Aditi Mallavarapu

CS 412

Question 2 a)

% printing option

more off;

% read files

D\_tr = csvread('spambasetrain.csv');

D\_ts = csvread('spambasetest.csv');

% construct x and y for training and testing

X\_tr = D\_tr(:, 1:end-1);

y\_tr = D\_tr(:, end);

X\_ts = D\_ts(:, 1:end-1);

y\_ts = D\_ts(:, end);

% number of training / testing samples

n\_tr = size(D\_tr, 1);

n\_ts = size(D\_ts, 1);

% add 1 as a feature

X\_tr = [ones(n\_tr, 1) X\_tr];

X\_ts = [ones(n\_ts, 1) X\_ts];

% perform gradient descent :: logistic regression

n\_vars = size(X\_tr, 2); % number of variables

lr = 1e-3; % learning rate

w = zeros(n\_vars, 1); % initialize parameter w

tolerance = 1e-2; % tolerance for stopping criteria

iter = 0; % iteration counter

max\_iter = 1000; % maximum iteration

while true

iter = iter + 1; % start iteration

%first\_term = sum(y\_tr,x\_tr);

% calculate gradient

grad = zeros(n\_vars, 1); % initialize gradient

for j=1:n\_vars

temp = (X\_tr'(j,:))';

temp\_exp = exp(w(j) \* temp);

temp\_grad = (y\_tr .\* temp) - ((temp .\* temp\_exp)./(1 .+ temp\_exp));

grad(j) = sum(temp\_grad');

%j++;

%grad(j) = % compute the gradient with respect to w\_j here

end

% take step

% w\_new = w + ..... % take a step using the learning rate

w\_new = w .+ (lr \* grad);

printf('iter = %d, mean abs gradient = %0.3f\n', iter, mean(abs(grad)));

%fflush(stdout);

% stopping criteria and perform update if not stopping

if mean(abs(grad)) < tolerance

%w = w\_new;

break;

else

w = w\_new;

end

if iter >= max\_iter

break;

end

end

% use w for prediction

pred = zeros(n\_ts, 1); % initialize prediction vector

for i=1:n\_ts

temp\_sum = sum(w' .\* X\_ts(i,:));

prob\_0 = 1 / (1+ exp(temp\_sum));

prob\_1 = exp(temp\_sum)/(1+ exp(temp\_sum));

% pred(i) = ..... % compute your prediction

if(prob\_1 >= prob\_0)

pred(i) = 1; %ham

else

pred(i) = 0; %spam

endif

end

% calculate testing accuracy

count =0;

for i= 1:n\_ts

if(pred(i)== y\_ts(i))

count++;

end

end

accuracy = count / n\_ts;

printf('Testing accuracy=%d',accuracy);

% repeat the similar prediction procedure to get training accuracy

pred\_tr = zeros(n\_tr, 1); % initialize prediction vector

for i=1:n\_tr

temp\_sum = sum(w' .\* X\_tr(i,:));

prob\_0 = 1 / (1+ exp(temp\_sum));

prob\_1 = exp(temp\_sum)/(1+ exp(temp\_sum));

% pred(i) = ..... % compute your prediction

if(prob\_1 >= prob\_0)

pred\_tr(i) = 1; %ham

else

pred\_tr(i) = 0; %spam

endif

end

% calculate testing accuracy

count =0;

for i= 1:n\_tr

if(pred\_tr(i)== y\_tr(i))

count++;

end

end

accuracy = count / n\_tr;

printf('\nTraining accuracy=%d',accuracy);

Output:

Testing accuracy=0.919623

Training accuracy=0.912112

Aditi Mallavarapu

CS 412

Question 2 b)

% printing option

more off;

% read files

D\_tr = csvread('spambasetrain.csv');

D\_ts = csvread('spambasetest.csv');

% construct x and y for training and testing

X\_tr = D\_tr(:, 1:end-1);

y\_tr = D\_tr(:, end);

X\_ts = D\_ts(:, 1:end-1);

y\_ts = D\_ts(:, end);

% number of training / testing samples

n\_tr = size(D\_tr, 1);

n\_ts = size(D\_ts, 1);

% add 1 as a feature

X\_tr = [ones(n\_tr, 1) X\_tr];

X\_ts = [ones(n\_ts, 1) X\_ts];

% perform gradient descent :: logistic regression

n\_vars = size(X\_tr, 2); % number of variables

lr = 1e-3; % learning rate

w = zeros(n\_vars, 1); % initialize parameter w

tolerance = 1e-2; % tolerance for stopping criteria

lambda = [0.00390625,0.015625,0.0625,0.25,1,4];

accuracy\_tr = size(length(lambda),1);

accuracy\_ts = size(length(lambda),1);

for k = 1:length(lambda)

iter = 0; % iteration counter

max\_iter = 1000; % maximum iteration

while true

iter = iter + 1; % start iteration

% calculate gradient

grad = zeros(n\_vars, 1); % initialize gradient

for j=1:n\_vars

temp = (X\_tr'(j,:))';

temp\_exp = exp(w(j) \* temp);

temp\_lambda = (lambda(k) \* w(j));

temp\_grad = (y\_tr .\* temp) - ((temp .\* temp\_exp)./(1 .+ temp\_exp));

grad(j) = sum(temp\_grad')- temp\_lambda;

%grad(j) = % compute the gradient with respect to w\_j here

end

% take step

% w\_new = w + ..... % take a step using the learning rate

w\_new = w .+ (lr \* grad);

% printf('lambda=%d ,iter = %d, mean abs gradient = %0.3f\n',lambda(k), iter, mean(abs(grad)));

%fflush(stdout);

% stopping criteria and perform update if not stopping

if mean(abs(grad)) < tolerance

%w = w\_new;

break;

else

w = w\_new;

end

if iter >= max\_iter

break;

end

end

% use w for prediction

pred = zeros(n\_ts, 1); % initialize prediction vector

for i=1:n\_ts

temp\_sum = sum(w' .\* X\_ts(i,:));

prob\_0 = 1 / (1+ exp(temp\_sum));

prob\_1 = exp(temp\_sum)/(1+ exp(temp\_sum));

% pred(i) = ..... % compute your prediction

if(prob\_1 >= prob\_0)

pred(i) = 1; %ham

else

pred(i) = 0; %spam

endif

end

% calculate testing accuracy

count =0;

for i= 1:n\_ts

if(pred(i)== y\_ts(i))

count++;

end

end

accuracy\_ts(k) = count / n\_ts;

printf('\nlambda=%d, Testing accuracy=%d',lambda(k),accuracy\_ts(k));

% repeat the similar prediction procedure to get training accuracy

pred\_tr = zeros(n\_tr, 1); % initialize prediction vector

for i=1:n\_tr

temp\_sum = sum(w' .\* X\_tr(i,:));

prob\_0 = 1 / (1+ exp(temp\_sum));

prob\_1 = exp(temp\_sum)/(1+ exp(temp\_sum));

% pred(i) = ..... % compute your prediction

if(prob\_1 >= prob\_0)

pred\_tr(i) = 1; %ham

else

pred\_tr(i) = 0; %spam

endif

end

% calculate testing accuracy

count =0;

for i= 1:n\_tr

if(pred\_tr(i)== y\_tr(i))

count++;

end

end

accuracy\_tr(k) = count / n\_tr;

printf('\nlambda=%d, Training accuracy=%d',lambda(k),accuracy\_tr(k));

end

k=[-8,-6,-4,-2,0,2];

plot(k,accuracy\_tr,"-;trainaccuracy;",k,accuracy\_ts,"-r;testaccuracy;");

Output:

lambda=0.00390625, Testing accuracy=0.919623

lambda=0.00390625, Training accuracy=0.912112

lambda=0.015625, Testing accuracy=0.919623

lambda=0.015625, Training accuracy=0.912112

lambda=0.0625, Testing accuracy=0.918899

lambda=0.0625, Training accuracy=0.912112

lambda=0.25, Testing accuracy=0.918899

lambda=0.25, Training accuracy=0.912422

lambda=1, Testing accuracy=0.918175

lambda=1, Training accuracy=0.913354

lambda=4, Testing accuracy=0.916727

lambda=4, Training accuracy=0.913043

s