Q1(a) Code

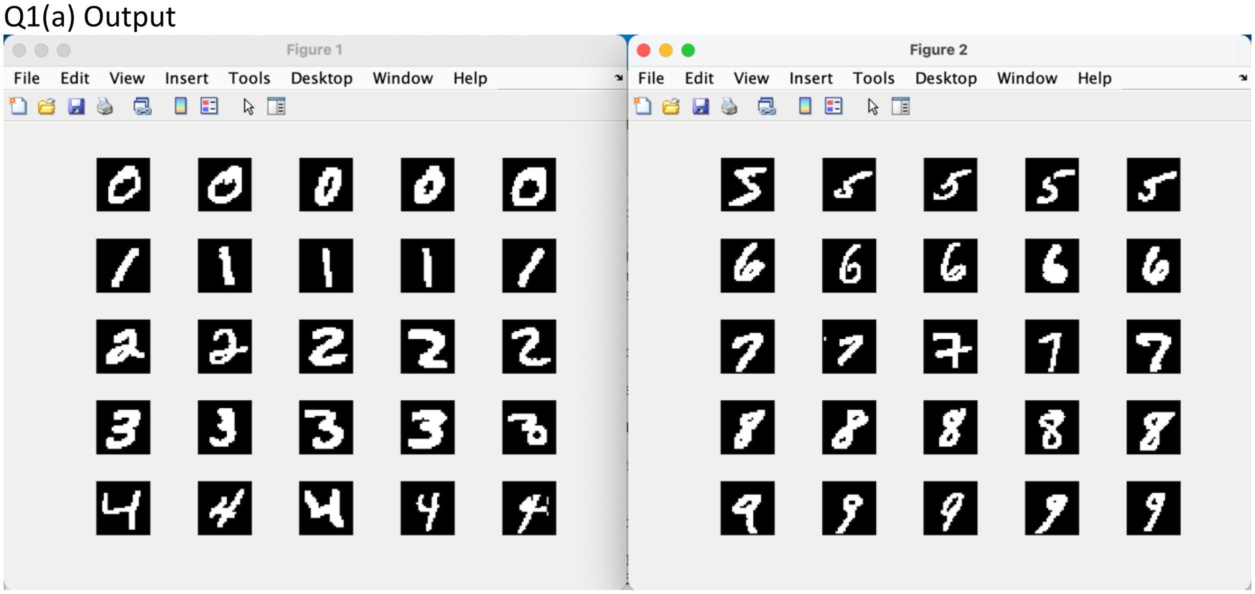
```matlab
% display digits data set
digitimages = loadMNISTImages('Digits_train-images.idx3-ubyte');
imagelabels = loadMNISTLabels('Digits_train-labels.idx1-ubyte');

digits0to9 = [];
for i = 1:length(imagelabels)
    digits0to9(i,1) = i;
    digits0to9(i,2) = imagelabels(i);
end
digits0to9=sortrows(digits0to9,2);

% fig1: 5x5 grid for digits 0-4
f1 = figure;
for digit0to4 = 1:5
    newdigits = digits0to9(digits0to9(:,2)>=digit0to4,:);
    for j = 1:5
        k = newdigits(j,1);
        subplot(5,5,(j+(5*(digit0to4-1))));
        imshow(digitimages(:,:,k));
    end
end

% fig2: 5x5 grid for digits 5-9
f2 = figure;
for digit5to9 = 1:5
    newdigits = digits0to9(digits0to9(:,2)>=(digit5to9+5),:);
    for j = 1:5
        k = newdigits(j,1);
        subplot(5,5,(j+(5*(digit5to9-1))));
        imshow(digitimages(:,:,k));
    end
end
```

Q1(a) Output



Q1(b) Code

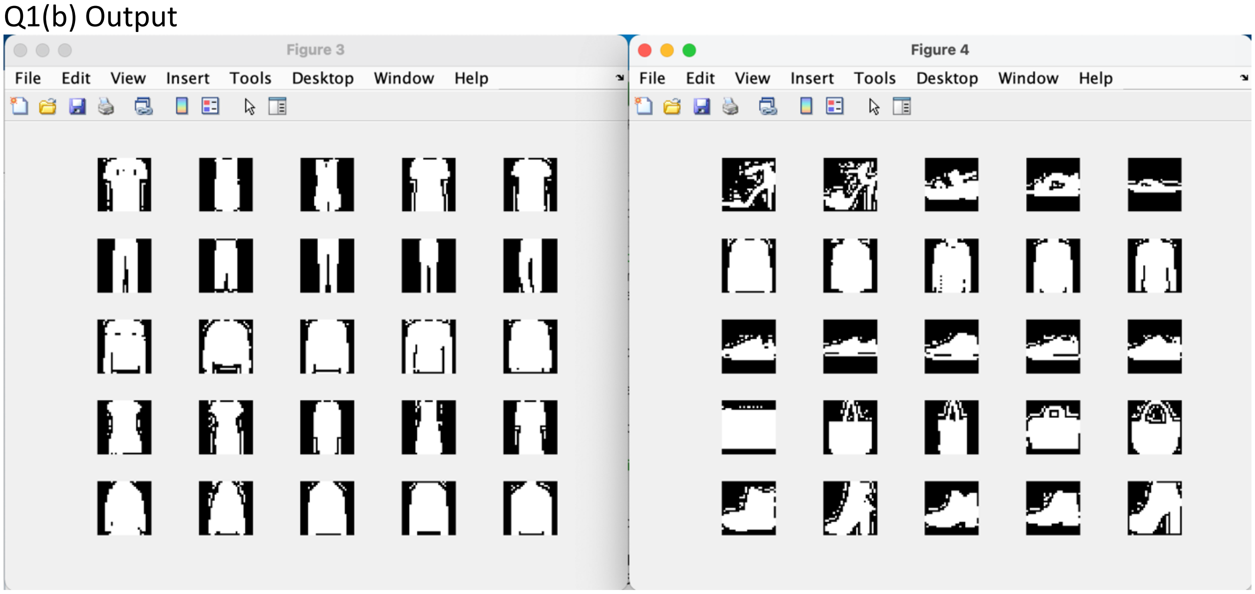
```
% display fashion data set
fashionimages = loadMNISTImages('fashion_train-images.idx3-ubyte');
fashionlabels = loadMNISTLabels('fashion_train-labels.idx1-ubyte');

fashionclothing = [];
for i = 1:length(fashionlabels)
    fashionclothing(i,1) = i;
    fashionclothing(i,2) = fashionlabels(i);
end
fashionclothing=sortrows(fashionclothing,2);

% fig3: 5x5 grid for fashion 1-5
f3 = figure;
for item0to4 = 1:5
    newitem = fashionclothing(fashionclothing(:,2)>=item0to4,:);
    for j = 1:5
        k = newitem(j,1);
        subplot(5,5,(j+(5*(item0to4-1))));
        imshow(fashionimages(:,:,k));
    end
end

% fig4: 5x5 grid for fashion 6-10
f4 = figure;
for item5to9 = 1:5
    newitem = fashionclothing(fashionclothing(:,2)>=(item5to9+5),:);
    for j = 1:5
        k = newitem(j,1);
        subplot(5,5,(j+(5*(item5to9-1))));
        imshow(fashionimages(:,:,k));
    end
end
```

Q1(b) Output



It will be easier to classify the digits than fashion images because the digits are made up of components that can be easily identified. For instance, the digit ‘9’ is made of a circle and a stick, and the digit ‘8’ is made of 2 circles, etc. The fashion images have a higher degree in variations. For example, heels and sneakers are both classified as shoes, but they look very different.

Q1(c) Code

phi_ReLU.m

```
function h = phi_ReLU(z)
% Usage: computes the activation vector h for weighted input vector z
% using the ReLU activation function componentwise; both z and h
% are column vectors of the same size (should work for any size)

h=max(0,z);
end
```

phi_Softmax.m

```
function y = phi_Softmax(z)
% Usage: computes the probability vector y for real input vector z
% using the Softmax function; both z and y
% are column vectors with size equal to the number of classes
% (should work for any size)

y = exp(z)/sum(exp(z));
end
```

jac_ReLU.m

```
function J_ReLU = jac_ReLU(z)
% Usage: computes the Jacobian matrix of the ReLU activation
% function evaluated in weighted input vector z;
% this is a diagonal matrix (should work for input z of any size)

J = [];
for i = 1:length(z)
    if z(i)>=0
        J = [J, 1];
    else
        J = [J, 0];
    end
end

J_ReLU = diag(J);
end
```

jac_Softmax.m

```
function J_Softmax = jac_Softmax(y)
% Usage: computes the Jacobian matrix of the Softmax
% function evaluated in weighted input vector z, but we give
% the Softmax output y as input to the jac_Softmax(y) function
% because the Jacobian formulas can easily be expressed
% as a function of y; this matrix is not diagonal but it is symmetric % (s

J_Softmax = diag(y);

for i=1:length(J_Softmax)
    for k=1:size(J_Softmax)
        if i == k
            J_Softmax(i,k) = y(i)-(y(i)^2);
        else
            J_Softmax(i,k) = -y(i)*y(k);
        end
    end
end

end
```

Q1(d)

Command Window

```
>> z=[-2 2]';
>> h = phi_ReLU(z)

h =

     0
     2

>> y = phi_Softmax(z)

y =

   0.017986209962092
   0.982013790037908

>> J_ReLU = jac_ReLU(z)

J_ReLU =

     0     0
     0     1

>> J_Softmax = jac_Softmax(y)

J_Softmax =

   0.017662706213291  -0.017662706213291
  -0.017662706213291   0.017662706213291
```

Calculator results:

$$h = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

$$y = \begin{bmatrix} \exp(-2)/(\exp(-2) + \exp(2)) \\ \exp(2)/(\exp(-2) + \exp(2)) \end{bmatrix} = \begin{bmatrix} 0.017986209 \\ 0.982013790 \end{bmatrix}$$

$$J\_ReLU = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$J\_Softmax = \begin{bmatrix} \hat{y}_1\text{-}\hat{y}_1{}^2 & \text{-}\hat{y}_2\hat{y}_1 \\ \text{-}\hat{y}_1\hat{y}_2 & \hat{y}_2\text{-}\hat{y}_2{}^2 \end{bmatrix} = \begin{bmatrix} 0.017662706 & -0.017662706 \\ -0.017662706 & 0.017662706 \end{bmatrix}$$

All the functions work properly for the input column vector z = [-2 2]’ as per calculator results.