公式章 1 节 

**HUNAN UNIV****ERSITY**

课 程 作 业

|  |  |
| --- | --- |
| **论文题目** | 基于matlab实现的支持向量机 |
|  | (SVM)算法 |
| 学生姓名 | 何景林 |
| 学生学号 | S230200192 |
| 专业班级 | 机械2304班 |
| 课程名称 | 工程优化方法 |
| 学院名称 | 机械与运载工程学院 |
| 课程老师 | 王琥 |
| 学院院长 | 丁荣军 |

2024年07月16日

基于matlab实现的支持向量机(SVM)算法

# 摘 要

本文基于matlab设计一个支持向量机(SVM)用来解决分类问题。设计的支持向量机主要包括二次惩罚、增广拉格朗日和InteriorPoint Barrier三种不同的优化方法以及线性、多项式、高斯和Sigmoid四种不同的核。同时使用在机器学习领域广泛使用的“一对多”的方法处理多类分类问题。使用的数据集是matlab自带的Fisher Iris数据，其包含三个类别、四个特征和 150 个样本。取样本中80% 的数据作为训练集，取样本的20%数据作为测试集。最后分析了不同优化方法、不同核化函数下、不同SVM分类器的收敛速率并对比不同优化方法、不同核化函数下的求解精度；结果表明，在该分类问题中，最佳内核是RBF，因为其精度一直很高；其中，增广拉格朗日法在所有内核中都具有最好的性能。

关键词：SVM；matlab；分类

目 录

[摘 要 I](#_Toc10461)

[插图索引 III](#_Toc16509)

[附表索引 IV](#_Toc23701)

[第1章 SVM简介 1](#_Toc18480)

[第2章 基于SVM方法的分类问题 2](#_Toc12600)

[2.1. 问题描述 2](#_Toc26246)

[2.2. SVM方法设计 2](#_Toc32158)

[2.2.1. 决策规则 2](#_Toc15036)

[2.2.2. 计算间隔宽度 2](#_Toc17886)

[2.2.3. 拉格朗日优化 3](#_Toc16373)

[2.2.4. 非完全线性可分的数据处理 4](#_Toc5740)

[2.2.5. 优化方法 6](#_Toc16195)

[2.2.6. 标签预测 8](#_Toc4281)

[2.2.7. 核函数 9](#_Toc1623)

[第3章 “一对多”的多种类分类 10](#_Toc15790)

[第4章 Matlab实现 12](#_Toc3206)

[第5章 结果与讨论 19](#_Toc18708)

[5.1. 结果 19](#_Toc16666)

[5.2. 收敛速率 19](#_Toc5416)

[5.2.1. 二次惩罚法 19](#_Toc21770)

[5.2.2. 增广拉格朗日法 21](#_Toc20464)

[5.2.3. InteriorPoint Barrier法 22](#_Toc9099)

[5.3. 精度比较 23](#_Toc16390)

[结 论 25](#_Toc14364)

[参考文献 26](#_Toc27896)

# 插图索引

[图 1 最大间隔超平面 1](#_Toc25540)

[图 2 铰链损失（蓝色）和零1个损失（绿色） 5](#_Toc8121)

[图 3 分类结果 19](#_Toc21823)

[图 4 不同SVM分类器收敛速率图 19](#_Toc18639)

[图 5 不同SVM分类器收敛速率图 20](#_Toc26103)

[图 6 不同SVM分类器收敛速率图 20](#_Toc12968)

[图 7 不同SVM分类器收敛速率图 20](#_Toc12778)

[图 8 不同SVM分类器收敛速率图 21](#_Toc22855)

[图 9 不同SVM分类器收敛速率图 21](#_Toc5738)

[图 10 不同SVM分类器收敛速率图 21](#_Toc3773)

[图 11 不同SVM分类器收敛速率图 22](#_Toc20758)

[图 12 不同SVM分类器收敛速率图 22](#_Toc12722)

[图 13 不同SVM分类器收敛速率图 22](#_Toc2824)

[图 14 不同SVM分类器收敛速率图 23](#_Toc12865)

[图 15 不同SVM分类器收敛速率图 23](#_Toc9296)

# 附表索引

[表 1 同内核的不同优化方法求解精度 23](#_Toc3642)

# SVM简介

支持向量机（Support Vector Machine，常简称为SVM）是一种监督式学习的方法，可广泛地应用于统计分类以及回归分析。它是将向量映射到一个更高维的空间里，在这个空间里建立有一个最大间隔超平面。在分开数据的超平面的两边建有两个互相平行的超平面，分隔超平面使两个平行超平面的距离最大化。假定平行超平面间的距离或差距越大，分类器的总误差越小。对于线性可分的任务，找到一个具有最大间隔超平面，如图1所示，这个超平面需要满足两个条件：一是能将两类数据点完全分开；二是距离超平面最近的数据点到超平面的距离最大。满足这两个条件的超平面被称为最优超平面，而距离超平面最近的数据点被称为支持向量。

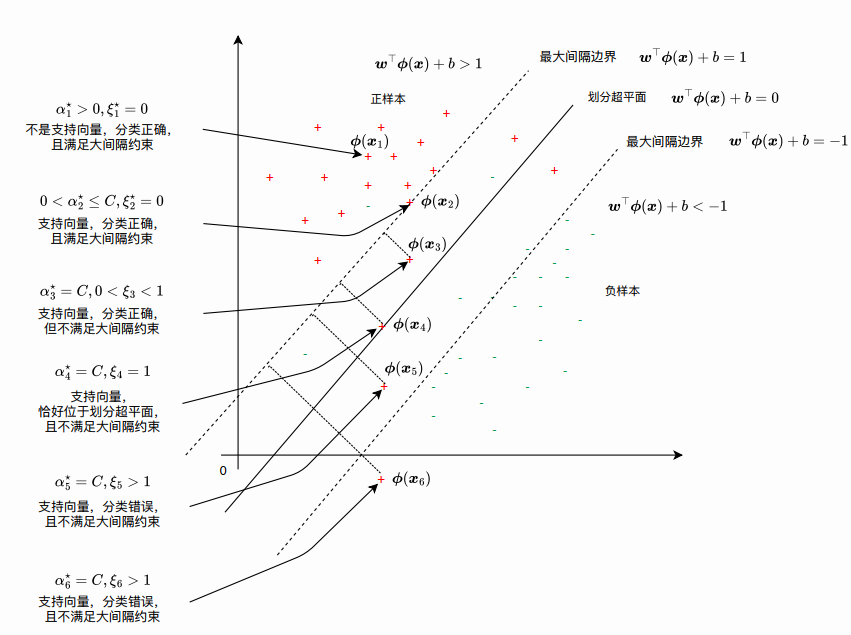


图 1 最大间隔超平面

在SVM中，我们通常使用核函数来将数据映射到低维空间。常见的核函数包括线性核、多项式核、径向基核等。选择合适的核函数对于SVM的性能至关重要。此外，为了处理非线性问题，SVM还引入了软间隔的概念，即在允许一定误差的情况下寻找最优超平面。

# 基于SVM方法的分类问题

## 问题描述

本文旨在基于matlab设计一个支持向量机(SVM)用来解决分类问题。我使用的数据集是matlab自带的Fisher Iris数据，其包含三个类别、四个特征和 150 个样本。取样本中80% 的数据作为训练集，取样本的20%数据作为测试集。设计的支持向量机主要包括Quadratic Penalty、Augmented Lagrangian和InteriorPoint Barrier三种不同的优化方法以及linear、polynomial、RBF和Sigmoid四种不同的核。最后使用在机器学习领域广泛使用的“一对多”的方法处理多类分类问题。

## SVM方法设计

### 决策规则

通过尽可能优化这些区域（为+1的区域和-1的区域）来优化分类器的间隔，定义函数为：





### 计算间隔宽度



式中，是的最近点，是最近点，r是标量倍数，w是垂直于边界的向量。

位于边缘的点需要满足：



则，





我们的目标是尽可能地分离出边界。因此，通过使用最大边际分类器，可以得到最优的w：



将最大化问题更改为最小化问题，定义为：



以上是一个具有线性约束的二次代价函数的例，我们需要最小化二次函数的参数，该约束条件可以重写为：



其中可以取+1或-1。

### 拉格朗日优化

通过引入拉格朗日乘子α（单约束），使难以满足的约束更容易被满足，用拉格朗日乘子来加强不等式约束，拉格朗日方程表示为：



微分拉格朗日方程和关于w的平稳条件表示为：



转换得，



### 非完全线性可分的数据处理

为了处理非完全线性可分的数据，我将稍微放宽方约束条件，以允许错误分类的点。使用软间隔为约束并引入松弛变量。



其中，是一个松弛变量，是惩罚项。

如果R取得非常大，我们需要更加注意以确保没有数据违反间隔。如果R的选择非常小，我们将最大化大多数数据的间隔，但允许一些数据超过一点。我们总是可以将解、和B初始化为某个满足约束的值。

最优作为w的函数，可以解析地表示为：



代入等式（13）得，



其中，是一个铰链损失作为替代损失，如图2所示，当大于1时，铰链损耗等于0。当小于1时，铰链损耗呈线性增加。

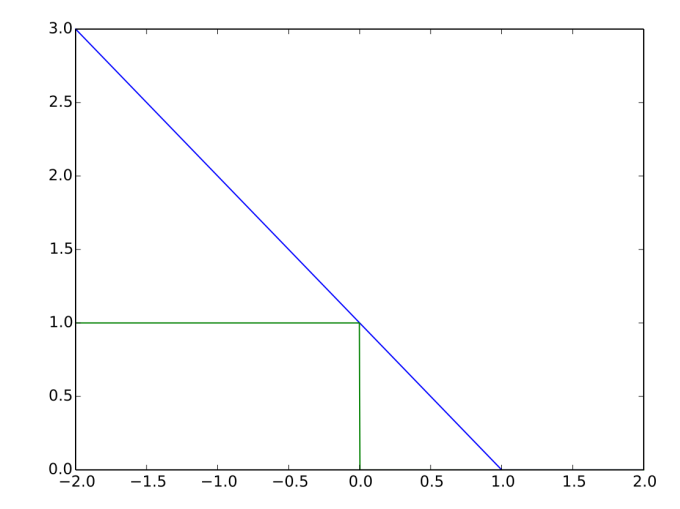


图 2 铰链损失（蓝色）和零1个损失（绿色）

在这个软间隔SVM中，边界错误一侧的数据点有一个惩罚，随着距离它的距离而增加。重新格式化为拉格朗日，如前面所述，我们需要最小化对，和，并最大化对（其中≥0，µ≥0），非完全线性可分离的数据的拉格朗日方程可以显示为：



关于w，b和和平稳点的区别是：



现在，将这些导数插入拉格朗日方程，软间隔对偶形式可以显示为：



最大化实际上取决于点积，因此，我将引入不同类型的内核来处理这个点积。带核的新方程式显示为：



该方程可以进一步重新格式化为二次规划形式：



其中， 。

二次规划（20）的优化可以通过使用Quadratic Penalty或Augmented Lagrangian法来完成。

支持向量S的集合可以通过找到指数来将（）R设置为2来确定。常数b可以计算为：



式中，为支持向量的数量。

### 优化方法

本文通过四种不同的优化方法优化二次函数（20）。它们是Quadratic Penalty Method，Augmented Lagrangian Method，InteriorPoint Barrier and quadratic programming package，作为方程（20）的优化方法。使用quadratic programming package作为参考，以查看编写的优化方法性能如何。这里的f (x)是二次规划函数（20）。对alpha的最初猜测是0.2\*ones（N，1）。等式约束为 ，其中，Aeq是测试集（Y’）的转置，x是训练集。不等式约束为，其中，cost为取2的向量。

* 采用牛顿方向的线搜索方法和回溯线搜索方法来解决所有实现方法的每一步的无约束问题。
* 使用 作为所有实现方法的停止标准。

#### 二次惩罚法

 ，为初始惩罚权重，公差为，更新参数t为2。

由于存在两个不等式约束： 和 ，其优化函数可以显示为：



其中是等式约束和两个，和，表示两个不等式约束和。这意味着当满足不等式约束时，不等式项将消失为：



优化函数的梯度可以表示为：



优化函数的海塞矩阵表示如下：



#### 增广拉格朗日法

µ= 10和ν0 是一个大小为（2N+1）的向量，作为一个固定的惩罚权值和初始拉格朗日乘子，公差为。

优化函数为：



优化函数的梯度为：



优化函数的海塞矩阵为：



#### InteriorPoint Barrier法

，为初始惩罚权重，公差为，更新参数t为2，最大的迭代次数是1000，对alpha的最初猜测是0.01\*1（N，1）。

优化函数为：



其中，



优化函数的梯度为：



其中，



优化函数的海塞矩阵为：



其中，



### 标签预测

新的点通过计算以下方程式进行分类：



### 核函数

分别引入了如下三种内核来优化方程（19）。

#### 多项式核函数



其中，d为多项式条件，等于2。

#### 高斯核函数



其中，当σ的值很大时，所有的数据点看起来都与内核相似。当σ的值较小时，只有几个点与测试点相似。

#### Sigmoid 核函数



其中，c和h均等于数据数的倒数。

# “一对多”的多种类分类

支持向量机（SVM）旨在解决一个二元分类问题。但是，我们使用的数据有三个类，这是一个多类的分类问题。在这里，使用有效地扩展到多类分类的“一对多”方法。这也是SVM多类分类中最早使用的。它构造了k个SVM模型，其中k是类的数量（在我们的例子中是3个）。第m个SVM是针对其他两个例子进行训练的。第m个SVM被考虑与正标签，其他的例子被考虑与负标签。

因此给定l个训练数据集 ，其中 和是x的类。第m个SVM解决了以下问题：



其中，训练数据由函数映射到更高维空间，C为惩罚参数。在这里，我们要想最小化，这意味着最大化 ，两组数据之间的间隔。因为我们的数据不是线性可分离的，所以我设置了一个极小的项 ，用于减少训练错误的数量。SVM是在正则化和训练错误之间寻找平衡。

求解方程(1)后，由于三个类，有三个决策边界。



x在决策函数的值最大的类中，



多类SVM分类器对所有分类器的预测精度可以通过以下方法计算：

* 每个SVM分类器将产生一个分数。比较不同SVM模型生成的所有分数，预测标签可以由最高分数确定。



* 则精度可以计算为：



# Matlab实现

基于前文所述的问题与方法，基于matlab实现过程如下：

**第一步：数据导入与划分**

%Fisher Iris 数据

load fisheriris

outputs1=[];

outputs2=[];

outputs3=[];

y=grp2idx(species);

N=max(size(species));

for i=1:N

if strcmp(species(i),'setosa')

outputs1=[outputs1;1];

elseif strcmp(species(i),'versicolor')

outputs1=[outputs1;-1];

elseif strcmp(species(i),'virginica')

outputs1=[outputs1;-1];

end

end

for i=1:N

if strcmp(species(i),'versicolor')

outputs2=[outputs2;1];

elseif strcmp(species(i),'setosa')

outputs2=[outputs2;-1];

elseif strcmp(species(i),'virginica')

outputs2=[outputs2;-1];

end

end

for i=1:N

if strcmp(species(i),'virginica')

outputs3=[outputs3;1];

elseif strcmp(species(i),'versicolor')

outputs3=[outputs3;-1];

elseif strcmp(species(i),'setosa')

outputs3=[outputs3;-1];

end

end

X=meas;

y1 = outputs1;

y2 = outputs2;

y3 = outputs3;

s=rng(3456);

rand\_num = randperm(size(X,1));

X\_train = X(rand\_num(1:round(0.8\*length(rand\_num))),:);

y\_train = y(rand\_num(1:round(0.8\*length(rand\_num))),:);

y1\_train = y1(rand\_num(1:round(0.8\*length(rand\_num))),:);

y2\_train = y2(rand\_num(1:round(0.8\*length(rand\_num))),:);

y3\_train = y3(rand\_num(1:round(0.8\*length(rand\_num))),:);

X\_test = X(rand\_num(round(0.8\*length(rand\_num))+1:end),:);

y\_test = y(rand\_num(round(0.8\*length(rand\_num))+1:end),:);

y1\_test = y1(rand\_num(round(0.8\*length(rand\_num))+1:end),:);

y2\_test = y2(rand\_num(round(0.8\*length(rand\_num))+1:end),:);

y3\_test = y3(rand\_num(round(0.8\*length(rand\_num))+1:end),:);

**第二步：定义参数**

主要包括是多项式核函数的参数、高斯核函数的参数、Sigmoid 核函数的参数

精度要求，超参数等。

global poly\_con gamma kappa1 kappa2 precision Cost

poly\_con=2; % 多项式核函数参数

gamma=1/size(X,1);% 高斯核函数参数

kappa1=1/size(X,1);kappa2=kappa1; % Sigmoid核函数

precision=10^-5;Cost=2;

**第三步：不同核函数和优化方法的SVM设计与选择**

% 选择核函数

kernel=char(Kernel\_Cell(1));

% SVM设计

[alpha1,Ker1,beta01]=SVM(X\_train,y1\_train,kernel);

[alpha2,Ker2,beta02]=SVM(X\_train,y2\_train,kernel);

[alpha3,Ker3,beta03]=SVM(X\_train,y3\_train,kernel);

scores1 = SVM\_pred(X\_test, X\_train, y1\_train,kernel,alpha1,beta01);

scores2 = SVM\_pred(X\_test, X\_train, y2\_train,kernel,alpha2,beta02);

scores3\_fig = SVM\_pred(X\_test, X\_train, y3\_train,kernel,alpha3,beta03);

**核函数**

线性核

function Y=Ker\_Linear(X1,X2)

Y=zeros(size(X1,1),size(X2,1));

for i=1:size(X1,1)

for j=1:size(X2,1)

Y(i,j)=dot(X1(i,:),X2(j,:));

end

end

Return

多项式核

function Y=Ker\_Polynomial(X1,X2)

global poly\_con

Y=zeros(size(X1,1),size(X2,1));

for i=1:size(X1,1)

for j=1:size(X2,1)

Y(i,j)=(1+dot(X1(i,:),X2(j,:))).^poly\_con;

end

end

return

高斯核

function Y=Ker\_RBF(X1,X2)

global gamma

Y=zeros(size(X1,1),size(X2,1));

for i=1:size(X1,1)

for j=1:size(X2,1)

Y(i,j)=exp(-gamma\*norm(X1(i,:)-X2(j,:))^2);

end

end

return

Sigmoid核

function Y=Ker\_Sigmoid(X1,X2)

global kappa1 kappa2

Y=zeros(size(X1,1),size(X2,1));

for i=1:size(X1,1)

for j=1:size(X2,1)

Y(i,j)=(kappa1\*dot(X1(i,:),X2(j,:))+kappa2);

end

end

return

**优化方法**

二次惩罚法

function [xMin, fMin, t, nIter, infoQP] = Quadratic\_Penalty(x0, mu, t, tol, maxIter,N,Aeq,lb,ub,H,f)

% 初始化

nIter = 0;stopCond = false;x\_k = x0;infoQP.xs = x\_k;infoQP.fs = [];

alpha0 = 1; opts.c1 = 1e-4;opts.c2 = 0.9;opts.rho = 0.5;tolNewton = 1e-12;maxIterNewton = 100;

% 循环

while (~stopCond && nIter < maxIter)

% disp(size(Aeq))

% disp(size(x\_k))

% 为 Q 创建函数处理程序

G.f =@(x) 0.5\*x'\*H\*x + f'\*x + (mu/2).\*sum((max(lb-x,0)).^2)+ (mu/2).\*sum((max(x-ub,0)).^2)+ (mu/2)\*sum((Aeq\*x).^2);

G.df =@(x) H\*x+f + mu\*((Aeq\*x)\*Aeq') + mu\*penalty\_grad(x,N,lb,ub);

G.d2f =@(x) H + mu\*(Aeq'\*Aeq);

lsFun = @(x\_k, p\_k, alpha0) backtracking(G, x\_k, p\_k, alpha0, opts);

x\_k\_1 = x\_k;

[x\_k, f\_k, nIterLS, infoIter] = descentLineSearch(G, 'newton', lsFun, alpha0, x\_k, tolNewton, maxIterNewton);

% 判断循环终止条件

if norm(x\_k - x\_k\_1) < tol; stopCond = true; end

mu = mu\*t;

infoQP.xs = [infoQP.xs x\_k];

infoQP.fs = [infoQP.fs f\_k];

nIter = nIter + 1;

end

% 赋值

xMin = x\_k;

fMin = G.f(x\_k);

增广拉格朗日

function [xMin, fMin, nIter, infoAL] = Augmented\_Lagrangian(x0, mu, v0, tol, maxIter,N,Aeq,lb,ub,H,f)

% 初始化

nIter = 0;stopCond = false;x\_k = x0;infoAL.xs = x\_k;infoAL.fs = [];

v\_k=v0;

alpha0 = 1;

opts.c1 = 1e-4;

opts.c2 = 0.9;

opts.rho = 0.5;

tolNewton = 1e-12;

maxIterNewton = 100;

% 循环

while (~stopCond && nIter < maxIter)

disp(strcat('Iteration ', int2str(nIter)));

% 为 Q 创建函数处理程序

% G.f = @(x) 0.5\*x'\*H\*x + f'\*x + v\_k.\*sum(Aeq\*x) + (mu/2)\*sum((Aeq\*x).^2)...

% +(mu/2).\*sum((max(lb-x,0)).^2) + v\_k.\*sum(max(lb-x,0))...

% + (mu/2).\*sum((max(x-ub,0)).^2)+ v\_k.\*sum(max(x-ub,0));

G.f = @(x) 0.5\*x'\*H\*x + f'\*x + sum(v\_k(1).\*(Aeq\*x)) + (mu/2)\*sum((Aeq\*x).^2)...

+(mu/2).\*sum((max(lb-x,0)).^2) + sum(v\_k(2:(N+1)).\*max(lb-x,0))...

+ (mu/2).\*sum((max(x-ub,0)).^2)+ sum(v\_k((N+2):(2\*N+1)).\*max(x-ub,0));

G.df = @(x) H\*x+f + (v\_k(1) + mu\*Aeq\*x)\*Aeq' + augmented\_grad(v\_k,mu,x,N,lb,ub);

G.d2f = @(x) H + mu\*(Aeq'\*Aeq);

lsFun = @(x\_k, p\_k, alpha0) backtracking(G, x\_k, p\_k, alpha0, opts);

x\_k\_1 = x\_k;

[x\_k, f\_k, nIterLS, infoIter] = descentLineSearch(G, 'newton', lsFun, alpha0, x\_k, tolNewton, maxIterNewton);

v\_k(1) = v\_k(1) + mu\*Aeq\*x\_k;

v\_k(2:(N+1)) = v\_k(2:(N+1)) + mu\*max(lb-x\_k,0);

v\_k((N+2):(2\*N+1)) = v\_k((N+2):(2\*N+1)) + mu\*max(x\_k-ub,0);

% 判断循环终止条件

if norm(x\_k - x\_k\_1) < tol; stopCond = true; end

infoAL.xs = [infoAL.xs x\_k];

infoAL.fs = [infoAL.fs f\_k];

nIter = nIter + 1;

end

% 赋值

xMin = x\_k;

fMin = G.f(x\_k);

InteriorPoint Barrier法

function [xMin, fMin, t, nIter, infoBarrier] = interiorPoint\_Barrier(F, phi, x0, t, mu, tol, maxIter)

% 初始化

nIter = 0;

stopCond = false;

x\_k = x0;

infoBarrier.xs = x\_k;

infoBarrier.fs = F.f(x\_k);

infoBarrier.inIter = 0;

infoBarrier.dGap = 1/t;

alpha0 = 1;

opts.c1 = 1e-4;

opts.c2 = 0.9;

opts.rho = 0.5;

tolNewton = 1e-12;

maxIterNewton = 100;

% 循环

while (~stopCond && nIter < maxIter)

disp(strcat('Iteration ', int2str(nIter)));

% 为 Q 创建函数处理程序

G.f = @(x) t\*F.f(x) + phi.f(x);

G.df = @(x) t\*F.df(x) + phi.df(x);

G.d2f = @(x) t\*F.d2f(x) + phi.d2f(x);

% 行搜索函数

%lsFun = @(x\_k, p\_k, alpha0) lineSearch(G, x\_k, p\_k, alpha0, opts);

lsFun = @(x\_k, p\_k, alpha0) backtracking(G, x\_k, p\_k, alpha0, opts);

[x\_k, f\_k, nIterLS, infoIter] = descentLineSearch(G, 'newton', lsFun, alpha0, x\_k, tolNewton, maxIterNewton);

if 1/t < tol; stopCond = true; end

t = mu\*t;

infoBarrier.xs = [infoBarrier.xs x\_k];

infoBarrier.fs = [infoBarrier.fs f\_k];

infoBarrier.inIter = [infoBarrier.inIter nIterLS];

infoBarrier.dGap = [infoBarrier.dGap 1/t];

nIter = nIter + 1;

end

% 赋值

xMin = x\_k;

fMin = F.f(x\_k);

**第四步：可视化**

function SVM\_plot(X,Y,alpha,beta0,kernel)

global Cost poly\_con gamma kappa1

figure

hold on

P = size(X,2);

if P ~=2

warning('# of input X should be 2 for the 2D visualization!!')

end

plot(X(Y==1,1),X(Y==1,2),'ro',...

'LineWidth', 4,...

'MarkerSize', 4);

plot(X(Y==-1,1),X(Y==-1,2),'bs',...

'LineWidth', 4,...

'MarkerSize', 4);

%

d = 0.02;

[x1Grid,x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),...

min(X(:,2)):d:max(X(:,2)));

xGrid = [x1Grid(:),x2Grid(:)];

scores = SVM\_pred(xGrid, X, Y,kernel,alpha,beta0);

contour(x1Grid,x2Grid,reshape(scores,size(x1Grid)),[0 0],'k',...

'LineWidth', 4);

xlabel('$X\_1$','FontSize', 18,...

'Interpreter','latex');

ylabel('$X\_2$', 'FontSize', 18,...

'Interpreter','latex');

switch kernel

case 'linear'

title({'SVM',strcat('Kernel:',kernel,';C=',num2str(Cost))}, 'FontSize', 18,...

'Interpreter','latex');

case 'ploynomial'

title({'SVM',strcat('Kernel:',kernel,';C=',num2str(Cost),';n=',num2str(poly\_con))}, 'FontSize', 18,...

'Interpreter','latex');

case 'RBF'

title({'SVM',strcat('Kernel:',kernel,';C=',num2str(Cost),';$\gamma$=',num2str(gamma))}, 'FontSize', 18,...

'Interpreter','latex');

case 'Sigmoid'

title({'SVM',strcat('Kernel:',kernel,';C=',num2str(Cost),';$\kappa$=',num2str(kappa1))}, 'FontSize', 18,...

'Interpreter','latex');

end

legend({'+1:setosa';'-1:versicolor'},'FontSize',16,'Location', 'Best');

hold off

return

# 结果与讨论

## 结果

使用数据的80%为训练集，20%为测试集，三个类别的分类结果如下图所示，左图为训练集数据，右图为训练数据加新数据。

|  |  |
| --- | --- |
|  |  |
| 图 3分类结果 | |

## 收敛速率

为量化的收敛速率，计算 对迭代k的比值进行比较。

### 二次惩罚法

#### 线性核

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **SVM分类器1** | **SVM分类器2** | **SVM分类器3** |
| 图 4 不同SVM分类器收敛速率图 | | |

#### 多项式核

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **SVM分类器1** | **SVM分类器2** | **SVM分类器3** |
| 图 5 不同SVM分类器收敛速率图 | | |

#### 高斯核

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **SVM分类器1** | **SVM分类器2** | **SVM分类器3** |
| 图 6 不同SVM分类器收敛速率图 | | |

#### Sigmoid核

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **SVM分类器1** | **SVM分类器2** | **SVM分类器3** |
| 图 7 不同SVM分类器收敛速率图 | | |

### 增广拉格朗日法

#### 线性核

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **SVM分类器1** | **SVM分类器2** | **SVM分类器3** |
| 图 8 不同SVM分类器收敛速率图 | | |

#### 多项式核

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **SVM分类器1** | **SVM分类器2** | **SVM分类器3** |
| 图 9 不同SVM分类器收敛速率图 | | |

#### 高斯核

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **SVM分类器1** | **SVM分类器2** | **SVM分类器3** |
| 图 10 不同SVM分类器收敛速率图 | | |

#### Sigmoid核

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **SVM分类器1** | **SVM分类器2** | **SVM分类器3** |
| 图 11 不同SVM分类器收敛速率图 | | |

### InteriorPoint Barrier法

#### 线性核

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **SVM分类器1** | **SVM分类器2** | **SVM分类器3** |
| 图 12 不同SVM分类器收敛速率图 | | |

#### 多项式核

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **SVM分类器1** | **SVM分类器2** | **SVM分类器3** |
| 图 13 不同SVM分类器收敛速率图 | | |

#### 高斯核

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **SVM分类器1** | **SVM分类器2** | **SVM分类器3** |
| 图 14 不同SVM分类器收敛速率图 | | |

#### Sigmoid核

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **SVM分类器1** | **SVM分类器2** | **SVM分类器3** |
| 图 15 不同SVM分类器收敛速率图 | | |

## 精度比较

不同内核的不同优化方法获得的精度如下所示：quadratic programming是matlab自带的，将其结果作为参考。

表 1 同内核的不同优化方法求解精度

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kernel | Quadratic Penalty | Augmented Lagrangian | InteriorPoint Barrier | Quadratic programming |
| Linear | 73.333 | 73.333 | 73.333 | 93.3333 |
| Polynomial | 73.333 | 73.333 | 40 | 26.6667 |
| RBF | 86.6667 | 86.6667 | 76.6667 | 96.6667 |
| Sigmoid | 40 | 76.6667 | 26.6667 | 33.3333 |

根据上表的结果可知，最佳内核是RBF，因为其精度一直很高。这也解释了为什么RBF内核是SVM中广泛使用且最常用的内核。其中，增广拉格朗日法在所有内核中都具有最好的性能。

# 结 论

基于matlab设计一个支持向量机(SVM)用来解决分类问题。设计的支持向量机主要包括二次惩罚、增广拉格朗日和InteriorPoint Barrier三种不同的优化方法以及线性、多项式、高斯和Sigmoid四种不同的核，结果表明：

1. 所设计的SVM分类器能够很好的对数据进行分类。
2. 使用的“一对多”的方法能够很好的处理多类分类问题。
3. 通过分析了不同优化方法、不同核化函数下、不同SVM分类器的收敛速率并对比不同优化方法、不同核化函数下的求解精度可知，在该分类问题中，最佳内核是高斯核，因为其精度一直很高；其中，增广拉格朗日法在所有内核中都具有最好的性能。

# 参考文献

1. Andrew Ng, CS229 Lecture notes , Support vector machines, Part V.
2. Tristan Fletcher, Support Vector Machines Explained
3. Numerical Optimisation Constraint optimisation: Penalty and augmented Lagrangian methods, Lecture 14 (based on Nocedal, Wright)
4. Numerical Optimisation Constraint optimisation: Interior point methods, Lecture 14 (based on Boyd, Vander bergher)