Deepayan Bhadra - hmwk3 Solutions

Q1(a): Function to test if 'grad' generates the gradient of x

```
s = rng;
function iscorrect = check_gradient(f,grad,x0)
b = phantom(64);
mu=0.5;
del = randn(size(x0,1),size(x0,2));
s=10e4;
for i=1:10;
fprintf('The ratio of perturbation to norm')
norm(del,'fro')/norm(x0,'fro')
fprintf('The relative error is')
% Absolute error divided by the exact value
err = (abs(feval(f,x0+del)-feval(f,x0)-sum(reshape(del,[],1).*reshape(feval(grad,x0),[],1))
del=del/10;
s = min(s,err);
end
```

Solution to Q1(b) to test gradient checker

```
rng(s)
x = sym('x',[3 5]);
A = randn(4,3);
b = randn(4,5);
f = @(x) 0.5*(norm(A*x-b,'fro'))^2;
x0 = randn(3,5);
grad = @(x) A'*(A*x-b);

output = check_gradient(f,grad,x0)
```

Output

The ratio of perturbation to norm ans =
1.3617
The relative error is err =
2.9125

The ratio of perturbation to norm ans =
0.1362
The relative error is err =
0.1796
The ratio of perturbation to norm ans =
0.0136
The relative error is err =
0.0155
The ratio of perturbation to norm ans =
0.0014
The relative error is err =
0.0015
The ratio of perturbation to norm ans =
1.3617e-04
The relative error is err =
1.5231e-04
The ratio of perturbation to norm ans =
1.3617e-05
The relative error is err =
1.5229e-05
The ratio of perturbation to norm ans =
1.3617e-06
The relative error is err =
1.5230e-06
The ratio of perturbation to norm ans =
1.3617e-07
The relative error is err =
1.5581e-07
The ratio of perturbation to norm ans =
1.3617e-08
The relative error is err =
2.6270e-08
The ratio of perturbation to norm ans =
1.3617e-09
The relative error is err =
9.9870e-08
The minimum relative error is s =
2.6270e-08

The gradient checker output is (T=1,F=0)

output = 1

Q2(a): Function to compute gradient of logistic regression function

```
function y = logreg_grad(x,D,c)
z = diag(c)*D*x;
idxN = z<0; idxP = z>=0; % logical indexing
y(idxN) = arrayfun(@(x) -1+(exp(x)/(1+exp(x))),z(idxN));
y(idxP) = arrayfun(@(x) -exp(-x)/(1+exp(-x)),z(idxP));
y = D'*diag(c)*y'; % grad f(CDx) = D'*C'*f'(CDx)
end
```

Q2(b):to test gradient with classification problem

```
[ D,c ] = create_classification_problem(1000, 10, 5);
x = randn(10,1);
f = @logreg_objective;
grad = @logreg_grad;
x = {x D c};
output = check_gradient(f,grad,x)
```

Output The ratio of perturbation to norm ans =
1.1581
The relative error is err =
0.6324
The ratio of perturbation to norm ans =
0.1158
The relative error is err =
0.0421
The ratio of perturbation to norm ans =
0.0116
The relative error is err =
0.0041
The ratio of perturbation to norm ans =
0.0012
The relative error is err =
4.0490e-04
The ratio of perturbation to norm ans =
1.1581e-04
The relative error is err =
4.0476e-05
The ratio of perturbation to norm ans =

1.1581e-05
The relative error is err =
4.0482e-06
The ratio of perturbation to norm ans =
1.1581e-06
The relative error is err =
4.0357e-07
The ratio of perturbation to norm ans =
1.1581e-07
The relative error is err =
6.8424e-08
The ratio of perturbation to norm ans =
1.1581e-08
The relative error is err =
4.1364e-07
The ratio of perturbation to norm ans =
1.1581e-09
The relative error is err =
8.3095e-06
The minimum relative error is s =
6.8424e-08

The gradient checker output is (T=1,F=0)

output = 1

Q3(a): Function to return the smoothed l1 norm

```
function f = l1_eps(x,eps)
x = reshape(x,[],1);
f = sum(sqrt(x.^2+eps^2));
end
```

Q3(b): Function to return the gradient of the smoothed l1 norm

```
function f = l1_grad(x,eps)
f =  x(:)./sqrt(x(:).^2+eps^2);
f = reshape(f,size(x));
end
```

Q3(c): Function to return the objective value of the given f(x)

```
function f = tv_objective(x,b,mu,eps)
f = mu*l1_eps(grad2d(x),eps)+0.5*(norm(x-b,'fro'))^2;
end
```

Q3(d): Function to return the gradient of objective value of the above f(x)

```
function f = tv_grad(x,b,mu,eps)
f = mu*div2d(l1_grad(grad2d(x),eps)) +(x-b);
end
```

Q3(e):to test gradient using noisy test image

```
b = phantom(64);
mu=0.5;
x = sym('x',[64 64]);
f1 = @tv_objective;
fprintf('The objective f(x) is');
f2 = @tv_grad
fprintf('The gradient of f(x) is');
x0 = randn(size(b,1),size(b,2));
x0 = {x0 b mu eps};
output = check_gradient(f1,f2,x0)
```

Output

The ratio of perturbation to norm ans =
0.9947
The relative error is err =
1.0213
The ratio of perturbation to norm ans =
0.0995
The relative error is err =
1.2607
The ratio of perturbation to norm ans =
0.0099

The relative error is err =
0.8976
The ratio of perturbation to norm ans =
9.9472e-04
The relative error is err =
0.0616
The ratio of perturbation to norm ans =
9.9472e-05
The relative error is err =
0.0074
The ratio of perturbation to norm ans =
9.9472e-06
The relative error is err =
2.4814e-04
The ratio of perturbation to norm ans =
9.9472e-07
The relative error is err =
2.5131e-05
The ratio of perturbation to norm ans =
9.9472e-08
The relative error is err =
3.7474e-06
The ratio of perturbation to norm ans =
9.9472e-09
The relative error is err =
4.6101e-05
The ratio of perturbation to norm ans =
9.9472e-10
The relative error is err =
1.4461e-04
The minimum relative error is s =
3.7474e-06

The gradient checker output is (T=1,F=0)

output = 0

Q4(a):Function to compute gradient of neural net objective

```
First we define some auxiliary functions

function y = softplus_grad(X)
y = zeros(size(X,1),size(X,2));
idxN = X<0; idxP = X>=0;
```

```
y(idxN) = exp(X(idxN))/(1+exp(X(idxN))); y(idxP) = 1/(1+exp(-X(idxP)));
end

function y = cross_entropy_grad(X,Y)
y = zeros(size(X,2),1);
m = size(X,1); n = size(X,2);
Xm = X - repmat(max(X,[],2), 1, size(X,2));
% subtracts the maximum of each row from that row.
softmax = exp(Xm)./repmat(sum(exp(Xm),2), 1, size(X,2));
% Element-wise division by exponential sum of each row

deriv = softmax.*Y
end

function dW = net_grad(W,D,L)
z{1} = D*W{1};
y{1} = softplus(z{1});
for i = 2:size(W,2)
z{i} = y{i-1}*W{i};
y{i} = softplus(z{i});
end
for j = 1:size(W,2)
```

Q4(b): to verify neural objective gradient with MNIST data

```
load_mnist;
W{1} = randn(784,20); % Random Gaussian Weights
W{2} = randn(20,15);
W{3} = randn(15,10);
vec = cell_to_vec(W);
D = x_train;
L = y_train; % Creating a one-hot label representation
f1 = @net_objective;
fprintf('The loss of the neural network is')
feval(f1,W,D,L)
f2 = @net_grad;
```

The loss of the neural network is
ans = 2.3793