刘程华 2018011687 计91

prob 1

我们根据公式(17)(18)进行LTWSVM1函数的编写，根据公式(20)(21)进行LTWSVM2函数的编写。根据公式(22)进行LinearTWSVM函数的编写。值得一提的是，因为公式(16)(19)中$\left(B^T B\right)^{-1}$不是一定存在的，我们将其处理为$\left(B^T B+\epsilon I\right)^{-1}$确保逆一定存在。

我们选用心脏病数据集Heart-statlog data。其1-14维数据分别为:

1. age
2. sex
3. chest pain type (4 values)
4. resting blood pressure
5. serum cholesterol in mg/dl
6. fasting blood sugar > 120 mg/dl
7. resting electrocardiographic results (values 0,1,2)
8. maximum heart rate achieved
9. exercise induced angina
10. oldpeak = ST depression induced by exercise relative to rest
11. the slope of the peak exercise ST segment
12. number of major vessels (0-3) colored by flourosopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
14. Absence (1) or presence (2) of heart disease

我们根据前13项信息来预测是否有心脏病。我们在实验中选取前70%的数据作为训练数据，后30%的数据作为检测数据。对参数C_1 ,C_2遍历后取准确率最高时的参数，得到准确率accuracy=83.9506

代码见附录

刘程华　　2018011687　　计91

2、设 m_k 为第k类训练样本的数量，属于第k类数据样本为 $X_k \in R^{m_k \times n}$

设类别总数为M，$\sum_{i=1}^{M} m_i = m$

设 $Y_k = [X_1^T, X_2^T \cdots X_{k-1}^T, X_{k+1}^T \cdots X_m^T]^T \in R^{(m-m_k)\times n}$

为每个类构造超平面，每个超平面远离~~离~~一个类别的样本数据

第k个超平面方程为：　$w_k x + b_k = 0$

$$\underset{w_k, b_k, \xi_k}{\text{minimize}} \ \frac{1}{2}\| Y_k w_k + e_{k1} b_k \|_2^2 + C_k e_{k2}^T \xi_k$$

$$\text{s.t.} \quad (X_k w_k + e_{k2} b_k) + \xi_k \geq e_{k2} \qquad \xi_k \geq 0$$

$e_{k1} \in R^{(m-m_k)}$ $e_{k2} \in R^{m_k}$ 是全为1的向量，$C_k > 0$ 是惩罚参数。

ξ_k 是松弛变量

最后分类时选最远的：$\text{Class}(k) = \text{argmax}(k=1,2\cdots M) \dfrac{|w_k x + b_k|}{\|w_k\|}$

$$L(w_k, b_k, \xi_k, \alpha_k, \beta_k) = \frac{1}{2}\| Y_k w_k + e_{k1} b_k \|_2^2 + C_k e_{k2}^T \xi_k - \alpha_k^T((X_k w_k + e_{k2} b_k) + \xi_k - e_{k2}) - \beta_k^T \xi_k$$

KKT条件：

$$\frac{\partial L}{\partial w_k} = 0 \Rightarrow Y_k^T(Y_k w_k + e_{k1} b_k) - \alpha_k^T X_k = 0 \quad ①$$

$$\frac{\partial L}{\partial b_k} = 0 \Rightarrow e_{k1}^T(Y_k w_k + e_k b_k) - \alpha_k^T e_{k2} = 0 \quad ②$$

$$\frac{\partial L}{\partial \xi_k} = 0 \Rightarrow C_k e_{k2}^T - \alpha_k^T - \beta_k^T = 0$$

$$(X_k w_k + e_{k2} b_k) + \xi_k - e_{k2} \geq 0 \qquad \xi_k \geq 0$$

$$\alpha_k^T((X_k w_k + e_{k2} b_k) + \xi_k - e_{k2}) = 0 \qquad \beta_k^T \xi_k = 0$$

$$\alpha_k \geq 0, \ \beta_k \geq 0$$

①② $\Rightarrow$

$$\begin{bmatrix} Y_k^T \\ e_{k1}^T \end{bmatrix} [Y_k, e_{k1}] \begin{bmatrix} w_k \\ b_k^T \end{bmatrix} - \underbrace{\alpha_k^T \begin{bmatrix} X_k \\ e_{k2} \end{bmatrix}}_{= [X_k^T \ e_{k2}^T] \alpha_k} = 0$$

记 $H_k=[\,Y_k \quad e_{k_1}]$ 记 $G_k=\begin{bmatrix} X_k \\ e_{k_2} \end{bmatrix}$ 记 $u_k=\begin{bmatrix} w_k \\ b_k^T \end{bmatrix}$

$$H_k^T H_k u_k - G_k^T \alpha_k = 0 \Rightarrow u_k = (H_k^T H_k)^{-1} G_k^T \alpha_k$$

同 hw6 上 公式 (17) 我们得到对偶问题

$$\max_{\alpha_k} \quad e_{k_2}^T \alpha_k - \frac{1}{2} \alpha_k G_k (H_k^T H_k)^{-1} G_k^T \alpha_k$$

$$0 \le \alpha_k \le C_k$$

3、 $z \in R^n$， $P_z(u) = e^{-\phi(\|u\|_2)}$ $\phi: R^n \to R$ 凸且增

$x = Az+b$ $A \in R^{n\times n}$ 非奇异.

$x_1 \cdots x_N \overset{iid}{\sim} X$

$$P_x(v) = \frac{1}{|\det A|} e^{-\phi(\|A^{-1}(v-b)\|_2)}$$ 参数为 (A, b)

设 Q 为正交阵

$$|\det(AQ)| = |\det A|\,|\det Q| = |\det A|$$

$$\|(AQ)^{-1}(v-b)\|_2 = \|Q^T A^{-1}(v-b)\|_2 = \|A^{-1}(v-b)\|_2$$

所以，参数为 (AQ, b) 与 (A, b) 的密度分布相同

A 可逆 $\Rightarrow A = U\Sigma V^T$ 令 $Q = VU^T$

$AQ = U\Sigma U^T \in S_{++}^n \Rightarrow$ 因为 AQ 和 A 的密度分布相同

我们不妨假设 $A \in S_{++}^n$

由独立同分布可知对数似然函数为

$$l(A,b) = \sum_{i=1}^N \log P_x(x_i) = \sum_{i=1}^N \log\left[\frac{1}{|\det(A)|} e^{-\phi(\|A^{-1}(x_i-b)\|_2)}\right]$$

$$= -N\log(|\det(A)|) - \sum_{i=1}^N \phi(\|A^{-1}(x_i-b)\|_2)$$

上述求最大值 $\Leftrightarrow \sum_{i=1}^{N} \phi(\|A^{-1}(x_i - b)\|_2)$ 求最小值

令 $B = A^{-1} \in S_{++}^{n}$, $C = A^{-1}b \in R^n$

得到： $\underset{B,C}{\text{minimize}}: \sum_{i=1}^{N} \phi(\|Bx_i - C\|_2)$ 是凸优化问题

对目标函数关于 B、C 求导为零得 B,C，变换后即可得到 A 和 b.

LTWSVM1:

```
function [ wA, bA, EXITFLAG ] = LTWSVM1( xA, xB, C1 )
%LTWSVM1 Solves the Linear Twin SVM Dual QPP for the first plane

[N1,D]=size(xA);
[N2,D]=size(xB);

H=[xA,ones(N1,1)];
G=[xB,ones(N2,1)];

alpha0=[rand(N2,1)];

% Quadratic term objective
obj_quad=G*pinv(H'*H+eps*eye(size(H'*H)))*G';
obj_quad=obj_quad+eps*eye(size(obj_quad)); %Conditioning
obj_quad=(obj_quad+obj_quad')/2; %Making symmetric

% Linear term objective
obj_linear=-ones(size(alpha0,1),1);


% Setup inwquality constraints
A_ineq_const=[];
b_ineq_const=[];

% Setup equality constraints
A_eq_const=[];
b_eq_const=[];

% Setup bounds
lb=zeros(size(alpha0,1),1);
ub=C1*ones(size(alpha0,1),1);

try
    % Setup options
    options = optimoptions('quadprog','Algorithm','interior-point-
convex','Display','none');

    % Solve QPP
    [X, FVAL, EXITFLAG]=quadprog(obj_quad, obj_linear, A_ineq_const,
b_ineq_const, A_eq_const, b_eq_const, lb, ub, [], options);

    % Compute solution
    u=-pinv(H'*H + eps*eye(size(H'*H)))*G'*X;

    wA=u(1:end-1,:);
    bA=u(end,:);
```

```
catch
    wA=rand(N1+N2,1);
    bA=rand;
end
end
```

LTWSVM2:

```
function [ wB, bB, EXITFLAG ] = LTWSVM2( xA, xB, C2 )
%LTWSVM2 Solves the Linear Twin SVM Dual QPP for the second plane

[N1,D]=size(xA);
[N2,D]=size(xB);

P=[xA,ones(N1,1)];
Q=[xB,ones(N2,1)];

alpha0=[rand(N1,1)];

% Quadratic term objective
obj_quad=P*pinv(Q'*Q+eps*eye(size(Q'*Q)))*P';
obj_quad=obj_quad+eps*eye(size(obj_quad)); %Conditioning
obj_quad=(obj_quad+obj_quad')/2; %Making symmetric

% Linear term objective
obj_linear=-ones(size(alpha0,1),1);

% Setup inwquality constraints
A_ineq_const=[];
b_ineq_const=[];

% Setup equality constraints
A_eq_const=[];
b_eq_const=[];

% Setup bounds
lb=zeros(size(alpha0,1),1);
ub=C2*ones(size(alpha0,1),1);

try
    % Setup options
    options = optimoptions('quadprog','Algorithm','interior-point-
convex','Display','none');

    % Solve QPP
    [X, FVAL, EXITFLAG]=quadprog(obj_quad, obj_linear, A_ineq_const,
b_ineq_const, A_eq_const, b_eq_const, lb, ub, [], options);

    % Compute solution
```

```
    u=-pinv(Q'*Q + eps*eye(size(Q'*Q)))*P'*X;

    wB=u(1:end-1,:);
    bB=u(end,:);

catch
    wB=rand(N1+N2,1);
    bB=rand;
end

end
```

LinearTWSVM:

```
function [ yPred, accuracy, model ] = LinearTWSVM( xTrain, yTrain, xTest, yTest,
C1, C2 )
%LINEARTWSVM
% This function implements the linear Twin SVM formulation (dual) for binary
classification.
% Inputs:
% xTrain: Training data (samplesXfeatures)
% yTrain: Training labels (samplesX1) - should be +1/-1
% xTest: Testing data (test_samplesXfeatures)
% yTest: Testing labels (test_samplesX1) - should be +1/-1
% C1, C2: Hyperparameters for the two hyperplanes

[N,D]=size(xTrain);

% Pre-process data to make zero mean and unit variance
trainmean=mean(xTrain);
trainvar=var(xTrain);
for i=1:size(xTrain,1)
    xTrain(i,:)=(xTrain(i,:)-trainmean)./trainvar; %Normalize train data
end
for i=1:size(xTest,1)
    xTest(i,:)=(xTest(i,:)-trainmean)./trainvar; %Normalize test data
end


% Separate data of the two classes
A=xTrain(yTrain==1,:);
B=xTrain(yTrain==-1,:);

% Obtain Twin SVM hyperplanes
[ wA, bA, EXITFLAG1 ] = LTWSVM1( A, B, C1 );
[ wB, bB, EXITFLAG2 ] = LTWSVM2( A, B, C2 );

if (EXITFLAG1~=1 || EXITFLAG2~=1)
```

```
    fprintf(1, 'Optimization did not converge! --- EXITFLAG1 = %d --- EXITFLAG2 =
%d', EXITFLAG1, EXITFLAG2);
end
if (all(wA)==0)
    wA=rand(D,1);bA=rand;
end
if (all(wB)==0)
    wB=rand(D,1); bB=rand;
end

model.wA=wA;
model.wB=wB;
model.bA=bA;
model.bB=bB;
model.trainMean=trainmean;
model.trainVar=trainvar;

% Compute test set predictions
yPred=zeros(size(xTest,1),1);
for i=1:size(xTest,1)
    sample=xTest(i,:);
    distA=(sample*wA + bA)/norm(wA);
    distB=(sample*wB + bB)/norm(wB);
    if (distA>distB)
        yPred(i)=-1;
    else
        yPred(i)=1;
    end
end

accuracy=(sum(yPred==yTest)/length(yTest))*100;

% Sanity check - if labels are predicted wrongly then flip
if (accuracy<50)
    yPred=-1*yPred;
    accuracy=(sum(yPred==yTest)/length(yTest))*100;
end
end
```

run:

```
% Sample script for running Twin Support Vector Machine

clc;clearvars;close all;
rng default;

% Read Data
run('data_heartstatlog.m');
```

```
% Get Data and Labels
features=data(:,1:end-1);
labels=data(:,end);

% Normalize labels
labels(labels==2)=-1;

% Separate training and test data (80:20 split)
total_samples=size(features,1);
train_samples=round(0.7*total_samples);

% Define training and test samples
xTrain=features(1:train_samples,:);
yTrain=labels(1:train_samples,:);
xTest=features(train_samples+1:end,:);
yTest=labels(train_samples+1:end,:);

% Define hyperparameter values
C1=0.1; C2=0.05;

% Run Twin SVM (Linear)
 [ yPred, accuracy ] = LinearTWSVM( xTrain, yTrain, xTest, yTest, C1, C2 );

 disp('Accuracy (Linear) is');
 accuracy
```