MS328 Assignment3

周李韬 518030910407

2020 年 4 月 19 日

SVM 方法

1. 选择几个点手动计算出 SVM 的解并与软件计算出的答案进行比对 (如果有可能还可以尝试考虑不可分引入核)；

2. 对真实分类数据尝试逻辑回归、线性判别分析、SVM 三种方法比较。(可以练习各种重抽样所得分类结果以及 SVM 中调参等。)

1 问题分析

SVM 的基本型如下所示。

$$\begin{aligned}&\min_{\boldsymbol{w},b}\frac{1}{2}\|\boldsymbol{w}\|^2\\&\text{s.t. } y_i\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i+b\right)\geqslant 1,\quad i=1,2,\ldots,m\end{aligned}$$

利用拉格朗日乘子法，得到优化问题的 Lagrangian

$$L(\boldsymbol{w},b,\boldsymbol{\alpha})=\frac{1}{2}\|\boldsymbol{w}\|^2+\sum_{i=1}^{m}\alpha_i\left(1-y_i\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i+b\right)\right)$$

我们可以求出 SVM 基本型的对偶函数。

$$\begin{aligned}g(\boldsymbol{\alpha})&=\inf_{\boldsymbol{w},b\in\mathcal{D}}L(\boldsymbol{w},b,\boldsymbol{\alpha})\\&=\inf_{\boldsymbol{w},b\in\mathcal{D}}\frac{1}{2}\|\boldsymbol{w}\|^2+\sum_{i=1}^{m}\alpha_i\left(1-y_i\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i+b\right)\right)\end{aligned}$$

对 Lagrangian 函数求偏导，简化得。

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i$$
$$0 = \sum_{i=1}^{m} \alpha_i y_i$$

将以上等式和约束代入对偶函数，可得到对偶问题。

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{x}_j$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y_i = 0$$
$$\alpha_i \geqslant 0, \quad i = 1, 2, \ldots, m$$

由于 SVM 是一个凸优化问题，我们只需求解以上对偶问题，将 α 代回求解 ω、β 即可得到 SVM 的解。上述对偶问题是一个二次规划问题，求解运算量正比于样本量。若希望提高求解的效率，注意到约束 $\sum_{i=1}^{m} \alpha_i y_i = 0$ 的存在，若固定 α_i 以外的变量，α_i 可以由其他变量导出。因此可采用迭代求解法 SMO (Sequential Minimal Optimization)，即每次选定两个优化变量 α_i, α_j，固定其他变量，求解简化后的优化问题，不断迭代完成求解。

注意到 KKT 条件是凸优化问题强对偶成立的充要条件，因此利用 KKT 条件我们还可以启发式地选择迭代变量，只要我们选取的 α_i, α_j 不满足 KKT 条件，迭代就是有效的。进一步，我们可以选择偏离 KKT 条件最大的 α_i，和对应的，与该 α_i 间隔最大的 α_j 进行优化，使每一次迭代的效果最好。

本实验中，考虑到 SMO 算法实现较为复杂，我们直接调用 sklearn 相关模块进行 SVM 求解。

2 实验 1：手动求解

我们尝试用一些简单的数据手工求解 SVM 问题。我们取二维空间上的样本 (1,2),(3,5) 标签为正，(2,1),(4,3) 标签为负。对应的 SVM 问题如下所示。

$$\min_{\omega,\beta} \frac{1}{2}(\omega_1^2 + \omega_2^2)$$
$$\text{s.t.} \begin{cases} \omega_1 + 2\omega_2 + b & \geq 1 \\ 3\omega_1 + 5\omega_2 + b & \geq 1 \\ 2\omega_1 + \omega_2 + b & \leq -1 \\ 4\omega_1 + 3\omega_2 + b & \leq -1 \end{cases}$$

该问题的 Lagrangian 如下。

$$L(\boldsymbol{\omega},\beta,\boldsymbol{\lambda})=\frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\omega}+\boldsymbol{\lambda}^T\begin{bmatrix}-1 & -2 & -1 & 1\\ -3 & -5 & -1 & 1\\ 2 & 1 & 1 & 1\\ 4 & 3 & 1 & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{\omega} & b & 1\end{bmatrix}^T$$

根据上一节中的结论我们有

$$\omega=\begin{bmatrix}\lambda_1+3\lambda_2-2\lambda_3-4\lambda_4\\ 2\lambda_1+5\lambda_2-\lambda_3-3\lambda_4\end{bmatrix}$$

$$\lambda_1+\lambda_2=\lambda_3+\lambda_4$$

所以

$$\begin{bmatrix}-1 & -2 & -1 & 1\\ -3 & -5 & -1 & 1\\ 2 & 1 & 1 & 1\\ 4 & 3 & 1 & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{\omega} & b & 1\end{bmatrix}^T=\begin{bmatrix}-5 & -13 & 4 & 10 & -1 & 1\\ -13 & -34 & 11 & 27 & -1 & 1\\ 4 & 11 & -5 & -11 & 1 & 1\\ 10 & 27 & -11 & -25 & 1 & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda} & b & 1\end{bmatrix}^T$$

我们根据 λ 的取值进行讨论。

1. 此外也不可能有 3 个 λ_i 值为 0，否则第四个 λ_i 也一定为 0，得到 $\omega=0$，解无效。

2. 由 $\lambda_1+\lambda_2=\lambda_3+\lambda_4$，若存在两个 $\lambda_i=0$，两个 λ_i 必须在等号两侧，才能满足 $\boldsymbol{\lambda}\geq 0$. 在四种情况下，联立 2 个 $g_i(\omega,\beta)=0$ 方程与另两个 λ_i 的相等关系可以求解。

3. 在 $\lambda_2=\lambda_4=0$ 时，有 $\lambda_1=\lambda_3=1$. 因此 $\omega=(-1,\ 1),b=0$，超平面为 $g(x_1,x_2)=-x_1+x_2$. 支持向量为 (1,2,1),(2,1,-1)。我们还发现第四个样本 (4,3,-1) 虽然 $\lambda_4=0$，但也是支持向量，说明 $\lambda_i>0$ 是对应样本为支持向量的充分非必要条件。

4. 在 $\lambda_2=\lambda_3=0$ 时，有 $\lambda_1=\lambda_4=\frac{1}{5}$，此时样本 3 预测值为正，出现矛盾。

5. 在 $\lambda_1=\lambda_3=0$ 时，有 $\lambda_2=\lambda_4=\frac{2}{5}$，此时样本 1 预测值为负，出现矛盾。

6. 在 $\lambda_1=\lambda_4=0$ 时，有 $\lambda_2=\lambda_3=\frac{2}{17}$，此时样本 1 预测值为负，出现矛盾。

7. 若只有一项 $\lambda_i=0$, 联立三个方程组可以求解，该种情况不会添加新的解。

下面我们调用 SVM 包检验计算结果。

[1]:
```
import numpy as np
X = np.array([[1, 2], [3, 5], [2, 1], [4, 3]])
y = np.array([2, 2, 1, 1])
from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(X, y)
print(clf.coef_, clf.intercept_)
print (clf.support_vectors_)
```

```
[[-1.  1.]] [1.03620816e-15]
[[2. 1.]
 [4. 3.]
 [1. 2.]]
```

b 十分接近 0，经检验结果符合手工计算。

3 实验 2：三种分类模型比较

我们采用与作业 2 中相同的Breast Cancer Wisconsin (Diagnostic) Data Set 数据集进行模型分类与比较。该数据集中采集了 569 位病人的胸部细胞特征信息. 胸部细胞共有十类特征, 如细胞半径, 灰度, 细胞面积等, 均为实数值. 每一类特征具有平均值, 标准差, 极值三个域. 即每位病人有 30 个数值信息. 每位病人被分类为患癌 (M=malignant) 和健康 (B=benign). 数据集中, 共有 357 个健康样本, 212 个患癌样本. 本实验中, 我们取十种特征的平均值作为样本特征进行回归分类。数据经过归一化处理。

[2]:
```
import numpy as np
import pandas as pd
import math
data = pd.read_csv('wdbc.data',header=None,index_col=0,usecols=[0,1]+[2+3*i for␣
↪i in range(10)])

# 读取数据
def read_sig(data):
    x = []
    y = []
    for line in data.values:
        x.append(line[1:])
```

```python
        if line[0] == 'M':
            y.append([1])
        else:
            y.append([0])
    x = np.array(x)
    y = np.array(y)
    return x,y

# 归一化
def normalize(x):
    return ((x-x.mean(axis=0)) /(x.max(axis=0)-x.min(axis=0)))
```

我们使用留出法对模型进行简单测试，我们调用 sklearn 的方法将数据集随机分为 75% 的训练集和 25% 的测试集，调用牛顿法逻辑回归模型，训练结果如下所示。

[3]:
```python
x1, y = read_sig(data)
x = normalize(x1)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25,␣
 ↪random_state=42)
```

我们统计查准率、查全率和 F1 值衡量模型的分类效果。

[4]:
```python
def test_analysis(y_test,result):
    result = np.c_[result,y_test]
    TP, FP, TN, FN = 0, 0, 0, 0
    for i in range(result.shape[0]):
        # print (result[i])
        if (result[i][0] == 0 and result[i][1] == 0):
            TN += 1
        if (result[i][0] == 0 and result[i][1] == 1):
            FN += 1
        if (result[i][0] == 1 and result[i][1] == 1):
            TP += 1
        if (result[i][0] == 1 and result[i][1] == 0):
            FP += 1
    print ("TN: ",TN,"  FN: ",FN,"  TP: ",TP,"  FP: ",FP)
```

```
    print ("查准率 P: ", TP/(TP+FP))
    print ("查全率 R: ", TP/(TP+FN))
    print ("F1: ", 2*TP/(2*TP+FN+FP))
    return
```

3.1 线性模型

[5]:
```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train,y_train)
print (linreg.coef_)
print (linreg.intercept_)
```

```
[[ 2.81953999 -2.38499303  0.35857664 -0.11909062  0.64583153 -0.45577087
   0.33115516  0.56861437  0.44343077  0.72510486]]
[0.38299295]
```

[6]:
```
test_result = linreg.predict(X_test)[:,0]
X_pred = (np.sign(test_result - 0.5) + 1)/2
X_pred.astype(np.int64)
test_analysis(X_pred,y_test)
```

```
TN:  88   FN:  1   TP:  52   FP:  2
查准率 P:  0.9629629629629629
查全率 R:  0.9811320754716981
F1:  0.9719626168224299
```

3.2 逻辑回归模型

[7]:
```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(X_train,y_train[:,0])
print(log_reg.coef_)
print(log_reg.intercept_)
```

```
[[ 3.3702681   2.83163283  2.62042043 -0.98995057  1.54871095 -0.2862578
   0.12955475  2.93917019  1.98480414  4.45575328]]
[-0.67471005]
```

[8]:
```
test_analysis(log_reg.predict(X_test),y_test[:,0])
```

```
TN:  89   FN:  0   TP:  51   FP:  3
查准率 P:  0.9444444444444444
查全率 R:  1.0
F1:  0.9714285714285714
```

3.3 SVM

[9]:
```
svm_clf = SVC(kernel='linear')
svm_clf.fit(X_train,y_train[:,0])
print(svm_clf.coef_)
print(clf.intercept_)
```

```
[[ 2.72724598  2.33189605  1.96317924 -1.1793112   1.41942215 -0.89588497
   0.54595555  2.62125575  2.13241358  3.28471177]]
[1.03620816e-15]
```

[10]:
```
test_analysis(svm_clf.predict(X_test),y_test)
```

```
TN:  89   FN:  0   TP:  52   FP:  2
查准率 P:  0.9629629629629629
查全率 R:  1.0
F1:  0.9811320754716981
```

我们发现线性模型、逻辑回归模型和 SVM 都在该数据集达到了较好的预测效果，其中，SVM 的查准率、查全率和 F1 值均表现最好。