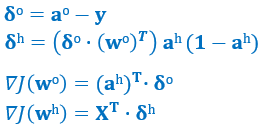
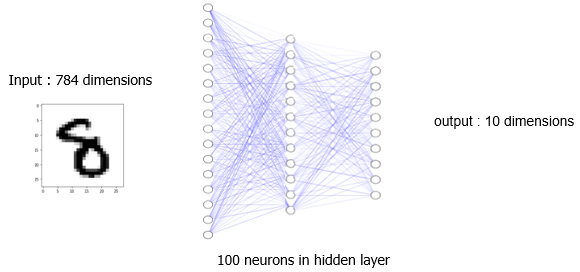
# Week11 : Multilayer Perceptron

* **Student ID** : 학번
* **Name** : 이름
* Write and run the code below (including Quiz) in jupyter notebook
* After completion, rename the file, and summit the file to e-class
* Submit file name : \*\*“Week11\_<StudentID>\_<Name>.ipynb”\*\*
  + Ex) Week11\_2020123456\_홍길동.ipynb
* Due : **Saturday 11:59pm**

# 1. Multilayer perceptron and backpropagation



### Sigmoid activation function

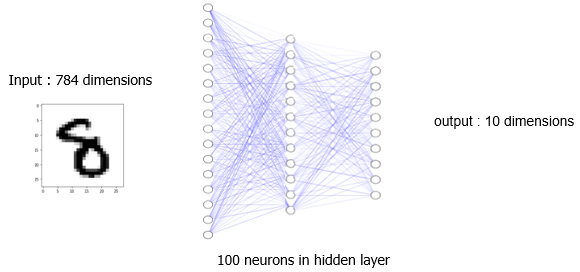


import numpy as np  
  
# sigmoid function  
def sigmoid(z):  
 return 1. / (1. + np.exp(-np.clip(z, -250, 250))) # np.clip - preventing overflow

# test sigmoid  
z = np.array([[1.0, 0.0, -1.0],  
 [-1.0, 0.0, 1.0]])  
  
print("sigmoid of z = \n", sigmoid(z))

sigmoid of z =   
 [[0.73105858 0.5 0.26894142]  
 [0.26894142 0.5 0.73105858]]

### Softmax function



# softmax function for 2D array  
def softmax(z):  
 exps = np.exp(z)  
 return None

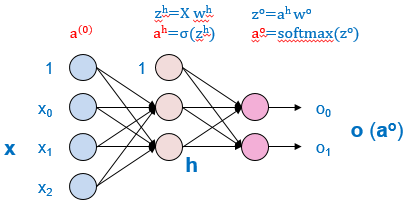
# test softmax  
z = np.array([[1.0, 0.0, -1.0],  
 [-1.0, 0.0, 1.0]])  
  
print("softmax of z = \n", softmax(z))  
print("sum of softmax values = \n", np.sum(softmax(z), axis=1, keepdims=True))

softmax of z =   
 [[0.66524096 0.24472847 0.09003057]  
 [0.09003057 0.24472847 0.66524096]]  
sum of softmax values =   
 [[1.]  
 [1.]]

### Example dataset

X = np.array([[0.5, 0.0, -0.5],  
 [-0.5, 0.0, 0.5]])  
y = np.array([[1, 0],  
 [0, 1]])

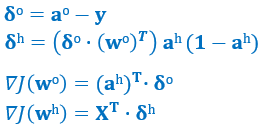
### Example network



### Initial parameters

# weights and bias of hidden layer. w\_h is (3, 2)  
w\_h = np.array([[1, -1],   
 [0, 0],  
 [-1, 1]])  
b\_h = [0.0, 0.0]  
  
# weights and bias of output layer. w\_o is (2, 2)  
w\_o = np.array([[1, -1],   
 [-1, 1]])  
b\_o = [0.0, 0.0]

### Forward computation



# input X  
print(X)

[[ 0.5 0. -0.5]  
 [-0.5 0. 0.5]]

# output of hidden layer  
z\_h = None  
a\_h = None  
  
print(z\_h)  
print(a\_h)

[[ 1. -1.]  
 [-1. 1.]]  
[[0.73105858 0.26894142]  
 [0.26894142 0.73105858]]

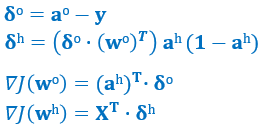
# output of output layer  
z\_o = None  
a\_o = None  
  
print(z\_o)  
print(a\_o)

[[ 0.46211716 -0.46211716]  
 [-0.46211716 0.46211716]]  
[[0.71590409 0.28409591]  
 [0.28409591 0.71590409]]

np.argmax(a\_o, axis=1)

array([0, 1], dtype=int64)

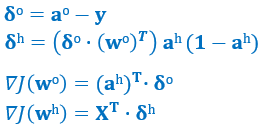
### Compute cost



# cross entropy loss  
cost = None  
  
print(cost)

0.334208933408766

### Compute gradients



# compute delta of output layer and hidden layer  
delta\_o = None  
delta\_h = None

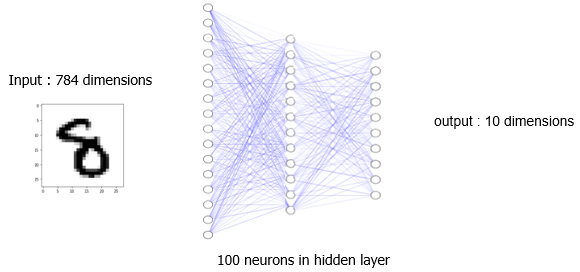
# compute gradient of output layer  
grad\_w\_o = None   
grad\_b\_o = None   
  
print(grad\_w\_o)  
print(grad\_b\_o)

[[-0.13128559 0.13128559]  
 [ 0.13128559 -0.13128559]]  
[0. 0.]

# compute gradient of hidden layer  
grad\_w\_h = None  
grad\_b\_h = None   
  
print(grad\_w\_h)  
print(grad\_b\_h)

[[-0.11171329 0.11171329]  
 [ 0. 0. ]  
 [ 0.11171329 -0.11171329]]  
[0. 0.]

### Update parameters - gradient descent



# learning rate  
alpha = 0.1  
  
# update parameters by gradient descent  
w\_o = None   
b\_o = None  
  
w\_h = None  
b\_h = None

print(w\_o)  
print(b\_o)  
print(w\_h)  
print(b\_h)

[[ 1.01312856 -1.01312856]  
 [-1.01312856 1.01312856]]  
[0. 0.]  
[[ 1.01117133 -1.01117133]  
 [ 0. 0. ]  
 [-1.01117133 1.01117133]]  
[0. 0.]

# 2. Image Classification using Multilayer Perceptron

### The MNIST image dataset

import matplotlib.pyplot as plt  
from scipy import io  
  
# load the MNIST dataset  
mnist = io.loadmat('mnist-original.mat')  
mnist

{'\_\_header\_\_': b'MATLAB 5.0 MAT-file Platform: posix, Created on: Sun Mar 30 03:19:02 2014',  
 '\_\_version\_\_': '1.0',  
 '\_\_globals\_\_': [],  
 'mldata\_descr\_ordering': array([[array(['label'], dtype='<U5'), array(['data'], dtype='<U4')]],  
 dtype=object),  
 'data': array([[0, 0, 0, ..., 0, 0, 0],  
 [0, 0, 0, ..., 0, 0, 0],  
 [0, 0, 0, ..., 0, 0, 0],  
 ...,  
 [0, 0, 0, ..., 0, 0, 0],  
 [0, 0, 0, ..., 0, 0, 0],  
 [0, 0, 0, ..., 0, 0, 0]], dtype=uint8),  
 'label': array([[0., 0., 0., ..., 9., 9., 9.]])}

# get X and y  
X = None  
y = None  
  
X = np.array(X).T  
X.shape

(70000, 784)

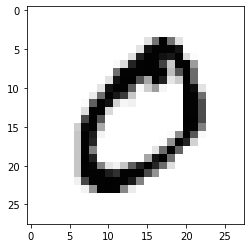
y = np.array(y).T.ravel()  
y.shape

(70000,)

# check data 0 (image 0)  
X[0]

array([ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 51, 159, 253,  
 159, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 48, 238,  
 252, 252, 252, 237, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 54,  
 227, 253, 252, 239, 233, 252, 57, 6, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10,  
 60, 224, 252, 253, 252, 202, 84, 252, 253, 122, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 163, 252, 252, 252, 253, 252, 252, 96, 189, 253, 167, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 51, 238, 253, 253, 190, 114, 253, 228, 47, 79, 255,  
 168, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 48, 238, 252, 252, 179, 12, 75, 121, 21, 0,  
 0, 253, 243, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 38, 165, 253, 233, 208, 84, 0, 0, 0,  
 0, 0, 0, 253, 252, 165, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 7, 178, 252, 240, 71, 19, 28, 0,  
 0, 0, 0, 0, 0, 253, 252, 195, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 57, 252, 252, 63, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 253, 252, 195, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 198, 253, 190, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 253, 196, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 76, 246, 252,  
 112, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 253, 252,  
 148, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 85,  
 252, 230, 25, 0, 0, 0, 0, 0, 0, 0, 0, 7, 135,  
 253, 186, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 85, 252, 223, 0, 0, 0, 0, 0, 0, 0, 0, 7,  
 131, 252, 225, 71, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 85, 252, 145, 0, 0, 0, 0, 0, 0, 0,  
 48, 165, 252, 173, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 86, 253, 225, 0, 0, 0, 0, 0,  
 0, 114, 238, 253, 162, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 85, 252, 249, 146, 48, 29,  
 85, 178, 225, 253, 223, 167, 56, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 85, 252, 252, 252,  
 229, 215, 252, 252, 252, 196, 130, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 28, 199,  
 252, 252, 253, 252, 252, 233, 145, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 25, 128, 252, 253, 252, 141, 37, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0], dtype=uint8)

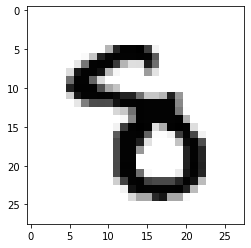
# show image 0  
ex1 = X[0]   
ex1\_image = None   
plt.imshow(ex1\_image, cmap='Greys')   
plt.show()



# label of image 0  
y[0]

0.0

# show image 50000  
ex1 = X[50000]   
ex1\_image = None   
plt.imshow(ex1\_image, cmap='Greys')   
plt.show()



# label of image 50000  
y[50000]

8.0

# train/test split  
X\_train, X\_test = X[:60000], X[60000:]  
y\_train, y\_test = y[:60000], y[60000:]

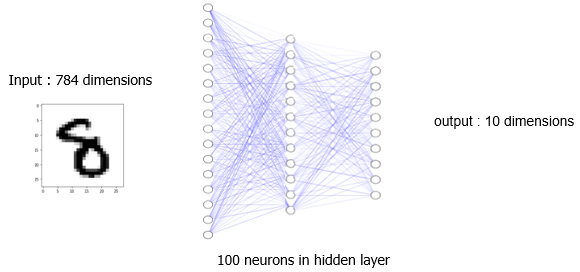
### One-hot encoding of class label

# function to encode class label to one-hot  
# ex> 2 --> 0 0 1 0 0 0 0 0 0 0  
def onehot(y, n\_classes):  
  
 # y is an array of labels  
 # n\_classes is number of different labels  
 onehot = np.zeros((y.shape[0], n\_classes))  
   
 for idx, val in enumerate(y.astype(int)):  
 onehot[idx, val] = 1.  
 return onehot

# test onehot encoding  
y = np.array([0, 1, 2, 0, 1, 2])  
print(onehot(y, 3))

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

### Generating batchs for Stochastic Gradient Descent



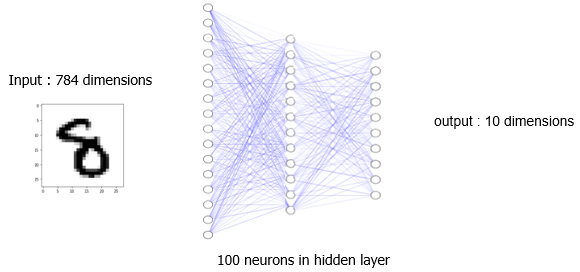
# total number of data and total index  
n\_data = X\_train.shape[0]  
indices = np.arange(n\_data)  
  
print("total number of data = ", n\_data)  
print("indices = ", indices)

total number of data = 60000  
indices = [ 0 1 2 ... 59997 59998 59999]

# test generating batch training data of size 10000  
batch\_size = 10000  
  
# for 0, 10000, 20000, ...  
for start\_idx in range(0, None, None):  
   
 # each batch has 10000 data  
 batch\_idx = indices[None]  
 print("indices =", batch\_idx, "batch data shape = ", X\_train[batch\_idx].shape)

indices = [ 0 1 2 ... 9997 9998 9999] batch data shape = (10000, 784)  
indices = [10000 10001 10002 ... 19997 19998 19999] batch data shape = (10000, 784)  
indices = [20000 20001 20002 ... 29997 29998 29999] batch data shape = (10000, 784)  
indices = [30000 30001 30002 ... 39997 39998 39999] batch data shape = (10000, 784)  
indices = [40000 40001 40002 ... 49997 49998 49999] batch data shape = (10000, 784)  
indices = [50000 50001 50002 ... 59997 59998 59999] batch data shape = (10000, 784)

### The Multilayer Perceptron class



import sys  
  
class NeuralNetMLP(object):  
 '''  
 This model has 1 hidden layer  
   
 n\_hidden : number of hidden units  
 epochs : number of epoches  
 alpha : learning rate  
 shuffle : if True, shuffle the training data each epoch   
 batch\_size : size of batch training set   
 seed : seed for random generation  
   
 z\_h, a\_h : z and output of hidden layer  
 z\_o, a\_o : z and output of output layer  
   
 n\_samples : number of total data   
 n\_features : number of features of a data  
 n\_output : numner of output (number of class labels)  
   
 w\_h, b\_h : parameter of hidden layer. (n\_features, n\_hidden), (n\_hidden)  
 w\_o, b\_o : parameter of output layer. (n\_hidden, n\_output), (n\_output)  
  
 '''  
 def \_\_init\_\_(self, n\_hidden=100, epochs=100, alpha=0.01,  
 shuffle=True, batch\_size=100, seed=None):  
  
 self.random = np.random.RandomState(seed)  
 self.n\_hidden = n\_hidden  
 self.epochs = epochs  
 self.alpha = alpha  
 self.shuffle = shuffle  
 self.batch\_size = batch\_size  
  
 # sigmoid function  
 def \_sigmoid(self, z):  
 return 1. / (1. + np.exp(-np.clip(z, -250, 250))) # np.clip - preventing overflow  
  
 # softmax function for 2D array  
 def \_softmax(self, z):  
 exps = np.exp(z)  
 return exps / np.sum(exps, axis=1, keepdims=True)  
  
 # forward computation  
 def \_forward(self, X):  
  
 # z and a of hidden layer. a = sigmoid(z)  
 # (n\_samples, n\_features) dot (n\_features, n\_hidden) -> (n\_samples, n\_hidden)  
 z\_h = None  
 a\_h = None  
  
 # z and a of output layer. a = softmax(z)  
 # (n\_samples, n\_hidden) dot (n\_hidden, n\_output) -> (n\_samples, n\_output)  
 z\_o = None  
 a\_o = None  
  
 return z\_h, a\_h, z\_o, a\_o  
  
 # compute cost - cross entropy  
 def \_compute\_cost(self, y\_enc, output):  
  
 # y\_enc : onehot endcoded y (n\_samples, n\_output (labels))  
 # output : a\_o of output layer (n\_samples, n\_output)  
 cost = None # output+1e-7 to prevent overflow  
   
 return cost  
  
 # predict class label  
 def predict(self, X):  
  
 # y\_pred : index of max output (n\_samples)  
 z\_h, a\_h, z\_o, a\_o = self.\_forward(X)  
 y\_pred = None  
  
 return y\_pred  
  
 # train the model  
 def fit(self, X\_train, y\_train):  
  
 # X\_train : (n\_samples, n\_features)  
 # y\_train : (n\_samples)  
 self.n\_samples = None  
 self.n\_features = None   
 self.n\_output = np.unique(y\_train).shape[0] # number of class labels  
  
 # initialize parameters  
 self.b\_h = np.zeros(self.n\_hidden)  
 self.w\_h = self.random.normal(loc=0.0, scale=0.1, size=(None, None))  
 self.b\_o = np.zeros(self.n\_output)  
 self.w\_o = self.random.normal(loc=0.0, scale=0.1, size=(None, None))  
  
 # one-hot encoding y\_train  
 y\_train\_enc = onehot(y\_train, self.n\_output)  
   
 # print the dimension of model  
 print("number of input = ", self.n\_features)   
 print("number of hidden = ", self.n\_hidden)   
 print("number of output = ", self.n\_output)   
  
 # record cost   
 self.history = []  
  
 # gradient descent for total epochs   
 for i in range(self.epochs):  
  
 indices = np.arange(self.n\_samples)  
 if self.shuffle:  
 self.random.shuffle(indices)  
  
 # for each batch  
 for start\_idx in range(0, indices.shape[0]-self.batch\_size+1, self.batch\_size):  
 batch\_idx = indices[start\_idx:start\_idx+self.batch\_size]  
  
 X = None  
 y = None  
   
 # forward computation  
 z\_h, a\_h, z\_o, a\_o = None  
  
 # compute deltas   
 delta\_o = None # [batch\_size, n\_output]  
 delta\_h = None # [batch\_size, n\_hidden]  
  
 # compute gradients   
 grad\_w\_o = None # [n\_hidden, n\_output]  
 grad\_b\_o = None  
 grad\_w\_h = np.dot(X.T, delta\_h) / self.batch\_size # [n\_features, n\_hidden]  
 grad\_b\_h = np.sum(delta\_h, axis=0) / self.batch\_size  
  
 # update parameters  
 self.w\_o = None # [n\_hidden, n\_output]  
 self.b\_o = None  
 self.w\_h = self.w\_h - self.alpha \* grad\_w\_h # [n\_features, n\_hidden]  
 self.b\_h = self.b\_h - self.alpha \* grad\_b\_h  
  
 # record costs every epoch  
 z\_h, a\_h, z\_o, a\_o = self.\_forward(X\_train)  
 cost = self.\_compute\_cost(y\_train\_enc, a\_o)  
 print('Iteration %5d: Cost %f ' % (i, cost))  
 self.history.append(cost)  
  
 return self

### Training MLP

# multilayer perceptron with 1 hidden layer, 100 hidden units.  
# stochastic gradient descent with batch size 100. learning rate = 0.01, epochs = 100  
  
nn = None

# train the network with 60000 training data  
None

number of input = 784  
number of hidden = 100  
number of output = 10  
Iteration 0: Cost 1.003301   
Iteration 1: Cost 0.678887   
Iteration 2: Cost 0.545278   
Iteration 3: Cost 0.475429   
Iteration 4: Cost 0.427215   
Iteration 5: Cost 0.391475   
Iteration 6: Cost 0.364307   
Iteration 7: Cost 0.340373   
Iteration 8: Cost 0.331018   
Iteration 9: Cost 0.313169   
Iteration 10: Cost 0.302126   
Iteration 11: Cost 0.289829   
Iteration 12: Cost 0.282312   
Iteration 13: Cost 0.274791   
Iteration 14: Cost 0.270024   
Iteration 15: Cost 0.261128   
Iteration 16: Cost 0.252893   
Iteration 17: Cost 0.247832   
Iteration 18: Cost 0.242523   
Iteration 19: Cost 0.235945   
Iteration 20: Cost 0.233016   
Iteration 21: Cost 0.229360   
Iteration 22: Cost 0.224466   
Iteration 23: Cost 0.220540   
Iteration 24: Cost 0.214813   
Iteration 25: Cost 0.212168   
Iteration 26: Cost 0.209461   
Iteration 27: Cost 0.207210   
Iteration 28: Cost 0.206466   
Iteration 29: Cost 0.196358   
Iteration 30: Cost 0.196094   
Iteration 31: Cost 0.192961   
Iteration 32: Cost 0.192299   
Iteration 33: Cost 0.192700   
Iteration 34: Cost 0.187659   
Iteration 35: Cost 0.184100   
Iteration 36: Cost 0.179010   
Iteration 37: Cost 0.177062   
Iteration 38: Cost 0.177148   
Iteration 39: Cost 0.177655   
Iteration 40: Cost 0.173867   
Iteration 41: Cost 0.172179   
Iteration 42: Cost 0.166742   
Iteration 43: Cost 0.167785   
Iteration 44: Cost 0.166801   
Iteration 45: Cost 0.166730   
Iteration 46: Cost 0.165294   
Iteration 47: Cost 0.162029   
Iteration 48: Cost 0.158945   
Iteration 49: Cost 0.152756   
Iteration 50: Cost 0.155105   
Iteration 51: Cost 0.151653   
Iteration 52: Cost 0.152560   
Iteration 53: Cost 0.149534   
Iteration 54: Cost 0.147722   
Iteration 55: Cost 0.145228   
Iteration 56: Cost 0.144978   
Iteration 57: Cost 0.145525   
Iteration 58: Cost 0.146033   
Iteration 59: Cost 0.138744   
Iteration 60: Cost 0.138564   
Iteration 61: Cost 0.134543   
Iteration 62: Cost 0.135022   
Iteration 63: Cost 0.136132   
Iteration 64: Cost 0.133649   
Iteration 65: Cost 0.135422   
Iteration 66: Cost 0.132457   
Iteration 67: Cost 0.131425   
Iteration 68: Cost 0.127472   
Iteration 69: Cost 0.126407   
Iteration 70: Cost 0.124049   
Iteration 71: Cost 0.125929   
Iteration 72: Cost 0.125756   
Iteration 73: Cost 0.125909   
Iteration 74: Cost 0.120564   
Iteration 75: Cost 0.122907   
Iteration 76: Cost 0.118462   
Iteration 77: Cost 0.117452   
Iteration 78: Cost 0.116132   
Iteration 79: Cost 0.116260   
Iteration 80: Cost 0.117891   
Iteration 81: Cost 0.117906   
Iteration 82: Cost 0.114939   
Iteration 83: Cost 0.112823   
Iteration 84: Cost 0.110029   
Iteration 85: Cost 0.111297   
Iteration 86: Cost 0.113982   
Iteration 87: Cost 0.112701   
Iteration 88: Cost 0.113167   
Iteration 89: Cost 0.107517   
Iteration 90: Cost 0.107753   
Iteration 91: Cost 0.108252   
Iteration 92: Cost 0.105177   
Iteration 93: Cost 0.103488   
Iteration 94: Cost 0.104936   
Iteration 95: Cost 0.103193   
Iteration 96: Cost 0.102950   
Iteration 97: Cost 0.102582   
Iteration 98: Cost 0.099303   
Iteration 99: Cost 0.099824

<\_\_main\_\_.NeuralNetMLP at 0x18c70b36c10>

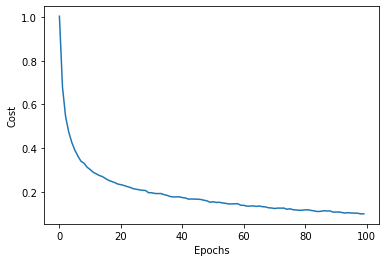
### Number of parameters

# check the total number of parameters  
print("shape of w\_h = ", nn.w\_h.shape)  
print("shape of b\_h = ", nn.b\_h.shape)  
print("shape of w\_o = ", nn.w\_o.shape)  
print("shape of b\_o = ", nn.b\_o.shape)  
print("total number of parameters = ", None)

shape of w\_h = (784, 100)  
shape of b\_h = (100,)  
shape of w\_o = (100, 10)  
shape of b\_o = (10,)  
total number of parameters = 79510

### Plot the cost change

import matplotlib.pyplot as plt  
  
# plot the loss - history  
None  
  
plt.ylabel('Cost')  
plt.xlabel('Epochs')  
plt.show()



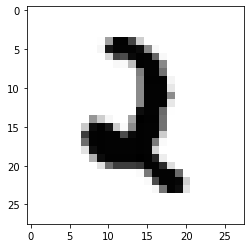
### Accuracy of the model

# training accuracy  
y\_train\_pred = None  
acc = None  
  
print('train 정확도: %.2f%%' % (acc \* 100))  
  
# test accuracy  
y\_test\_pred = None  
acc = None  
  
print('test 정확도: %.2f%%' % (acc \* 100))

train 정확도: 97.26%  
test 정확도: 95.55%

### Classification test

# show image 63000  
  
ex = X[63000]  
ex\_image = ex.reshape(28, 28)  
plt.imshow(ex\_image, cmap='Greys')  
plt.show()



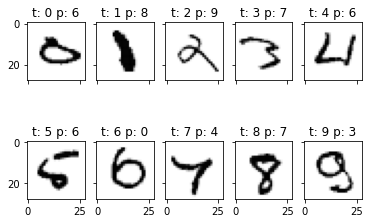
# classification - predict label of image 63000  
pred = None  
print("The image is number : ", pred[0])

The image is number : 2

### Examples of incorrect classification

# check the incorrect results  
mistake\_img = X\_test[y\_test != y\_test\_pred]  
true\_lab = y\_test[y\_test != y\_test\_pred]  
pred\_lab = y\_test\_pred[y\_test != y\_test\_pred]  
  
print("total %d images are incorrectly classified" % mistake\_img.shape[0])  
print("samples(t:true label, p:predicted label):")   
  
# show the misclassified image examples  
fig, ax = plt.subplots(nrows=2, ncols=5, sharex=True, sharey=True)  
ax = ax.flatten()  
for i in range(10):  
 img = mistake\_img[true\_lab == i][0].reshape(28, 28)  
 ax[i].imshow(img, cmap='Greys')  
 ax[i].set\_title('t: %d p: %d' % (true\_lab[true\_lab == i][0], pred\_lab[true\_lab == i][0]))  
  
plt.show()

total 445 images are incorrectly classified  
samples(t:true label, p:predicted label):



# 3. Multilayer perceptron using scikit learn

### Standardize data

from sklearn.preprocessing import StandardScaler  
  
# standardize data  
sc = StandardScaler()  
sc.fit(X\_train)  
X\_train\_std = sc.transform(X\_train)  
X\_test\_std = sc.transform(X\_test)

y\_train.shape

(60000,)

### Train MLPClassifier

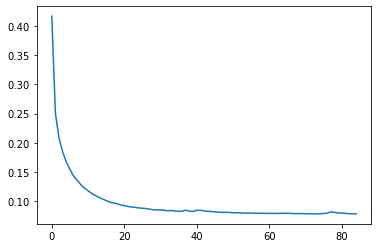
from sklearn.neural\_network import MLPClassifier  
  
# Multilayer perceptron from scikit learn with 1 hidden layer, 100 hidden units  
# stochastic gradient descent with batch size 100. learning rate = 0.01, L2 regularization parameter = 1e-1,   
mlp = None  
  
None

Iteration 1, loss = 0.41689056  
Iteration 2, loss = 0.25026337  
Iteration 3, loss = 0.20809725  
Iteration 4, loss = 0.18444219  
Iteration 5, loss = 0.16713313  
Iteration 6, loss = 0.15427066  
Iteration 7, loss = 0.14313532  
Iteration 8, loss = 0.13586182  
Iteration 9, loss = 0.12837024  
Iteration 10, loss = 0.12232528  
Iteration 11, loss = 0.11778416  
Iteration 12, loss = 0.11328087  
Iteration 13, loss = 0.10974739  
Iteration 14, loss = 0.10637758  
Iteration 15, loss = 0.10370598  
Iteration 16, loss = 0.10095428  
Iteration 17, loss = 0.09835926  
Iteration 18, loss = 0.09663742  
Iteration 19, loss = 0.09553043  
Iteration 20, loss = 0.09343205  
Iteration 21, loss = 0.09240505  
Iteration 22, loss = 0.09081009  
Iteration 23, loss = 0.09006995  
Iteration 24, loss = 0.08928516  
Iteration 25, loss = 0.08832028  
Iteration 26, loss = 0.08784616  
Iteration 27, loss = 0.08727906  
Iteration 28, loss = 0.08623176  
Iteration 29, loss = 0.08509248  
Iteration 30, loss = 0.08503916  
Iteration 31, loss = 0.08472869  
Iteration 32, loss = 0.08454073  
Iteration 33, loss = 0.08343270  
Iteration 34, loss = 0.08390219  
Iteration 35, loss = 0.08299714  
Iteration 36, loss = 0.08250653  
Iteration 37, loss = 0.08278885  
Iteration 38, loss = 0.08456384  
Iteration 39, loss = 0.08274158  
Iteration 40, loss = 0.08234007  
Iteration 41, loss = 0.08436894  
Iteration 42, loss = 0.08443855  
Iteration 43, loss = 0.08331360  
Iteration 44, loss = 0.08248065  
Iteration 45, loss = 0.08223552  
Iteration 46, loss = 0.08144645  
Iteration 47, loss = 0.08121637  
Iteration 48, loss = 0.08067893  
Iteration 49, loss = 0.08085716  
Iteration 50, loss = 0.08066155  
Iteration 51, loss = 0.08004372  
Iteration 52, loss = 0.08013447  
Iteration 53, loss = 0.07977591  
Iteration 54, loss = 0.07965036  
Iteration 55, loss = 0.07944386  
Iteration 56, loss = 0.07946776  
Iteration 57, loss = 0.07925961  
Iteration 58, loss = 0.07914650  
Iteration 59, loss = 0.07929907  
Iteration 60, loss = 0.07894741  
Iteration 61, loss = 0.07908353  
Iteration 62, loss = 0.07883906  
Iteration 63, loss = 0.07907499  
Iteration 64, loss = 0.07903927  
Iteration 65, loss = 0.07924408  
Iteration 66, loss = 0.07911657  
Iteration 67, loss = 0.07904405  
Iteration 68, loss = 0.07849868  
Iteration 69, loss = 0.07857384  
Iteration 70, loss = 0.07875606  
Iteration 71, loss = 0.07825730  
Iteration 72, loss = 0.07837397  
Iteration 73, loss = 0.07842687  
Iteration 74, loss = 0.07808086  
Iteration 75, loss = 0.07820606  
Iteration 76, loss = 0.07881078  
Iteration 77, loss = 0.07923678  
Iteration 78, loss = 0.08155256  
Iteration 79, loss = 0.08076600  
Iteration 80, loss = 0.07974082  
Iteration 81, loss = 0.07954405  
Iteration 82, loss = 0.07903510  
Iteration 83, loss = 0.07838956  
Iteration 84, loss = 0.07840940  
Iteration 85, loss = 0.07808378  
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.

MLPClassifier(alpha=0.1, batch\_size=100, hidden\_layer\_sizes=100,  
 learning\_rate\_init=0.01, max\_iter=100, random\_state=0,  
 solver='sgd', verbose=10)

### Plot the cost change

# plot the loss. use loss\_curve\_  
None  
plt.show()



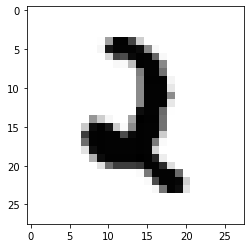
### Accuracy of the model

# Train and test accuracy  
acc = None  
print("Train accuracy : %.4f" % acc)  
acc = None  
print("Train accuracy : %.4f" % acc)

Train accuracy : 0.9978  
Train accuracy : 0.9780

### Classification test

# show image 63000  
ex = X[63000]  
ex\_image = ex.reshape(28, 28)  
plt.imshow(ex\_image, cmap='Greys')  
plt.show()



# classification - predict label of image 63000  
pred = None  
print("The image is number : ", pred[0])

The image is number : 2

### Number of parameters

# check the total number of parameters  
# parameters are mlp.coefs\_ and mlp.intercepts\_  
print("shape of w[0] ", mlp.coefs\_[0].shape)  
print("shape of b[0] ", mlp.intercepts\_[0].shape)  
print("shape of w[1] ", mlp.coefs\_[1].shape)  
print("shape of b[1] ", mlp.intercepts\_[1].shape)  
print("total number of parameters = ", None)

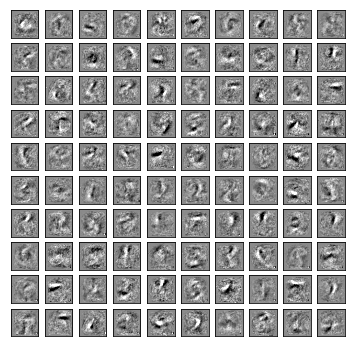
shape of w[0] (784, 100)  
shape of b[0] (100,)  
shape of w[1] (100, 10)  
shape of b[1] (10,)  
total number of parameters = 79510

### Visualize parameters

# weights of hidden layer  
mlp.coefs\_[0].shape

(784, 100)

# display weights of hidden layer (784, 100)  
  
fig, axes = plt.subplots(10, 10, figsize=(6, 6))  
plt.figsize = 20  
  
# use global min / max to ensure all weights are shown on the same scale  
vmin, vmax = mlp.coefs\_[0].min(), mlp.coefs\_[0].max()  
for coef, ax in zip(mlp.coefs\_[0].T, axes.ravel()):  
 ax.matshow(coef.reshape(28, 28), cmap=plt.cm.gray, vmin=.5 \* vmin, vmax=.5 \* vmax)  
 ax.set\_xticks(())  
 ax.set\_yticks(())  
  
plt.show()



# weights of output layer  
mlp.coefs\_[1].shape

(100, 10)

# display weights of output layer (100, 10)  
fig, axes = plt.subplots(1, 10)  
  
# use global min / max to ensure all weights are shown on the same scale  
vmin, vmax = mlp.coefs\_[1].min(), mlp.coefs\_[1].max()  
for coef, ax in zip(mlp.coefs\_[1].T, axes.ravel()):  
 ax.matshow(coef.reshape(10, 10), cmap=plt.cm.gray, vmin=.5 \* vmin, vmax=.5 \* vmax)  
 ax.set\_xticks(())  
 ax.set\_yticks(())  
  
plt.show()



# Quiz 1 : Learning Nonlinear Decision Boundary

## Train multilayer perceptron with the following moon dataset

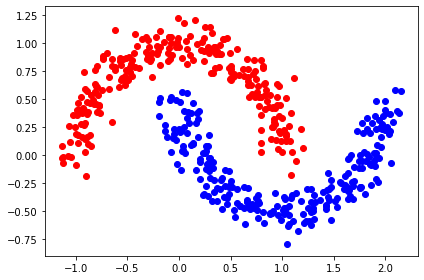
* Use the NeuraNetMLP with 100 neurons in hidden layer, learning rate 0.1, batch size 10

1. Train the model up to 1000 epochs
2. Plot the cost change during training
3. Show the decision boundary

* Repeat above using the MLPClassifier in scikit learn with 100 neurons each in 2 hidden layer, learning rate 0.01

### Dataset

from sklearn.datasets import make\_moons, make\_circles  
import matplotlib.pyplot as plt  
  
X, y = make\_moons(n\_samples=500, noise=0.1, random\_state=0)  
  
plt.scatter(X[y==0, 0], X[y==0, 1], c='r')  
plt.scatter(X[y==1, 0], X[y==1, 1], c='b')  
plt.tight\_layout()  
  
plt.show()



### Training MLP

nn = None  
None

number of input = 2  
number of hidden = 100  
number of output = 2  
Iteration 0: Cost 0.706873   
Iteration 1: Cost 0.526284   
Iteration 2: Cost 0.421231   
Iteration 3: Cost 0.390352   
Iteration 4: Cost 0.343257   
Iteration 5: Cost 0.333388   
Iteration 6: Cost 0.313195   
Iteration 7: Cost 0.334875   
Iteration 8: Cost 0.292701   
Iteration 9: Cost 0.300982   
Iteration 10: Cost 0.279281   
Iteration 11: Cost 0.305351   
Iteration 12: Cost 0.269271   
Iteration 13: Cost 0.291806   
Iteration 14: Cost 0.268628   
Iteration 15: Cost 0.318180   
Iteration 16: Cost 0.261627   
Iteration 17: Cost 0.266763   
Iteration 18: Cost 0.304177   
Iteration 19: Cost 0.263069   
Iteration 20: Cost 0.259577   
Iteration 21: Cost 0.261148   
Iteration 22: Cost 0.267490   
Iteration 23: Cost 0.273016   
Iteration 24: Cost 0.260017   
Iteration 25: Cost 0.258877   
Iteration 26: Cost 0.262971   
Iteration 27: Cost 0.279397   
Iteration 28: Cost 0.269703   
Iteration 29: Cost 0.265642   
Iteration 30: Cost 0.260187   
Iteration 31: Cost 0.261477   
Iteration 32: Cost 0.262154   
Iteration 33: Cost 0.275471   
Iteration 34: Cost 0.281458   
Iteration 35: Cost 0.272425   
Iteration 36: Cost 0.274063   
Iteration 37: Cost 0.262809   
Iteration 38: Cost 0.266768   
Iteration 39: Cost 0.302180   
Iteration 40: Cost 0.264002   
Iteration 41: Cost 0.265034   
Iteration 42: Cost 0.304505   
Iteration 43: Cost 0.258693   
Iteration 44: Cost 0.258986   
Iteration 45: Cost 0.275207   
Iteration 46: Cost 0.318041   
Iteration 47: Cost 0.280207   
Iteration 48: Cost 0.260853   
Iteration 49: Cost 0.258779   
Iteration 50: Cost 0.262389   
Iteration 51: Cost 0.258944   
Iteration 52: Cost 0.290632   
Iteration 53: Cost 0.260213   
Iteration 54: Cost 0.271557   
Iteration 55: Cost 0.260700   
Iteration 56: Cost 0.258799   
Iteration 57: Cost 0.297051   
Iteration 58: Cost 0.259502   
Iteration 59: Cost 0.270984   
Iteration 60: Cost 0.375649   
Iteration 61: Cost 0.278359   
Iteration 62: Cost 0.263691   
Iteration 63: Cost 0.259023   
Iteration 64: Cost 0.258945   
Iteration 65: Cost 0.258848   
Iteration 66: Cost 0.278315   
Iteration 67: Cost 0.259704   
Iteration 68: Cost 0.259863   
Iteration 69: Cost 0.269956   
Iteration 70: Cost 0.265670   
Iteration 71: Cost 0.301326   
Iteration 72: Cost 0.264019   
Iteration 73: Cost 0.259503   
Iteration 74: Cost 0.296939   
Iteration 75: Cost 0.261846   
Iteration 76: Cost 0.269587   
Iteration 77: Cost 0.259123   
Iteration 78: Cost 0.265916   
Iteration 79: Cost 0.265007   
Iteration 80: Cost 0.259419   
Iteration 81: Cost 0.259175   
Iteration 82: Cost 0.258975   
Iteration 83: Cost 0.260633   
Iteration 84: Cost 0.277096   
Iteration 85: Cost 0.259786   
Iteration 86: Cost 0.272731   
Iteration 87: Cost 0.259309   
Iteration 88: Cost 0.258837   
Iteration 89: Cost 0.263146   
Iteration 90: Cost 0.325005   
Iteration 91: Cost 0.259000   
Iteration 92: Cost 0.345236   
Iteration 93: Cost 0.263355   
Iteration 94: Cost 0.264182   
Iteration 95: Cost 0.264555   
Iteration 96: Cost 0.282428   
Iteration 97: Cost 0.260948   
Iteration 98: Cost 0.258774   
Iteration 99: Cost 0.269134   
Iteration 100: Cost 0.277830   
Iteration 101: Cost 0.265707   
Iteration 102: Cost 0.289504   
Iteration 103: Cost 0.259682   
Iteration 104: Cost 0.260587   
Iteration 105: Cost 0.263418   
Iteration 106: Cost 0.258716   
Iteration 107: Cost 0.261267   
Iteration 108: Cost 0.282943   
Iteration 109: Cost 0.304654   
Iteration 110: Cost 0.262807   
Iteration 111: Cost 0.276564   
Iteration 112: Cost 0.289401   
Iteration 113: Cost 0.265608   
Iteration 114: Cost 0.260973   
Iteration 115: Cost 0.317274   
Iteration 116: Cost 0.262646   
Iteration 117: Cost 0.258734   
Iteration 118: Cost 0.260500   
Iteration 119: Cost 0.264851   
Iteration 120: Cost 0.259202   
Iteration 121: Cost 0.260371   
Iteration 122: Cost 0.272064   
Iteration 123: Cost 0.275228   
Iteration 124: Cost 0.296590   
Iteration 125: Cost 0.267665   
Iteration 126: Cost 0.265999   
Iteration 127: Cost 0.280314   
Iteration 128: Cost 0.259058   
Iteration 129: Cost 0.263760   
Iteration 130: Cost 0.260889   
Iteration 131: Cost 0.271782   
Iteration 132: Cost 0.296544   
Iteration 133: Cost 0.260824   
Iteration 134: Cost 0.259352   
Iteration 135: Cost 0.294497   
Iteration 136: Cost 0.288526   
Iteration 137: Cost 0.262419   
Iteration 138: Cost 0.264171   
Iteration 139: Cost 0.259133   
Iteration 140: Cost 0.311772   
Iteration 141: Cost 0.277648   
Iteration 142: Cost 0.263352   
Iteration 143: Cost 0.258820   
Iteration 144: Cost 0.282825   
Iteration 145: Cost 0.320449   
Iteration 146: Cost 0.259438   
Iteration 147: Cost 0.259100   
Iteration 148: Cost 0.314372   
Iteration 149: Cost 0.304206   
Iteration 150: Cost 0.265676   
Iteration 151: Cost 0.262376   
Iteration 152: Cost 0.305925   
Iteration 153: Cost 0.266362   
Iteration 154: Cost 0.262651   
Iteration 155: Cost 0.270953   
Iteration 156: Cost 0.261996   
Iteration 157: Cost 0.259434   
Iteration 158: Cost 0.264916   
Iteration 159: Cost 0.270233   
Iteration 160: Cost 0.295330   
Iteration 161: Cost 0.261470   
Iteration 162: Cost 0.266090   
Iteration 163: Cost 0.259443   
Iteration 164: Cost 0.259117   
Iteration 165: Cost 0.276419   
Iteration 166: Cost 0.259906   
Iteration 167: Cost 0.262919   
Iteration 168: Cost 0.259287   
Iteration 169: Cost 0.265285   
Iteration 170: Cost 0.273788   
Iteration 171: Cost 0.271488   
Iteration 172: Cost 0.272785   
Iteration 173: Cost 0.271338   
Iteration 174: Cost 0.260084   
Iteration 175: Cost 0.260864   
Iteration 176: Cost 0.313072   
Iteration 177: Cost 0.259058   
Iteration 178: Cost 0.264667   
Iteration 179: Cost 0.260508   
Iteration 180: Cost 0.259751   
Iteration 181: Cost 0.273782   
Iteration 182: Cost 0.258994   
Iteration 183: Cost 0.287088   
Iteration 184: Cost 0.305909   
Iteration 185: Cost 0.258878   
Iteration 186: Cost 0.281019   
Iteration 187: Cost 0.262308   
Iteration 188: Cost 0.260499   
Iteration 189: Cost 0.260001   
Iteration 190: Cost 0.288294   
Iteration 191: Cost 0.259943   
Iteration 192: Cost 0.260890   
Iteration 193: Cost 0.266301   
Iteration 194: Cost 0.261007   
Iteration 195: Cost 0.258752   
Iteration 196: Cost 0.259882   
Iteration 197: Cost 0.259300   
Iteration 198: Cost 0.258777   
Iteration 199: Cost 0.280688   
Iteration 200: Cost 0.259120   
Iteration 201: Cost 0.259966   
Iteration 202: Cost 0.264405   
Iteration 203: Cost 0.260905   
Iteration 204: Cost 0.259043   
Iteration 205: Cost 0.258790   
Iteration 206: Cost 0.262043   
Iteration 207: Cost 0.258946   
Iteration 208: Cost 0.283967   
Iteration 209: Cost 0.284075   
Iteration 210: Cost 0.263461   
Iteration 211: Cost 0.279197   
Iteration 212: Cost 0.260206   
Iteration 213: Cost 0.260725   
Iteration 214: Cost 0.262738   
Iteration 215: Cost 0.258658   
Iteration 216: Cost 0.262467   
Iteration 217: Cost 0.259660   
Iteration 218: Cost 0.294912   
Iteration 219: Cost 0.258929   
Iteration 220: Cost 0.295099   
Iteration 221: Cost 0.259033   
Iteration 222: Cost 0.260843   
Iteration 223: Cost 0.259180   
Iteration 224: Cost 0.262177   
Iteration 225: Cost 0.267486   
Iteration 226: Cost 0.266231   
Iteration 227: Cost 0.262899   
Iteration 228: Cost 0.277190   
Iteration 229: Cost 0.287860   
Iteration 230: Cost 0.258808   
Iteration 231: Cost 0.267791   
Iteration 232: Cost 0.258817   
Iteration 233: Cost 0.258942   
Iteration 234: Cost 0.261115   
Iteration 235: Cost 0.264982   
Iteration 236: Cost 0.260635   
Iteration 237: Cost 0.286799   
Iteration 238: Cost 0.258803   
Iteration 239: Cost 0.265765   
Iteration 240: Cost 0.266306   
Iteration 241: Cost 0.259928   
Iteration 242: Cost 0.274468   
Iteration 243: Cost 0.286236   
Iteration 244: Cost 0.272141   
Iteration 245: Cost 0.259851   
Iteration 246: Cost 0.259123   
Iteration 247: Cost 0.264094   
Iteration 248: Cost 0.276206   
Iteration 249: Cost 0.258793   
Iteration 250: Cost 0.261459   
Iteration 251: Cost 0.264792   
Iteration 252: Cost 0.279749   
Iteration 253: Cost 0.268105   
Iteration 254: Cost 0.280098   
Iteration 255: Cost 0.289063   
Iteration 256: Cost 0.258813   
Iteration 257: Cost 0.275540

Iteration 258: Cost 0.261254   
Iteration 259: Cost 0.263607   
Iteration 260: Cost 0.264762   
Iteration 261: Cost 0.259347   
Iteration 262: Cost 0.287488   
Iteration 263: Cost 0.263379   
Iteration 264: Cost 0.259979   
Iteration 265: Cost 0.258861   
Iteration 266: Cost 0.259332   
Iteration 267: Cost 0.270629   
Iteration 268: Cost 0.264361   
Iteration 269: Cost 0.259346   
Iteration 270: Cost 0.262782   
Iteration 271: Cost 0.259106   
Iteration 272: Cost 0.265422   
Iteration 273: Cost 0.261720   
Iteration 274: Cost 0.268900   
Iteration 275: Cost 0.258846   
Iteration 276: Cost 0.284011   
Iteration 277: Cost 0.258679   
Iteration 278: Cost 0.261090   
Iteration 279: Cost 0.260848   
Iteration 280: Cost 0.267003   
Iteration 281: Cost 0.260256   
Iteration 282: Cost 0.265091   
Iteration 283: Cost 0.276448   
Iteration 284: Cost 0.262988   
Iteration 285: Cost 0.273146   
Iteration 286: Cost 0.259105   
Iteration 287: Cost 0.271235   
Iteration 288: Cost 0.258732   
Iteration 289: Cost 0.262613   
Iteration 290: Cost 0.259694   
Iteration 291: Cost 0.280337   
Iteration 292: Cost 0.258588   
Iteration 293: Cost 0.258578   
Iteration 294: Cost 0.259300   
Iteration 295: Cost 0.260794   
Iteration 296: Cost 0.299029   
Iteration 297: Cost 0.264980   
Iteration 298: Cost 0.264449   
Iteration 299: Cost 0.274110   
Iteration 300: Cost 0.258854   
Iteration 301: Cost 0.261530   
Iteration 302: Cost 0.260766   
Iteration 303: Cost 0.263611   
Iteration 304: Cost 0.264894   
Iteration 305: Cost 0.259657   
Iteration 306: Cost 0.258631   
Iteration 307: Cost 0.260950   
Iteration 308: Cost 0.273435   
Iteration 309: Cost 0.325518   
Iteration 310: Cost 0.259918   
Iteration 311: Cost 0.267976   
Iteration 312: Cost 0.262659   
Iteration 313: Cost 0.259820   
Iteration 314: Cost 0.262262   
Iteration 315: Cost 0.274735   
Iteration 316: Cost 0.266773   
Iteration 317: Cost 0.266096   
Iteration 318: Cost 0.258985   
Iteration 319: Cost 0.264807   
Iteration 320: Cost 0.268170   
Iteration 321: Cost 0.263310   
Iteration 322: Cost 0.273806   
Iteration 323: Cost 0.263198   
Iteration 324: Cost 0.264809   
Iteration 325: Cost 0.262076   
Iteration 326: Cost 0.265591   
Iteration 327: Cost 0.260746   
Iteration 328: Cost 0.259004   
Iteration 329: Cost 0.260491   
Iteration 330: Cost 0.262077   
Iteration 331: Cost 0.272650   
Iteration 332: Cost 0.258932   
Iteration 333: Cost 0.270361   
Iteration 334: Cost 0.258507   
Iteration 335: Cost 0.259385   
Iteration 336: Cost 0.260512   
Iteration 337: Cost 0.272471   
Iteration 338: Cost 0.277848   
Iteration 339: Cost 0.267938   
Iteration 340: Cost 0.263043   
Iteration 341: Cost 0.258797   
Iteration 342: Cost 0.258656   
Iteration 343: Cost 0.291962   
Iteration 344: Cost 0.258628   
Iteration 345: Cost 0.266935   
Iteration 346: Cost 0.258590   
Iteration 347: Cost 0.281816   
Iteration 348: Cost 0.279310   
Iteration 349: Cost 0.262632   
Iteration 350: Cost 0.258658   
Iteration 351: Cost 0.264723   
Iteration 352: Cost 0.282412   
Iteration 353: Cost 0.272721   
Iteration 354: Cost 0.294454   
Iteration 355: Cost 0.265236   
Iteration 356: Cost 0.263335   
Iteration 357: Cost 0.261179   
Iteration 358: Cost 0.259212   
Iteration 359: Cost 0.287259   
Iteration 360: Cost 0.263177   
Iteration 361: Cost 0.258711   
Iteration 362: Cost 0.263006   
Iteration 363: Cost 0.259243   
Iteration 364: Cost 0.259257   
Iteration 365: Cost 0.287783   
Iteration 366: Cost 0.258941   
Iteration 367: Cost 0.258991   
Iteration 368: Cost 0.264498   
Iteration 369: Cost 0.258471   
Iteration 370: Cost 0.261419   
Iteration 371: Cost 0.262258   
Iteration 372: Cost 0.268795   
Iteration 373: Cost 0.277912   
Iteration 374: Cost 0.266504   
Iteration 375: Cost 0.261356   
Iteration 376: Cost 0.307014   
Iteration 377: Cost 0.263428   
Iteration 378: Cost 0.259728   
Iteration 379: Cost 0.258971   
Iteration 380: Cost 0.266504   
Iteration 381: Cost 0.259588   
Iteration 382: Cost 0.263792   
Iteration 383: Cost 0.269846   
Iteration 384: Cost 0.260380   
Iteration 385: Cost 0.269684   
Iteration 386: Cost 0.259432   
Iteration 387: Cost 0.280679   
Iteration 388: Cost 0.258769   
Iteration 389: Cost 0.258975   
Iteration 390: Cost 0.258259   
Iteration 391: Cost 0.260336   
Iteration 392: Cost 0.286197   
Iteration 393: Cost 0.258287   
Iteration 394: Cost 0.258519   
Iteration 395: Cost 0.265583   
Iteration 396: Cost 0.258162   
Iteration 397: Cost 0.265189   
Iteration 398: Cost 0.258923   
Iteration 399: Cost 0.258074   
Iteration 400: Cost 0.268972   
Iteration 401: Cost 0.262083   
Iteration 402: Cost 0.262912   
Iteration 403: Cost 0.260241   
Iteration 404: Cost 0.257911   
Iteration 405: Cost 0.258507   
Iteration 406: Cost 0.263606   
Iteration 407: Cost 0.257840   
Iteration 408: Cost 0.265654   
Iteration 409: Cost 0.258479   
Iteration 410: Cost 0.279277   
Iteration 411: Cost 0.269215   
Iteration 412: Cost 0.295957   
Iteration 413: Cost 0.260219   
Iteration 414: Cost 0.259905   
Iteration 415: Cost 0.264058   
Iteration 416: Cost 0.270302   
Iteration 417: Cost 0.257444   
Iteration 418: Cost 0.287013   
Iteration 419: Cost 0.257007   
Iteration 420: Cost 0.281739   
Iteration 421: Cost 0.258593   
Iteration 422: Cost 0.262682   
Iteration 423: Cost 0.258991   
Iteration 424: Cost 0.268889   
Iteration 425: Cost 0.259546   
Iteration 426: Cost 0.262237   
Iteration 427: Cost 0.256497   
Iteration 428: Cost 0.255877   
Iteration 429: Cost 0.269761   
Iteration 430: Cost 0.255708   
Iteration 431: Cost 0.284601   
Iteration 432: Cost 0.255189   
Iteration 433: Cost 0.254734   
Iteration 434: Cost 0.254476   
Iteration 435: Cost 0.256901   
Iteration 436: Cost 0.260199   
Iteration 437: Cost 0.253375   
Iteration 438: Cost 0.253714   
Iteration 439: Cost 0.267827   
Iteration 440: Cost 0.260370   
Iteration 441: Cost 0.270153   
Iteration 442: Cost 0.254454   
Iteration 443: Cost 0.250269   
Iteration 444: Cost 0.263360   
Iteration 445: Cost 0.277650   
Iteration 446: Cost 0.311382   
Iteration 447: Cost 0.246963   
Iteration 448: Cost 0.247491   
Iteration 449: Cost 0.245789   
Iteration 450: Cost 0.244874   
Iteration 451: Cost 0.242320   
Iteration 452: Cost 0.280136   
Iteration 453: Cost 0.239217   
Iteration 454: Cost 0.249157   
Iteration 455: Cost 0.249845   
Iteration 456: Cost 0.233681   
Iteration 457: Cost 0.262088   
Iteration 458: Cost 0.230295   
Iteration 459: Cost 0.230241   
Iteration 460: Cost 0.230828   
Iteration 461: Cost 0.222796   
Iteration 462: Cost 0.219146   
Iteration 463: Cost 0.220251   
Iteration 464: Cost 0.213053   
Iteration 465: Cost 0.212527   
Iteration 466: Cost 0.239437   
Iteration 467: Cost 0.207512   
Iteration 468: Cost 0.200872   
Iteration 469: Cost 0.199104   
Iteration 470: Cost 0.205866   
Iteration 471: Cost 0.202389   
Iteration 472: Cost 0.185861   
Iteration 473: Cost 0.189744   
Iteration 474: Cost 0.192454   
Iteration 475: Cost 0.184443   
Iteration 476: Cost 0.184404   
Iteration 477: Cost 0.168979   
Iteration 478: Cost 0.184352   
Iteration 479: Cost 0.162014   
Iteration 480: Cost 0.158025   
Iteration 481: Cost 0.159114   
Iteration 482: Cost 0.153825   
Iteration 483: Cost 0.151357   
Iteration 484: Cost 0.186105   
Iteration 485: Cost 0.142006   
Iteration 486: Cost 0.140373   
Iteration 487: Cost 0.136007   
Iteration 488: Cost 0.140116   
Iteration 489: Cost 0.130975   
Iteration 490: Cost 0.128107   
Iteration 491: Cost 0.125424   
Iteration 492: Cost 0.123039   
Iteration 493: Cost 0.121893   
Iteration 494: Cost 0.129842   
Iteration 495: Cost 0.123509   
Iteration 496: Cost 0.118583   
Iteration 497: Cost 0.112985   
Iteration 498: Cost 0.109748   
Iteration 499: Cost 0.108378   
Iteration 500: Cost 0.107525   
Iteration 501: Cost 0.104898   
Iteration 502: Cost 0.105852   
Iteration 503: Cost 0.100888   
Iteration 504: Cost 0.099142   
Iteration 505: Cost 0.098524   
Iteration 506: Cost 0.098445   
Iteration 507: Cost 0.101310   
Iteration 508: Cost 0.094144   
Iteration 509: Cost 0.091601   
Iteration 510: Cost 0.097609   
Iteration 511: Cost 0.089672   
Iteration 512: Cost 0.087523   
Iteration 513: Cost 0.086338   
Iteration 514: Cost 0.085127   
Iteration 515: Cost 0.097012   
Iteration 516: Cost 0.082951   
Iteration 517: Cost 0.083783   
Iteration 518: Cost 0.080907   
Iteration 519: Cost 0.079540   
Iteration 520: Cost 0.078468   
Iteration 521: Cost 0.077783   
Iteration 522: Cost 0.076597   
Iteration 523: Cost 0.075881   
Iteration 524: Cost 0.074937

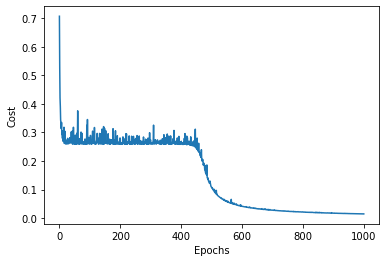
Iteration 525: Cost 0.074105   
Iteration 526: Cost 0.072943   
Iteration 527: Cost 0.072365   
Iteration 528: Cost 0.071708   
Iteration 529: Cost 0.070474   
Iteration 530: Cost 0.069787   
Iteration 531: Cost 0.068916   
Iteration 532: Cost 0.070324   
Iteration 533: Cost 0.067480   
Iteration 534: Cost 0.066996   
Iteration 535: Cost 0.067191   
Iteration 536: Cost 0.066526   
Iteration 537: Cost 0.065000   
Iteration 538: Cost 0.064517   
Iteration 539: Cost 0.063495   
Iteration 540: Cost 0.065866   
Iteration 541: Cost 0.062687   
Iteration 542: Cost 0.061657   
Iteration 543: Cost 0.061097   
Iteration 544: Cost 0.060570   
Iteration 545: Cost 0.062158   
Iteration 546: Cost 0.060028   
Iteration 547: Cost 0.060830   
Iteration 548: Cost 0.058541   
Iteration 549: Cost 0.058095   
Iteration 550: Cost 0.059528   
Iteration 551: Cost 0.057560   
Iteration 552: Cost 0.056468   
Iteration 553: Cost 0.059003   
Iteration 554: Cost 0.055523   
Iteration 555: Cost 0.055842   
Iteration 556: Cost 0.054731   
Iteration 557: Cost 0.054660   
Iteration 558: Cost 0.054168   
Iteration 559: Cost 0.053404   
Iteration 560: Cost 0.053613   
Iteration 561: Cost 0.052626   
Iteration 562: Cost 0.053179   
Iteration 563: Cost 0.052001   
Iteration 564: Cost 0.065626   
Iteration 565: Cost 0.051703   
Iteration 566: Cost 0.051307   
Iteration 567: Cost 0.050816   
Iteration 568: Cost 0.053637   
Iteration 569: Cost 0.049700   
Iteration 570: Cost 0.049259   
Iteration 571: Cost 0.049181   
Iteration 572: Cost 0.048589   
Iteration 573: Cost 0.052734   
Iteration 574: Cost 0.049014   
Iteration 575: Cost 0.047930   
Iteration 576: Cost 0.048791   
Iteration 577: Cost 0.047845   
Iteration 578: Cost 0.047225   
Iteration 579: Cost 0.047396   
Iteration 580: Cost 0.049237   
Iteration 581: Cost 0.047144   
Iteration 582: Cost 0.048257   
Iteration 583: Cost 0.045294   
Iteration 584: Cost 0.044972   
Iteration 585: Cost 0.044774   
Iteration 586: Cost 0.044436   
Iteration 587: Cost 0.044260   
Iteration 588: Cost 0.044612   
Iteration 589: Cost 0.043708   
Iteration 590: Cost 0.043675   
Iteration 591: Cost 0.043144   
Iteration 592: Cost 0.042937   
Iteration 593: Cost 0.042834   
Iteration 594: Cost 0.046677   
Iteration 595: Cost 0.042237   
Iteration 596: Cost 0.042307   
Iteration 597: Cost 0.044512   
Iteration 598: Cost 0.041523   
Iteration 599: Cost 0.041374   
Iteration 600: Cost 0.041417   
Iteration 601: Cost 0.040866   
Iteration 602: Cost 0.040897   
Iteration 603: Cost 0.040853   
Iteration 604: Cost 0.040363   
Iteration 605: Cost 0.039995   
Iteration 606: Cost 0.040028   
Iteration 607: Cost 0.039624   
Iteration 608: Cost 0.039417   
Iteration 609: Cost 0.039228   
Iteration 610: Cost 0.038966   
Iteration 611: Cost 0.038858   
Iteration 612: Cost 0.038774   
Iteration 613: Cost 0.040091   
Iteration 614: Cost 0.039649   
Iteration 615: Cost 0.038016   
Iteration 616: Cost 0.037934   
Iteration 617: Cost 0.037688   
Iteration 618: Cost 0.037510   
Iteration 619: Cost 0.037544   
Iteration 620: Cost 0.037108   
Iteration 621: Cost 0.036941   
Iteration 622: Cost 0.037442   
Iteration 623: Cost 0.036787   
Iteration 624: Cost 0.038578   
Iteration 625: Cost 0.038092   
Iteration 626: Cost 0.036116   
Iteration 627: Cost 0.035961   
Iteration 628: Cost 0.036611   
Iteration 629: Cost 0.035840   
Iteration 630: Cost 0.035709   
Iteration 631: Cost 0.035664   
Iteration 632: Cost 0.035343   
Iteration 633: Cost 0.035137   
Iteration 634: Cost 0.035190   
Iteration 635: Cost 0.035065   
Iteration 636: Cost 0.035612   
Iteration 637: Cost 0.035106   
Iteration 638: Cost 0.034273   
Iteration 639: Cost 0.035399   
Iteration 640: Cost 0.034238   
Iteration 641: Cost 0.033903   
Iteration 642: Cost 0.033742   
Iteration 643: Cost 0.034737   
Iteration 644: Cost 0.033488   
Iteration 645: Cost 0.033490   
Iteration 646: Cost 0.033761   
Iteration 647: Cost 0.033087   
Iteration 648: Cost 0.032902   
Iteration 649: Cost 0.032764   
Iteration 650: Cost 0.032637   
Iteration 651: Cost 0.032533   
Iteration 652: Cost 0.033065   
Iteration 653: Cost 0.032253   
Iteration 654: Cost 0.032124   
Iteration 655: Cost 0.032003   
Iteration 656: Cost 0.032058   
Iteration 657: Cost 0.032508   
Iteration 658: Cost 0.032335   
Iteration 659: Cost 0.031514   
Iteration 660: Cost 0.031426   
Iteration 661: Cost 0.031984   
Iteration 662: Cost 0.031720   
Iteration 663: Cost 0.031154   
Iteration 664: Cost 0.031376   
Iteration 665: Cost 0.033160   
Iteration 666: Cost 0.031382   
Iteration 667: Cost 0.030702   
Iteration 668: Cost 0.030510   
Iteration 669: Cost 0.030409   
Iteration 670: Cost 0.030529   
Iteration 671: Cost 0.030173   
Iteration 672: Cost 0.031599   
Iteration 673: Cost 0.030678   
Iteration 674: Cost 0.031381   
Iteration 675: Cost 0.029789   
Iteration 676: Cost 0.030970   
Iteration 677: Cost 0.031728   
Iteration 678: Cost 0.029605   
Iteration 679: Cost 0.029386   
Iteration 680: Cost 0.029411   
Iteration 681: Cost 0.029129   
Iteration 682: Cost 0.029152   
Iteration 683: Cost 0.029483   
Iteration 684: Cost 0.030322   
Iteration 685: Cost 0.028982   
Iteration 686: Cost 0.030127   
Iteration 687: Cost 0.028881   
Iteration 688: Cost 0.028587   
Iteration 689: Cost 0.028444   
Iteration 690: Cost 0.029079   
Iteration 691: Cost 0.029948   
Iteration 692: Cost 0.028407   
Iteration 693: Cost 0.028235   
Iteration 694: Cost 0.027923   
Iteration 695: Cost 0.027852   
Iteration 696: Cost 0.027923   
Iteration 697: Cost 0.027631   
Iteration 698: Cost 0.027667   
Iteration 699: Cost 0.027623   
Iteration 700: Cost 0.028020   
Iteration 701: Cost 0.028030   
Iteration 702: Cost 0.027554   
Iteration 703: Cost 0.027162   
Iteration 704: Cost 0.027087   
Iteration 705: Cost 0.027076   
Iteration 706: Cost 0.029063   
Iteration 707: Cost 0.027821   
Iteration 708: Cost 0.026705   
Iteration 709: Cost 0.026700   
Iteration 710: Cost 0.026544   
Iteration 711: Cost 0.026619   
Iteration 712: Cost 0.026394   
Iteration 713: Cost 0.026689   
Iteration 714: Cost 0.026241   
Iteration 715: Cost 0.026142   
Iteration 716: Cost 0.026331   
Iteration 717: Cost 0.026253   
Iteration 718: Cost 0.026225   
Iteration 719: Cost 0.026483   
Iteration 720: Cost 0.025794   
Iteration 721: Cost 0.025686   
Iteration 722: Cost 0.025812   
Iteration 723: Cost 0.025765   
Iteration 724: Cost 0.025691   
Iteration 725: Cost 0.025418   
Iteration 726: Cost 0.025595   
Iteration 727: Cost 0.025238   
Iteration 728: Cost 0.025176   
Iteration 729: Cost 0.025116   
Iteration 730: Cost 0.025091   
Iteration 731: Cost 0.025103   
Iteration 732: Cost 0.024996   
Iteration 733: Cost 0.025366   
Iteration 734: Cost 0.024792   
Iteration 735: Cost 0.024750   
Iteration 736: Cost 0.024740   
Iteration 737: Cost 0.025469   
Iteration 738: Cost 0.024825   
Iteration 739: Cost 0.024727   
Iteration 740: Cost 0.024386   
Iteration 741: Cost 0.024931   
Iteration 742: Cost 0.024814   
Iteration 743: Cost 0.024332   
Iteration 744: Cost 0.024123   
Iteration 745: Cost 0.024177   
Iteration 746: Cost 0.023943   
Iteration 747: Cost 0.023881   
Iteration 748: Cost 0.023894   
Iteration 749: Cost 0.023789   
Iteration 750: Cost 0.024458   
Iteration 751: Cost 0.024439   
Iteration 752: Cost 0.023570   
Iteration 753: Cost 0.023588   
Iteration 754: Cost 0.023442   
Iteration 755: Cost 0.023400   
Iteration 756: Cost 0.023402   
Iteration 757: Cost 0.023306   
Iteration 758: Cost 0.023206   
Iteration 759: Cost 0.023158   
Iteration 760: Cost 0.023090   
Iteration 761: Cost 0.023317   
Iteration 762: Cost 0.022971   
Iteration 763: Cost 0.023033   
Iteration 764: Cost 0.022904   
Iteration 765: Cost 0.022845   
Iteration 766: Cost 0.023067   
Iteration 767: Cost 0.022757   
Iteration 768: Cost 0.022743   
Iteration 769: Cost 0.023469   
Iteration 770: Cost 0.022536   
Iteration 771: Cost 0.022450   
Iteration 772: Cost 0.022640   
Iteration 773: Cost 0.022379   
Iteration 774: Cost 0.022295   
Iteration 775: Cost 0.022763   
Iteration 776: Cost 0.022172   
Iteration 777: Cost 0.022209   
Iteration 778: Cost 0.022154   
Iteration 779: Cost 0.022030   
Iteration 780: Cost 0.021967   
Iteration 781: Cost 0.021934   
Iteration 782: Cost 0.022237   
Iteration 783: Cost 0.022549   
Iteration 784: Cost 0.021957   
Iteration 785: Cost 0.021797

Iteration 786: Cost 0.021786   
Iteration 787: Cost 0.021652   
Iteration 788: Cost 0.021729   
Iteration 789: Cost 0.021818   
Iteration 790: Cost 0.021461   
Iteration 791: Cost 0.021408   
Iteration 792: Cost 0.021356   
Iteration 793: Cost 0.021619   
Iteration 794: Cost 0.021240   
Iteration 795: Cost 0.021208   
Iteration 796: Cost 0.021264   
Iteration 797: Cost 0.021096   
Iteration 798: Cost 0.021072   
Iteration 799: Cost 0.021787   
Iteration 800: Cost 0.020959   
Iteration 801: Cost 0.020999   
Iteration 802: Cost 0.021144   
Iteration 803: Cost 0.020855   
Iteration 804: Cost 0.021009   
Iteration 805: Cost 0.020747   
Iteration 806: Cost 0.020666   
Iteration 807: Cost 0.021135   
Iteration 808: Cost 0.020581   
Iteration 809: Cost 0.020532   
Iteration 810: Cost 0.020768   
Iteration 811: Cost 0.020546   
Iteration 812: Cost 0.020510   
Iteration 813: Cost 0.020465   
Iteration 814: Cost 0.020588   
Iteration 815: Cost 0.020344   
Iteration 816: Cost 0.020212   
Iteration 817: Cost 0.020165   
Iteration 818: Cost 0.020205   
Iteration 819: Cost 0.020149   
Iteration 820: Cost 0.020059   
Iteration 821: Cost 0.020070   
Iteration 822: Cost 0.020393   
Iteration 823: Cost 0.019982   
Iteration 824: Cost 0.020158   
Iteration 825: Cost 0.019838   
Iteration 826: Cost 0.020244   
Iteration 827: Cost 0.019827   
Iteration 828: Cost 0.019792   
Iteration 829: Cost 0.020359   
Iteration 830: Cost 0.020101   
Iteration 831: Cost 0.019570   
Iteration 832: Cost 0.019527   
Iteration 833: Cost 0.019577   
Iteration 834: Cost 0.019518   
Iteration 835: Cost 0.019442   
Iteration 836: Cost 0.019448   
Iteration 837: Cost 0.019352   
Iteration 838: Cost 0.019287   
Iteration 839: Cost 0.019301   
Iteration 840: Cost 0.019241   
Iteration 841: Cost 0.019977   
Iteration 842: Cost 0.019181   
Iteration 843: Cost 0.020050   
Iteration 844: Cost 0.019495   
Iteration 845: Cost 0.019020   
Iteration 846: Cost 0.019171   
Iteration 847: Cost 0.019672   
Iteration 848: Cost 0.018889   
Iteration 849: Cost 0.019561   
Iteration 850: Cost 0.018870   
Iteration 851: Cost 0.019245   
Iteration 852: Cost 0.018741   
Iteration 853: Cost 0.019069   
Iteration 854: Cost 0.020066   
Iteration 855: Cost 0.019096   
Iteration 856: Cost 0.018912   
Iteration 857: Cost 0.018566   
Iteration 858: Cost 0.018527   
Iteration 859: Cost 0.018598   
Iteration 860: Cost 0.018485   
Iteration 861: Cost 0.019256   
Iteration 862: Cost 0.018513   
Iteration 863: Cost 0.018346   
Iteration 864: Cost 0.018422   
Iteration 865: Cost 0.018269   
Iteration 866: Cost 0.018650   
Iteration 867: Cost 0.018199   
Iteration 868: Cost 0.018168   
Iteration 869: Cost 0.018324   
Iteration 870: Cost 0.018127   
Iteration 871: Cost 0.018082   
Iteration 872: Cost 0.018116   
Iteration 873: Cost 0.018011   
Iteration 874: Cost 0.017954   
Iteration 875: Cost 0.017987   
Iteration 876: Cost 0.018011   
Iteration 877: Cost 0.017855   
Iteration 878: Cost 0.017831   
Iteration 879: Cost 0.017802   
Iteration 880: Cost 0.018269   
Iteration 881: Cost 0.018060   
Iteration 882: Cost 0.017694   
Iteration 883: Cost 0.017820   
Iteration 884: Cost 0.017708   
Iteration 885: Cost 0.017855   
Iteration 886: Cost 0.017580   
Iteration 887: Cost 0.017930   
Iteration 888: Cost 0.017669   
Iteration 889: Cost 0.017735   
Iteration 890: Cost 0.017458   
Iteration 891: Cost 0.018055   
Iteration 892: Cost 0.017370   
Iteration 893: Cost 0.018005   
Iteration 894: Cost 0.018890   
Iteration 895: Cost 0.017553   
Iteration 896: Cost 0.017421   
Iteration 897: Cost 0.017258   
Iteration 898: Cost 0.017196   
Iteration 899: Cost 0.017155   
Iteration 900: Cost 0.017131   
Iteration 901: Cost 0.017335   
Iteration 902: Cost 0.017119   
Iteration 903: Cost 0.017539   
Iteration 904: Cost 0.016993   
Iteration 905: Cost 0.017025   
Iteration 906: Cost 0.016941   
Iteration 907: Cost 0.017015   
Iteration 908: Cost 0.016881   
Iteration 909: Cost 0.016928   
Iteration 910: Cost 0.016819   
Iteration 911: Cost 0.017021   
Iteration 912: Cost 0.016793   
Iteration 913: Cost 0.016811   
Iteration 914: Cost 0.016699   
Iteration 915: Cost 0.016712   
Iteration 916: Cost 0.016716   
Iteration 917: Cost 0.016634   
Iteration 918: Cost 0.016841   
Iteration 919: Cost 0.016591   
Iteration 920: Cost 0.016801   
Iteration 921: Cost 0.016611   
Iteration 922: Cost 0.016487   
Iteration 923: Cost 0.016464   
Iteration 924: Cost 0.016490   
Iteration 925: Cost 0.016419   
Iteration 926: Cost 0.016391   
Iteration 927: Cost 0.016348   
Iteration 928: Cost 0.016304   
Iteration 929: Cost 0.016355   
Iteration 930: Cost 0.016376   
Iteration 931: Cost 0.016291   
Iteration 932: Cost 0.016197   
Iteration 933: Cost 0.016236   
Iteration 934: Cost 0.016164   
Iteration 935: Cost 0.016481   
Iteration 936: Cost 0.016097   
Iteration 937: Cost 0.016074   
Iteration 938: Cost 0.016384   
Iteration 939: Cost 0.016101   
Iteration 940: Cost 0.016028   
Iteration 941: Cost 0.016096   
Iteration 942: Cost 0.016318   
Iteration 943: Cost 0.016135   
Iteration 944: Cost 0.016100   
Iteration 945: Cost 0.015856   
Iteration 946: Cost 0.015932   
Iteration 947: Cost 0.015820   
Iteration 948: Cost 0.015837   
Iteration 949: Cost 0.015767   
Iteration 950: Cost 0.015942   
Iteration 951: Cost 0.015804   
Iteration 952: Cost 0.015689   
Iteration 953: Cost 0.015809   
Iteration 954: Cost 0.015702   
Iteration 955: Cost 0.015607   
Iteration 956: Cost 0.015647   
Iteration 957: Cost 0.015562   
Iteration 958: Cost 0.015546   
Iteration 959: Cost 0.015517   
Iteration 960: Cost 0.015558   
Iteration 961: Cost 0.015602   
Iteration 962: Cost 0.015527   
Iteration 963: Cost 0.015411   
Iteration 964: Cost 0.015656   
Iteration 965: Cost 0.015535   
Iteration 966: Cost 0.015408   
Iteration 967: Cost 0.015376   
Iteration 968: Cost 0.015400   
Iteration 969: Cost 0.015279   
Iteration 970: Cost 0.015245   
Iteration 971: Cost 0.015353   
Iteration 972: Cost 0.015197   
Iteration 973: Cost 0.015288   
Iteration 974: Cost 0.015151   
Iteration 975: Cost 0.015430   
Iteration 976: Cost 0.015139   
Iteration 977: Cost 0.015339   
Iteration 978: Cost 0.015125   
Iteration 979: Cost 0.015083   
Iteration 980: Cost 0.015065   
Iteration 981: Cost 0.015058   
Iteration 982: Cost 0.014985   
Iteration 983: Cost 0.014965   
Iteration 984: Cost 0.015049   
Iteration 985: Cost 0.014954   
Iteration 986: Cost 0.014934   
Iteration 987: Cost 0.015049   
Iteration 988: Cost 0.014953   
Iteration 989: Cost 0.014870   
Iteration 990: Cost 0.014856   
Iteration 991: Cost 0.014804   
Iteration 992: Cost 0.014857   
Iteration 993: Cost 0.014733   
Iteration 994: Cost 0.014781   
Iteration 995: Cost 0.014704   
Iteration 996: Cost 0.014677   
Iteration 997: Cost 0.014677   
Iteration 998: Cost 0.014920   
Iteration 999: Cost 0.014600

<\_\_main\_\_.NeuralNetMLP at 0x18c0002d190>

### Plot the cost change

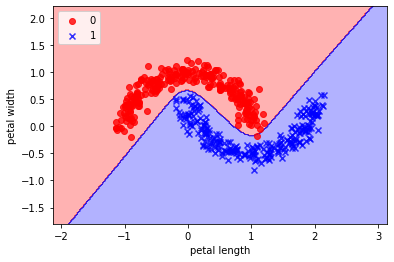
import matplotlib.pyplot as plt  
  
None  
  
plt.ylabel('Cost')  
plt.xlabel('Epochs')  
plt.show()



### Plot the decision boundary

# A function for plotting decision regions  
from matplotlib.colors import ListedColormap  
import matplotlib.pyplot as plt  
  
def plot\_decision\_regions(X, y, classifier, resolution=0.02):  
  
 # setup marker generator and color map  
 markers = ('o', 'x', 's', '^', 'v')  
 colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')  
 cmap = ListedColormap(colors[:len(np.unique(y))])  
  
 # plot the decision surface  
 x1\_min, x1\_max = X[:, 0].min() - 1, X[:, 0].max() + 1  
 x2\_min, x2\_max = X[:, 1].min() - 1, X[:, 1].max() + 1  
 xx1, xx2 = np.meshgrid(np.arange(x1\_min, x1\_max, resolution),  
 np.arange(x2\_min, x2\_max, resolution))  
 Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)  
 Z = Z.reshape(xx1.shape)  
 plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)  
 plt.xlim(xx1.min(), xx1.max())  
 plt.ylim(xx2.min(), xx2.max())  
  
 # plot class samples  
 for idx, cl in enumerate(np.unique(y)):  
 plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],  
 alpha=0.8, c=colors[idx], marker=markers[idx], label=cl)

# plot decision boundary of the model   
None  
  
plt.xlabel('petal length')  
plt.ylabel('petal width')  
plt.legend(loc='upper left')  
plt.show()



### Training scikit learn MLPClassifier

from sklearn.neural\_network import MLPClassifier  
  
mlp = None  
None

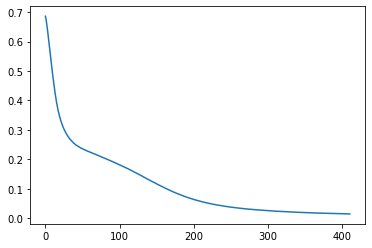
Iteration 1, loss = 0.68638631  
Iteration 2, loss = 0.67412337  
Iteration 3, loss = 0.65612239  
Iteration 4, loss = 0.63577343  
Iteration 5, loss = 0.61363586  
Iteration 6, loss = 0.59085751  
Iteration 7, loss = 0.56849187  
Iteration 8, loss = 0.54620165  
Iteration 9, loss = 0.52419985  
Iteration 10, loss = 0.50303243  
Iteration 11, loss = 0.48230069  
Iteration 12, loss = 0.46261200  
Iteration 13, loss = 0.44373352  
Iteration 14, loss = 0.42555398  
Iteration 15, loss = 0.40917172  
Iteration 16, loss = 0.39356959  
Iteration 17, loss = 0.38009799  
Iteration 18, loss = 0.36763780  
Iteration 19, loss = 0.35643458  
Iteration 20, loss = 0.34636153  
Iteration 21, loss = 0.33746254  
Iteration 22, loss = 0.32889278  
Iteration 23, loss = 0.32147256  
Iteration 24, loss = 0.31485476  
Iteration 25, loss = 0.30846778  
Iteration 26, loss = 0.30254339  
Iteration 27, loss = 0.29736602  
Iteration 28, loss = 0.29234218  
Iteration 29, loss = 0.28764119  
Iteration 30, loss = 0.28342652  
Iteration 31, loss = 0.27922445  
Iteration 32, loss = 0.27557925  
Iteration 33, loss = 0.27202908  
Iteration 34, loss = 0.26861374  
Iteration 35, loss = 0.26581648  
Iteration 36, loss = 0.26257948  
Iteration 37, loss = 0.25995294  
Iteration 38, loss = 0.25736855  
Iteration 39, loss = 0.25507705  
Iteration 40, loss = 0.25280815  
Iteration 41, loss = 0.25095811  
Iteration 42, loss = 0.24881259  
Iteration 43, loss = 0.24721144  
Iteration 44, loss = 0.24540117  
Iteration 45, loss = 0.24382689  
Iteration 46, loss = 0.24214907  
Iteration 47, loss = 0.24069184  
Iteration 48, loss = 0.23926855  
Iteration 49, loss = 0.23790789  
Iteration 50, loss = 0.23704855  
Iteration 51, loss = 0.23538617  
Iteration 52, loss = 0.23411499  
Iteration 53, loss = 0.23309814  
Iteration 54, loss = 0.23201898  
Iteration 55, loss = 0.23062638  
Iteration 56, loss = 0.22953895  
Iteration 57, loss = 0.22847354  
Iteration 58, loss = 0.22740476  
Iteration 59, loss = 0.22633879  
Iteration 60, loss = 0.22535362  
Iteration 61, loss = 0.22432220  
Iteration 62, loss = 0.22341290  
Iteration 63, loss = 0.22231726  
Iteration 64, loss = 0.22115787  
Iteration 65, loss = 0.22013811  
Iteration 66, loss = 0.21922026  
Iteration 67, loss = 0.21822424  
Iteration 68, loss = 0.21717236  
Iteration 69, loss = 0.21600266  
Iteration 70, loss = 0.21499357  
Iteration 71, loss = 0.21400768  
Iteration 72, loss = 0.21280003  
Iteration 73, loss = 0.21185311  
Iteration 74, loss = 0.21080436  
Iteration 75, loss = 0.20969392  
Iteration 76, loss = 0.20870348  
Iteration 77, loss = 0.20756914  
Iteration 78, loss = 0.20639311  
Iteration 79, loss = 0.20534639  
Iteration 80, loss = 0.20461181  
Iteration 81, loss = 0.20352266  
Iteration 82, loss = 0.20239580  
Iteration 83, loss = 0.20117306  
Iteration 84, loss = 0.20010732  
Iteration 85, loss = 0.19931037  
Iteration 86, loss = 0.19793899  
Iteration 87, loss = 0.19676383  
Iteration 88, loss = 0.19575672  
Iteration 89, loss = 0.19462754  
Iteration 90, loss = 0.19365514  
Iteration 91, loss = 0.19247297  
Iteration 92, loss = 0.19132385  
Iteration 93, loss = 0.19026829  
Iteration 94, loss = 0.18914580  
Iteration 95, loss = 0.18799324  
Iteration 96, loss = 0.18684970  
Iteration 97, loss = 0.18567945  
Iteration 98, loss = 0.18452243  
Iteration 99, loss = 0.18350683  
Iteration 100, loss = 0.18223700  
Iteration 101, loss = 0.18117282  
Iteration 102, loss = 0.18019626  
Iteration 103, loss = 0.17879161  
Iteration 104, loss = 0.17747119  
Iteration 105, loss = 0.17636235  
Iteration 106, loss = 0.17514326  
Iteration 107, loss = 0.17414034  
Iteration 108, loss = 0.17289979  
Iteration 109, loss = 0.17170881  
Iteration 110, loss = 0.17038112  
Iteration 111, loss = 0.16904015  
Iteration 112, loss = 0.16785967  
Iteration 113, loss = 0.16675244  
Iteration 114, loss = 0.16544796  
Iteration 115, loss = 0.16411884  
Iteration 116, loss = 0.16265157  
Iteration 117, loss = 0.16137447  
Iteration 118, loss = 0.16007239  
Iteration 119, loss = 0.15872114  
Iteration 120, loss = 0.15747183  
Iteration 121, loss = 0.15629544  
Iteration 122, loss = 0.15508533  
Iteration 123, loss = 0.15370369  
Iteration 124, loss = 0.15247450  
Iteration 125, loss = 0.15080697  
Iteration 126, loss = 0.14969092  
Iteration 127, loss = 0.14857938  
Iteration 128, loss = 0.14700100  
Iteration 129, loss = 0.14585761  
Iteration 130, loss = 0.14422801  
Iteration 131, loss = 0.14316262  
Iteration 132, loss = 0.14142121  
Iteration 133, loss = 0.14009959  
Iteration 134, loss = 0.13889005  
Iteration 135, loss = 0.13753491  
Iteration 136, loss = 0.13620686  
Iteration 137, loss = 0.13474636  
Iteration 138, loss = 0.13323255  
Iteration 139, loss = 0.13196816  
Iteration 140, loss = 0.13090604  
Iteration 141, loss = 0.12956850  
Iteration 142, loss = 0.12795648  
Iteration 143, loss = 0.12656455  
Iteration 144, loss = 0.12545713  
Iteration 145, loss = 0.12411798  
Iteration 146, loss = 0.12264839  
Iteration 147, loss = 0.12149154  
Iteration 148, loss = 0.12048148  
Iteration 149, loss = 0.11876588  
Iteration 150, loss = 0.11749016  
Iteration 151, loss = 0.11629327  
Iteration 152, loss = 0.11477164  
Iteration 153, loss = 0.11354168  
Iteration 154, loss = 0.11225203  
Iteration 155, loss = 0.11093578  
Iteration 156, loss = 0.10963942  
Iteration 157, loss = 0.10838388  
Iteration 158, loss = 0.10710185  
Iteration 159, loss = 0.10589132  
Iteration 160, loss = 0.10465231  
Iteration 161, loss = 0.10340508  
Iteration 162, loss = 0.10217480  
Iteration 163, loss = 0.10099641  
Iteration 164, loss = 0.10009673  
Iteration 165, loss = 0.09856364  
Iteration 166, loss = 0.09751288  
Iteration 167, loss = 0.09628563  
Iteration 168, loss = 0.09512673  
Iteration 169, loss = 0.09414918  
Iteration 170, loss = 0.09272265  
Iteration 171, loss = 0.09147594  
Iteration 172, loss = 0.09061375  
Iteration 173, loss = 0.08947331  
Iteration 174, loss = 0.08814675  
Iteration 175, loss = 0.08702357  
Iteration 176, loss = 0.08645326  
Iteration 177, loss = 0.08505745  
Iteration 178, loss = 0.08391044  
Iteration 179, loss = 0.08290054  
Iteration 180, loss = 0.08181692  
Iteration 181, loss = 0.08091713  
Iteration 182, loss = 0.07992400  
Iteration 183, loss = 0.07888516  
Iteration 184, loss = 0.07782672  
Iteration 185, loss = 0.07696134  
Iteration 186, loss = 0.07592904  
Iteration 187, loss = 0.07498496  
Iteration 188, loss = 0.07408467  
Iteration 189, loss = 0.07308582  
Iteration 190, loss = 0.07219311  
Iteration 191, loss = 0.07149010  
Iteration 192, loss = 0.07054244  
Iteration 193, loss = 0.06973417  
Iteration 194, loss = 0.06888374  
Iteration 195, loss = 0.06793995  
Iteration 196, loss = 0.06717581  
Iteration 197, loss = 0.06638085  
Iteration 198, loss = 0.06573251  
Iteration 199, loss = 0.06481976  
Iteration 200, loss = 0.06399488  
Iteration 201, loss = 0.06313418  
Iteration 202, loss = 0.06243298  
Iteration 203, loss = 0.06173262  
Iteration 204, loss = 0.06103378  
Iteration 205, loss = 0.06033204  
Iteration 206, loss = 0.05962154  
Iteration 207, loss = 0.05903157  
Iteration 208, loss = 0.05818525  
Iteration 209, loss = 0.05767059  
Iteration 210, loss = 0.05708616  
Iteration 211, loss = 0.05630769  
Iteration 212, loss = 0.05572759  
Iteration 213, loss = 0.05490518  
Iteration 214, loss = 0.05449050  
Iteration 215, loss = 0.05371767  
Iteration 216, loss = 0.05309928  
Iteration 217, loss = 0.05250039  
Iteration 218, loss = 0.05191411  
Iteration 219, loss = 0.05134550  
Iteration 220, loss = 0.05073237  
Iteration 221, loss = 0.05019346  
Iteration 222, loss = 0.04969485  
Iteration 223, loss = 0.04913338  
Iteration 224, loss = 0.04853824  
Iteration 225, loss = 0.04801126  
Iteration 226, loss = 0.04779308  
Iteration 227, loss = 0.04713466  
Iteration 228, loss = 0.04657168  
Iteration 229, loss = 0.04605879  
Iteration 230, loss = 0.04552508  
Iteration 231, loss = 0.04511532  
Iteration 232, loss = 0.04463795  
Iteration 233, loss = 0.04421282  
Iteration 234, loss = 0.04372316  
Iteration 235, loss = 0.04325297  
Iteration 236, loss = 0.04283034  
Iteration 237, loss = 0.04238022  
Iteration 238, loss = 0.04201714  
Iteration 239, loss = 0.04161297  
Iteration 240, loss = 0.04121922  
Iteration 241, loss = 0.04081166  
Iteration 242, loss = 0.04038459  
Iteration 243, loss = 0.03998973  
Iteration 244, loss = 0.03962900  
Iteration 245, loss = 0.03933237  
Iteration 246, loss = 0.03891012  
Iteration 247, loss = 0.03857667  
Iteration 248, loss = 0.03818380  
Iteration 249, loss = 0.03784015  
Iteration 250, loss = 0.03749573  
Iteration 251, loss = 0.03723423  
Iteration 252, loss = 0.03684477  
Iteration 253, loss = 0.03652686  
Iteration 254, loss = 0.03614207  
Iteration 255, loss = 0.03589778  
Iteration 256, loss = 0.03569958  
Iteration 257, loss = 0.03550023  
Iteration 258, loss = 0.03494815  
Iteration 259, loss = 0.03461801  
Iteration 260, loss = 0.03429847  
Iteration 261, loss = 0.03403393  
Iteration 262, loss = 0.03379388  
Iteration 263, loss = 0.03347087  
Iteration 264, loss = 0.03323785  
Iteration 265, loss = 0.03292847  
Iteration 266, loss = 0.03267669  
Iteration 267, loss = 0.03236858  
Iteration 268, loss = 0.03209024  
Iteration 269, loss = 0.03181918  
Iteration 270, loss = 0.03163016  
Iteration 271, loss = 0.03131366  
Iteration 272, loss = 0.03112260  
Iteration 273, loss = 0.03080689  
Iteration 274, loss = 0.03062426  
Iteration 275, loss = 0.03044824  
Iteration 276, loss = 0.03008123  
Iteration 277, loss = 0.02996450  
Iteration 278, loss = 0.02968293  
Iteration 279, loss = 0.02939249  
Iteration 280, loss = 0.02920667  
Iteration 281, loss = 0.02900543  
Iteration 282, loss = 0.02879748  
Iteration 283, loss = 0.02865532  
Iteration 284, loss = 0.02833979  
Iteration 285, loss = 0.02809932  
Iteration 286, loss = 0.02800238  
Iteration 287, loss = 0.02780984  
Iteration 288, loss = 0.02757085  
Iteration 289, loss = 0.02740278  
Iteration 290, loss = 0.02723293  
Iteration 291, loss = 0.02700705  
Iteration 292, loss = 0.02673227  
Iteration 293, loss = 0.02656344  
Iteration 294, loss = 0.02640086  
Iteration 295, loss = 0.02617981  
Iteration 296, loss = 0.02596889  
Iteration 297, loss = 0.02581035  
Iteration 298, loss = 0.02561031  
Iteration 299, loss = 0.02544506  
Iteration 300, loss = 0.02532313  
Iteration 301, loss = 0.02514990  
Iteration 302, loss = 0.02503723  
Iteration 303, loss = 0.02478291  
Iteration 304, loss = 0.02464823  
Iteration 305, loss = 0.02443777  
Iteration 306, loss = 0.02428021  
Iteration 307, loss = 0.02410657  
Iteration 308, loss = 0.02397925  
Iteration 309, loss = 0.02386305  
Iteration 310, loss = 0.02368288  
Iteration 311, loss = 0.02350678  
Iteration 312, loss = 0.02334511  
Iteration 313, loss = 0.02320893  
Iteration 314, loss = 0.02305903  
Iteration 315, loss = 0.02291093  
Iteration 316, loss = 0.02276177  
Iteration 317, loss = 0.02265898  
Iteration 318, loss = 0.02253300  
Iteration 319, loss = 0.02232386  
Iteration 320, loss = 0.02225028  
Iteration 321, loss = 0.02215497  
Iteration 322, loss = 0.02201018  
Iteration 323, loss = 0.02185725  
Iteration 324, loss = 0.02170322  
Iteration 325, loss = 0.02172075  
Iteration 326, loss = 0.02148360  
Iteration 327, loss = 0.02131039  
Iteration 328, loss = 0.02124368  
Iteration 329, loss = 0.02106802  
Iteration 330, loss = 0.02099726

Iteration 331, loss = 0.02084015  
Iteration 332, loss = 0.02070063  
Iteration 333, loss = 0.02057292  
Iteration 334, loss = 0.02048105  
Iteration 335, loss = 0.02033480  
Iteration 336, loss = 0.02020452  
Iteration 337, loss = 0.02010853  
Iteration 338, loss = 0.02000349  
Iteration 339, loss = 0.01986784  
Iteration 340, loss = 0.01975998  
Iteration 341, loss = 0.01963795  
Iteration 342, loss = 0.01956331  
Iteration 343, loss = 0.01944035  
Iteration 344, loss = 0.01935272  
Iteration 345, loss = 0.01924468  
Iteration 346, loss = 0.01919674  
Iteration 347, loss = 0.01906321  
Iteration 348, loss = 0.01897222  
Iteration 349, loss = 0.01891405  
Iteration 350, loss = 0.01879020  
Iteration 351, loss = 0.01868349  
Iteration 352, loss = 0.01851653  
Iteration 353, loss = 0.01846123  
Iteration 354, loss = 0.01837644  
Iteration 355, loss = 0.01827596  
Iteration 356, loss = 0.01814103  
Iteration 357, loss = 0.01805101  
Iteration 358, loss = 0.01796370  
Iteration 359, loss = 0.01788139  
Iteration 360, loss = 0.01779407  
Iteration 361, loss = 0.01771349  
Iteration 362, loss = 0.01762351  
Iteration 363, loss = 0.01762686  
Iteration 364, loss = 0.01746587  
Iteration 365, loss = 0.01734642  
Iteration 366, loss = 0.01730601  
Iteration 367, loss = 0.01718104  
Iteration 368, loss = 0.01708140  
Iteration 369, loss = 0.01700696  
Iteration 370, loss = 0.01693507  
Iteration 371, loss = 0.01688764  
Iteration 372, loss = 0.01676253  
Iteration 373, loss = 0.01666569  
Iteration 374, loss = 0.01659526  
Iteration 375, loss = 0.01655323  
Iteration 376, loss = 0.01644555  
Iteration 377, loss = 0.01637253  
Iteration 378, loss = 0.01632746  
Iteration 379, loss = 0.01623379  
Iteration 380, loss = 0.01623290  
Iteration 381, loss = 0.01609984  
Iteration 382, loss = 0.01601667  
Iteration 383, loss = 0.01592608  
Iteration 384, loss = 0.01584991  
Iteration 385, loss = 0.01582054  
Iteration 386, loss = 0.01571604  
Iteration 387, loss = 0.01567182  
Iteration 388, loss = 0.01557538  
Iteration 389, loss = 0.01550015  
Iteration 390, loss = 0.01545884  
Iteration 391, loss = 0.01542730  
Iteration 392, loss = 0.01536880  
Iteration 393, loss = 0.01529645  
Iteration 394, loss = 0.01525873  
Iteration 395, loss = 0.01513995  
Iteration 396, loss = 0.01505392  
Iteration 397, loss = 0.01502007  
Iteration 398, loss = 0.01492674  
Iteration 399, loss = 0.01489953  
Iteration 400, loss = 0.01479204  
Iteration 401, loss = 0.01473401  
Iteration 402, loss = 0.01468193  
Iteration 403, loss = 0.01460206  
Iteration 404, loss = 0.01455794  
Iteration 405, loss = 0.01447843  
Iteration 406, loss = 0.01442326  
Iteration 407, loss = 0.01436888  
Iteration 408, loss = 0.01431737  
Iteration 409, loss = 0.01424268  
Iteration 410, loss = 0.01419381  
Iteration 411, loss = 0.01416255  
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.

MLPClassifier(alpha=0.001, hidden\_layer\_sizes=(100, 100),  
 learning\_rate\_init=0.01, max\_iter=1000, random\_state=0,  
 solver='sgd', verbose=10)

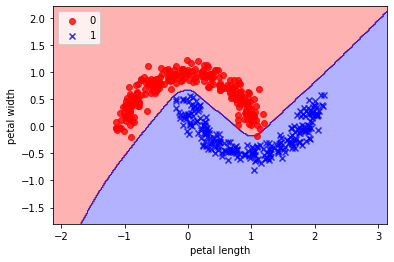
### Plot the cost change

# plot the loss  
None  
plt.show()



### Plot the decision boundary

# plot decision boundary of the model   
None  
  
plt.xlabel('petal length')  
plt.ylabel('petal width')  
plt.legend(loc='upper left')  
plt.show()



# Quiz 2 : Image Classification

## Train multilayer perceptron with the Fashion MNIST dataset

* Use the MLPClassifier in scikit learn with 128 neurons each in 2 hidden layer

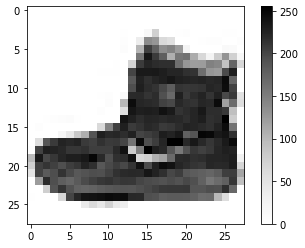
1. Import tensorflow as tf, and load the Fashion MNIST dataset using tf.keras.datasets.fashion\_mnist.load\_data()
2. Train the model up to 50 epochs using X\_train
3. Plot the cost change during training
4. Show the train and test accuracies
5. Show the classification result of the test data X\_test[3]

### Load and prepare the Fashion MNIST dataset

import tensorflow as tf  
  
# read fashion MNIST data   
fashion\_mnist = tf.keras.datasets.fashion\_mnist  
  
(X\_train, y\_train), (X\_test, y\_test) = fashion\_mnist.load\_data()  
  
# shape of X\_train   
X\_train.shape

(60000, 28, 28)

# show the image data 0  
None  
plt.colorbar()  
plt.show()



# scaling X  
X\_train, X\_test = X\_train / 255.0, X\_test / 255.0

# class labels (y\_train)  
y\_train

array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)

### Show first 25 images and labels

# names for class labels  
class\_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',   
 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']  
  
# name of the class label of train data 0  
class\_names[y\_train[0]]

'Ankle boot'

# show first 25 data and label  
plt.figure(figsize=(8,8))  
  
for i in range(25):  
 plt.subplot(5,5,i+1)  
 plt.xticks([])  
 plt.yticks([])  
 plt.imshow(None, cmap=plt.cm.binary)  
 plt.xlabel(class\_names[y\_train[i]])  
plt.show()



### Train the model

from sklearn.neural\_network import MLPClassifier  
  
# build the model with 2 hidden layers, 128 units each, max iteration 50  
mlp = None

# flatten the data  
X\_train\_1d = X\_train.reshape(60000, 784)  
X\_test\_1d = X\_test.reshape(10000, 784)

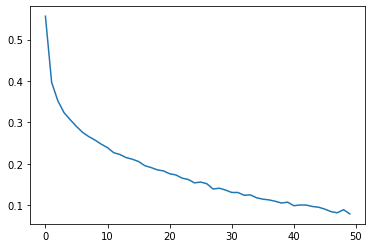
# checking the execution time  
import time  
start\_time = time.time()  
  
# training the model  
None  
  
print("Time : ", time.time()-start\_time)

Iteration 1, loss = 0.55645299  
Iteration 2, loss = 0.39674814  
Iteration 3, loss = 0.35206005  
Iteration 4, loss = 0.32335400  
Iteration 5, loss = 0.30622575  
Iteration 6, loss = 0.29004210  
Iteration 7, loss = 0.27561103  
Iteration 8, loss = 0.26541692  
Iteration 9, loss = 0.25696521  
Iteration 10, loss = 0.24717210  
Iteration 11, loss = 0.23872071  
Iteration 12, loss = 0.22663507  
Iteration 13, loss = 0.22229734  
Iteration 14, loss = 0.21481390  
Iteration 15, loss = 0.21088833  
Iteration 16, loss = 0.20530210  
Iteration 17, loss = 0.19528614  
Iteration 18, loss = 0.19084128  
Iteration 19, loss = 0.18524678  
Iteration 20, loss = 0.18267485  
Iteration 21, loss = 0.17578112  
Iteration 22, loss = 0.17292266  
Iteration 23, loss = 0.16521429  
Iteration 24, loss = 0.16189670  
Iteration 25, loss = 0.15376023  
Iteration 26, loss = 0.15568037  
Iteration 27, loss = 0.15147741  
Iteration 28, loss = 0.13891280  
Iteration 29, loss = 0.14083972  
Iteration 30, loss = 0.13635171  
Iteration 31, loss = 0.13054728  
Iteration 32, loss = 0.13045263  
Iteration 33, loss = 0.12375475  
Iteration 34, loss = 0.12478579  
Iteration 35, loss = 0.11766188  
Iteration 36, loss = 0.11438345  
Iteration 37, loss = 0.11258786  
Iteration 38, loss = 0.10925650  
Iteration 39, loss = 0.10504857  
Iteration 40, loss = 0.10706785  
Iteration 41, loss = 0.09853899  
Iteration 42, loss = 0.10030816  
Iteration 43, loss = 0.10004245  
Iteration 44, loss = 0.09662907  
Iteration 45, loss = 0.09482950  
Iteration 46, loss = 0.09017948  
Iteration 47, loss = 0.08417489  
Iteration 48, loss = 0.08146402  
Iteration 49, loss = 0.08894281  
Iteration 50, loss = 0.07869972  
Time : 47.49512028694153

C:\Users\win\anaconda3\lib\site-packages\sklearn\neural\_network\\_multilayer\_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (50) reached and the optimization hasn't converged yet.  
 warnings.warn(

### Plot the cost change

# plot the loss  
None  
plt.show()



### Accuracy of the model

# Train and test accuracy  
acc = None  
print("Train accuracy : %.4f" % acc)  
acc = None  
print("Train accuracy : %.4f" % acc)

Train accuracy : 0.9655  
Train accuracy : 0.8888

### Classification test

# prediction of all test data  
predictions = None

# show the image of test data 3  
plt.figure(figsize=(1, 1))  
plt.imshow(None, cmap=plt.cm.binary)  
plt.show()



# show the true class name and the predicted class name of test data 3  
print('True lable = %s' % None)  
print('Predicted = %s' % None)

True lable = Trouser  
Predicted = Trouser