% Gradient Descent

clear; clc;

w = [1; 1];

f = @(w) ((w(1)\*exp(w(2))-2\*w(2)\*exp(-w(1)))^2);

eta = 0.1;

for iter = 1:20

w = w - eta\*hw5\_gd(w);

err = f(w);

if err < 1e-14

iter

w

break

end

end

w = [1; 1];

for iter = 1:15

temp = hw5\_gd(w);

w(1) = w(1) - eta\*temp(1);

temp = hw5\_gd(w);

w(2) = w(2) - eta\*temp(2);

end

f(w)

% error function

function err = hw5\_gd(w)

u = double(w(1));

v = double(w(2));

du = 2\*(exp(v)+2\*v\*exp(-u))\*(u\*exp(v)-2\*v\*exp(-u));

dv = 2\*(u\*exp(v)-2\*v\*exp(-u))\*(u\*exp(v)-2\*exp(-u));

err = double([du; dv]);

end

clear; clc;

N = 100;

d = 2;

th = 0.01;

eta = 0.01;

sdg = @(xn,yn,w) (-yn\*xn/(1+exp(yn\*xn\*w)))';

err = @(xn,yn,w) double(log(1+exp(-yn\*xn\*w)));

total\_iter = 0;

M = 10000;

total\_Eout = 0;

for run = 1 : 100

x = unifrnd(-1, 1, N, d);

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

ps = unifrnd(-1, 1, 2, d);

ps = [ones(2,1) ps];

bd = null(ps);

y = sign(x\*bd);

w = double([0; 0; 0]);

iter = 0;

error = 1;

while error >= th

pre = double(w);

index = randi(1,N,N,1);

for i = 1:length(index)

xi = x(index(i),:);

yi = y(index(i));

w = w-eta\*sdg(xi,yi,w);

end

error = norm(pre-w);

iter = iter+1;

end

total\_iter = total\_iter+iter;

xout = unifrnd(-1, 1, M, d);

xout = [ones(M, 1) xout];

yout = sign(xout\*bd);

Eout = 0;

for j = 1:M

xoutj = xout(j,:);

youtj = yout(j);

temp = err(xoutj,youtj,w);

Eout = Eout + temp;

end

total\_Eout = total\_Eout + Eout;

end

aver\_iter = total\_iter/100;

Eout = total\_Eout/M/100;