CpE 520: HW#12

West Virginia University

Ali Zafari

Table of Contents

In the first problem of this homework assignment a Radial Basis Function neural network will be trained on MNIST dataset and its classification performance will be reported.

The second problem discussed how to find the partial derivatives to be used on gradient descent algorithm.

```
clc; clear; close all;
load('HW12_code_workspace.mat');
```

MNIST Dataset

0.1 Downloading MNIST Dataset and Import into *Matlab*

The code snippet below will download MNIST dataset from web and extract the downloaded files into Matlab matrices as test and train images and labels.

```
mnist_train_image = 'train-images-idx3-ubyte';
mnist_train_label = 'train-labels-idx1-ubyte';
mnist_test_image  = 't10k-images-idx3-ubyte';
mnist_test_label  = 't10k-labels-idx1-ubyte';
train_set_number  = 60000;
test_set_number   = 10000;

downloadMNIST(mnist_train_image, mnist_train_label, mnist_test_image,
    mnist_test_label);
```

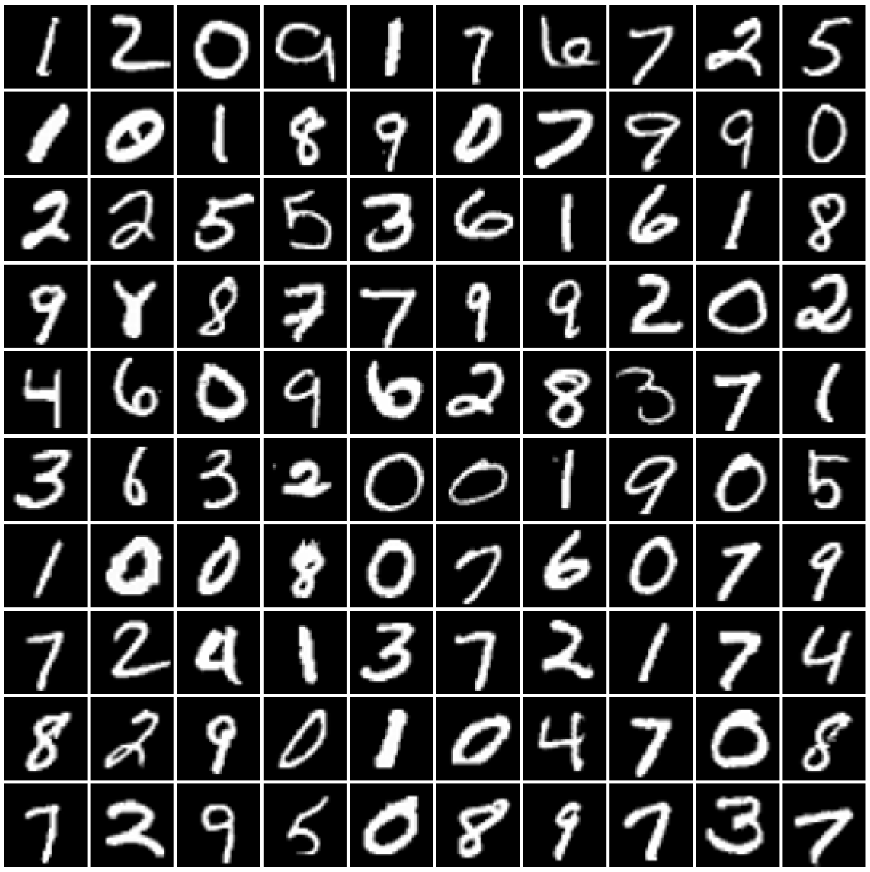
```
Downloading MNIST dataset...
MNIST dataset downloded.
Unzipping started...
Unzipping completed.
```

```
[train_images, train_labels] = readMNIST(mnist_train_image, mnist_train_label,
    train_set_number);
[test_images,  test_labels] = readMNIST(mnist_test_image, mnist_test_label,
    test_set_number);

clear mnist_train_image mnist_train_label mnist_test_image mnist_test_label
    train_set_number test_set_number
```

0.2 Plotting a Sample of MNIST Dataset

A set of 100 randomly chosen samples of the training dataset is plotted here.

```
random_indices = randi(60000, [1 100]);
montage(train_images(:, :, random_indices), 'BorderSize', [2 2], '
    BackgroundColor', 'white');
```



0.3 Number of Occurances for Each Digit in Training Set

Here we investigate how many of each digit is there in the training set.

```
for i=0:9
    disp([num2str(i),': ', num2str(sum(train_labels == i))]);
end
```

```
0: 5923
1: 6742
2: 5958
3: 6131
4: 5842
5: 5421
6: 5918
7: 6265
8: 5851
9: 5949
```

0.4 Rashaping Images Into Vectors

For further calculations it is more desirable to work with vectors of 784 elements instead of 28x28 size images.

```
train_images_reshaped = reshape(train_images, [size(train_images, 1)*size(
    train_images, 2) size(train_images, 3)]);
test_images_reshaped = reshape(test_images, [size(test_images, 1)*size(
    test_images, 2) size(test_images, 3)]);
```

Problem 1

In this problem we implement an RBF network with 40 hidden units and 10 binary outputs.

1.1 Finding Centroids Using K-means Clustring Algorithm

By running a K-means algorithm on the training dataset, we have found 40 centroids to be used later in the radial basis functions.

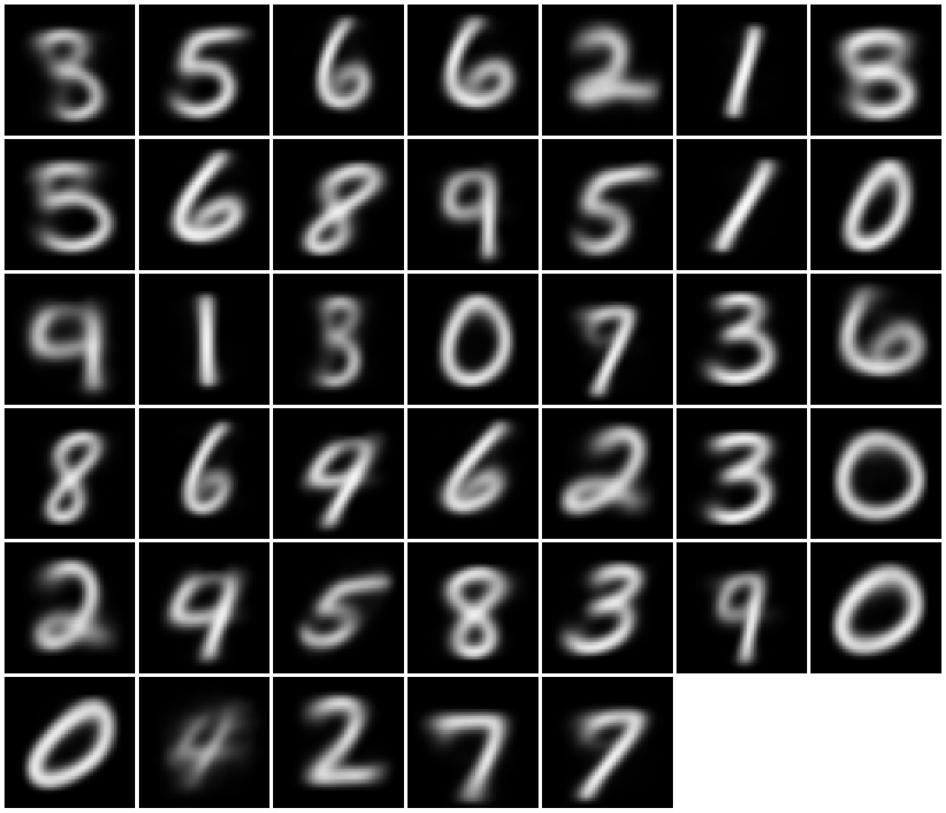
```matlab
K = 40;
[idx, centroids] = kmeans(train_images_reshaped', K, "Display", "iter", "MaxIter
    ", 1000, 'OnlinePhase','on');
```

```
iter    phase      num          sum
   1        1    60000    2.17743e+06
   2        1    17400    2.03712e+06
   3        1     8921    1.99709e+06
   4        1     5806    1.97847e+06
   5        1     4445    1.96579e+06
   6        1     3544    1.95701e+06
   7        1     2930    1.95091e+06
   8        1     2449    1.94653e+06
   9        1     2030    1.94354e+06
  10        1     1667     1.9415e+06
  11        1     1454    1.93993e+06
  12        1     1216    1.93875e+06
  13        1     1124    1.93772e+06
  14        1     1058     1.9368e+06
  15        1      960    1.93605e+06
  16        1      933    1.93534e+06
  17        1      853    1.93469e+06
  18        1      884    1.93397e+06
  19        1      893    1.93322e+06
  20        1      919    1.93239e+06
  21        1      947    1.93141e+06
  22        1      949    1.93028e+06
  23        1     1020     1.9288e+06
  24        1     1066      1.927e+06
  25        1      998    1.92553e+06
  26        1      908    1.92448e+06
  27        1      863    1.92363e+06
  28        1      775    1.92301e+06
  29        1      714    1.92253e+06
  30        1      619    1.92211e+06
  31        1      576    1.92178e+06
  32        1      526     1.9215e+06
  33        1      529    1.92122e+06
  34        1      530    1.92094e+06
  35        1      497    1.92068e+06
  36        1      510    1.92042e+06
  37        1      512    1.92016e+06
  38        1      462    1.91993e+06
  39        1      447    1.91972e+06
  40        1      460    1.91951e+06
  41        1      447    1.91928e+06
  42        1      462    1.91906e+06
  43        1      414    1.91887e+06
  44        1      421    1.91865e+06
  45        1      410    1.91843e+06
  46        1      380    1.91824e+06
  47        1      377    1.91804e+06
  48        1      366    1.91786e+06
  49        1      400    1.91762e+06
  50        1      411    1.91738e+06
  51        1      407    1.91712e+06
  52        1      459    1.91682e+06
  53        1      486    1.91647e+06
  54        1      503    1.91609e+06
  55        1      507    1.91573e+06
  56        1      508    1.91536e+06
  57        1      530    1.91497e+06
  58        1      533    1.91458e+06
  59        1      496    1.91424e+06
```

```
    60       1      490         1.91391e+06
    61       1      468         1.91358e+06
    62       1      475         1.91328e+06
    63       1      472         1.91298e+06
    64       1      470         1.91269e+06
    65       1      433         1.91247e+06
    66       1      397         1.91227e+06
    67       1      373          1.9121e+06
    68       1      342         1.91197e+06
    69       1      271         1.91188e+06
    70       1      240         1.91182e+06
    71       1      245         1.91175e+06
    72       1      207          1.9117e+06
    73       1      198         1.91165e+06
    74       1      192         1.91159e+06
    75       1      206         1.91154e+06
    76       1      209         1.91148e+06
    77       1      214          1.9114e+06
    78       1      202         1.91134e+06
    79       1      208         1.91127e+06
    80       1      191         1.91119e+06
    81       1      184         1.91113e+06
    82       1      189         1.91107e+06
    83       1      191           1.911e+06
    84       1      190         1.91094e+06
    85       1      177         1.91089e+06
    86       1      176         1.91083e+06
    87       1      166         1.91077e+06
    88       1      158         1.91072e+06
    89       1      162         1.91067e+06
    90       1      174         1.91062e+06
    91       1      172         1.91057e+06
    92       1      166         1.91052e+06
    93       1      148         1.91047e+06
    94       1      145         1.91043e+06
    95       1      105         1.91041e+06
    96       1      116         1.91038e+06
    97       1      101         1.91036e+06
    98       1      123         1.91033e+06
    99       1      132          1.9103e+06
   100       1      126         1.91027e+06
   101       1      104         1.91025e+06
   102       1       86         1.91023e+06
   103       1       79         1.91022e+06
   104       1       69         1.91021e+06
   105       1       58          1.9102e+06
   106       1       50         1.91019e+06
   107       1       50         1.91018e+06
   108       1       41         1.91018e+06
   109       1       24         1.91018e+06
   110       1       24         1.91017e+06
   111       1       28         1.91017e+06
   112       1       29         1.91017e+06
   113       1       31         1.91017e+06
   114       1       25         1.91016e+06
   115       1       16         1.91016e+06
   116       1       18         1.91016e+06
   117       1       16         1.91016e+06
   118       1       15         1.91016e+06
   119       1       11         1.91015e+06
   120       1        8         1.91015e+06
   121       1        2         1.91015e+06
   122       1        1         1.91015e+06
   123       2        1         1.91015e+06
   124       2        1         1.91014e+06
   125       2        1         1.91014e+06
   126       2        1         1.91013e+06
   127       2        1         1.91013e+06
   128       2        1         1.91012e+06
   129       2        1         1.91011e+06
   130       2        1         1.91011e+06
   131       2        1          1.9101e+06
   132       2        1          1.9101e+06
   133       2        1         1.91009e+06
   134       2        1         1.91009e+06
   135       2        1         1.91008e+06
   136       2        1         1.91008e+06
   137       2        1         1.91008e+06
   138       2        1         1.91008e+06
   139       2        1         1.91008e+06
   140       2        1         1.91008e+06
   141       2        0         1.91008e+06
Best total sum of distances = 1.91008e+06
```

The 40 centroids are plotted below:

```
iMontage(centroids');
```



The 2 line codes below will change the labels into a one-hot-encoded matrix.

```
train_labels_one_hot_encoded = full(ind2vec(train_labels'+1));
test_labels_one_hot_encoded  = full(ind2vec(test_labels'+1));
```

1.2 Finding Weights Using Psuedo-Inverse

We first find the matrix of th radial basis functions' outputs and then by using the psuedo-inverse of it, the weights will be known as the equation below:

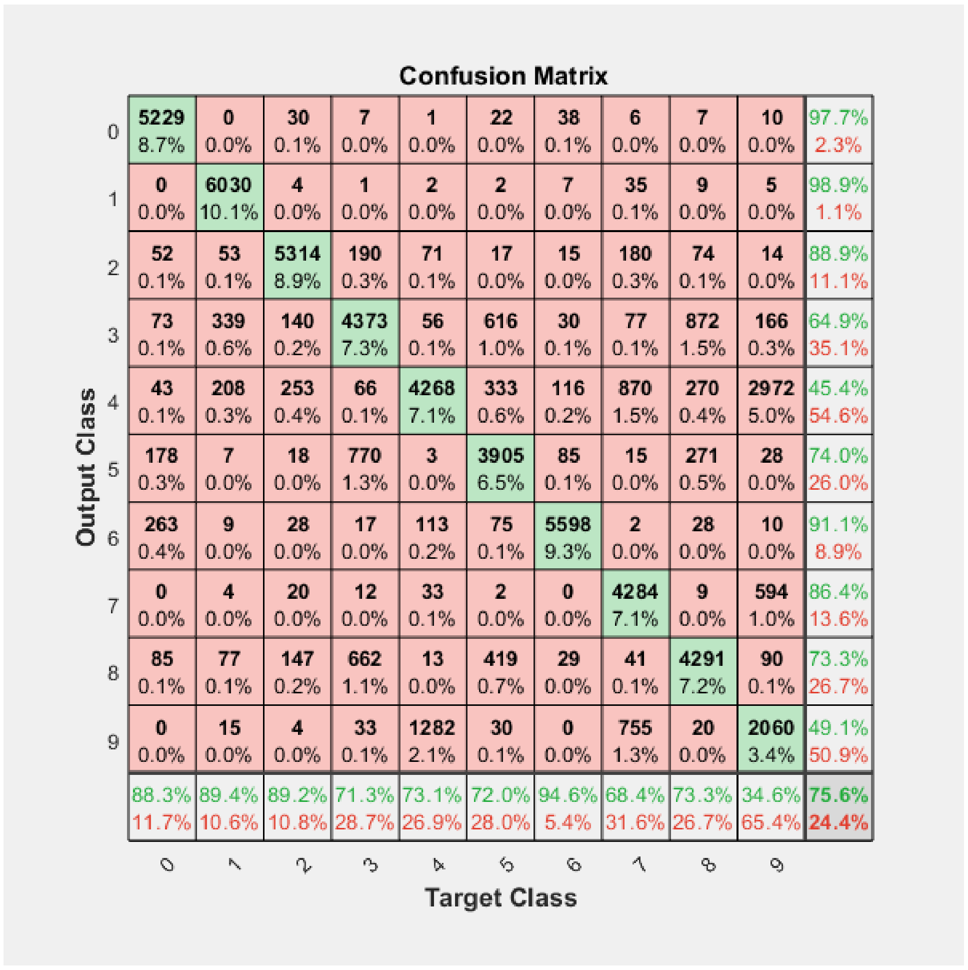
$$\mathbf{W} = \mathbf{\Phi}^{\dagger}\mathbf{y}$$

```
sigma = max(max(dist(centroids)))/sqrt(2*K);
N = size(train_images_reshaped, 2);
hidden = zeros(N, K);

for i = 1:40
    hidden(:, i) = exp(vecnorm(train_images_reshaped'-centroids(i, :), 2, 2)
        .^2/(-2*sigma*sigma));
end

weights = (inv(hidden'*hidden) * hidden') * (train_labels_one_hot_encoded');
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  1.264945e-56.
```

With the code snippet below, we first predict the labels of training dataset with the RBF network, then plot the confusion matrix to see how is the classification performance.

```
predicted_train_labels_one_hot_encoded = hidden * weights;

[~, train_labels_predicted] = max(predicted_train_labels_one_hot_encoded');

categorized_label_train = categorical(train_labels, 0:9, {'0' '1' '2' '3' '4' '5
    ' '6' '7' '8' '9'});
categorized_label_train_predicted = categorical(train_labels_predicted', 1:10, {
    '0' '1' '2' '3' '4' '5' '6' '7' '8' '9'});

plotconfusion(categorized_label_train, categorized_label_train_predicted);
```


Confusion Matrix
Output Class
Target Class

Output \ Target	0	1	2	3	4	5	6	7	8	9	
0	5229 8.7%	0 0.0%	30 0.1%	7 0.0%	1 0.0%	22 0.0%	38 0.1%	6 0.0%	7 0.0%	10 0.0%	97.7% 2.3%
1	0 0.0%	6030 10.1%	4 0.0%	1 0.0%	2 0.0%	2 0.0%	7 0.0%	35 0.1%	9 0.0%	5 0.0%	98.9% 1.1%
2	52 0.1%	53 0.1%	5314 8.9%	190 0.3%	71 0.1%	17 0.0%	15 0.0%	180 0.3%	74 0.1%	14 0.0%	88.9% 11.1%
3	73 0.1%	339 0.6%	140 0.2%	4373 7.3%	56 0.1%	616 1.0%	30 0.1%	77 0.1%	872 1.5%	166 0.3%	64.9% 35.1%
4	43 0.1%	208 0.3%	253 0.4%	66 0.1%	4268 7.1%	333 0.6%	116 0.2%	870 1.5%	270 0.4%	2972 5.0%	45.4% 54.6%
5	178 0.3%	7 0.0%	18 0.0%	770 1.3%	3 0.0%	3905 6.5%	85 0.1%	15 0.0%	271 0.5%	28 0.0%	74.0% 26.0%
6	263 0.4%	9 0.0%	28 0.0%	17 0.0%	113 0.2%	75 0.1%	5598 9.3%	2 0.0%	28 0.0%	10 0.0%	91.1% 8.9%
7	0 0.0%	4 0.0%	20 0.0%	12 0.0%	33 0.1%	2 0.0%	0 0.0%	4284 7.1%	9 0.0%	594 1.0%	86.4% 13.6%
8	85 0.1%	77 0.1%	147 0.2%	662 1.1%	13 0.0%	419 0.7%	29 0.0%	41 0.1%	4291 7.2%	90 0.1%	73.3% 26.7%
9	0 0.0%	15 0.0%	4 0.0%	33 0.1%	1282 2.1%	30 0.1%	0 0.0%	755 1.3%	20 0.0%	2060 3.4%	49.1% 50.9%
	88.3% 11.7%	89.4% 10.6%	89.2% 10.8%	71.3% 28.7%	73.1% 26.9%	72.0% 28.0%	94.6% 5.4%	68.4% 31.6%	73.3% 26.7%	34.6% 65.4%	75.6% 24.4%

1.3 Plotting Learning Curves

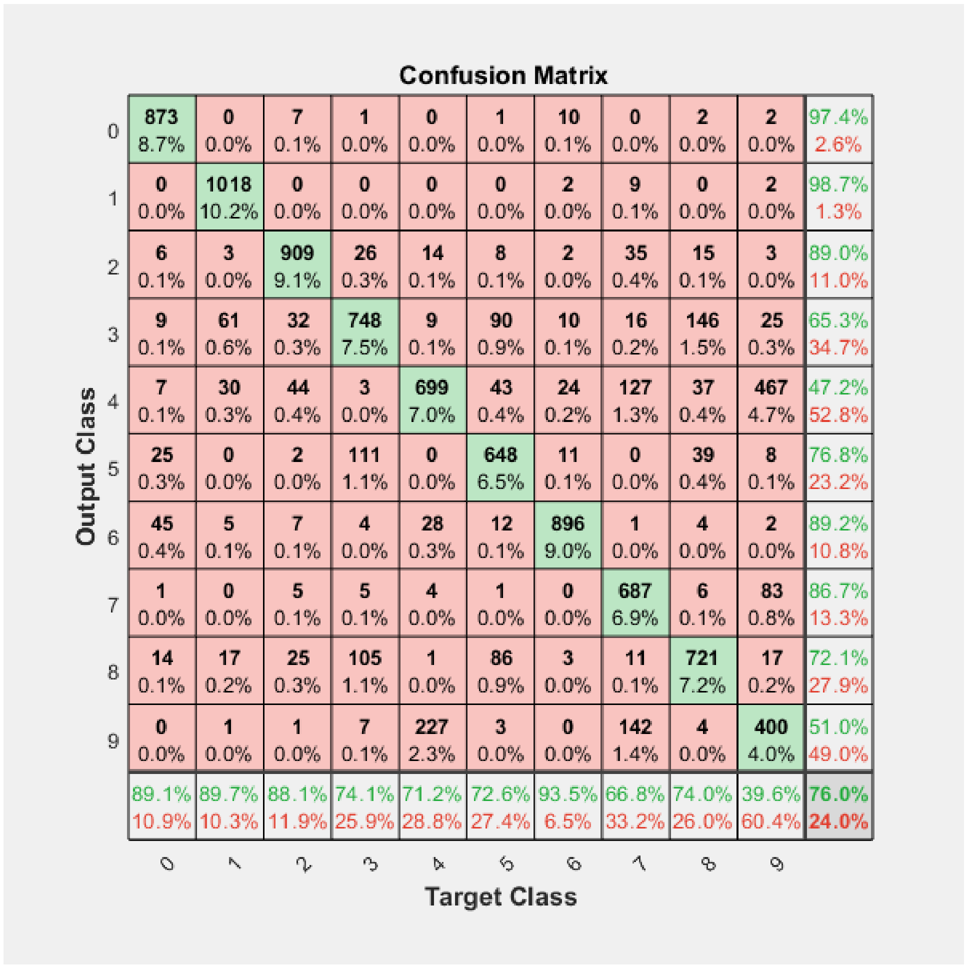
By using psuedo-inverse algorithm we have found the weights at once, so there is no learning curve available to be displayed.

1.4 Classification Performance

With the code snippet below, we first predict the labels of test dataset with the RBF network, then plot the confusion matrix to see how is the classification performance.

As it is clear, performance on the whole test set of MNIST is about 76% which is acceptable in comparison with the time that feed forward neural networks need to be trained and acheive a comparable performance.

```
N_test = size(test_images_reshaped, 2);
hidden_test = zeros(N_test, K);

for i = 1:40
    hidden_test(:, i) = exp(vecnorm(test_images_reshaped'-centroids(i, :), 2, 2)
          .^2/(-2*sigma*sigma));
end

predicted_test_labels_one_hot_encoded = hidden_test*weights;

[~, I] = max(predicted_test_labels_one_hot_encoded');

categorized_label_test = categorical(test_labels, 0:9, {'0' '1' '2' '3' '4' '5'
     '6' '7' '8' '9'});
categorized_label_test_predicted = categorical(I', 1:10, {'0' '1' '2' '3' '4' '5
     ' '6' '7' '8' '9'});

plotconfusion(categorized_label_test, categorized_label_test_predicted);
```



Confusion Matrix

Output Class \ Target Class	0	1	2	3	4	5	6	7	8	9	
0	873 8.7%	0 0.0%	7 0.1%	1 0.0%	0 0.0%	1 0.0%	10 0.1%	0 0.0%	2 0.0%	2 0.0%	97.4% 2.6%
1	0 0.0%	1018 10.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.0%	9 0.1%	0 0.0%	2 0.0%	98.7% 1.3%
2	6 0.1%	3 0.0%	909 9.1%	26 0.3%	14 0.1%	8 0.1%	2 0.0%	35 0.4%	15 0.1%	3 0.0%	89.0% 11.0%
3	9 0.1%	61 0.6%	32 0.3%	748 7.5%	9 0.1%	90 0.9%	10 0.1%	16 0.2%	146 1.5%	25 0.3%	65.3% 34.7%
4	7 0.1%	30 0.3%	44 0.4%	3 0.0%	699 7.0%	43 0.4%	24 0.2%	127 1.3%	37 0.4%	467 4.7%	47.2% 52.8%
5	25 0.3%	0 0.0%	2 0.0%	111 1.1%	0 0.0%	648 6.5%	11 0.1%	0 0.0%	39 0.4%	8 0.1%	76.8% 23.2%
6	45 0.4%	5 0.1%	7 0.1%	4 0.0%	28 0.3%	12 0.1%	896 9.0%	1 0.0%	4 0.0%	2 0.0%	89.2% 10.8%
7	1 0.0%	0 0.0%	5 0.1%	5 0.1%	4 0.0%	1 0.0%	0 0.0%	687 6.9%	6 0.1%	83 0.8%	86.7% 13.3%
8	14 0.1%	17 0.2%	25 0.3%	105 1.1%	1 0.0%	86 0.9%	3 0.0%	11 0.1%	721 7.2%	17 0.2%	72.1% 27.9%
9	0 0.0%	1 0.0%	1 0.0%	7 0.1%	227 2.3%	3 0.0%	0 0.0%	142 1.4%	4 0.0%	400 4.0%	51.0% 49.0%
	89.1% 10.9%	89.7% 10.3%	88.1% 11.9%	74.1% 25.9%	71.2% 28.8%	72.6% 27.4%	93.5% 6.5%	66.8% 33.2%	74.0% 26.0%	39.6% 60.4%	76.0% 24.0%

Problem 2

At first we rewrite the instantaneous error function as the equations below:

$$E(x^{(k)}) = \frac{1}{2}\{y^{(k)} - \sum_{i=1}^{m} w_i \exp[-\frac{||x^{(k)} - \mu_i||^2}{2\sigma_i^2}]\}^2$$

2.1 Calculating $\partial\mathbf{E}/\partial\mathbf{w_j}$

We can immediately find this derivative by th compact form of the squared error equation.

$$\begin{aligned}\frac{\partial E}{\partial w_j} &= \frac{1}{2}(2)(y^{(k)} - \sum_{i=1}^{m} w_i \exp[-\frac{||x^{(k)} - \mu_i||^2}{2\sigma_i^2}]) \exp[-\frac{||x^{(k)} - \mu_j||^2}{2\sigma_j^2}] \\ &= (y^{(k)} - f(x^{(k)}))\phi_j(x^{(k)})\end{aligned}$$

2.2 Calculating $\partial\mathbf{E}/\partial\mu_\mathbf{j}$

$$\begin{aligned}\frac{\partial E}{\partial \mu_j} &= \frac{1}{2}(2)(y^{(k)} - \sum_{i=1}^{m} w_i \exp[-\frac{||x^{(k)} - \mu_i||^2}{2\sigma_i^2}])\frac{\partial}{\partial \mu_j}\{w_j \exp[-\frac{||x^{(k)} - \mu_j||^2}{2\sigma_j^2}]\} \\ &= (y^{(k)} - \sum_{i=1}^{m} w_i \exp[-\frac{||x^{(k)} - \mu_i||^2}{2\sigma_i^2}])w_j \exp[-\frac{||x^{(k)} - \mu_j||^2}{2\sigma_j^2}]\frac{\partial}{\partial \mu_j}\{-\frac{||x^{(k)} - \mu_j||^2}{2\sigma_j^2}\} \\ &= (y^{(k)} - \sum_{i=1}^{m} w_i \exp[-\frac{||x^{(k)} - \mu_i||^2}{2\sigma_i^2}])w_j \exp[-\frac{||x^{(k)} - \mu_j||^2}{2\sigma_j^2}](-\frac{2(x^{(k)} - \mu_j)}{2\sigma_j^2}) \\ &= (y^{(k)} - f(x^{(k)}))\phi_j(x^{(k)})(-\frac{(x^{(k)} - \mu_j)}{\sigma_j^2})\end{aligned}$$

2.3 Calculating $\partial\mathbf{E}/\partial\sigma_\mathbf{j}$

$$\begin{aligned}\frac{\partial E}{\partial \sigma_j} &= \frac{1}{2}(2)(y^{(k)} - \sum_{i=1}^{m} w_i \exp[-\frac{||x^{(k)} - \mu_i||^2}{2\sigma_i^2}])\frac{\partial}{\partial \sigma_j}\{w_j \exp[-\frac{||x^{(k)} - \mu_j||^2}{2\sigma_j^2}]\} \\ &= (y^{(k)} - \sum_{i=1}^{m} w_i \exp[-\frac{||x^{(k)} - \mu_i||^2}{2\sigma_i^2}])w_j \exp[-\frac{||x^{(k)} - \mu_j||^2}{2\sigma_j^2}]\frac{\partial}{\partial \sigma_j}\{-\frac{||x^{(k)} - \mu_j||^2}{2\sigma_j^2}\} \\ &= (y^{(k)} - \sum_{i=1}^{m} w_i \exp[-\frac{||x^{(k)} - \mu_i||^2}{2\sigma_i^2}])w_j \exp[-\frac{||x^{(k)} - \mu_j||^2}{2\sigma_j^2}](-(-2)\frac{||x^{(k)} - \mu_j||^2}{2\sigma_j^3}) \\ &= (y^{(k)} - f(x^{(k)}))\phi_j(x^{(k)})(\frac{(x^{(k)} - \mu_j)}{\sigma_j^3})\end{aligned}$$

Appendix

A.1 Saving Workspace Variables for Future Use

```
save('HW12_code_workspace.mat');
```

A.2 Definition of Auxiliary Functions

```
function downloadMNIST(mnist_train_image, mnist_train_label, mnist_test_image,
    mnist_test_label)

if exist('train-images-idx3-ubyte','file') ~= 2
    disp('Downloading MNIST dataset...');
    websave([mnist_train_image,'.gz'],...
        ['http://yann.lecun.com/exdb/mnist/', ...
        mnist_train_image, '.gz']);
    websave([mnist_train_label,'.gz'],...
        ['http://yann.lecun.com/exdb/mnist/', ...
        mnist_train_label, '.gz']);
    websave([mnist_test_image,'.gz'],...
        ['http://yann.lecun.com/exdb/mnist/', ...
        mnist_test_image, '.gz']);
    websave([mnist_test_label,'.gz'],...
        ['http://yann.lecun.com/exdb/mnist/', ...
        mnist_test_label, '.gz']);
    disp('MNIST dataset downloded.');

    disp('Unzipping started...');
    gunzip([mnist_train_image, '.gz'])
    gunzip([mnist_train_label, '.gz'])
    gunzip([mnist_test_image, '.gz'])
    gunzip([mnist_test_label, '.gz'])
    delete([mnist_train_image, '.gz'])
    delete([mnist_train_label, '.gz'])
    delete([mnist_test_image, '.gz'])
    delete([mnist_test_label, '.gz'])
    disp('Unzipping completed.');
else
    disp('MNIST dataset already downloaded.')
end

end

function [imgs, labels] = readMNIST(imgFile, labelFile, num_of_digits_to_read)

```

```
fileID = fopen(imgFile, 'r', 'b');
header = fread(fileID, 1, 'int32');

if header ~= 2051
    error('Invalid image file header');
end

count = fread(fileID, 1, 'int32');

if count < num_of_digits_to_read
    error('Trying to read too many digits');
end

rows_num = fread(fileID, 1, 'int32');
cols_num = fread(fileID, 1, 'int32');

imgs = zeros([rows_num cols_num num_of_digits_to_read]);

for i = 1:num_of_digits_to_read
    for row = 1:rows_num
        imgs(row, :, i) = fread(fileID, cols_num, 'uint8');
    end
end

fclose(fileID);

fileID = fopen(labelFile, 'r', 'b');
header = fread(fileID, 1, 'int32');

if header ~= 2049
    error('Invalid label file header');
end

count = fread(fileID, 1, 'int32');

if count < num_of_digits_to_read
    error('Trying to read too many digits');
end

labels = fread(fileID, num_of_digits_to_read, 'uint8');

fclose(fileID);

imgs = double(imgs)./255.0;

end

function iMontage(images)
```

```
montage(reshape(images, [28 28 size(images, 2)]), 'BackgroundColor', 'white', '
    BorderSize', [2 2]);
end
```