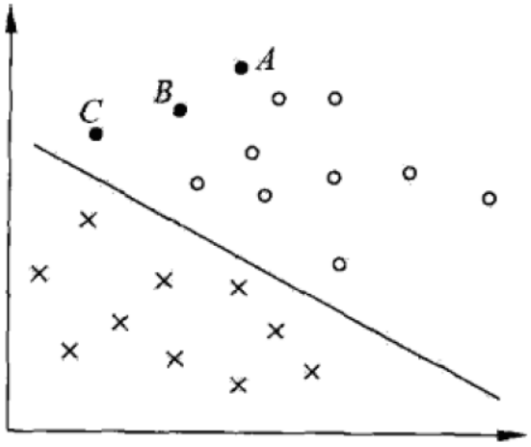


Fig 4.2: functional margin and geometric margin

在图 4.2 中，有 A,B,C 三个点, 表示 3 个实例, 均在分离超平面的正类侧, 预测它们的类, 点 A 距分离超平面较远, 若预测该点为正类, 就比较确信预测是正确的, 点 C 距分离超平面较劲, 若预测该点为正类就不么确信, 点 B 介于点 A 与 C 之间, 预测其为正类的确信度也在 A 与 C 之间

- 函数间隔：一般来说, 一个点距离分离超平面的远近可以表示分类预测的确信程度. 在超平面 $w^T\phi(x)+b=0$ 确定的情况下, $|w^T\phi(x)+b|$ 能够相对地表示点 x 距离超平面的远近, 而 $w^T\phi(x)+b$ 的符号与类标记 y 的符号是否一致能够表示分类是否正确. 所以可用量 $y_i\cdot(w^T\cdot\phi(x_i)+_b$ 来表示分类的正确性及确信度.

  定义超平面 (w,b) 关于样本点 (x_i,y_i) 的函数间隔为

$$\hat{\gamma}_i = y_i(w\cdot x_i+b) \tag{4.4}$$

- 几何间隔: 函数间隔可以表示分类预测的正确性及确信度. 但是选择分离超平面时, 只有函数间隔还不够, 因为只要成比例的缩放 w 和 b，例如将它们改为 $2w$ 和 $2b$, 超平面并没有改变, 但函数间隔却成为原来的 2 倍, 这一事实启示我们, 可以对分离超平面的法向量 $\boldsymbol{w}$ 加某些约束, 如规范化, $\parallel w \parallel = 1$, 使得间隔是确定的. 这是函数间隔称为几何间隔.

  w,b 等比例缩放, 则 $y_i * y$ 的值同样缩放, 从而：

$$\gamma_i = \frac{y_i\cdot y(x_i)}{\parallel w \parallel} = \frac{y_i\cdot(w^T\cdot\phi(x_i)+_b)}{\parallel w \parallel} \tag{4.5}$$

4. maximum margin to select separable hyperplane

- assume minimum functional margin and geometric margin

  定义函数间隔平面 (w, b) 关于 T 中所有样本点 (x_i, y_i) 的函数间隔之最小值, 即

$$\hat{\gamma} = \min_{i=1,\dots,N} \hat{\gamma}_i \tag{4.6}$$

  定义几何间隔平面 (w, b) 关于 T 中所有样本点 (x_i, y_i) 的几何间隔之最小值, 即

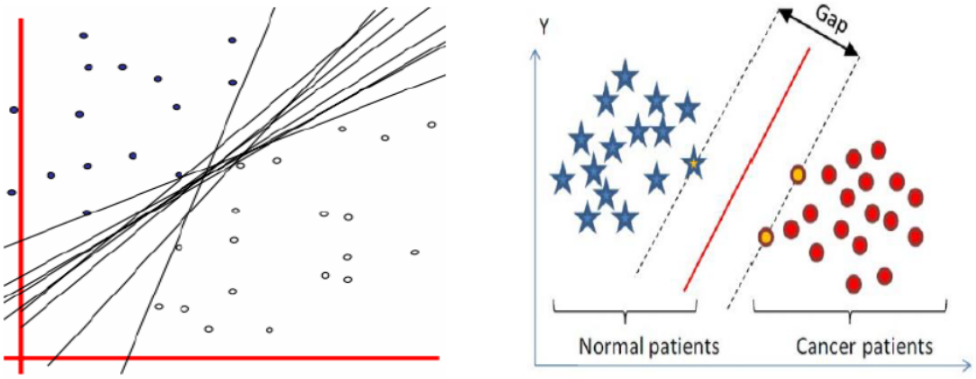
$$\gamma = \min_{i=1,\dots,N} \gamma_i \tag{4.7}$$

- original target function

$$\arg\max_{w,b} \gamma = \arg\max_{w,b} \frac{\hat{\gamma}}{\| w \|} \Rightarrow \tag{4.8}$$

$$\arg\max_{w,b} \left\{ \frac{1}{\| w \|} \min_i \left[y_i \cdot (w^T \cdot \phi(x_i) + b) \right] \right\} \tag{4.9}$$

$$\text{s.t.} \quad \frac{y_i \cdot (w^T \cdot \phi(x_i) + b)}{\| w \|} \geq \gamma \tag{4.10}$$



(a) many optional hyperplane (b) margin separable hyperplane

Fig 4.3: maximum margin

- optimal problem

  函数间隔 $\hat{\gamma} = y(w \cdot x + b)$ 不影响最优化问题的解. 事实上, 假设将 w 和 b 按比例改变为 λw λb, 这时函数间隔成为 $\lambda\hat{\gamma}$.

  函数间隔的改变对上面最优化问题的不等式约束没有影响, 对目标函数的优化也没有影响. 即它产生一个等价的优化问题.

取函数间隔最小值 $\hat{\gamma}=1$. 将 $\hat{\gamma}=1$ 代入上面的最优化问题, 注意到最大化 $\frac{1}{\|w\|}$ 和最小化 $\frac{1}{2}\|w\|^2$ 是等价的. 则得到下面的 线性可分支持向量机 的最优化问题:

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 \tag{4.11}$$

$$\text{s.t.} \quad y_i(w\cdot x_i+b)-1\geq 0, \quad i=1,2,...,N \tag{4.12}$$

这是一个凸二次规划 (convex quadratic programming) 问题: 目标函数 $f(x)$ 是二次函数且约束函数 $g_i(w)$ 是仿射函数时, 约束最优化问题 (凸优化问题) 成为凸二次规划问题

5. dual algorithm

   (a) Lagrange multiplier method to build Lagrange function

$$L(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i(y_i(w^T\cdot\phi(x_i)+b)-1) \tag{4.13}$$

   among, $\alpha=(\alpha_1,\alpha_2,...,\alpha_N)^T$ is Lagrange multiplier vector.

   (b) 原问题是极小极大问题

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

   原始问题的对偶问题, 是极大极小问题

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

   (c) 得到对偶问题的解

      (1) 求 $\min\limits_{w,b}L(w,b,\alpha)$

      将拉格朗日函数 $L(w,b,\alpha)$ 分别对 w,b 求偏导数并令其等于 0.

$$\nabla_w L(w,b,\alpha)=w-\sum_{i=1}^{N}\alpha_i y_i x_i=0 \tag{4.14}$$

$$\nabla_b L(w,b,\alpha)=\sum_{i=1}^{N}\alpha_i y_i=0 \tag{4.15}$$

$$\text{得} \tag{4.16}$$

$$w=\sum_{i=1}^{N}\alpha_i x_i y_i \tag{4.17}$$

$$\sum_{i=1}^{N}\alpha_i y_i=0 \tag{4.18}$$

(2) 将式 4.17 代入 Lagrange 函数 4.13, 并利用式 4.18, 即得

$$L(w,b,\alpha) = \frac{1}{2} \parallel w \parallel^2 - \sum_{i=1}^{n} \alpha_i (y_i(w^T \cdot \phi(x_i) + b) - 1) \tag{4.19}$$

$$= \frac{1}{2} w^T w - w^T \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) - b \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i \tag{4.20}$$

$$= \frac{1}{2} w^T \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) - w^T \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) - b \cdot 0 + \sum_{i-1}^{n} \alpha_i \tag{4.21}$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \left(\sum_{i=1}^{n} \alpha_i y_i \phi(x_i) \right)^T \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) \tag{4.22}$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \phi^T(x_i) \phi(x_j) \tag{4.23}$$

即

$$\min_{w,b} L(w,b,\alpha) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_i \alpha_j y_i y_j \phi^T(x_i) \phi(x_j) + \sum_{i=1}^{N} \alpha_i \tag{4.24}$$

(3) 求 $\min\limits_{w,b} L(w,b,\alpha)$ 对 α 的极大, 即是对偶问题

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\phi(x_i) \phi(x_j) \right) \tag{4.25}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{4.26}$$

$$\alpha_i \geq 0, \quad i = 1, 2, ..., n \tag{4.27}$$

(4) 整理目标函数: 添加负号

$$\min_{\alpha} \quad \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\phi(x_i) \phi(x_j) \right) - \sum_{i=1}^{n} \alpha_i \tag{4.28}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{4.29}$$

$$\alpha_i \geq 0, \quad i = 1, 2, ..., n \tag{4.30}$$

4.1.6 Linear divisible support Vector Machine Learning algorithm

1. 构造并求解约束最优化问题

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \left(\phi(x_i)\phi(x_j)\right) - \sum_{i=1}^{n}\alpha_i \tag{4.31}$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{4.32}$$

$$\alpha_i \geq 0, \quad i = 1, 2, ..., n \tag{4.33}$$

2. 求得最优解 α^*, $\alpha^* = (\alpha_1^*, \alpha_1^*, ..., \alpha_1^*)^T$

3. 计算

$$w^* = \sum_{i=1}^{N}\alpha_i^* y_i \phi(x_i) \tag{4.34}$$

$$b^* = y_i - \sum_{i=1}^{N}\alpha_i^* y_i (\phi(x_i)\cdot\phi(x_j)) \tag{4.35}$$

4. 求得分离超平面

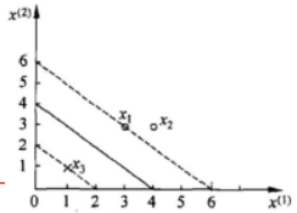
$$w^*\phi(x) + b^* = 0 \tag{4.36}$$

5. 分类决策函数

$$f(x) = sign(w^*\phi(x) + b^*) \tag{4.37}$$

6. example

举例



- 给定3个数据点：正例点$x_1=(3,3)^T$，$x_2=(4,3)^T$，负例点$x_3=(1,1)^T$，求线性可分支持向量机。
- 目标函数：

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

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

$$s.t.\ \ \alpha_1 + \alpha_2 - \alpha_3 = 0$$

$$\alpha_i \geq 0,\quad i = 1,2,3$$

将约束带入目标函数，化简计算

- 将 $\alpha_1 + \alpha_2 = \alpha_3$
- 带入目标函数，得到关于α_1，α_2的函数：

$$s(\alpha_1,\alpha_2) = 4\alpha_1^2 + \frac{13}{2}\alpha_2^2 + 10\alpha_1\alpha_2 - 2\alpha_1 - 2\alpha_2$$

- 对α_1，α_2求偏导并令其为0，易知$s(\alpha_1,\alpha_2)$在点(1.5,-1)处取极值。而该点不满足条件$\alpha_2 \geq 0$，所以，最小值在边界上达到。
- 当$\alpha_1=0$时，最小值s(0,2/13)=-2/13=-0.1538
- 当$\alpha_2=0$时，最小值s(1/4,0)=-1/4=-0.25
- 于是，$s(\alpha_1,\alpha_2)$在$\alpha_1=1/4$，$\alpha_2=0$时达到最小，此时，$\alpha_3=\alpha_1+\alpha_2=1/4$

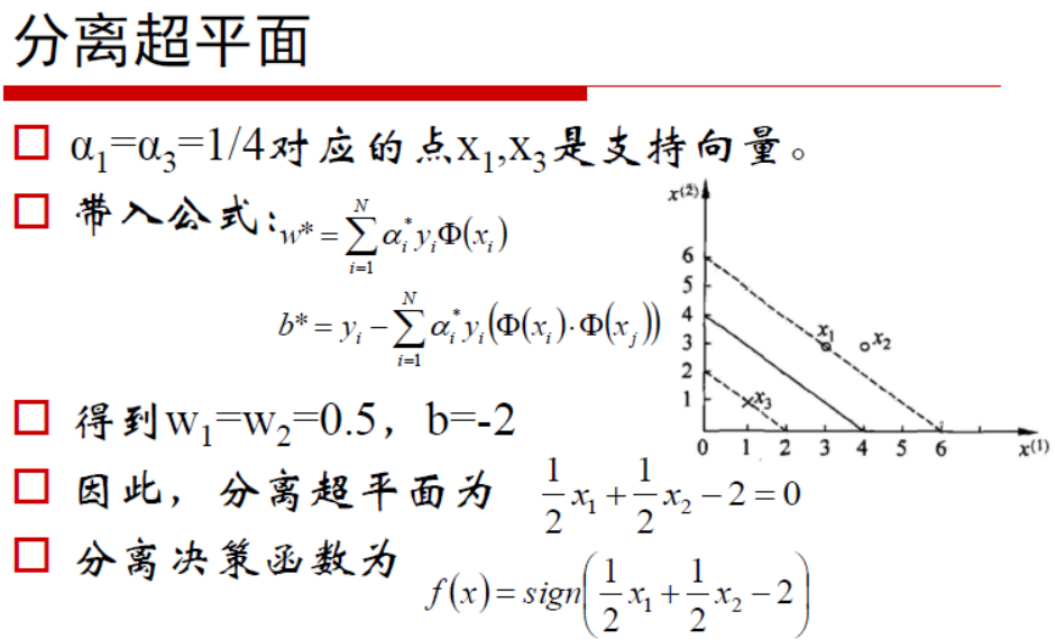


Fig 4.4: the example of solve lagest margin separable hyperplane with svm method

4.2 linear supprot vector machine and soft margin maximization

4.2.1 concept

1. 不一定分类完全正确的超平面就是最好的
2. 样本数据本身线性不可分
3. picture

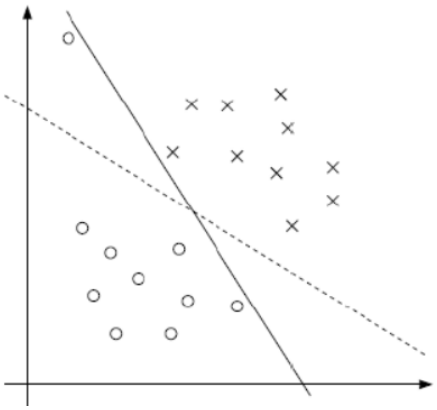


Fig 4.5: non separable dataset

4.2.2 derivation target function

1. 若数据线性不可分, 则增加松弛因子 $\xi \geq 0$, 使函数间隔加上松弛变量大于等于 1. 这样, 约束条件变成

$$y_i(w \cdot x_i + b) \geq 1 - \xi \tag{4.38}$$

2. target function with adding slack variable ξ:

$$\min_{w,b} \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{N} \xi_i \tag{4.39}$$

3. this moment, the target function of linear svm

$$\min_{w,b,\xi} \quad \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{N} \xi_i \tag{4.40}$$

$$\text{s.t.} \quad y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad i = 1, 2, ..., n \tag{4.41}$$

$$\xi_i \geq 0, \quad i = 1, 2, ..., n \tag{4.42}$$

4.2.3 dual property

1. Lagrange 函数

$$L(w, b, \xi, \alpha, \mu) = \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i (y_i(w \cdot x_i + b) - 1 + \xi_i) - \sum_{i=1}^{n} \mu_i \xi_i \tag{4.43}$$

2. 对 w, b, ξ 求偏导

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i \phi(x_n) \tag{4.44}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow 0 = \sum_{i=1}^{n} \alpha_i y_i \tag{4.45}$$

$$\frac{\partial L}{\partial \xi} = 0 \Rightarrow C - \alpha_i - \mu_i = 0 \tag{4.46}$$

3. 将式 4.44 、式 4.45 、式 4.46 代入式 4.43 中, 得到

$$\min w, b, \xi L(w, b, \xi, \alpha, \mu) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^{n} \alpha_i \tag{4.47}$$

4. 对上式 4.47 求关于 α 的极大, 得到

$$\max_{\alpha} \quad -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^{n}\alpha_i \tag{4.48}$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{4.49}$$

$$C - \alpha_i - \mu_i = 0 \tag{4.50}$$

$$\alpha_i \geq 0 \tag{4.51}$$

$$\mu_i \geq 0, \quad i = 1, 2, .., n \tag{4.52}$$

5. 将对偶问题 4.48 $\sim$ 4.52 进行变换: 利用等式约束 4.50 消去 μ_i, 从而只留下变量 α_i, 并将约束 4.50 $\sim$ 4.52 写成

$$0 \leq \alpha_i \leq C \tag{4.53}$$

6. 整理, 得到对偶问题:

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{n}\alpha_i \tag{4.54}$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{4.55}$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, ..., n \tag{4.56}$$

4.2.4 Linear support vector machine learning algorithm

1. 构造并求解约束最优化问题

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{n}\alpha_i \tag{4.57}$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{4.58}$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, ..., n \tag{4.59}$$

2. 求得最优解 α^*, $\alpha^* = (\alpha_1^*, \alpha_2^*, ..., \alpha_n^*)^T$

3. 计算

$$w^* = \sum_{i=1}^{n}\alpha_i^* y_i x_i \tag{4.60}$$

计算 b 的值有两种方式：

- 选择 α^* 的一个分量 α_j^* 适合条件 $0 < \alpha_j^* < C$, 计算

$$b^* = y_j - \sum_{i=1}^{n} y_i \alpha_i^* (x_i \cdot x_j) \tag{4.61}$$

- 实践中往往取支持向量的所有值取平均, 作为 b^*

$$b^* = \frac{\max\limits_{i:y_i=-1} w^* \cdot x_i + \min\limits_{i:y_i=1} w^* \cdot x_i}{2} \tag{4.62}$$

4. 求得分离超平面

$$w^* x + b^* = 0 \tag{4.63}$$

5. 分类决策函数

$$f(x) = sign(w^* x + b^*) \tag{4.64}$$

4.2.5 KKT of soft margin

$$\nabla_w L(w^*, b^*, \xi^*, \alpha^*, \mu^*) = w^* - \sum_{i=1}^{N} \alpha_i^* y_i x_i = 0 \tag{4.65}$$

$$\nabla_b L(w^*, b^*, \xi^*, \alpha^*, \mu^*) = -\sum_{i=1}^{N} \alpha_i^* y_i = 0 \tag{4.66}$$

$$\nabla_\xi L(w^*, b^*, \xi^*, \alpha^*, \mu^*) = C - \alpha^* - \mu^* = 0 \tag{4.67}$$

$$\alpha_i^* (y_i (w^* \cdot x_i + b^*) - 1 + \xi_i^*) = 0 \tag{4.68}$$

$$\mu_i^* \xi_i^* = 0 \tag{4.69}$$

$$y_i (w^* x_i + b^*) - 1 + \xi^* \geq = 0 \tag{4.70}$$

$$\xi_i^* \geq 0 \tag{4.71}$$

$$\alpha_i^* \geq 0 \tag{4.72}$$

$$\mu_i^* \geq 0, \quad i = 1, 2, ..., N \tag{4.73}$$

4.2.6 support vector

1. 在线性不可分的情况下, 将对偶问题 4.57 ~ 4.59 的解 $\alpha^* = (\alpha_1^*, \alpha_2^*, ..., \alpha_n^*)^T$ 中对应于 $\alpha_i^* > 0$ 的样本点 (x_i, y_i) 的实例 x_i 称为支持向量 (软间隔的支持向量).

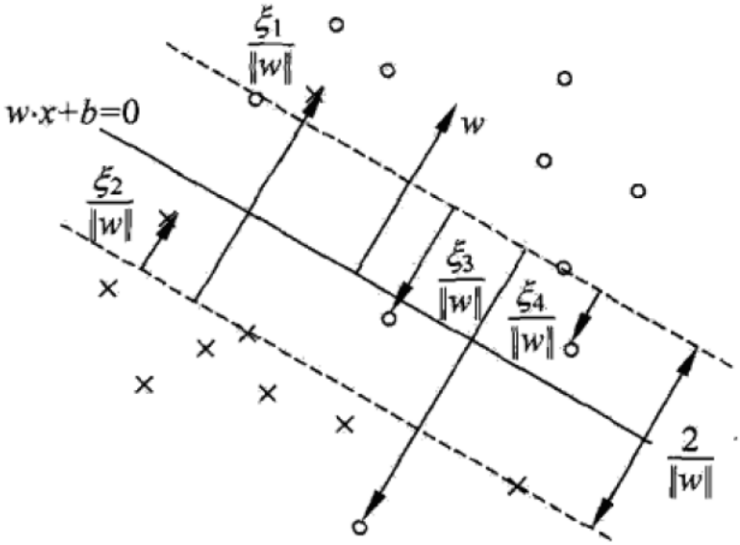


Fig 4.6: support vector soft margin linear svm

这时的支持向量要比线性可分时的情况复杂一些. 图中, 分离超平面由实线表示, 间隔边界由虚线表示, 正例点由 “。” 表示, 负例点由”×” 表示. 图中还标出了实例 x_i 到间隔边界的距离 $\frac{\xi}{\|w\|}$

软间隔的支持向量 x_i 或者在间隔边界上, 或者在间隔边界与分离超平面之间, 或者在分离超平面误分一侧.

KKT 条件中式 4.68、4.69、4.70 得出目标函数中 α_i 取值的意义：

(a) 在两条间隔线外面的点, 对应前面的系数 α_i 为 0,

$$\xi = 0, y_i(w^*x + b) - 1 + \xi > 0 \Leftrightarrow \alpha_i = 0 \tag{4.74}$$

(b) 在两条间隔线里面的对应 α_i 为 C,

- 支持向量落在间隔边界和分离超平面之间

$$\begin{aligned} &0 < \xi < 1 \Leftrightarrow \\ &\mu = 0, \alpha_i = C, y_i(w^*x + b) - 1 + \xi = 0, 0 < y_i(w^*x + b) < 1 \end{aligned} \tag{4.75}$$

- 支持向量落在分离超平面上

$$\begin{aligned} &\xi = 1 \Leftrightarrow \\ &\mu = 0, \alpha_i = C, y_i(w^*x + b) - 1 + \xi = 0, y_i(w^*x + b) = 0 \end{aligned} \tag{4.76}$$

- 支持向量位于分离超平面误分一侧

$$\begin{aligned}&\xi > 1 \Leftrightarrow \\ &\mu = 0, \alpha_i = C, y_i(w^* x + b) - 1 + \xi = 0, y_i(w^* x + b) < 0\end{aligned} \tag{4.77}$$

(c) 在两条间隔线上的对应的系数 α_i 在 0 和 C 之间

$$\begin{aligned}&\xi = 0, y_i(w^* x + b) - 1 = 0 \Leftrightarrow \\ &0 \le \mu \le C, 0 \le \alpha \le C, y_i(w^* x + b) - 1 + \xi = 0\end{aligned} \tag{4.78}$$

4.2.7 loss function

1. $l_{0/1}$ loss function

   在软间隔允许某些样本不满足约束

$$y_i(w^T x_i + b) \ge 1 \tag{4.79}$$

   在最大化间隔时, 不满足的样本应尽可能少, 则优化目标为

$$\min_{w,b} \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{m} l_{0/1}(y_i(w^T x_i + b) - 1) \tag{4.80}$$

   其中, $C > 0$ 是一个常数, $l_{0/1}$ 是"0/1 损失函数"

$$l_{0/1}(z) = \begin{cases} 1, & \text{if } z < 0; \\ 0, & \text{otherwise} \end{cases} \tag{4.81}$$

   显然, 当 C 为无穷大时, 式 4.80 迫使所有样本满足约束 4.79, 于是式 4.80 等价于式 4.11; 当 C 取有限制时, 式 4.80 允许一些样本不满足约束

2. surrogate loss 替代损失

   因为 $l_{0/1}$ 非凸、非连续, 数学性质不好, 使得式 4.80 不易直接求解.

   于是, 通常用其他一些函数来代替 $l_{0/1}$, 称为" 替代损失"

   替代损失函数一般具有较好的数学性质, 如它们通常是凸的连续函数且是 $l_{0/1}$ 的上届.

3. $l_{0/1}$ 损失函数、hinge 损失函数、指数损失函数、对率损失函数

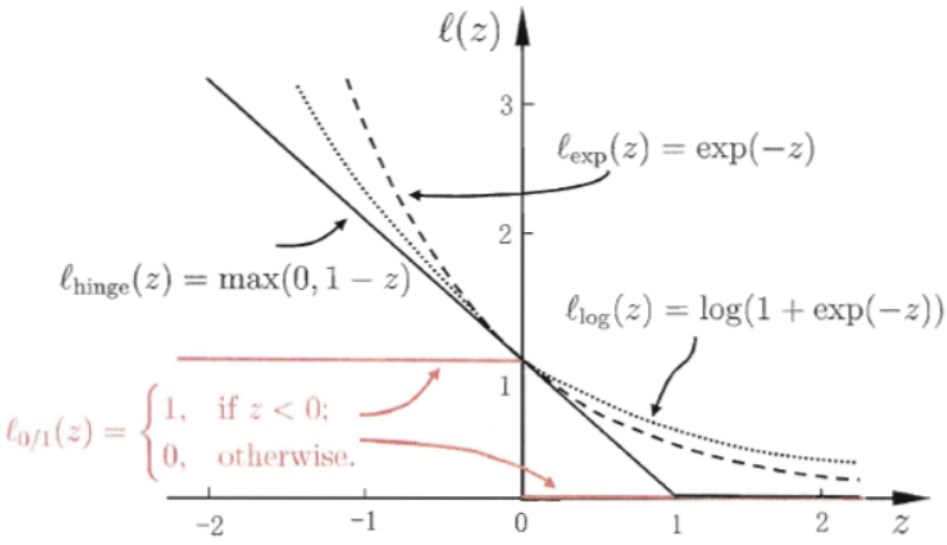


图 6.5 三种常见的替代损失函数: hinge损失、指数损失、对率损失

Fig 4.7: three surrogate loss function

4. hinge 损失：

$$l_{hinge}(z) = \max(0, 1-z) \tag{4.82}$$

指数损失函数：

$$l_{exp}(z) = \exp(-z) \tag{4.83}$$

对率损失：

$$l_{log}(z) = \log(1+\exp(-1)) \tag{4.84}$$

5. 若采用 hinge 损失, 则式 4.80 变成

$$\min_{w,b} \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{m} \max(0, 1 - y_i(w^T x_i + b)) \tag{4.85}$$

6. 引入松弛变量 $\xi_i \geq 0$, 可将式 4.85 重写为

$$\min_{w,b,\xi_i} \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{m} \xi_i \tag{4.86}$$

7. 图 4.7 中, 横轴是函数间隔 $y(w \cdot x + b)$，纵轴是损失.

   0-1 损失函数是二分类问题的真正的损失函数, 而合页损失函数是 0-1 损失函数的上界.

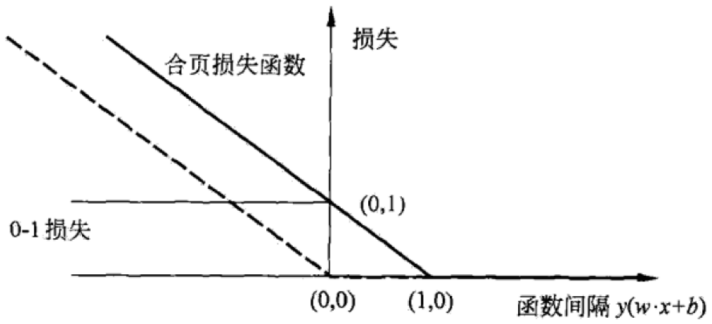


Fig 4.8: hinge loss function

   图 4.8 中虚线显示的是感知机的损失函数 $[-y_i(w \cdot x_i + b)]_+$.

   当样本点 (x_i, y_i) 点被分类正确时, 损失是 0, 否则损失是 $-y_i(w \cdot x_i + b)$.

   相比之下, 合页损失函数不仅要分类正确, 而且确信度足够高时损失才是 0. 也就是说, 合页损失函数对学习有更高的要求.

4.3 kernel function

如果样本线性不可分, 通过 $\phi : X \to F$ 函数映射将输入样本映射到另外一个高维空间并使其线性可分.

将内积 $< x_i, x_j >$ 变成 $< \phi(x_i), \phi(x_j) >$

求解 $< \phi(x_i), \phi(x_j) >$ 有两种方法：

1. 先找到这种映射 $\phi(x)$, 然后将输入空间中的样本映射到新的空间中, 最后在新空间中取求内积 $< \phi(x_i), \phi(x_j) >$

2. 核技巧：不显示的定义映射函数 $\phi(x)$, 值定义核函数 $K(x_i, x_j)$

4.3.1 define directily reflect function

以多项式 $x_1 + x_2 + x_1^2 + x_2^2 + c = 0$ 为例

将其进行变换, $c_1 = x_1, c_2 = x_2, c_3 = x_1^2, c_4 = x_2^2$, 得到：$c_1 + c_2 + c_3 + c_4 = 0$, 也就是说通过把输入空间从二维向四维映射后, 样本有线性不可分变成了线性可分, 但是这种转化带来的直接问他是维度变高了, 这意味着, 首先可能导致后续计算变复杂, 其次可能出现维度灾难.

对于学习器而言就是: 特征空间维数可能最终无法计算, 而它的泛化能力(学习器对训练样本以外数据的适应性) 会随着维度的增大而大大降低, 这也违反了” 奥坎姆剃刀”, 最终可能使内积 $< \phi(x_i), \phi(x_j) >$ 无法求出, 于是也就失去这种转化的优势了

4.3.2 kernel function

1. concept

   definition one: 核是一个函数 K, 对于所有的 $x_1, x_2 \in X$ 满足, $K < x_1, x_2 >=< \phi(x_1), \phi(x_2) >$, 这里的 ϕ 为从 X 到内积特征空间 F 的映射

2. picture

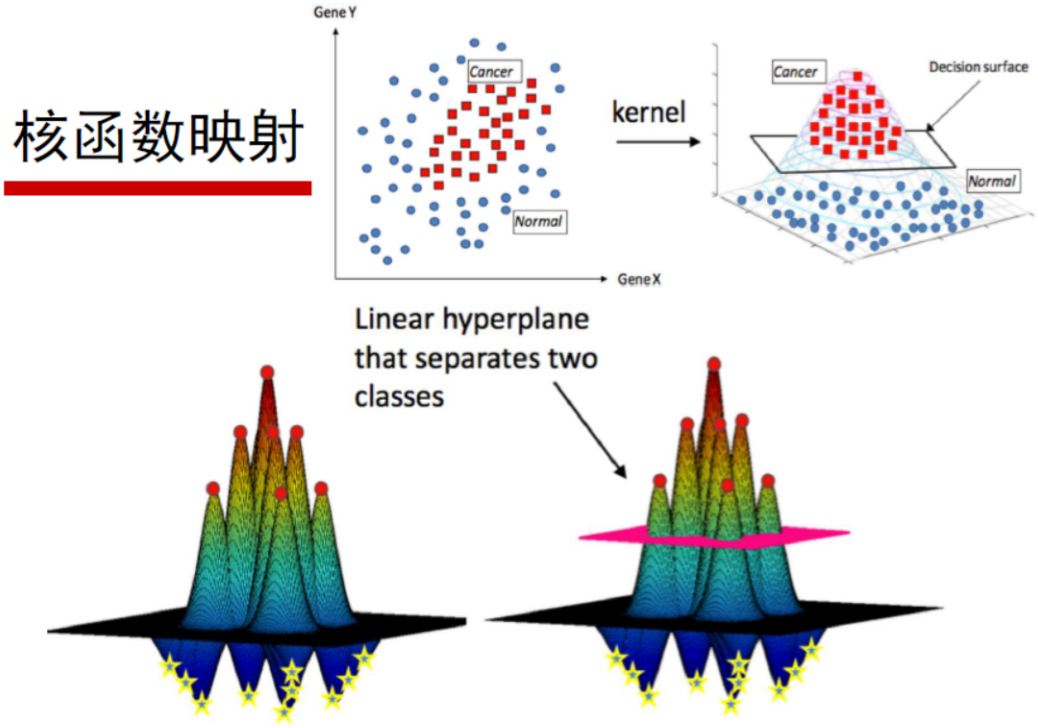


Fig 4.9: reflect kernel function

3. condition

   $K(x, y)$ 什么时候才是核函数呢?

   假设有输入有输入空间 $X = x_1, x_2, ..., x_n$ 且 $K(x, y)$ 为对称函数, 那么对于所有样本得到下面矩阵: $k = K(x_i, x_j)(i, j = 0, ..., n)$, 显然, 这个是对称矩阵, 那么对于对称矩阵一定存在一个正交矩阵, 使得 $P^T kP = \Lambda$, 这

里 Λ 是包含 k 的特征值 λ_i 的对角矩阵, 特征值 λ_i 对应的特征向量为 $\nu_i = (\nu_{i1}, \nu_{i2}, ..., \nu_{in})^T$, 其中 n 为样本数, 对输入空间做如下映射 ϕ

$$\phi(x_i) \Rightarrow (\sqrt{\lambda_1}\nu_{1i}, \sqrt{\lambda_2}\nu_2 i, ..., \sqrt{\lambda_n}\nu_n i)^T \in R^n (i = 1, 2, ..., n) \tag{4.87}$$

$$\phi(x) = (\phi(x_1), \phi(x_2), ..., \phi(x_n)) \tag{4.88}$$

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \tag{4.89}$$

于是有 $< \phi(x_i), \phi(x_j) >= \sum_{t=1}^{n} \lambda_t \nu_{ti} \nu_{tj} = (V \Lambda V^T)_{i,j} = k_{i,j} = K(x_i, x_j)$ (其中 V 为特征向量组成的矩阵, Λ 为相应特征值组成的三角矩阵), 也就是说 K 是对应于映射的核函数.

4. example

有 $k = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix}$, 由 $|k - \lambda E| = 0$ 解得特征值: $\lambda_1 = \lambda_2 = 4$, $\lambda_3 = 2$, 对 2 重特征根 4 求 $(A - 4E)x = 0$ 的基础解系、正交化、单位化后得到特征向量: $\nu_1 = (1, 0, 0)^T$ 和 $\nu_2 = (0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})^T$, 对 λ_3 的特征向量单位化后得到 $\nu_3 = (0, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})^T$, 于是有 $V = (\nu_1, \nu_2, \nu_3) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{vmatrix}$ 满足 $V^{-1}kV = \Lambda = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 2 \end{bmatrix}$, 对所有输入样本做映射得：

$$\phi(x_1) = (\sqrt{4} \times 1, \sqrt{4} \times 0, \sqrt{2} \times 0)^T = (2, 0, 0)^T; \tag{4.90}$$

$$\phi(x_2) = (\sqrt{4} \times 0, \sqrt{4} \times \frac{1}{\sqrt{2}}, \sqrt{2} \times -\frac{1}{\sqrt{2}})^T = (0, \sqrt{2}, -1)^T; \tag{4.91}$$

$$\phi(x_3) = (\sqrt{4} \times 0, \sqrt{4} \times \frac{1}{\sqrt{2}}, \sqrt{2} \times \frac{1}{\sqrt{2}})^T = (0, \sqrt{2}, 1)^T \tag{4.92}$$

随便选两个做内积, 如 $< \phi(x_2), \phi(x_3) >= 0 \times 0 + \sqrt{2} \times \sqrt{2} + (-1) \times 1 = 1 = k(x_2, x_3)$

由此可见：$K(x, y)$ 就是对应于特征映射 $\phi(x)$ 的核函数

5. conclusion

Tab 4.1: common kernel function

名称	表达式	参数
线性核	$\kappa(x_i, x_j) = x_i^T x_j$	
多项式核	$\kappa(x_i, x_j) = (x_i^T x_j)^d$	$d \geq 1$ 为多项式的次数
高斯核	$\kappa(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$	$\sigma > 0$ 为高斯核的带宽 (width)
拉普拉斯核	$\kappa(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|}{\sigma})$	$\sigma > 0$
Sigmoid 核	$\kappa(x_i, x_j) = \tanh(\beta x_i^T x_j + \theta)$	tanh 为双曲正切函数, $\beta > 0, \theta < 0$

- 定理 1：存在有限输入空间 X, K(x,y) 为 X 上的对称函数, 那么 K(x,y) 是核函数的冲要条件是矩阵 $k = (K(x_i, x_j))(i, j = 0, ..., n)$ 半正定, 此时相当于对输入空间想特征空间进行了 隐式 的 $\phi(x)$ 映射. 对于上面的映射 $\phi(x)$, 令 $\phi_i(x_j) = \sqrt{\lambda_i}\nu_{ij}$，于是 $\phi(x) = (\phi_1(x), \phi_2(x), ..., \phi_n(x))$
- 令 χ 为输入空间, $\kappa(\cdot, \cdot)$ 是定义在 $\chi \times \chi$ 上的对称函数, 则 κ 是核函数当且仅当对于任意数据 $D = \{x_1, x_2, ..., x_n\}$, kernel matrix K 总是半正定的

$$\boldsymbol{K} = \begin{bmatrix} \kappa(x_1, x_1) & \cdots & \kappa(x_1, x_j) & \cdots & \kappa(x_1, x_m) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \kappa(x_i, x_1) & \cdots & \kappa(x_i, x_j) & \cdots & \kappa(x_i, x_m) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \kappa(x_m, x_1) & \cdots & \kappa(x_m, x_j) & \cdots & \kappa(x_m, x_m) \end{bmatrix} \tag{4.93}$$

  事实证明, 只要一个对称函数所对应的核矩阵半正定, 它就能作为核函数使用. 事实上, 一个半正定核矩阵, 总能找到一个与之对应的映射 ϕ, 换言之, 任何一个核函数都隐式得定义了一个称为“再生核希尔伯特空间”的特征空间

6. common kernel function

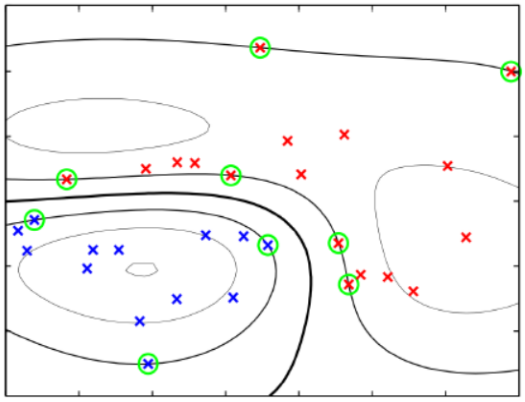
7. Gussian kernel function

   在实际应用中, 往往依赖先验领域知识 / 交叉验证等方案才能选择有效的核函数.

   没有更多先验信息, 则使用高斯核函数

高斯核

- 粗线是分割超“平面”
- 其他线是y(x)的等高线
- 绿色圈点是支持向量点



高斯核是无穷维的 $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + R_n$

$$\kappa(x_1, x_2) = e^{-\frac{\|x_1 - x_2\|^2}{2\sigma^2}} = e^{-\frac{(x_1 - x_2)^2}{2\sigma^2}} = e^{-\frac{x_1^2 + x_2^2 - 2x_1x_2}{2\sigma^2}} = e^{-\frac{x_1^2 + x_2^2}{2\sigma^2}} \cdot e^{\frac{x_1x_2}{\sigma^2}}$$

$$= e^{-\frac{x_1^2 + x_2^2}{2\sigma^2}} \cdot \left(1 + \frac{1}{\sigma^2} \cdot \frac{x_1x_2}{1!} + \left(\frac{1}{\sigma^2}\right)^2 \cdot \frac{(x_1x_2)^2}{2!} + \left(\frac{1}{\sigma^2}\right)^3 \cdot \frac{(x_1x_2)^3}{3!} + \cdots + \left(\frac{1}{\sigma^2}\right)^n \cdot \frac{(x_1x_2)^n}{n!} + \cdots\right)$$

$$= e^{-\frac{x_1^2 + x_2^2}{2\sigma^2}} \cdot \left(1 \cdot 1 + \frac{1}{1!}\frac{x_1}{\sigma} \cdot \frac{x_2}{\sigma} + \frac{1}{2!} \cdot \frac{x_1^2}{\sigma^2} \cdot \frac{x_2^2}{\sigma^2} + \frac{1}{3!} \cdot \frac{x_1^3}{\sigma^3} \cdot \frac{x_2^3}{\sigma^3} + \cdots + \frac{1}{n!} \cdot \frac{x_1^n}{\sigma^n} \cdot \frac{x_2^n}{\sigma^n} + \cdots\right)$$

$$= \Phi(x_1)^T \cdot \Phi(x_2)$$

- 其中 $\Phi(x) = e^{-\frac{x^2}{2\sigma^2}}\left(1, \sqrt{\frac{1}{1!}}\frac{x}{\sigma}, \sqrt{\frac{1}{2!}}\frac{x^2}{\sigma^2}, \sqrt{\frac{1}{3!}}\frac{x^3}{\sigma^3}, \cdots, \sqrt{\frac{1}{n!}}\frac{x^n}{\sigma^n}, \cdots\right)$

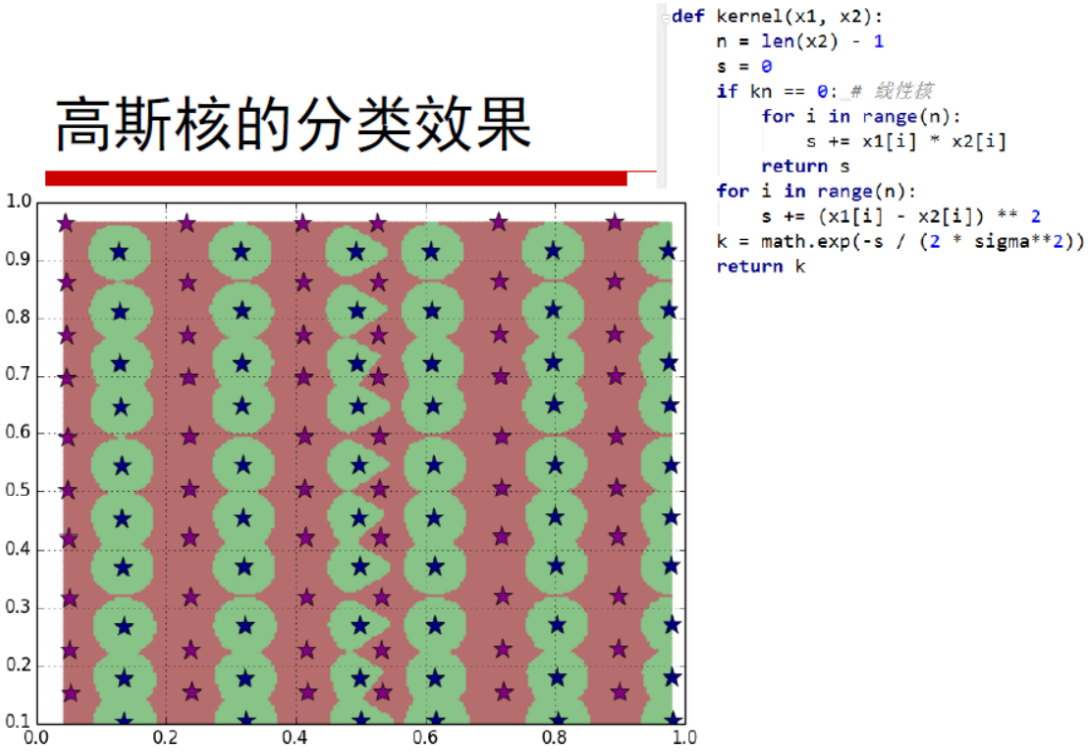


Fig 4.10: gussian kernel function

8. kernel function in svm

选定核 $K(x_i, x_j)$ 后, 原问题变成

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_i \cdot x_j) - \sum_{i=1}^{n}\alpha_i \tag{4.94}$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{4.95}$$

$$0 \le \alpha_i \le C, \quad i = 1, 2, ..., n \tag{4.96}$$

优化问题得最优解, 核要满足 Mercer 条件, 即矩阵 $k = K(x_i, x_j), (i, j = 0, 1, ..., n)$ 在所有训练上半正定, 说明这个优化是凸优化, 这个条件保证了最大化间隔优化问题有唯一解.

最后求得 α^*, b^*, 那么从输入空间向特征空间隐式映射后所得到的最大间隔

超平面也就出来了:

$$f(x) = \sum_{i=1}^{n} \alpha^* y_i K(x_i, x_j) + b^* \tag{4.97}$$

且有几何间隔

$$\gamma = \frac{1}{\parallel w^* \parallel} = \left(\sum_{i \in \text{support vector}} \alpha^* \right)^{\frac{1}{2}} \tag{4.98}$$

4.4 sequential minimal optimization algorithm

4.4.1 concept

1. 用于 SVM 中系数的求解算法
2. 场合: 有多个 Lagrange multiplier
3. 原理: 每次只选择其中两个乘子做优化, 其他因子认为是常数.
   - 将 N 个解问题, 转换成两个变量的求解问题: 并且目标函数是凸的

4.4.2 process

1. 考察目标函数

$$\min_{\alpha} \quad \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{n} \alpha_i \tag{4.99}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{4.100}$$

$$0 \le \alpha_i \le C, \quad i = 1, 2, ..., n \tag{4.101}$$

2. 假设 α_1 和 α_2 是变量, 其他是定值:

$$\begin{aligned} \min_{\alpha_1, \alpha_2} \quad & W(\alpha_1, \alpha_2) \\ & = \frac{1}{2} \kappa_{11} \alpha_1^2 + \frac{1}{2} \kappa_{22} \alpha_2^2 + y_1 y_2 \alpha_1 \alpha_2 \kappa_{12} - (\alpha_1 + \alpha_2) \\ & + y_1 \alpha_1 \sum_{i=3}^{N} y_1 \alpha_i \kappa_{i1} + y_2 \alpha_2 \sum_{i=3}^{N} y_i \alpha_i \kappa_{i2} \end{aligned} \tag{4.102}$$

$$\text{s.t.} \quad \alpha_1 y_1 + \alpha_2 y_2 = -\sum_{i=3}^{N} y_i \alpha_i = \xi \tag{4.103}$$

$$0 \le \alpha_i \le C \tag{4.104}$$

其中, ξ 是常数, 目标函数中省略了不含 α_1, α_2 的常数项

3. 为了求解两个变量的二次规划问题, 首先分析约束条件, 然后在此约束条件下求极小.

   由于只有两个变量, 约束可以用二维空间中的图形表示

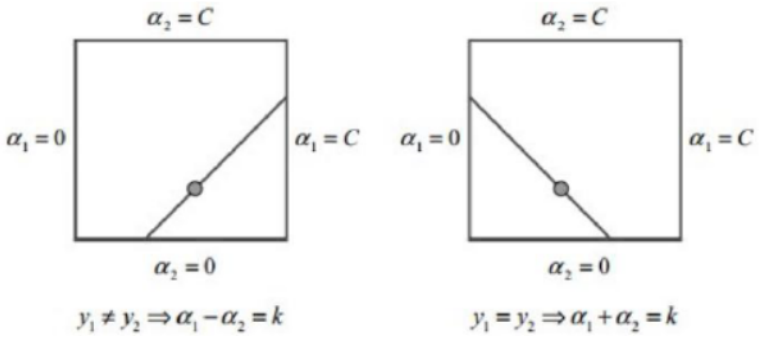


Fig 4.11: two variable optimization

4. 不等式约束 4.104 使得 (α_1, α_2) 在盒子 $[0, C] \times [0, C]$ 内, 等式约束 4.103 使 α_1, α_2 在平行于盒子 $[0, C] \times [0, C]$ 的对角线的直线上.

   因此要求的是目标函数在一条平行于对角线的线段上的最优值. 这使得两个变量的最优化问题成为实质上的单变量的最优化问题, 不妨考虑为变量 α_2 的最优化问题

5. 假设问题 4.102 ∼ 4.104 的初始可行解为 $\alpha_1^{old}, \alpha_2^{old}$, 最优解为 $\alpha_1^{new}, \alpha_2^{new}$, 并且假设在沿着约束方向未经剪辑时 α_2 的最优解为 $\alpha_2^{new,unc}$. 注意：此处最优值是在图中直线上移动的

6. 综合式 4.103 和 4.104 这两个约束条件, 求取 α_2^{new} 的取值范围

   由于 α_2^{new} 需满足不等式约束 4.104, 所以最优值 α_2^{new} 的取值范围必须满足条件

$$L \leq \alpha_2^{new} \leq H \tag{4.105}$$

   其中, L 和 H 是 α_2^{new} 所在的对角线段端点的界.

   (a) $y_1 \neq y_2$ 时, 根据 $\alpha_1^{new} y_1 + \alpha_2^{new} y_2 = \alpha_1^{old} y_1 + \alpha_2^{old} y_2 = \xi$ 可得 $\alpha_1^{old} - \alpha_2^{old} = \xi$, 所以有

$$L = \max(0, -\xi) = \max(0, \alpha_2^{old} - \alpha_1^{old}) \tag{4.106}$$

$$H = \min(C, C - \xi) = \min(C, C + \alpha_2^{old} - \alpha_1^{old}) \tag{4.107}$$

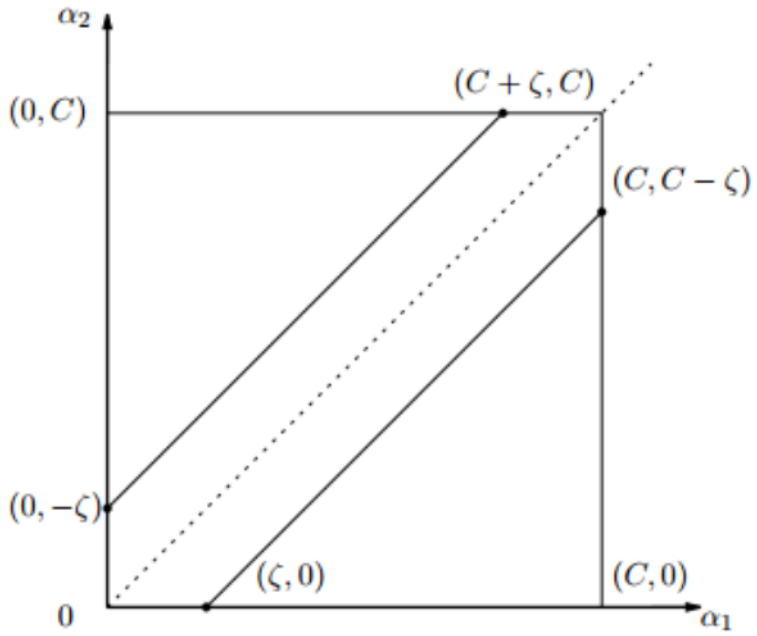


Fig 4.12: y1 not equal y2 smo

(b) $y_1 = y_2$ 时, 根据 $\alpha_1^{new} y_1 + \alpha_2^{new} y_2 = \alpha_1^{old} y_1 + \alpha_2^{old} y_2 = \xi$ 可得 $\alpha_1^{old} + \alpha_2^{old} = \xi$, 所以有

$$L = \max(0, \xi - C) = \max(0, \alpha_2^{old} + \alpha_1^{old} - C) \tag{4.108}$$

$$H = \min(C, \xi) = \min(C, \alpha_2^{old} + \alpha_1^{old}) \tag{4.109}$$

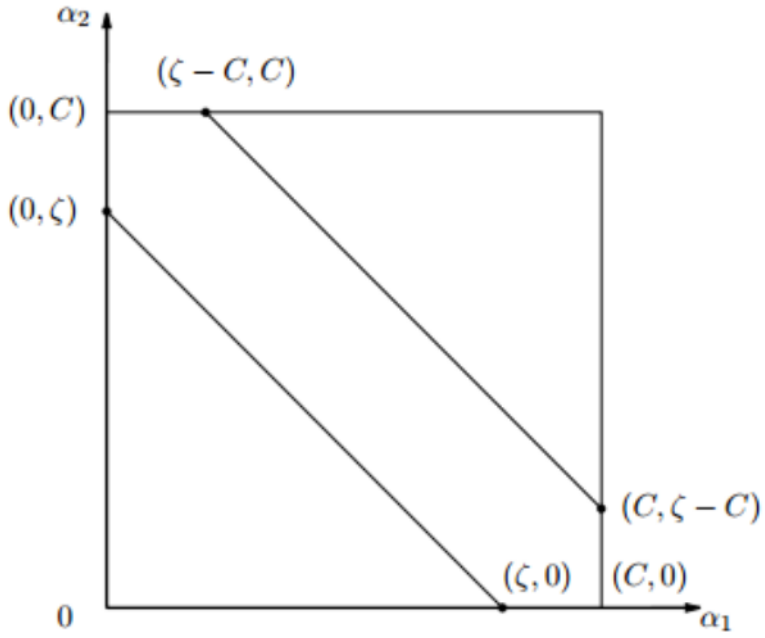


Fig 4.13: y1 equal y2 smo

7. 下面首先求沿着约束方向未经剪辑即未考虑不等式约束 4.104 时 α_2 的最优解 $\alpha_2^{new,unc}$; 然后再求剪辑后 α_2 的解 α_2^{new}.

   以下 smo 此时求解 α_1 和 α_2 的迭代公式

   为叙述简单记, 记 (此处 $g(x)$ 中 i 和下面 E 中所表示的 j 一致)

$$g(x) = \sum_{i=1}^{N} \alpha_2 y_i K(x_i, x) + b \tag{4.110}$$

   令

$$E_i = g(x_i) - y_i = \left(\sum_{j=1}^{N} \alpha_j y_j K(x_j, x_i) + b \right) - y_i \quad i = 1, 2 \tag{4.111}$$

   令

$$\eta = K_{11} + K_{22} - 2K12 = \| \phi(x_1) - \phi(x_2) \|^2 \tag{4.112}$$

   $\phi(x)$ 是输入空间到特征空间的映射, $E_i, \quad i = 1, 2$

   则最优化问题 4.102 $\sim$ 4.104 沿着约束方向未经剪辑时的解是

$$\alpha_2^{new,unc} = \alpha_2^{old} + \frac{y_2(E_1 - E_2)}{\eta} \tag{4.113}$$

   经剪辑后 α_2 的解是

$$\alpha_2^{new} = \begin{cases} H, & \alpha_2^{new,unc} > H \\ \alpha_2^{new,unc}, & L \le \alpha_2^{new,unc} \le H \\ L, & \alpha_2^{new,unc} < L \end{cases} \tag{4.114}$$

   由 α_2^{new} 求得 α_1^{new} 是

$$\alpha_1^{new} = \alpha_1^{old} + y_1 y_2 (\alpha_2^{old} - \alpha_2^{new}) \tag{4.115}$$

   上述公式具体证明过程看博客 https://blog.csdn.net/v_JULY_v/article/details/7624837 https://www.cnblogs.com/jerrylead/archive/2011/03/18/1988419.html https://blog.csdn.net/the_lastest/article/details/78637565 结合李航的统计学和周志华的机器学习

8. 如何选择乘子 α_1 和 α_2 (参考统计学习方法, 介绍详细)

   SMO 算法在每个子问题中选择两个变量优化, 其中至少一个变量是违反 KKT 条件的.

- 对于 α_1, 即第一个乘子, 可以通过 3 种不满足 KKT 的条件来找
- 对于第二个乘子 α_2 可以寻找满足条件：$\max |E_i - E_j|$ 的乘子

9. 计算阈值 b 和差值 E_i b 在满足下述条件：

$$b = \begin{cases} b_1 & if\ 0 < \alpha_1^{new} < C \\ b_1 & if\ 0 < \alpha_1^{new} < C \\ (b_1 + b_2)/2 & otherwise \end{cases} \tag{4.116}$$

下更新 b:

$$b_1^{new} = b^{old} - E_1 - y_1(\alpha_1^{new} - \alpha_1^{old})K(x_1, x_1) - y_2(\alpha_2^{new} - \alpha_2 old)K(x_1, x_2) \tag{4.117}$$

$$b_2^{new} = b^{old} - E_2 - y_1(\alpha_1^{new} - \alpha_1^{old})K(x_1, x_2) - y_2(\alpha_2^{new} - \alpha_2 old)K(x_2, x_2) \tag{4.118}$$

且每次更新完两个乘子的优化后, 都需要再重新计算 b, 及对应的 E_i 值

最后更新完所有 α_i, y 和 b, 这样模型就出来了, 从而即可求出咱们开头提出的分类函数：

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i < x_i, x > + b \tag{4.119}$$

10. summary of smo algorithm step

输入：训练数据集 $T = \{(x_1, y_1), (x_2, y_2), .., (x_N, y_N)\}$, 其中, $x_i \in \chi = \boldsymbol{R}^n$, $y_i \in \gamma = -1, +1$, $i = 1, 2, ..., N$, 精度 ε

输出：近似解 $\hat{\alpha}$

(1) 取初值 $\alpha^{(0)} = 0$, 令 $k = 0$

(2) 选取优化变量 α_1^k, α_2^k, 解析求解两个变量的最优化问题 4.102 $\sim$ 4.104, 求得最优解 $\alpha_1^{k+1}, \alpha_2^{k+1}$, 更新 α 为 $\alpha^{(k+1)}$

i. 第一步选取一对 α_i 和 α_j, 选取方法使用启发方法

ii. 第二步, 固定除 α_i 和 α_j 之外的其他参数, 确定 W 极值条件下的 α_i, α_j 由 α_i 表示

detail explaintation:

i. 假定在某一次迭代中, 需要更新 x_1, x_2 对应的拉格朗日乘子 α_1, α_2, 那么这个小规模的二次规划问题写为：

$$L_s = \max_{\alpha}\{(\alpha_1+\alpha_2)+\sum_{i=3}^{n}\alpha_i - \frac{1}{2}\left\|\alpha_1 y_1 \phi(\boldsymbol{x_1}) + \alpha_1 y_1 \phi(\boldsymbol{x_1}) + \sum_{i=3}^{n}\alpha_i y_i \phi(\boldsymbol{x_i})\right\|^2\}$$
$$s.t. \alpha_1 y_1 + \alpha_2 y_2 = -\sum_{i=3}^{n}\alpha_i y_i, 0 < \alpha_i < C, \forall i$$

Fig 4.14: update alpha smo summary

ii. 那么在每次迭代中, 如何更新乘子呢？引用http://staff.ustc.edu.cn/~ketang/PPT/PRLec5.pdf的两张 ppt 说明下：

更新拉格朗日乘子α_1, α_2

– 步骤1：计算上下界L和H

- $L = \max(0, \alpha_2^{old} - \alpha_1^{old}), H = \min(C, C + \alpha_2^{old} - \alpha_1^{old}), if\ y_1 \neq y_2$
- $L = \max(0, \alpha_2^{old} + \alpha_1^{old} - C), H = \min(C, \alpha_2^{old} + \alpha_1^{old}), if\ y_1 = y_2$

– 步骤2：计算L_s的二阶导数

- $\eta = 2\phi(\boldsymbol{x_1})^t\phi(\boldsymbol{x_2}) - \phi(\boldsymbol{x_1})^t\phi(\boldsymbol{x_1}) - \phi(\boldsymbol{x_2})^t\phi(\boldsymbol{x_2})$

– 步骤3：更新L_s

$$\alpha_2^{new} = \alpha_2^{old} - \frac{y_2(e_1 - e_2)}{\eta}$$
$$e_i = g^{old}(\boldsymbol{x_i}) - y_i$$

– 步骤4：计算变量α_2

$$\alpha^{temp} = \begin{cases} H, & \text{if}\alpha_2^{new} \geq H \\ \alpha_2^{new}, & \text{if } L \leq \alpha_2^{new} \leq H \\ L, & \text{if } \alpha_2^{new} \leq L \end{cases}$$

– 步骤5：更新α_1

$$\alpha_1^{new} = \alpha_1^{old} + y_1 y_2(\alpha_2^{old} - \alpha^{temp})$$

Fig 4.15: detail update alpha smo summary

iii. 知道了如何更新乘子, 那么选取哪些乘子进行更新呢？具体选择方法有以下两个步骤：

- step1: 先“扫描”所有乘子, 把第一个违反 KKT 条件的作为更新对象, 令为 a_1
- step2: 在所有不违反 KKT 条件的乘子中, 选择使 $|E_1 - E_2|$ 最大的 α_2 进行更新, 使得能最大限度增大目标函数的值 (类似于

梯度下降, 此外 $E_i = \mu_i - y_i$, 而 $\boldsymbol{\mu} = \vec{w} \cdot \vec{x} - b$, 求出来的 E 代表函数 μ_i 对输入 x_i 的预测值与真实输出类标记 y_i 之差)

iv. 每次更新完两个乘子的优化后, 都需要再重新计算 b, 及对应的 E_i 值

(3) 若在精度 ε 范围内满足停机条件

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{4.120}$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, .., N \tag{4.121}$$

$$y_i \cdot g(x_i) = \begin{cases} \geq 1, & \{x_i | \alpha_i = 0\} \\ = 1, & \{x_i | 0 < \alpha_i < C\} \\ \leq 1, & \{x_i | \alpha_i = C\} \end{cases} \tag{4.122}$$

其中,

$$g(x_i) = \sum_{j=1}^{N} \alpha_j y_j K(x_j, x_i) + b \tag{4.123}$$

则转 (4), 否则令 $k = k + 1$, 转 (2)

(4) 取 $\hat{\alpha} = \alpha^{(k+1)}$

与通常的分解算法比较，尽管 SMO 可能需要更多的迭代次数，但每次迭代的计算量比较小，所以该算法表现出较好的快速收敛性，且不需要存储核矩阵，也没有矩阵运算。