Homework 05 SVM

王晓捷 (11521053)

May 25, 2016

1 引言

支持向量机（support vector machines，SVM）是一种二类分类模型。它的基本模型是定义在特征空间上的间隔最大的线性分类器，间隔最大使它有别于感知机；支持机还包括核技巧，这使其成为实质上的非线性分类器。支持向量机的学习策略就是间隔最大化，可形式化为一个求解凸二次规划（convex quadratic programming）的问题。支持向量机的学习算法是就是求解凸二次规划的最优化算法[1]。

支持向量机构建一个或多个高维的超平面$\omega \cdot x + b = 0$来分类数据点，这个超平面即为分类边界。直观来说，好的分类边界要距离最近的训练数据点越远越好，因为这样可以减少分类器的泛化误差。支持向量就是离分隔超平面最近的那些点。在SVM 中，分类超平面与最近的训练数据点之间的距离称为间隔（margin），分为函数间隔和几何间隔[2]。定义超平面(ω, b)关于样本点(x_i, y_i)的函数间隔为

$$\hat{\gamma_i} = y_i(\frac{\omega_i}{\|\omega\|} \cdot x_i + \frac{b}{\|\omega\|}) \tag{1.1}$$

定义超平面(ω, b)关于训练数据集T的函数间隔为超平面(ω, b)关于T中所有样本点(x_i, y_i) 的函数间隔的最小值，即

$$\hat{\gamma} = \min_{i=1,...,N} \hat{\gamma_i} \tag{1.2}$$

另设超平面(ω, b)关于样本点(x_i, y_i)的几何间隔为γ_i，有

$$\gamma_i = y_i(\frac{\omega_i}{\|\omega\|} \cdot x_i + \frac{b}{\|\omega\|}) \tag{1.3}$$

定义超平面(ω, b)关于训练数据集T的几何间隔为超平面(ω, b)关于T中所有样本点(x_i, y_i)的几何间隔的最小值，即

$$\gamma = \min_{i=1,...,N} \gamma_i \tag{1.4}$$

可看出，函数间隔与几何间隔之间的关系，即$\gamma_i = \frac{\hat{\gamma_i}}{\|\omega\|}$。

SVM的目标就是找出分类器定义中的ω和b，为此必须找到具有最小几何间隔的数据点，然后对该间隔最大化，即

$$arg \max_{\omega,b} \{\min_n (y_i(\boldsymbol{\omega^T x} + b)) \cdot \frac{1}{\|\omega\|}\} \tag{1.5}$$

直接求解1.5是相当困难的。通过假设所有支持向量的函数间隔都为1，即$y_i(\boldsymbol{\omega^T x} + b) = 1$，从而得到下面的线性可分SVM学习的最优化问题：

$$\min_{\omega,b} \quad \frac{1}{2}\|\omega\|^2 \tag{1.6}$$

$$s.t. \quad y_i(\omega \cdot x_i + b) - 1 \geq 0, i = 1, 2, ..., N \tag{1.7}$$

引入拉格朗日乘子，得到(1.6)-(1.7)的对偶问题形式为：

$$max_\alpha \quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \tag{1.8}$$

$$s.t. \quad \alpha_i \geq 0, \quad i = 1, ..., m \tag{1.9}$$

$$\sum_{i=1}^{m} \alpha_i y^(i) = 0 \tag{1.10}$$

其中，可得

$$\omega = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \tag{1.11}$$

另外，(1.8)中的$\langle x^{(i)}, x^{(j)} \rangle$既可以转化为使用各种核函数来计算，从而将线性不可分的数据转换为线性可分的数据。可用的核函数有线性核、指数核、高斯核等。其中：

- 线性核：$kernel(x_i, x_j) = \langle x_i, x_j \rangle$
- 指数核：$kernel(x_i, x_j) = exp(-\gamma |x_i - x_j|^2), \quad \gamma > 0$
- 高斯核：$kernel(x_i, x_j) = exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$

然而并不是所有的数据都很“干净”，大多数情况下会存在一些数据处于分割面的错误一侧。这时，可以通过引入松弛变量来进行优化目标，此时需要优化的问题形式变为：

$$\max_{alpha} \quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \tag{1.12}$$

$$s.t. \quad 0 \le \alpha_i \le C, \quad i = 1, ..., m \tag{1.13}$$

$$\sum_{i=1}^{m} \alpha_i y^(i) = 0 \tag{1.14}$$

该问题的KKT条件为：

$$\alpha_i = 0 \quad \Rightarrow \quad y^{(i)}(\omega^T x^{(i)} + b) \ge 1 \tag{1.15}$$

$$\alpha_i = C \quad \Rightarrow \quad y^{(i)}(\omega^T x^{(i)} + b) \le 1 \tag{1.16}$$

$$0 < \alpha_i < C \quad \Rightarrow \quad y^{(i)}(\omega^T x^{(i)} + b) = 1 \tag{1.17}$$

优化该问题，可以使用二次规划算法进行求解。还可以使用SMO算法来求解。具体实现细节在下一节进行描述。

2 方法概述

2.1 利用CVXOPT包求解二次规划问题

标准凸二次规划问题形式为：

$$min_x \quad \frac{1}{2} x^T P x + q^T x \tag{2.1}$$

$$s.t. \quad Gx \le h \tag{2.2}$$

$$Ax = b \tag{2.3}$$

CVXOPT中有方法$solvers.qp(P, q, G, h, A, b)$可以求解凸二次规划问题。故需将(1.12)-(1.14)中对应的P,q,G,h,A,b找到：

- $P_{i,j} = y^{(i)} y^{(j)} \langle x^{(i)} x^{(j)}$
- $q = np.ones((m, 1)) \times -1$
- $A = y$
- $b = 0$
- 若没有outliers:
  - $G = np.Identity(m) \times -1$
  - $h = np.zeros((m, 1))$
- 若有outlier:
  - $G = np.vstack(G1, G2)$
    其中$G1 = np.diag(m) \times -1$，$G2 = np.Identity(m)$
  - $h = np.vstack(h1, h2)$
    其中$h1 = np.zeros((m, 1))$，$h2 = np.ones((m, 1)) \times C$

然后调用CVXOPT.$solvers.qp(P, q, G, h, A, b)$即可求得$(\omega, b)$。

2.2 SMO算法

1996年，John Platt[3]发布了一个称谓SMO(Sequential Minimal Optimization，序列最小优化)的强大算法，用于训练SVM。SMO将大优化问题分解为多个小优化问题来求解。这些小优化问题往往很容易求解，并且对它们进行顺序求解的结果与将它们作为整体来求解的结果是完全一致的。在结果完全相同的同时，SMO 算法的求解时间短很多。

SMO算法的工作原理是：每次循环中选择两个α来进行优化处理。一旦找到一对合适的α，那么久增大其中一个，同时减小另一个。这里的“合适”就是指两个α必须要符合一定的条件，条件之一就是这两个α必须要在间隔边界之外，而其第二个条件则是这两个α还没有进行过区间化处理或者不在边界上。该方法主要包括两个α的选择、α的优化。假设选择的α分别为α_i和α_j。

首先，选择α_i和α_j，需要两个循环，选择α_i的为外循环，选择α_j的为内循环。

- 最外层循环：首先，在所有样本中选择违反KKT条件的一个乘子（即α_i）作为最外层循环，用内层循环方法选择另外一个乘子（即α_j）并进行这两个乘子的优化。在所有样本都遍历一遍后，接着从所有非边界样本中选择违反KKT条件的一个乘子（α_i）作为最外层循环，再用内层循环方法选择另外一个乘子（即α_j）并进行这两个乘子的优化。最后，如果上述非边界样本中没有违反KKT条件的样本，则再从整个样本中去找，直到所有样本中没有需要改变的乘子。

- 内层循环：用于确定第二个乘子（即α_j）的方法:

  - 首先在非界乘子中寻找使$|E_i - E_j|$最大的样本；
  - 如果没有找到,则从随机位置查找非界乘子样本；
  - 如果还没有找到，则从随机位置查找整个样本（包含界上和非界乘子）。

其次，对α_i和α_j的优化：

- 针对α_j: $\alpha_j^{new} = \alpha_j^{old} + \frac{y_j(E_i - E_j)}{\eta}$。由于每个$\alpha$的取值都限制在$[0, C]$，需要对其进行一定的滑动。

- 针对α_i: $\alpha_i^{new} = \alpha_i^{old} + \frac{y_i y_j}{\alpha_j^{old} - \alpha_j^{new}}$

对α_i和α_j进行优化后，需要对b进行优化：

$$b1 = b - E_i - y_i(\alpha_i^{new} - \alpha_i^{old})k(x_i, x_i) - y_j(\alpha_j^{new} - \alpha_j^{old})k(x_i, x_j) \tag{2.4}$$

$$b2 = b - E_i - y_i(\alpha_i^{new} - \alpha_i^{old})k(x_i, x_j) - y_j(\alpha_j^{new} - \alpha_j^{old})k(x_j, x_j) \tag{2.5}$$

如果$0 < \alpha_i < C$，则$b^{new} = b1$；如果$0 < \alpha_j < C$，则$b^{new} = b2$；其他情况下，$b^{new} = (b1 + b2)/2$。

3 实验结果

实验部分分别采用了调用CVXOPT中凸二次优化算法包和SMO两种方法，并对结果进行了一定的比较分析。

线性可分数据由2 个均值分别为$(0, 2)$ 和$(2, 0)$，协方差为$([0.8, 0.6], [0.6, 0.8])$的2维混合高斯模型生成，每个模型生成100个数据，分别标记为1和-1。其中，每个模型生成的数据的前80%用于训练，后20%用于测试。使用线性核对其进行分类与预测。

线性不可分数据中使用了2种数据，每种数据都使用了多种核函数进行分类训练与预测，从而得出不同的非线性可分的数据需要使用的核函数的类型也是不同的。每种类别的数据的前80%用于训练，后20%用于测试。两种非线性可分数据：

- 半径为1的上半圆标记为类别1，半径为4的上半圆标记为类别-1。每个类别分别随机生成200个数据，其中160个数据用来训练，剩余40个数据用来测试。

- 4个2维混合高斯模型，每个模型均值不同，协方差相同，每个模型随机生成50个数据。将前两个模型产生的数据标记为1，后面两个模型标记为-1。

3.1 线性可分数据

针对线性可分数据使用线性核的具体实验结果如图1所示。

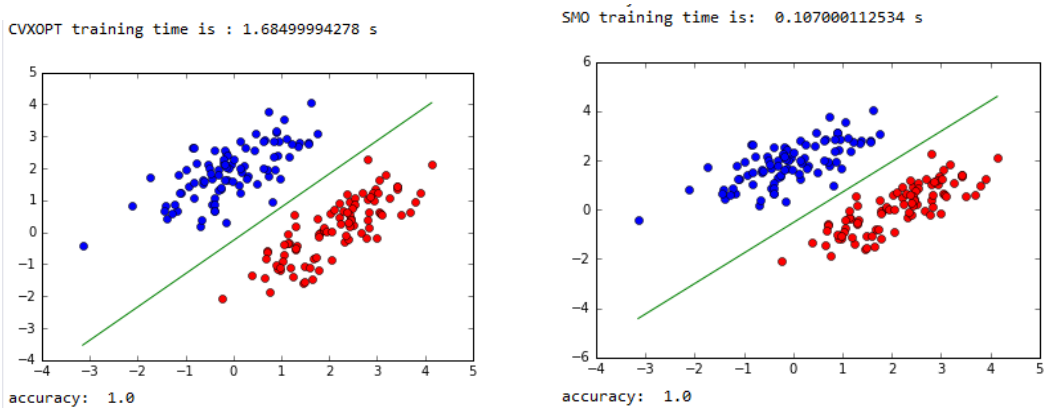

图 1: 线性可分数据实验结果。左侧图调用CVXOPT；右侧使用SMO方法。图片顶部标注了参数训练时间，底部标注了在测试集上进行测试的准确率。

3.2 线性不可分数据

- 半圆数据。

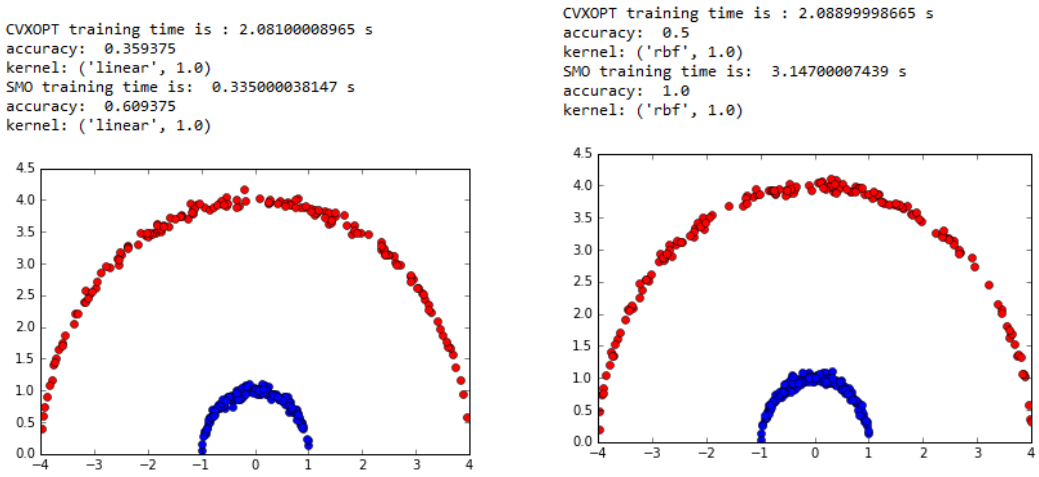

图 2: 半圆非线性可分数据实验结果。左侧使用的线性核，右侧使用了指数核。

- 混合高斯模型数据。

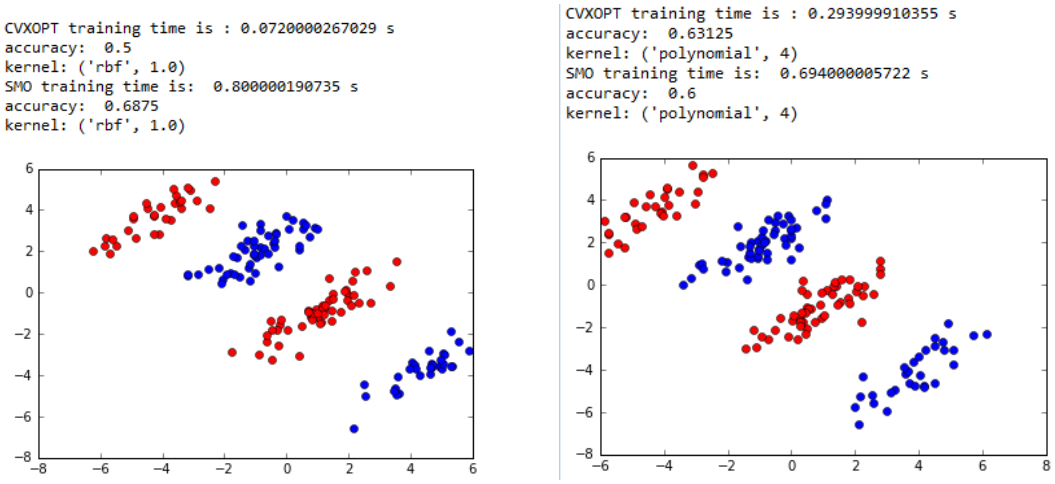

图 3: 混合高斯非线性可分数据实验结果。左侧使用的指数核，右侧使用了多项式核。

4 小结与讨论

由第3部分的实验结果可以观察出一下现象：

- 从图1可看出，针对线性可分数据，两种方法都能够很好地找出分隔超平面，并且在测试数据上的准确率为100%。这可以说明SMO的算法实现没有问题。另外，可以看到两种方法的训练时间，调用CVXOPT 的方法比SMO的训练时间要多了10多倍。说明在针对线性可分数据，使用线性核SMO 算法很高效。
- 半圆生成的非线性可分数据实验结果如图2所示。左侧使用线性核，训练时间SMO 更少，但是准确率两者都很差，SMO方法的准确率较高一些。说明，非线性数据不适合使用线性核。右侧为使用指数核，调用CVXOPT 的方法的训练时间与使用线性核差别不大，但是SMO方法使用非线性核时的训练时间明显提高，比使用CVXOPT方法的训练时间还要多。不过SMO的准确率可以达到100%，是使用CVXOPT 的准确率的两倍。
- 由4个混合高斯模型生成的非线性可分数据的实验结果如图3所示。左侧使用指数核，准确率均不高。左侧使用多项式核，准确率仍然不是很好。说明不同的线性数据需要不同的核函数，这两种核函数都不太适合这组非线性可分数据。另外，从训练时间可以看出，SMO方法的训练时间最长。
-

综上，针对线性可分数据，使用线性核，SMO方法可以快速高效地计算出分隔超平面，并且对测试数据可以达到很高的准确率。针对非线性可分数据，需要根据数据的情况选择不同的核函数，从而将其改变成线性可分数据，不同的核函数改造能力不同。在使用非线性核函数时，SMO算法的训练时间明显变长。

参考文献

[1] Peter Harrington. Machine learning in action. Manning, 2012.

[2] Andrew Ng. Cs229 lecture notes. CS229 Lecture notes, 1(1):1–3, 2000.

[3] John C Platt. 12 fast training of support vector machines using sequential minimal optimization. Advances in kernel methods, pages 185–208, 1999.