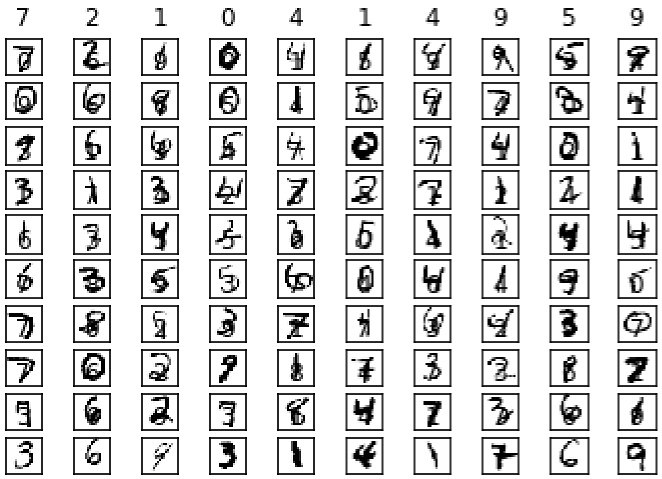
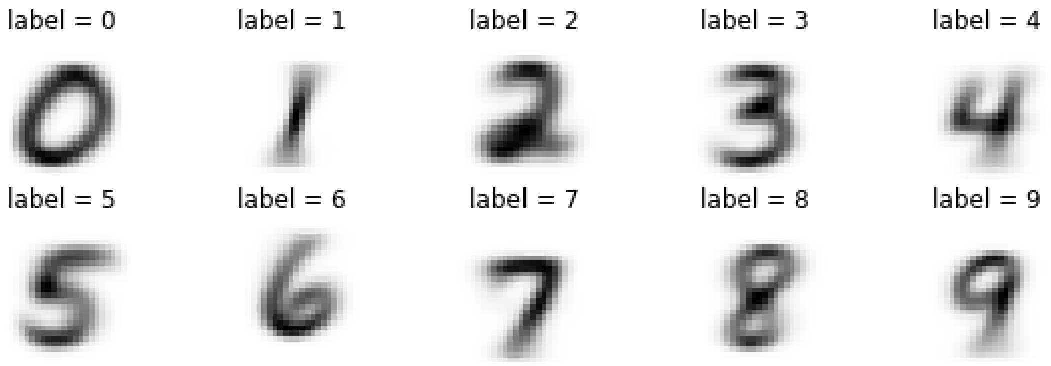
In [2]:

```
import matplotlib.pyplot as plt
import numpy as np

file_data   = "mnist_test.csv"
handle_file = open(file_data, "r")
data        = handle_file.readlines()
handle_file.close()

size_row    = 28    # height of the image
size_col    = 28    # width of the image

num_image   = len(data)
count       = 0     # count for the number of images

#
# make a matrix each column of which represents an images in a vector form
#
list_image  = np.empty((size_row * size_col, num_image), dtype=float)
list_label  = np.empty(num_image, dtype=int)

for line in data:

    line_data   = line.split(',')
    label       = line_data[0]
    im_vector   = np.asfarray(line_data[1:])

    list_label[count]       = label
    list_image[:, count]    = im_vector

    count += 1

#
# plot first 100 images out of 10,000 with their labels
#
f1 = plt.figure(1)

for i in range(100):

    label       = list_label[i]
    im_vector   = list_image[:, i]
    im_matrix   = im_vector.reshape((size_row, size_col))

    plt.subplot(10, 10, i+1)
    plt.title(label)
    plt.imshow(im_matrix, cmap='Greys', interpolation='None')

    frame   = plt.gca()
    frame.axes.get_xaxis().set_visible(False)
    frame.axes.get_yaxis().set_visible(False)

plt.show()
```



In [12]:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
from keras.datasets import mnist


# 1.Plot the average image
# load the dataset
file_data   = "mnist_test.csv"
handle_file = open(file_data, "r")
data        = handle_file.readlines()
handle_file.close()

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

#plotting them in an increasing order of labels
plt.figure(figsize=(10,3))
for i in range(10):
    avgImg = np.average(train_images[train_labels==i],0)
    plt.subplot(2, 5, i+1)
    plt.imshow(avgImg, cmap='Greys')
    plt.title('label = {}'.format(i))
    plt.axis('off')
```



In [70]:

```
from random import random
from keras.datasets import mnist
from math import exp

# Initialize a network
def initialize_network(n_inputs, n_hidden, n_outputs):
    network = list()
    hidden_layer = [{'weights':[random() for i in range(n_inputs + 1)]} for i in
    network.append(hidden_layer)
    output_layer = [{'weights':[random() for i in range(n_hidden + 1)]} for i in
    network.append(output_layer)
    return network

#nist(1)
network = initialize_network(2, 1, 2)
for layer in network:
    print(layer)

#we can devide forward propagation to three steps,the first one is:
# 1.calculate neurons activation for an input
def activate(weights, inputs):
    activation = 0
    for i in range(len(weights)-1):
        activation += weights[i] * inputs[i]
    return activation


# 2.ansfer neuron activation
def transfer(activation):
    return 1.0 / (1.0 + exp(-activation))

# Forward propagate input to a network output
def forward_propagate(network, row):
    inputs = row
    for layer in network:
        new_inputs = []
        for neuron in layer:
            activation = activate(neuron['weights'], inputs)
            neuron['output'] = transfer(activation)
            new_inputs.append(neuron['output'])
        inputs = new_inputs
    print("These are inputs of our feed forward process:",inputs)
    print("The average values for each label in the increasing order of the label
    for i in range(10):
        avg =  np.mean([train_labels==i],0)

        print(i,avg,"\n")
    return inputs

# test forward propagation
network = [[{'weights': [0.13436424411240122, 0.8474337369372327, 0.763774618976
        [{'weights': [0.2550690257394217, 0.49543508709194095]}, {'weights': [0.4
row = [1, 0, None]

output = forward_propagate(network, row)
print("the result of the output test is",output)
```

```
[{'weights': [0.11352903830196448, 0.13629583109956966, 0.29789449831522374]}]
[{'weights': [0.2335546293467009, 0.5693375798215953]}, {'weights': [0.63042721
06844898, 0.6632367391487167]}]
These are inputs of our feed forward process: [0.5339700092077766, 0.5596697192
194184]
The average values for each label in the increasing order of the label
0 [0. 1. 0. ... 0. 0. 0.]

1 [0. 0. 0. ... 0. 0. 0.]

2 [0. 0. 0. ... 0. 0. 0.]

3 [0. 0. 0. ... 0. 0. 0.]

4 [0. 0. 1. ... 0. 0. 0.]

5 [1. 0. 0. ... 1. 0. 0.]

6 [0. 0. 0. ... 0. 1. 0.]

7 [0. 0. 0. ... 0. 0. 0.]

8 [0. 0. 0. ... 0. 0. 1.]

9 [0. 0. 0. ... 0. 0. 0.]

the result of the output test is [0.5339700092077766, 0.5596697192194184]
```

In [ ]: