1. Code:

test=load('test79.mat');

test=test.d79;

train=load('train79.mat');

train=train.d79;

label = vertcat(ones(1000,1)\*1, ones(1000,1)\*-1);

N=2000;

% SVM

lambda = logspace(-8,1,20); % lambda = 1/C

svmModel = fitclinear(train,label,'Regularization','ridge','lambda',lambda);

svmResult = predict(svmModel,train);

svmLossList=zeros(length(lambda),1);

% prediction accuracy rate

for i=1:length(lambda)

diff=svmResult(:,i)-label;

svmLossList(i)=(transpose(diff)\*diff)/(4\*N);

end

% LSLC

w=lsqlin(train,label);

testResult=sign(test\*w);

svmLoss = 1/2\*(sum(abs(testResult-label)))/2000

For Least Square Linear Classifier (LSLC): error rate = 0.0655.

Figure 1 SVM's error rate trend with C's value

The SVM’s performance is universally better than LSLC.

1. Code:

For Least Squares classifier using gradient descent:

test=load('test79.mat');

test=test.d79;

train=load('train79.mat');

train=train.d79;

label = vertcat(ones(1000,1)\*1, ones(1000,1)\*-1);

N=2000;

d=784;

w = ones(d+1, 1)\*0; % Bias trick

train=[ones(N,1),train];

test=[ones(N,1),test];

nIterations = 2000;

it = 0;

learningRate = 1\*10e-11;

objFun = @(X, Y, w) ( transpose(w)\*transpose(X)\*X\*w-2\*w\*transpose(w)\*transpose(X)\*Y+transpose(Y)\*Y)/N;

gradient = @(X, Y, w) 2\*transpose(X)\*X\*w-2\*transpose(X)\*Y;

wBackup = ones(d+1, 1)\*0

while it <= nIterations

% computes objective

R = objFun(train, label, w);

grad = gradient(train, label, w);

wBackup=w;

w = w - learningRate\*grad;

it = it + 1;

R % print

if isnan(w(1))

break;

end

if sum(abs(w-wBackup))<10e-5 % Converge standard

break;

end

if it > nIterations

break;

end

end

result = sign(test\*w);

loss = 1/2\*(sum(abs(result-label)))/N

The final error rate is :

0.0510

The object function is highly unstable, thus larger learning rate will not converge.

For standard least squares classifier using pseudo-inverse:

test=load('test79.mat');

test=test.d79;

train=load('train79.mat');

train=train.d79;

label = vertcat(ones(1000,1)\*1, ones(1000,1)\*-1);

N=2000;

d=784;

w = pinv(train'\*train)\*train'\*label;

result = sign(test\*w);

loss = 1/2\*(sum(abs(result-label)))/N

The final error rate is:

0.0650

1. Code:

testRaw=load('test79.mat');

testRaw=testRaw.d79;

trainRaw=load('train79.mat');

trainRaw=trainRaw.d79;

labelRaw = vertcat(ones(1000,1)\*1, ones(1000,1)\*-1);

trainPCA\_coeff =pca(trainRaw,'NumComponents', 400);

testPCA\_coeff =pca(testRaw,'NumComponents', 400);

trainPCA = trainRaw\*trainPCA\_coeff;

testPCA = testRaw\*testPCA\_coeff;

NList = [25:25:1000];

lslcLossList=ones(length(NList),1);

svmLossList=ones(length(NList),1);

for i = 1:length(NList)

N= NList(i);

train = [trainPCA(1:N,:);trainPCA(1001:1000+N,:)];

test = [testPCA(1:N,:);testPCA(1001:1000+N,:)];

label = [ones(N,1)\*1; ones(N,1)\*(-1)];

% SVM

svmModel = fitclinear(train,label);

svmResult = predict(svmModel,test);

diff=abs(svmResult-label)/2;

svmLossList(i)=sum(diff)/(N\*2);

% LSLC

lslcW = lsqlin(train, label);

lslcResult = sign (test\*lslcW);

diff =abs(lslcResult-label)/2;

lslcLossList(i)=sum(diff)/(N\*2);

end

Figure 2 SVM and LSLC’s error rate’s changing trend with dataset size

1. Code:

N=50; % Change on demand

d=784;

testRaw=load('test79.mat');

testRaw=testRaw.d79;

trainRaw=load('train79.mat');

trainRaw=trainRaw.d79;

label = vertcat(ones(N,1)\*1, ones(N,1)\*-1);

train = [trainRaw(1:N,:);trainRaw(1001:1000+N,:)];

test = [testRaw(1:N,:);testRaw(1001:1000+N,:)];

w = ones(d+1, 1)\*0; % Bias trick

train=[ones(N\*2,1),train];

test=[ones(N\*2,1),test];

nIterations = 2000;

it = 0;

learningRate = 1\*10e-11;

objFun = @(X, Y, w) ( transpose(w)\*transpose(X)\*X\*w-2\*w\*transpose(w)\*transpose(X)\*Y+transpose(Y)\*Y)/N;

gradient = @(X, Y, w) 2\*transpose(X)\*X\*w-2\*transpose(X)\*Y;

loss\_rates=[];

wBackup = ones(d+1, 1)\*0

while it <= nIterations

% computes objective

R = objFun(train, label, w);

grad = gradient(train, label, w);

wBackup=w;

w = w - learningRate\*grad;

it = it + 1;

if isnan(w(1))

break;

end

if sum(abs(w-wBackup))<10e-6

break;

end

if it > nIterations

break;

end

result = sign(test\*w);

loss = 1/2\*(sum(abs(result-label)))/N

loss\_rates=[loss\_rates;loss];

end

Figure 3 Linear regression's error rate's trend with iterations (maxIterations=2000)