

HSSVM: 超球 SVM 求解多分类问题

熊 俊

Last updated: 2010-3-7

摘要

HSSVM 是一个用超球 SVM (Hyper-Sphere Support Vector Machines) 模型求解多分类问题的工具包, 采用 Java 语言实现。开发该程序的主要目的, 是利用超球 SVM 求解模型代替传统上借助于解二分类问题的经典 SVM 模型来求解多分类问题。本文将论述该程序的主要实现细节, 包括相关算法及设计原理的描述。

关键字: 多分类问题 超球 SVM SMO 算法 核矩阵 Cache

1. 概述

为方便使用及理解 HSSVM 工具包, 本文将论述该工具包的主要设计原理及相关实现细节, 包括超球 SVM 原理简述、程序总体算法描述、求解超球二次规划问题的 SMO 算法及核矩阵的存储机制等。具体使用本工具, 请参阅工具包内 README-zh.txt 文件。

本文具体章节安排如下: 第 2 节, 将简要介绍超球 SVM 相关原理及理论导出, 程序的总体算法描述; 第 3 节, 将论述超球二次规划问题的求解过程, 包括超球 SMO 算法、超球 KKT 条件、工作集选择 (WSS) 算法; 第 4 节, 描述程序中核矩阵的两种存储方式——UTM (上三角矩阵) 和 LRU (最近最少使用) Cache; 第 5 节, 将附上超球 SMO 算法、KKT 条件及 WSS 算法等的详细导出过程。

2. 问题总述

2.1 超球 SVM 原理

对于 $k > 2$ 的分类问题, 数学描述为: 给定 k 个 n 维空间的集合 A^m , $m=1, \dots, k$, 每个集合包含 l^m 个样本点 χ_i^m , $i=1, \dots, l^m$, 对每个集合寻找一个超球 (a^m, R^m) , a^m 为球心, R^m 为半径平方并使其尽量小, 使得该最小超球尽可能包含所有的同类样本点 χ_i^m 。考虑到存在一些孤立点 (离球心较远的心), 应允许这些点落在超球面外, 寻求最小超球过程可演化为原始的优化问题:

$$\min: R^m + C^m \sum_i^m \xi_i^m \quad (2.1)$$

$$s.t. \left\| \chi_i^m - a^m \right\|^2 \leq R^m + \xi_i^m$$

$$\xi_i^m \geq 0 \quad i=1, \dots, m$$

式 (2.1) 中 ξ_i^m 为松弛变量, C^m 为惩罚系数, 控制对错分样本的惩罚程度, 实现在超球大小和错分样本数量之间的折衷。此问题属于解二次规划问题, 利用 Lagrange 优化方法求解。构造 Lagrange 函数如下:

$$L(R^m, a^m, \xi_i^m, \alpha_i^m, \beta_i^m) = R^m + C^m \sum_i^m \xi_i^m - \sum_i^m \alpha_i^m (R^m + \xi_i^m - \left\| \chi_i^m - a^m \right\|^2) - \sum_i^m \beta_i^m \xi_i^m$$

$\alpha_i^m, \beta_i^m \geq 0$ 为 Lagrange 乘子, 对 Lagrange 函数求偏导并令其为零, 整理得

$$\sum_{i=1}^m \alpha_i^m = 1, \quad a^m = \sum_{i=1}^m \alpha_i^m \chi_i^m, \quad 0 \leq \alpha_i^m \leq C^m$$

这样, 原始问题转化为简单的对偶问题

$$\min L(\alpha_i) = \sum_{i=1, j=1}^m \alpha_i^m \alpha_j^m K(\chi_i^m, \chi_j^m) - \sum_{i=1}^m \alpha_i^m K(\chi_i^m, \chi_i^m) \quad (2.2)$$

$$s.t. \sum_{i=1}^m \alpha_i^m = 1, \quad 0 \leq \alpha_i^m \leq C$$

对于非线性情况, 通过引入核函数:

$$K(\chi_i, \chi_j) = \phi(\chi_i)^T \phi(\chi_j) \quad (2.3)$$

将样本数据空间映射到高维特征空间进行超球描述, 由此解决了通常情况下即使排除孤立点后, 数据依然不呈球状分布, 而造成构造最小超球的困难。

为简化描述及求解, 以下将只针对某一类别的超球进行讨论, 则 (2.2) 式亦可转化为如下表示:

$$\min L(\alpha) = \sum_{i=1, j=1}^l \alpha_i \alpha_j K(\chi_i, \chi_j) - \sum_{i=1}^l \alpha_i K(\chi_i, \chi_i) \quad (2.4)$$

$$s.t. \sum_{i=1}^l \alpha_i = 1, \quad 0 \leq \alpha_i \leq C$$

对每个类别都求解 (2.4) 所示的二次规划问题, 即产生 m 个最小超球, 每个超球代表一类样本, 最后得到关于 Lagrange 乘子向量 α 的全局最优解。最优解向量 α 中各分量通常大部分为零或趋于零, 少数大于零的 α_i 所对应的样本即为支持向量, 位于最小超球的球面上, 决定了超球的大小与位置。球心可表示为:

$$a = \sum_{i=1}^l \alpha_i \phi(\chi_i) \quad (2.5)$$

超球半径平方 R 可通过任意一个支持向量 χ 与球心的距离来确定：

$$R = \|\phi(\chi) - a\|^2 = K(\chi, \chi) - 2 \sum_{i=1}^l \alpha_i K(\chi, \chi_i) + \sum_{i,j=1}^l \alpha_i \alpha_j K(\chi_i, \chi_j) \quad (2.6)$$

对于待测样本 z ，计算其到球心 a 距离的平方如下：

$$D^2 = \|\phi(z) - a\|^2 = K(z, z) - 2 \sum_{i=1}^l \alpha_i K(z, \chi_i) + \sum_{i,j=1}^l \alpha_i \alpha_j K(\chi_i, \chi_j) \quad (2.7)$$

因此，根据计算出的 D^2 与 R 的比较，即可确定待测样本 z 所属的类别。

2.2 算法总述

超球 SVM 求解主体算法过程描述如下：

算法 1 (HSSVM 训练预测过程)

1. 读取训练样本数据，并分类；
2. 对第 m 类训练样本 $m=1$ to k do {
 - 利用 SMO 算法解二次规划问题，得到解向量 α (Lagrange 乘子向量)；
 - 由 α 计算出 R (超球半径平方)，得到球模型；
 - 将各球模型写入文件；
- } // 结束循环
3. 从文件读取球模型；
4. 读取测试样本数据；
5. 对第 i 个待测试样本 $i=1$ to n do {
 - 对第 j 个球模型 $j=1$ to m do {
 - 计算样本 i 到球 j 的距离平方 $D_{i,j}^2$ ；
 - 若 $D_{i,j}^2 - R_j \leq 0$ 则 i 落入球内，否则为落入球外；
 - } // 结束 j 循环
 - 统计样本 i 所落入的超球数量 N ；
 - 若 $N=1$ ，则 i 归入所落入的唯一球；
 - 若 $N \neq 1$ ，则 i 归入 $\arg \min |(D_{i,j}^2 - R_j) / R_j|$ 所确定的球；
- } // 结束 i 循环
6. 结束

本节主要理论来源请见[6][7].

3 解二次规划 (QP) 问题

解二次规划为本程序的核心问题, (2.4) 的形式及约束条件与经典 SVM 略有不同, 转化为矩阵形式为:

$$\begin{aligned} \min_{\alpha} \quad & L(\alpha) = \alpha^T H \alpha - k^T \alpha \\ \text{s.t.} \quad & e^T \alpha = 1, \quad 0 \leq \alpha \leq C e \end{aligned} \quad (3.1)$$

其中 $H \equiv (K(\chi_i, \chi_j))_{i,j}$, $K(\chi_i, \chi_j) = \phi(\chi_i)^T \phi(\chi_j)$ 为核函数, 简记 $K(\chi_i, \chi_j)$ 为 k_{ij} ,

$\alpha = (\alpha_1, \alpha_2, \dots, \alpha_l)^T$ 为 Lagrange 乘子向量, $e = (1, 1, \dots, 1)^T$ 为单位向量, $k = (k_{11}, k_{22}, \dots, k_{ll})^T$ 。

本程序中采用了 SMO 算法 (Sequential Minimal Optimization) [3] 求解 (3.1)。SMO 算法基于 Osuna 等提出的分解迭代算法 [5] (Decomposition method) 思想, 在解 QP 问题时, 每次仅选取两个 Lagrange 乘子进行迭代, 这样可用解析的方式求解每个最小规模的优化问题。循环地对其他 Lagrange 乘子进行同样的最小优化, 直到符合 KKT 条件, 即得目标函数最优可行解。

3.1 停机准则 (Stopping Criteria)

KKT 条件 (Karush-Kuhn-Tucker) 作为常用的停机准则, 是最优可行解的充分必要条件。若 α 是 (3.1) 的最优解, 当且仅当存在一个 b 值及两个非负向量 μ, ν 满足如下条件:

$$2H\alpha - k + be = \nu - \mu \quad (3.2)$$

$$\mu^T (\alpha - Ce) = 0, \quad \nu^T \alpha = 0, \quad \nu \geq 0, \quad \mu \geq 0$$

定义

$$u_i = (2H\alpha - k)_i = 2 \sum_{j=1}^l \alpha_j k_{ij} - k_{ii} \quad (3.3)$$

上述 (3.2) 亦可表述为:

$$u_i + b \geq 0 \quad \text{若 } \alpha_i < C \quad (3.4a)$$

$$u_i + b \leq 0 \quad \text{若 } \alpha_i > 0 \quad (3.4b)$$

定义集合:

$$I_{up}(\alpha^k) = \{i | \alpha_i < C\}, \quad I_{low}(\alpha^k) = \{i | \alpha_i > 0\} \quad (3.5)$$

若 α 是 (3.1) 的最优可行解, 当且仅当:

$$m(\alpha) \leq M(\alpha) \quad (3.6)$$

此处， $m(\alpha) = \max_{i \in I_{up}} (-u_i)$ ， $M(\alpha) = \min_{i \in I_{low}} (-u_i)$ ，考虑精度 ε ，则有如下停机条件：

$$m(\alpha^k) - M(\alpha^k) \leq \varepsilon \quad (3.7)$$

而当违反 KKT 条件时应有：

$$\underset{i \in I_{up}}{u_i} < \underset{i \in I_{low}}{u_i} \quad (3.8)$$

以上详细导出过程详见附录。

3.2 SMO 算法

因 (3.1) 式形式上与用 SMO 算法解经典 SVM 有相似之处，程序中 QP 问题的求解方法参阅了[1]中相关成果。

算法 2 (SMO 算法——解超球 QP 问题)

1. 给定精度 ε ，令 $k = 1$ ， $\alpha^1 = \{1, 0, \dots, 0\}$

2. 如果 α^k 在精度 ε 内满足停机准则 (3.7)，则停止；

否则，由 WSS 算法选择 $B = \{i, j\}$ 。定义 $N = \{1, \dots, l\} \setminus B$ ， α_B^k 和 α_N^k 为 α^k 的子集，分别对应于集合 B 和 N。

3. 当 $d = k_{ii} + k_{jj} - 2k_{ij} > 0$ 时，求解以下 QP 子问题，得到 $\alpha_B = (\alpha_i \ \alpha_j)^T$ ：

$$\begin{aligned} \min_{\alpha_B} & \alpha_B^T H_{BB} \alpha_B + (-k_B^T + 2H_{BN} \alpha_N^K)^T \alpha_B \\ \text{s.t.} & \alpha_i + \alpha_j = \text{const}, \quad 0 \leq \alpha_i, \alpha_j \leq C \end{aligned} \quad (3.9)$$

$$\text{其中 } H_{BB} = \begin{bmatrix} k_{ii} & k_{ij} \\ k_{ij} & k_{jj} \end{bmatrix}, \quad k_B = \begin{pmatrix} k_{ii} \\ k_{jj} \end{pmatrix}$$

4. 将 α_B^{k+1} 设定为最优解，且 $\alpha_N^{k+1} = \alpha_N^k$ ， $k \leftarrow k+1$ ，转第 2 步

注：1) α^1 初始值选择：根据 (3.1) 式中约束条件，Lagrange 乘子中各分量之和应为 1；

2) 条件 $d = k_{ii} + k_{jj} - 2k_{ij} > 0$ 是必需的，否则程序异常。根据相关理论，如果核矩阵 H

为正定或半正定的，则对任何 $i \neq j$ ， $k_{ii} + k_{jj} - 2k_{ij} > 0$ 。程序中，求解过程只处理凸二

次规划问题，选用 RBF 核函数。

3) (3.9) 详细导出见[附录](#)

选定工作集 $B=\{i, j\}$ 后，由 (3.9) 可开始迭代求解 α_i, α_j ，求解算法如下：

算法 3 (迭代求解 α_i, α_j)

1. 由 $d=k_{ii}-2k_{ij}+k_{jj}$ ， u_i^* 及 u_j^* ，解析求解 (3.9) 得：

$$\alpha_i^{uc} = \alpha_i^* + \frac{u_j^* - u_i^*}{2d} \quad (3.10a)$$

2. 修正 α_i^{uc} ，定义可行域：

$$L = \max(0, \alpha_i + \alpha_j - C), \quad H = \min(\alpha_i + \alpha_j, C) \quad (3.10b)$$

则

$$\alpha_i = \begin{cases} L, & \alpha_i^{uc} < L \\ \alpha_i^{uc}, & L \leq \alpha_i^{uc} \leq H \\ H, & \alpha_i^{uc} > H \end{cases} \quad (3.10c)$$

3. 求得 $\alpha_j = \alpha_i^* + \alpha_j^* - \alpha_i$

注： $\alpha_i + \alpha_j = \alpha_i^* + \alpha_j^* = \text{常数}$ ， α_i^* ， α_j^* 为上一次迭代结束后所得值。

算法 4 (更新 u_i, u_j)

$$u_i = u_i^* + 2(\alpha_j - \alpha_j^*)k_{ij} + 2(\alpha_i - \alpha_i^*)k_{ii} \quad (3.11a)$$

$$u_j = u_j^* + 2(\alpha_j - \alpha_j^*)k_{jj} + 2(\alpha_i - \alpha_i^*)k_{ij} \quad (3.11b)$$

其中 u_i^* 及 u_j^* 为上一次迭代结束后所得结果。

以上详细导出过程详见[附录](#)。

3.3 WSS 算法

超球工作集 B 的选择有如下算法：

算法 5 (超球 QP 问题 WSS 算法)

1. 对所有 t, s，定义如下：

$$a_{ts} = k_{tt} - 2k_{ts} + k_{ss} > 0, \quad b_{ts} = -u_t + u_s > 0$$

2. $B = \{i, j\}$ ，选择如下：

$$i \in \arg \min_t \{u_t \mid t \in I_{up}(\alpha^k)\} \quad (3.12a)$$

$$j \in \arg \max_t \left\{ \frac{b_{it}^2}{a_{it}} \mid t \in I_{low}(\alpha^k), u_i < u_t \right\}$$

或者表示为：

$$i \in \arg \max_t \{-u_t \mid t \in I_{up}(\alpha^k)\} \quad (3.12b)$$

$$j \in \arg \min_t \left\{ -\frac{b_{it}^2}{a_{it}} \mid t \in I_{low}(\alpha^k), -u_i > -u_t \right\}$$

导出详见[附录](#)。

4 核矩阵存储

由于采用了 SMO 算法，迭代过程中可免去存储核矩阵而耗费大量存储空间。然后，这样也使得迭代过程每次用到 $K(\chi_i, \chi_j)$ 时都得重新再计算，耗费不必要的时间，且由于工作集的选择呈现出明显的 LRU（最近最少使用）特性，即如果此次选中了 i, j ，则以后的几次工作集选择中 i, j 仍然可能会被再次选中。因而，在程序中我们选择了两种方式或部分或全部地存储核矩阵，以加快计算。

4.1 UTM（Upper Triangular Matrix）方式

利用核矩阵 H 的对称特性我们可以利用 UTM（上三角矩阵）进行压缩存储，即只存储核矩阵对角线及其以上的数据（或者亦可用下三角阵矩阵存储）。上三角矩阵可以用一维数组进行存储，若设某类样本量为 n ，则该一维数组长度为：

$$l = [0.5n(n+1)] \quad (4.1)$$

若已知两样本 i, j ，则由以下公式即可在一维数组中定位所需 $K(\chi_i, \chi_j)$ 值：

$$T(i, j) = [0.5(2n - i - 1)i + j] \quad (4.2)$$

考虑具体程序设计语言细节，则存储核矩阵所需内存为：

$$\text{UTM 内存} = \frac{4n(n+1)}{10^6} \text{ MB} \quad (n \text{ 为某类样本量}) \quad (4.3)$$

UTM 存储时，由于核矩阵的规模仍然是以样本量的平方增长，所以当某类样本数据量很大时（如某类样本量为 2000 时，需内存约 16M，而当样本量达到 4000 时，所需内存空间约 64M），所需的存储空间会快速增长。因而，我们在程序中限定了 UTM 所能使用内存大小（目前程序暂设定为 24M，约允许某类拥有 2450 个样本）。程序执行时预先计算构建

UTM 所需的内存量，若 UTM 所需内存超过预设值，则使用 LRU Cache 来存储核矩阵的部分数据，此时可通过调整 Cache 因子，以达到内存空间耗费较少的目的。

4.2 LRU Cache 方式

由于采用了[算法 5](#)，程序的实际运行中工作集 i, j 的选择呈现 LRU 特性，因而程序中使用了具有 LRU 特性的 Cache 结构，如下图所示：

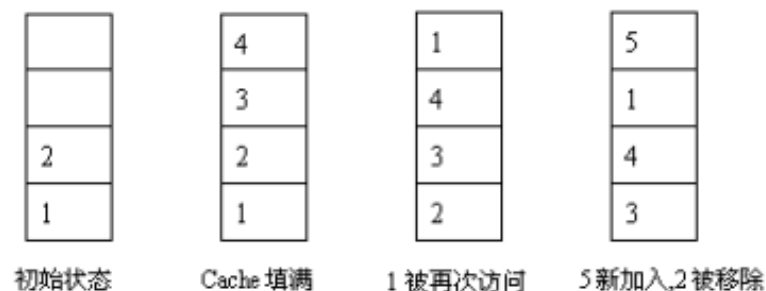


图 1 LRU Cache 原理

上图 1 中从左至右展示了 Cache 的状态变化情况。如图所示，Cache 容量为 4 行，采用 FIFO（先进先出）队列存储。程序初始时预先计算出核矩阵的 2 行存入 Cache 中；后续计算中，若找不到所需的核矩阵数据行，则计算所需的核矩阵行，使用并存入 Cache，直至 Cache 被填满；若某次计算所需的核矩阵行在 Cache 中被找到，则访问，并将被访问的数据行置于队列顶部，以表示最近刚被使用过；若后续又有新的数据行加入，则处于队列尾部的数据行将被移除，以表示最近最少使用。

程序中设定了一个 factor 因子，表示 Cache 中存储的核矩阵的多少行数据，该因子也决定了 Cache 的大小。理论上， $0 < \text{factor} \leq 1$ ，如当 $\text{factor} = 0.05$ ，某类样本量 $n = 4000$ 时，核矩阵 H 规模为 4000×4000 个数据，Cache 中存储 H 的行数为 $\text{rows} = n * \text{factor} = 200$ ，此时 Cache 所需内存可用以下公式进行计算：

$$\text{Cache 内存} = \frac{8n^2 * \text{factor}}{10^6} \text{ MB} \quad (n \text{ 为某类样本量}) \quad (4.4)$$

因而，通过调整 factor 值，理论上我们可以处理极大数目的样本量。当然，factor 越小，Cache 存储核矩阵行数越少，则命中率越低，每次迭代中程序需要进行核函数的计算次数增加，求解过程越慢，因而需要根据具体情况选择合适的 factor 值。

当样本量较小时，程序会优先构建 UTM，这样能大大提升计算速度。只有当某类样本量超过 UTM 允许使用的内存值，程序才会考虑构建 LRU Cache，以取得内存耗费及计算性能的平衡。

5 附录

5.1 超球 QP 问题 KKT 条件导出

根据 (3.1)，定义其 Lagrange 目标函数如下：

$$L(\alpha, \mu, \nu, b) = \alpha^T H \alpha - k^T \alpha + b(e^T \alpha - 1) - \nu^T \alpha + \mu^T (\alpha - C e) \quad (5.1)$$

其中 μ, ν, b 均为 Lagrange 乘子， μ, ν 为向量，且 $\nu \geq 0$ ， $\mu \geq 0$ 。对 (5.1) 各变量求偏导，有：

$$\nabla L(\alpha) = 2H\alpha - k + be - \nu + \mu = 0 \Leftrightarrow 2H\alpha - k + be = \nu - \mu$$

$$\nabla L(\mu) = \alpha - Ce \leq 0 \Rightarrow \mu^T (\alpha - Ce) = 0, \quad \mu \geq 0$$

$$\nabla L(\nu) = -\alpha \geq 0 \Rightarrow \nu^T \alpha = 0, \quad \nu \geq 0$$

所以，(3.1) KKT 条件可表述如下：

$$e^T \alpha = 1, 0 \leq \alpha \leq Ce \quad (5.2a)$$

$$2H\alpha - k + be = \nu - \mu \quad (5.2b)$$

$$\mu^T (\alpha - Ce) = 0 \quad (5.2c)$$

$$\nu^T \alpha = 0 \quad (5.2d)$$

$$\mu \geq 0, \nu \geq 0 \quad (5.2e)$$

定义 $u_i = \nabla L(\alpha)_i = (2H\alpha - k)_i$ ，则有 $u_i = 2 \sum_{j=1}^I \alpha_j k_{ij} - k_{ii}$

当 $\alpha_i = 0$ 时，由 (5.2c)，知此时 $\mu = 0$ ，且 $\nu \geq 0$ ，代入 (5.2b) 则有：

$$2H\alpha - k + be \geq 0, \quad \text{即 } u_i + b \geq 0 \quad (5.3a)$$

当 $\alpha_i = C$ 时，由 (5.2d)，知此时 $\nu = 0$ ，且 $\mu \geq 0$ ，则有：

$$2H\alpha - k + be \leq 0, \quad \text{即 } u_i + b \leq 0 \quad (5.3b)$$

当 $0 < \alpha_i < C$ 时，由 (5.2c)，(5.2d)，知此时 $\mu = 0$ ， $\nu = 0$ ，所以有：

$$2H\alpha - k + be = 0 \Rightarrow b = k_{ii} - 2 \sum_{j=1}^I \alpha_j k_{ij} = -u_i \quad (5.3c)$$

为避免每次迭代 b 值的计算及 α_i 不同取值情况下判断的复杂性，考虑简化 KKT 条件。定义集合：

$$I_{up}(\alpha) = \{i | \alpha_i < C\} \quad , \quad I_{low}(\alpha) = \{i | \alpha_i > 0\} \quad (5.4a)$$

且同时定义：

$$I_0 = \{i | \alpha_i = 0\} \quad , \quad I_1 = \{i | 0 < \alpha_i < C\} \quad , \quad I_2 = \{i | \alpha_i = C\} \quad (5.4b)$$

则

$$I_{up}(\alpha) = I_0 \cup I_1 \quad , \quad I_{low}(\alpha) = I_1 \cup I_2 \quad (5.4c)$$

$I_{up}(\alpha)$ 与 $I_{low}(\alpha)$ 共同交集为 I_1

则可合并 (5.3a) (5.3b) (5.3c) 三种情况，KKT 条件改为如下表述方式：

$$u_i + b \geq 0 \quad i \in I_{up}(\alpha) \quad (5.5a)$$

$$u_i + b \leq 0 \quad i \in I_{low}(\alpha) \quad (5.5b)$$

(5.5a) 两边变号与 (5.5b) 相加，消去 b 值，即产生如下 KKT 条件的简单表述：

$$\underset{i \in I_{up}}{u_i} \geq \underset{i \in I_{low}}{u_i} \quad (5.6)$$

或：

$$-\underset{i \in I_{up}}{u_i} \leq -\underset{i \in I_{low}}{u_i} \quad (5.7)$$

考虑精度 ε ，使 $\underset{i \in I_{up}}{u_i}$ 最小值大于等于 $\underset{i \in I_{low}}{u_i}$ 最大值，即为所要的结果 ([3.7](#))

5.2 超球 SMO 求解过程

5.2.1 超球 QP 子问题导出：

设 $\alpha = \begin{pmatrix} \alpha_B \\ \alpha_N \end{pmatrix}$ ， $H = \begin{pmatrix} H_{BB} & H_{BN} \\ H_{NB} & H_{NN} \end{pmatrix}$ ， $k = \begin{pmatrix} k_B \\ k_N \end{pmatrix}$ ，因 H 是对称的，所以 $H_{BN} = H_{NB}^T$ ，根据分

解算法 ([3.1](#)) 可改写为：

$$\begin{aligned} \min L(\alpha) &= \begin{pmatrix} \alpha_B^T & \alpha_N^T \end{pmatrix} \begin{pmatrix} H_{BB} & H_{BN} \\ H_{NB} & H_{NN} \end{pmatrix} \begin{pmatrix} \alpha_B \\ \alpha_N \end{pmatrix} - \begin{pmatrix} k_B^T & k_N^T \end{pmatrix} \begin{pmatrix} \alpha_B \\ \alpha_N \end{pmatrix} \\ &= \alpha_B^T H_{BB} \alpha_B + 2\alpha_B^T H_{BN} \alpha_N - k_B^T \alpha_B + \alpha_N^T H_{NN} \alpha_N - k_N^T \alpha_N \end{aligned} \quad (5.8)$$

$$s.t. \quad \alpha_B^T e_B + \alpha_N^T e_N = 1, \quad 0 \leq \alpha \leq Ce$$

因为在迭代过程中，需要调整的是 α_B ，而 α_N 是固定不变的，所以 $\alpha_N^T H_{NN} \alpha_N - k_N^T \alpha_N$ 为常数*，则问题 (5.8) 等价于：

$$\min L(\alpha_B) = \alpha_B^T H_{BB} \alpha_B + 2\alpha_B^T H_{BN} \alpha_N - k_B^T \alpha_B \quad (5.9)$$

$$s.t. \quad \alpha_B^T e_B = \text{常数}, \quad 0 \leq \alpha_B \leq Ce$$

在 SMO 算法中, 若设 $B=\{i, j\}$, 则 $\alpha_B = (\alpha_i \quad \alpha_j)^T$, $k_B = (k_{ii} \quad k_{jj})^T$, $H_{BB} = \begin{bmatrix} k_{ii} & k_{ij} \\ k_{ij} & k_{jj} \end{bmatrix}$,

代入 (5.9) 即可得到 (3.9) 所示子问题。

★: 在 SMO 算法的某一次迭代过程中, 因为只选取两个乘子 α_i, α_j 进行调整, 由于等式约束 ($\alpha_i + \alpha_j = \text{常数}$) 的限制, 这样调整 α_i 必然引起 α_j 的变化, 而其他所有的 α_N 是固定不变的, 因而 $\alpha_N^T H_{NN} \alpha_N - k_N^T \alpha_N$ 可视为常数。

5.2.2 解析解的导出

为推导过程中表述简单起见, 且不失一般性, 设 $\alpha_B = (\alpha_1 \quad \alpha_2)^T$, $k_B = (k_{11} \quad k_{22})^T$,

$$H_{BB} = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}, \quad H_{BN} = \begin{bmatrix} k_{13} & k_{14} & \dots & k_{1n} \\ k_{23} & k_{24} & \dots & k_{2n} \end{bmatrix}, \quad \alpha_N = (\alpha_3 \quad \alpha_4 \quad \dots \quad \alpha_n)^T, \text{ 则问题 (5.9)}$$

最终可化为如下形式:

$$L(\alpha_1, \alpha_2) = k_{11}\alpha_1^2 + 2k_{12}\alpha_1\alpha_2 + k_{22}\alpha_2^2 + \nu_1\alpha_1 + \nu_2\alpha_2 \quad (5.10)$$

$$s.t. \quad \alpha_1 + \alpha_2 = h = \text{常数}, \quad 0 \leq \alpha_1, \alpha_2 \leq C$$

其中

$$\nu_1 = 2 \sum_{i=3}^n \alpha_i^* k_{1i} - k_{11} = u_1^* - 2k_{11}\alpha_1^* - 2k_{12}\alpha_2^* \quad (5.11)$$

$$\nu_2 = 2 \sum_{i=3}^n \alpha_i^* k_{2i} - k_{22} = u_2^* - 2k_{12}\alpha_1^* - 2k_{22}\alpha_2^*$$

以上带*的变元均为上一次迭代结束时的值。以 $h - \alpha_2$ 替换 α_1 , 则 (5.10) 变为如下一元形式:

$$L(\alpha_2) = (k_{11} - 2k_{12} + k_{22})\alpha_2^2 + (2hk_{12} - 2hk_{11} + \nu_2 - \nu_1)\alpha_2 + h^2k_{11} + h\nu_1 \quad (5.12)$$

求目标函数的极值点, 则有:

$$\nabla L(\alpha_2) = 2(k_{11} - 2k_{12} + k_{22})\alpha_2 + 2hk_{12} - 2hk_{11} + \nu_2 - \nu_1 = 0 \quad (5.13)$$

因预设条件 α_2 二阶导数 $d = k_{11} - 2k_{12} + k_{22} > 0$, 所以:

$$\alpha_2 = \frac{h(k_{11} - k_{12})}{d} + \frac{\nu_1 - \nu_2}{2d} \quad (5.14)$$

将 (5.11) 及 $\alpha_1^* + \alpha_2^* = h$ 代入 (5.14), 化简即得:

$$\alpha_2 = \alpha_2^* + \frac{u_1^* - u_2^*}{2d} \quad (5.15)$$

5.2.3 推导可行域:

由 $\alpha_1 = h - \alpha_2$, 且 $0 \leq \alpha_1, \alpha_2 \leq C$, 所以有:

$$0 \leq h - \alpha_2 \leq C \Leftrightarrow h - C \leq \alpha_2 \leq h \Leftrightarrow \alpha_1 + \alpha_2 - C \leq \alpha_2 \leq \alpha_1 + \alpha_2$$

由此定义 α_2 下界为 $\max(0, \alpha_1 + \alpha_2 - C)$, 上界为 $\min(\alpha_1 + \alpha_2, C)$, 得可行域 (3.10b)。

将 α_2 在该可行域内进行修正, 即可得到 (3.10c)。最后可根据 $\alpha_1 + \alpha_2 = h$ 求得 α_1

5.2.4 推导 u_i, u_j 迭代公式:

设某次迭代中已选取工作集为 $B = \{i, j\}$, 以下所有加*的变量均为上一次迭代结束时值。

现将 (3.3) 式展开可得:

$$\begin{aligned} u_i &= 2(\alpha_1 k_{i1} + \alpha_2 k_{i2} + \dots + \alpha_i k_{ii} + \dots + \alpha_j k_{ij} + \dots + \alpha_l k_{il}) - k_{ii} \\ &= 2\alpha_i k_{ii} + 2\alpha_j k_{ij} + 2 \sum_{t \neq i, j}^l \alpha_t^* k_{it} - k_{ii} \end{aligned} \quad (5.16)$$

又因为

$$u_i^* = 2 \sum_{t=1}^l \alpha_t^* k_{it} - k_{ii} = 2\alpha_i^* k_{ii} + 2\alpha_j^* k_{ij} + 2 \sum_{t \neq i, j}^l \alpha_t^* k_{it} - k_{ii},$$

则有

$$2 \sum_{t \neq i, j}^l \alpha_t^* k_{it} - k_{ii} = u_i^* - 2\alpha_i^* k_{ii} - 2\alpha_j^* k_{ij} \quad (5.17)$$

将 (5.17) 代入 (5.16), 所以有

$$u_i = 2\alpha_i k_{ii} + 2\alpha_j k_{ij} + u_i^* - 2\alpha_i^* k_{ii} - 2\alpha_j^* k_{ij}$$

整理后即可得到(3.11), 同理 u_j 也可类似求出。

5.3 WSS 算法导出

引: 已知某函数 $f(x)$, 由泰勒公式有:

$$f(x+d) - f(x) = f'(x)d + \frac{1}{2!}f''(x)d^2 + \frac{1}{3!}f'''(x)d^3 + \dots + \frac{1}{n!}f^{(n)}(x)d^n + \dots \quad (5.18)$$

其中 d 为 $f(x)$ 在 x 点的变化量, 若 d 足够小, 则上式可表示为:

$$f(x+d)-f(x) \approx f'(x)d + \frac{1}{2!}f''(x)d^2 \quad (5.19)$$

工作集选择即是要在每次迭代时找到最合适的违反对 (Violating pair)。合适的工作集, 能使目标函数(3.1)收敛更快, 以便在较少的迭代次数得到所需的解向量。程序中采用第二序信息 (Second Order Information) 方法选择工作集 (相关理论请参阅[2])。具体推导过程如下:

违反对 (Violating Pair): 如果 $i \in I_{up}(\alpha)$, $j \in I_{low}(\alpha)$, 并有 $u_i < u_j$, 那么 $\{i, j\}$ 即为一个违反对。

定理 1 (Hush and Scovel, 2003):

如果核矩阵 H (如前所述) 是半正定的, 当且仅当所选工作集 $B=\{i, j\}$ 是一个违反对时, (3.1) 所示目标函数值严格递减 (如 $L(\alpha^{k+1}) < L(\alpha^k), \forall k$)。

因而, 由 (3.1) 及 (5.19), 有:

$$L(\alpha^k + d) - L(\alpha^k) = \nabla L(\alpha^k)^T d + \frac{1}{2} d^T \nabla^2 L(\alpha^k) d \quad (5.20)$$

其中 d 为第 $k+1$ 次迭代时乘子向量 α^{k+1} 相对于 α^k 的变化量, $d=(d_B, d_N)$ 。根据 5.2 节所讨论的结果, 由 (5.20) 可得到下式:

$$L(\alpha^k + d) - L(\alpha^k) = \nabla L(\alpha^k)_B^T d_B + \frac{1}{2} d_B^T \nabla^2 L(\alpha^k)_{BB} d_B \quad (5.21)$$

如前所述, $d_B=(d_i, d_j)^T$, $\nabla L(\alpha^k)_B = u_B = (u_i, u_j)^T$, $\nabla^2 L(\alpha^k)_{BB} = 2H_{BB}$, 并根据定理 1, 我们得到如下目标函数:

$$Sub(B) = \min_{d_B} \left(\frac{1}{2} d_B^T \nabla^2 L(\alpha^k)_{BB} d_B + \nabla L(\alpha^k)_B^T d_B \right) \quad (5.22)$$

$$= \min_{d_B} (d_B^T H_{BB} d_B + u_B^T d_B) \quad (5.23)$$

$$s.t. \quad e_B^T d_B = 0; \quad (5.24)$$

$$\text{若 } \alpha_i^k = 0, \quad d_i \geq 0, \quad i \in B; \text{ 若 } \alpha_i^k = C, \quad d_i \leq 0, \quad i \in B.$$

注: 关于 (5.23) 的导出: $\alpha^K + d$ 作为选取了工作集 $B=\{i, j\}$ 后 α^K 的调整后的值, 若

设 $d=(d_B, d_N)$, 显然 $d_N=0$, 则 $e_N^T d_N=0$ 。根据约束条件 $e^T(\alpha^K + d)=1$, $e^T \alpha^K = 1$,

$$\text{从而有 } e^T d = 0, \text{ 即 } (e_B^T, e_N^T) \begin{pmatrix} d_B \\ d_N \end{pmatrix} = 0 \Leftrightarrow e_B^T d_B + e_N^T d_N = 0 \Rightarrow e_B^T d_B = 0$$

将 (5.23) 展开则有:

$$Sub(B) = \min_{d_B} (k_{ii}d_i^2 + 2k_{ij}d_id_j + k_{jj}d_j^2 + u_id_i + u_jd_j) \quad (5.25)$$

由 $e_B^T d_B = 0$, 则有 $d_i = -d_j$, 代入 (5.25), 即得到:

$$Sub(B) = \min_{d_B} [(k_{ii} - 2k_{ij} + k_{jj})d_j^2 + (-u_i + u_j)d_j] \quad (5.26)$$

因 $B = \{i, j\}$ 是一个违反对, 即有 $-u_i + u_j > 0$, 且 $k_{ii} + k_{jj} - 2k_{ij} > 0$, 所以定义:

$$a_{ij} = k_{ii} + k_{jj} - 2k_{ij} > 0, \quad b_{ij} = -u_i + u_j > 0 \quad (5.27)$$

(5.26)中对 d_j 求导并令导数为 0, 则有目标函数最小值为:

$$Sub(B) = -\frac{b_{ij}^2}{4a_{ij}} < 0 \quad (5.28)$$

且该最小值点在:

$$d_j = -\frac{b_{ij}}{2a_{ij}} \quad (5.29)$$

当由 WSS 算法具体求工作集 B 中的 j 时, 可对(5.28)进行缩放, 即用 $-\frac{b_{ij}^2}{a_{ij}}$ 取代(5.28)作为

目标函数最小值, 由此便得到 (3.12) 所示结果。

致谢

程序中的算法, 大量运用了前人的研究成果, 他们的辛劳与智慧, 是我们的工作得以进行的基础, 感谢他们。

程序的设计与实现过程中, 得到了合作者王凯、唐小彪的大力支持, 正是他们使得我有机会接触这一领域。他们慷慨提供的大量资料、热心地回答各种理论与应用方面的问题、程序成型后所进行的大量辛苦的测试与验证、及由此提出的各种修改意见, 使得该软件的产生与发布成为可能。

参考文献

- [1] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines, 2009. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [2] R.-E.Fan, P.-H.Chen, and C.-J.Lin. Working set selection using second order information for training SVM. Journal of Machine Learning Research, 6:1889-1918, 2005. URL: <http://www.csie.ntu.edu.tw/~cjlin/papers/quadworkset.pdf>.

- [3] J.C.Platt. Fast training of support vector machines using sequential minimal optimization. In C.J.C.Burges, and A.J.Smola, editors, Advances in Kernel Methods-Support Vector Learning, Cambridge, MA, 1998. MITPress.
- [4] S.S.Keerthi, S.K.Shevade, C.Bhattacharyya, and K.R.K.Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. Neural Computation, 13:637–649, 2001.
- [5] Edgar Osuna, Robert Freund, Federico Girosi. An Improved Training Algorithm for Support Vector Machines. The Proc. of IEEE NNSP'97, Amelia Island, Fl, 24-26 Sep., 1995.
- [6] 朱美琳 ,刘向东 ,陈世福.用球结构的支持向量机解决多分类问题 [J].南京大学学报 (自然科学版) , 2003, 39 (2) : 153-158
- [7] 袁胜发,褚福磊.球结构支持向量机在转轴碰摩位置识别中的应用.振动与冲击,2009,28(8):70-73
- [8] 王凯.改进型超球 SVM 求解过程.2009,10
- [9] 邓乃扬,田英杰. 数据挖掘中的新方法----支持向量机.科学出版社,2004.
- [10]张学工.关于统计学习理论与支持向量机.自动化学报,2000,26(1): 32- 41.