Lecture 7
Νευρωνικά Δίκτυα
(Artificial Neural Networks)

Δημιουργία νευρωνικού δικτύου ενός επιπέδου
single layer neural network

- Ένα μόνο perceptron περιορίζεται σε γραμμικά διαχωρίσιμα δεδομένα.
- Με ένα σύνολο perceptron που συνεργάζονται, μπορούμε να ξεπεράσουμε αυτόν τον περιορισμό.

- Κατεβάστε το αρχείο “`simple_neural_network.py`” από τη σελίδα του μαθήματος.
- Φορτώστε το αρχείο από το Spyder IDE.

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each dimension
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()

# Define the number of neurons in the output layer
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

Χρησιμοποιώντας τα ίδια πακέτα.

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each dimension
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()

# Define the number of neurons in the output layer
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

Χρησιμοποιήστε τα δεδομένα εισόδου από το αρχείο data_simple_nn.txt που σας παρέχεται στη σελίδα του μαθήματος.

Διαχωρίστε τα δεδομένα σε σημεία δεδομένων και ετικέτες.

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each dimension
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()

# Define the number of neurons in the output layer
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

Τα δεδομένα σας είναι αυτά. →

```
1.0 4.0 0 0
1.1 3.9 0 0
1.2 4.1 0 0
0.9 3.7 0 0
7.0 4.0 0 1
7.2 4.1 0 1
6.9 3.9 0 1
7.1 4.2 0 1
4.0 1.0 1 0
4.1 0.9 1 0
4.2 1.1 1 0
3.9 0.8 1 0
4.0 7.0 1 1
4.2 7.2 1 1
3.9 7.1 1 1
4.1 6.8 1 1
```

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each dimension
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()

# Define the number of neurons in the output layer
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

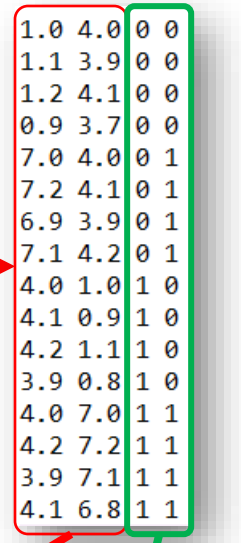
Τα δεδομένα σας είναι αυτά.

```
1.0 4.0 0 0
1.1 3.9 0 0
1.2 4.1 0 0
0.9 3.7 0 0
7.0 4.0 0 1
7.2 4.1 0 1
6.9 3.9 0 1
7.1 4.2 0 1
4.0 1.0 1 0
4.1 0.9 1 0
4.2 1.1 1 0
3.9 0.8 1 0
4.0 7.0 1 1
4.2 7.2 1 1
3.9 7.1 1 1
4.1 6.8 1 1
```

Σημεία: πρώτες 2 στήλες

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each dimension
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()

# Define the number of neurons in the output layer
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

Τα δεδομένα σας είναι αυτά.

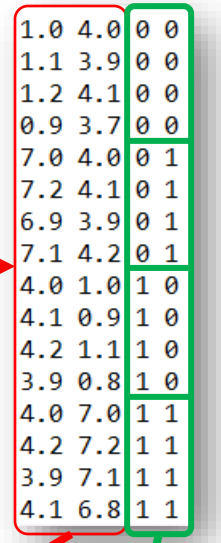


Σημεία: πρώτες 2 στήλες

Ετικέτες: τελευταίες 2 στήλες

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each dimension
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()

# Define the number of neurons in the output layer
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

Τα δεδομένα σας είναι αυτά.



Σημεία: πρώτες 2 στήλες

Ετικέτες: τελευταίες 2 στήλες

4 διακριτές κατηγορίες

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each dimension
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()

# Define the number of neurons in the output layer
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

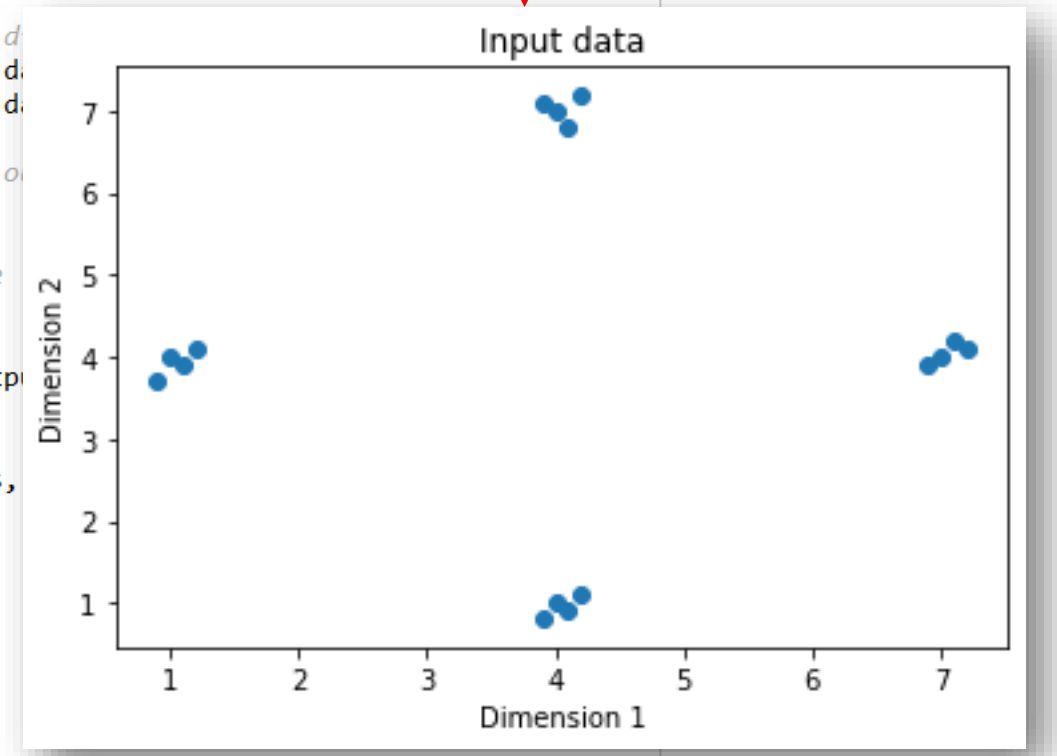
Αποτυπώστε τα σημεία σε ένα γράφημα.

1.0 4.0	0 0
1.1 3.9	0 0
1.2 4.1	0 0
0.9 3.7	0 0
7.0 4.0	0 1
7.2 4.1	0 1
6.9 3.9	0 1
7.1 4.2	0 1
4.0 1.0	1 0
4.1 0.9	1 0
4.2 1.1	1 0
3.9 0.8	1 0
4.0 7.0	1 1
4.2 7.2	1 1
3.9 7.1	1 1
4.1 6.8	1 1

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each d
dim1_min, dim1_max = data[:,0].min(), da
dim2_min, dim2_max = data[:,1].min(), da

# Define the number of neurons in the o
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_outpu

# Train the neural network
error_progress = nn.train(data, labels,

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

Θα πρέπει να πάρετε αυτό το αποτέλεσμα.

1.0 4.0	0 0
1.1 3.9	0 0
1.2 4.1	0 0
0.9 3.7	0 0
7.0 4.0	0 1
7.2 4.1	0 1
6.9 3.9	0 1
7.1 4.2	0 1
4.0 1.0	1 0
4.1 0.9	1 0
4.2 1.1	1 0
3.9 0.8	1 0
4.0 7.0	1 1
4.2 7.2	1 1
3.9 7.1	1 1
4.1 6.8	1 1



```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each dimension
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()

# Define the number of neurons in the output layer
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

Καθορίστε τις μέγιστες και ελάχιστες τιμές που μπορεί να λάβει κάθε διάσταση.

1.0 4.0	0 0
1.1 3.9	0 0
1.2 4.1	0 0
0.9 3.7	0 0
7.0 4.0	0 1
7.2 4.1	0 1
6.9 3.9	0 1
7.1 4.2	0 1
4.0 1.0	1 0
4.1 0.9	1 0
4.2 1.1	1 0
3.9 0.8	1 0
4.0 7.0	1 1
4.2 7.2	1 1
3.9 7.1	1 1
4.1 6.8	1 1

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each dimension
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()

# Define the number of neurons in the output layer
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

1.0 4.0	0 0
1.1 3.9	0 0
1.2 4.1	0 0
0.9 3.7	0 0
7.0 4.0	0 1
7.2 4.1	0 1
6.9 3.9	0 1
7.1 4.2	0 1
4.0 1.0	1 0
4.1 0.9	1 0
4.2 1.1	1 0
3.9 0.8	1 0
4.0 7.0	1 1
4.2 7.2	1 1
3.9 7.1	1 1
4.1 6.8	1 1

Καθορίστε τον αριθμό των νευρώνων στο επίπεδο εξόδου.

Αυτό είναι 2 τώρα.

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each dimension
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()

# Define the number of neurons in the output layer
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

1.0 4.0	0 0
1.1 3.9	0 0
1.2 4.1	0 0
0.9 3.7	0 0
7.0 4.0	0 1
7.2 4.1	0 1
6.9 3.9	0 1
7.1 4.2	0 1
4.0 1.0	1 0
4.1 0.9	1 0
4.2 1.1	1 0
3.9 0.8	1 0
4.0 7.0	1 1
4.2 7.2	1 1
3.9 7.1	1 1
4.1 6.8	1 1

Ορίστε ένα νευρωνικό δίκτυο ενός επιπέδου χρησιμοποιώντας τις παραπάνω παραμέτρους.

2 είσοδοι, 2 έξοδοι.

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each dimension
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()

# Define the number of neurons in the output layer
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

1.0 4.0	0 0
1.1 3.9	0 0
1.2 4.1	0 0
0.9 3.7	0 0
7.0 4.0	0 1
7.2 4.1	0 1
6.9 3.9	0 1
7.1 4.2	0 1
4.0 1.0	1 0
4.1 0.9	1 0
4.2 1.1	1 0
3.9 0.8	1 0
4.0 7.0	1 1
4.2 7.2	1 1
3.9 7.1	1 1
4.1 6.8	1 1

Εκπαιδεύουμε το perceptron με τα δεδομένα από το training σετ.

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each dimension
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()

# Define the number of neurons in the output layer
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

1.0 4.0	0 0
1.1 3.9	0 0
1.2 4.1	0 0
0.9 3.7	0 0
7.0 4.0	0 1
7.2 4.1	0 1
6.9 3.9	0 1
7.1 4.2	0 1
4.0 1.0	1 0
4.1 0.9	1 0
4.2 1.1	1 0
3.9 0.8	1 0
4.0 7.0	1 1
4.2 7.2	1 1
3.9 7.1	1 1
4.1 6.8	1 1

Εκπαιδεύουμε το perceptron με τα δεδομένα από το training σετ.

```
Epoch: 20; Error: 4.0;
Epoch: 40; Error: 4.0;
Epoch: 60; Error: 4.0;
Epoch: 80; Error: 4.0;
Epoch: 100; Error: 4.0;
The maximum number of train epochs is reached
```

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_nn.txt')

# Separate it into datapoints and labels
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values for each dimension
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()

# Define the number of neurons in the output layer
num_output = labels.shape[1]

# Define a single-layer neural network
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

1.0 4.0	0 0
1.1 3.9	0 0
1.2 4.1	0 0
0.9 3.7	0 0
7.0 4.0	0 1
7.2 4.1	0 1
6.9 3.9	0 1
7.1 4.2	0 1
4.0 1.0	1 0
4.1 0.9	1 0
4.2 1.1	1 0
3.9 0.8	1 0
4.0 7.0	1 1
4.2 7.2	1 1
3.9 7.1	1 1
4.1 6.8	1 1

Σχεδιάστε την πρόοδο της εκπαίδευσης χρησιμοποιώντας τη μέτρηση σφάλματος (error_progress).

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Load input data
text = np.loadtxt('data_simple_
# Separate it into datapoints a
data = text[:, 0:2]
labels = text[:, 2:]

# Plot input data
plt.figure()
plt.scatter(data[:,0], data[:,1
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Minimum and maximum values fo
dim1_min, dim1_max = data[:,0].
dim2_min, dim2_max = data[:,1].

# Define the number of neurons
num_output = labels.shape[1]

# Define a single-layer neural
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)

# Train the neural network
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Plot the training progress
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()

plt.show()
```

1.0	4.0	0	0
1.1	3.9	0	0
1.2	4.1	0	0
0.9	3.7	0	0
7.0	4.0	0	1
7.2	4.1	0	1
6.9	3.9	0	1
7.1	4.2	0	1
4.0	1.0	1	0
4.1	0.9	1	0
4.2	1.1	1	0
3.9	0.8	1	0
4.0	7.0	1	1
4.2	7.2	1	1
3.9	7.1	1	1
4.1	6.8	1	1

Θα πρέπει να πάρετε περίπου αυτό το αποτέλεσμα. Τι παρατηρείτε;

Δημιουργία νευρωνικού δικτύου ενός επιπέδου

single layer neural network

- Ας ορίσουμε μερικά δείγματα σημείων δεδομένων δοκιμής και ας εκτελέσουμε το δίκτυο σε αυτά τα σημεία.
- Το **nn** τώρα περιέχει το εκπαιδευμένο (διαμορφωμένο) νευρωνικό μας δίκτυο.

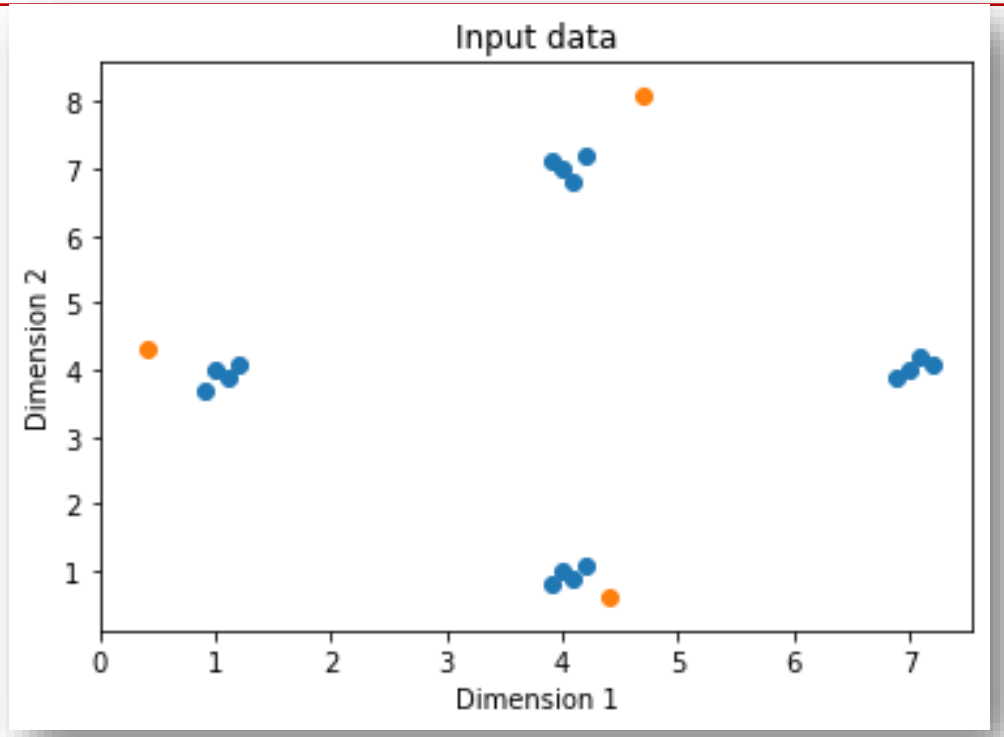
```
44 # Run the classifier on test datapoints
45 print('\nTest results:')
46 data_test = [[0.4, 4.3], [4.4, 0.6], [4.7, 8.1]]
47 for item in data_test:
48     print(item, '-->', nn.sim([item])[0])
```

Δημιουργία νευρωνικού δικτύου ενός επιπέδου

single layer neural network

- Ας ορίσουμε μερικά δείγματα σημείων δεδομένων δοκιμής και ας εκτελέσουμε το δίκτυο σε αυτά τα σημεία.
- Το **nn** τώρα περιέχει το εκπαιδευμένο (διαμορφωμένο) νευρωνικό μας δίκτυο.

```
44 # Run the classifier on test datapoints
45 print('\nTest results:')
46 data_test = [[0.4, 4.3], [4.4, 0.6], [4.7, 8.1]]
47 for item in data_test:
48     print(item, '-->', nn.sim([item])[0])
```



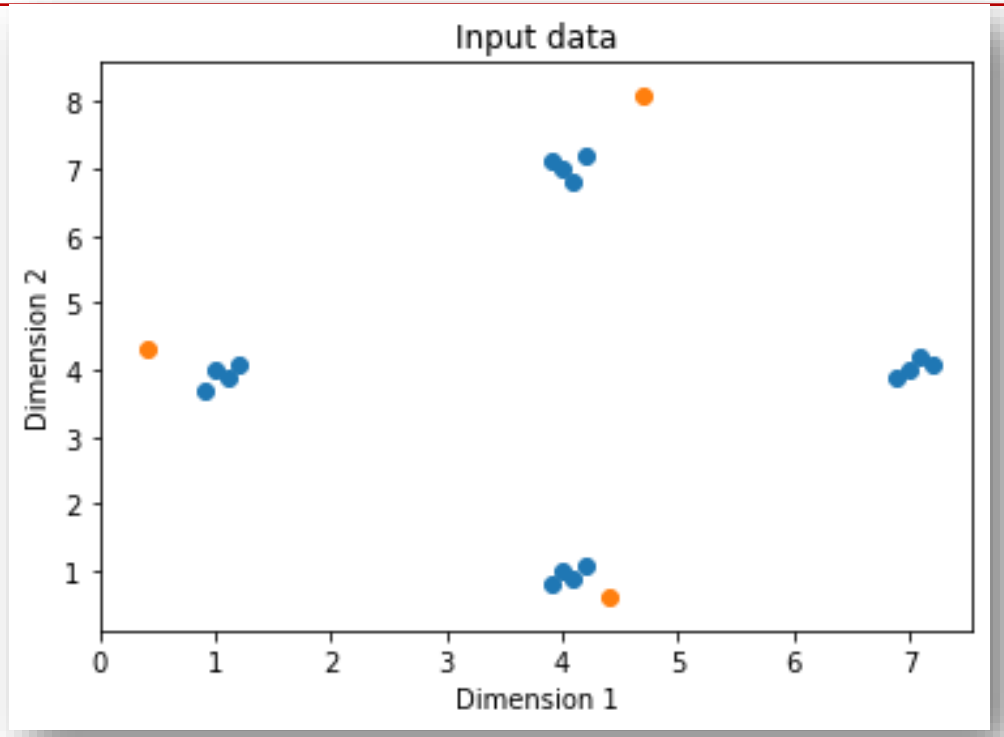
Δεδομένα εκπαίδευσης (train data) με **μπλε**
Δεδομένα δοκιμής(test data) με **πορτοκαλί**

Δημιουργία νευρωνικού δικτύου ενός επιπέδου

single layer neural network

Για να πάρετε αυτό το γράφημα, απλά προσθέστε στον κώδικα τις παρακάτω γραμμές

```
 8 # Separate it into datapoints and labels
 9 data = text[:, 0:2]
10 labels = text[:, 2:]
11 data_test = np.array([[0.4, 4.3], [4.4, 0.6], [4.7, 8.1]])
12
13 # Plot input data
14 plt.figure()
15 plt.scatter(data[:,0], data[:,1])
16 plt.scatter(data_test[:,0], data_test[:,1])
17 plt.xlabel('Dimension 1')
18 plt.ylabel('Dimension 2')
19 plt.title('Input data')
```



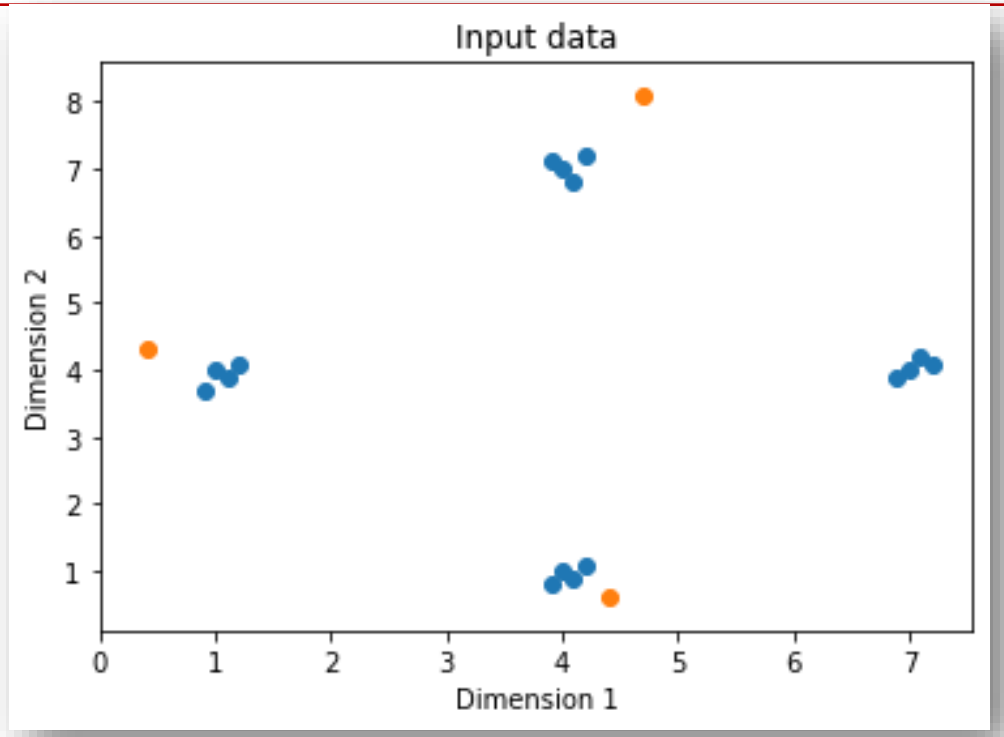
Δεδομένα εκπαίδευσης (train data) με **μπλε**
Δεδομένα δοκιμής(test data) με **πορτοκαλί**

Δημιουργία νευρωνικού δικτύου ενός επιπέδου

single layer neural network

1.0	4.0	0	0
1.1	3.9	0	0
1.2	4.1	0	0
0.9	3.7	0	0
7.0	4.0	0	1
7.2	4.1	0	1
6.9	3.9	0	1
7.1	4.2	0	1
4.0	1.0	1	0
4.1	0.9	1	0
4.2	1.1	1	0
3.9	0.8	1	0
4.0	7.0	1	1
4.2	7.2	1	1
3.9	7.1	1	1
4.1	6.8	1	1

Train data



Δεδομένα εκπαίδευσης (train data) με **μπλε**
Δεδομένα δοκιμής(test data) με **πορτοκαλί**

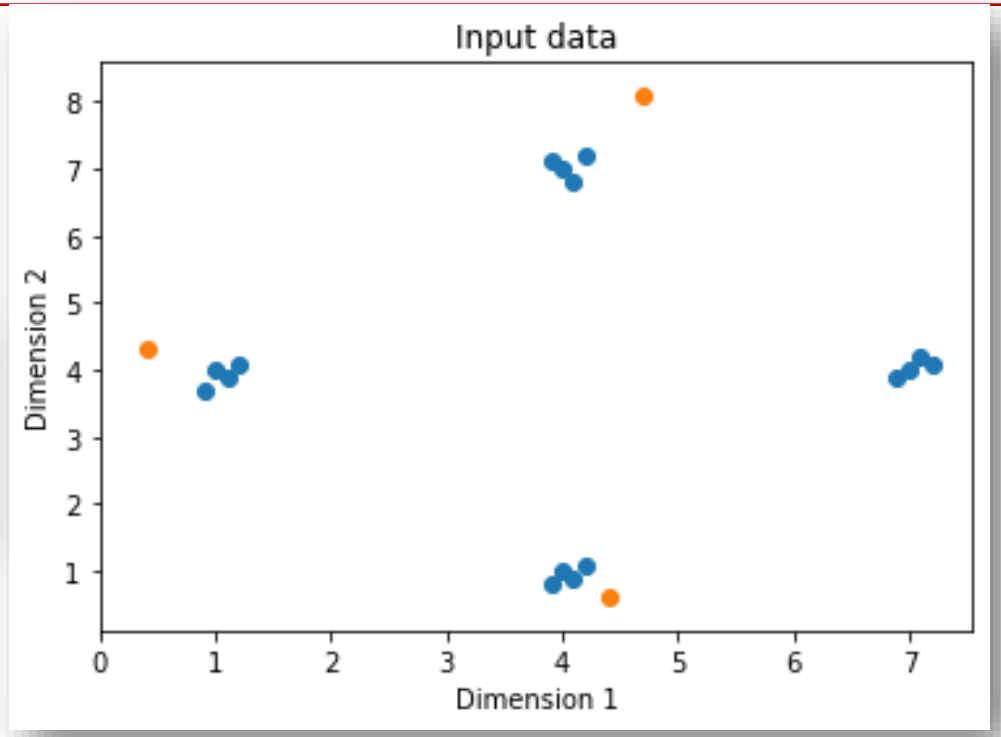
Δημιουργία νευρωνικού δικτύου ενός επιπέδου

single layer neural network

1.0 4.0	0 0
1.1 3.9	0 0
1.2 4.1	0 0
0.9 3.7	0 0
7.0 4.0	0 1
7.2 4.1	0 1
6.9 3.9	0 1
7.1 4.2	0 1
4.0 1.0	1 0
4.1 0.9	1 0
4.2 1.1	1 0
3.9 0.8	1 0
4.0 7.0	1 1
4.2 7.2	1 1
3.9 7.1	1 1
4.1 6.8	1 1

Train data

```
Test results:
[0.4, 4.3] --> [0. 0.]
[4.4, 0.6] --> [1. 0.]
[4.7, 8.1] --> [1. 1.]
```



Δεδομένα εκπαίδευσης (train data) με **μπλε**
Δεδομένα δοκιμής(test data) με **πορτοκαλί**

Δημιουργία πολυεπίπεδου νευρωνικού δικτύου
multilayer neural network

- Για να επιτρέψουμε μεγαλύτερη ακρίβεια, πρέπει να δώσουμε περισσότερη ελευθερία στο νευρωνικό δίκτυο.
- Απαιτούνται περισσότερα από ένα επίπεδα perceptrons για την εξαγωγή των υποκείμενων μοτίβων στα δεδομένα εκπαίδευσης.
- Κατεβάστε το αρχείο "multilayer_neural_network.py" από τη σελίδα του μαθήματος και φορτώστε το στο Spyder IDE.
- Σε αυτό το παράδειγμα, θα δούμε πώς να χρησιμοποιήσουμε ένα multilayer νευρωνικό δίκτυο ως regressor (πρόβλεψη τιμής).

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Generate some training data
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points) # Returns evenly spaced numbers over a specified interval.
y = 3 * np.square(x) + 5
y /= np.linalg.norm(y)

# Create data and labels
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)

# Plot input data
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Define a multilayer neural network with 2 hidden layers;
# First hidden layer consists of 10 neurons
# Second hidden layer consists of 6 neurons
# Output layer consists of 1 neuron
nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])

# Set the training algorithm to gradient descent
nn.trainf = nl.train.train_gd

# Train the neural network
error_progress = nn.train(data, labels, epochs=2000, show=100, goal=0.01)

# Run the neural network on training datapoints
output = nn.sim(data)
y_pred = output.reshape(num_points)

```

Για τους σκοπούς αυτής της άσκησης μπορούμε εύκολα να δημιουργήσουμε κάποια δεδομένα χρησιμοποιώντας το πακέτο numpy.

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Generate some training data
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points) # Returns evenly spaced numbers over a specified interval.
y = 3 * np.square(x) + 5
y /= np.linalg.norm(y)

# Create data and labels
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)

# Plot input data
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Define a multilayer neural network with 2 hidden layers;
# First hidden layer consists of 10 neurons
# Second hidden layer consists of 6 neurons
# Output layer consists of 1 neuron
nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])

# Set the training algorithm to gradient descent
nn.trainf = nl.train.train_gd

# Train the neural network
error_progress = nn.train(data, labels, epochs=2000, show=100, goal=0.01)

# Run the neural network on training datapoints
output = nn.sim(data)
y_pred = output.reshape(num_points)

```

Για τους σκοπούς αυτής της άσκησης μπορούμε εύκολα να δημιουργήσουμε κάποια δεδομένα χρησιμοποιώντας το πακέτο numpy.

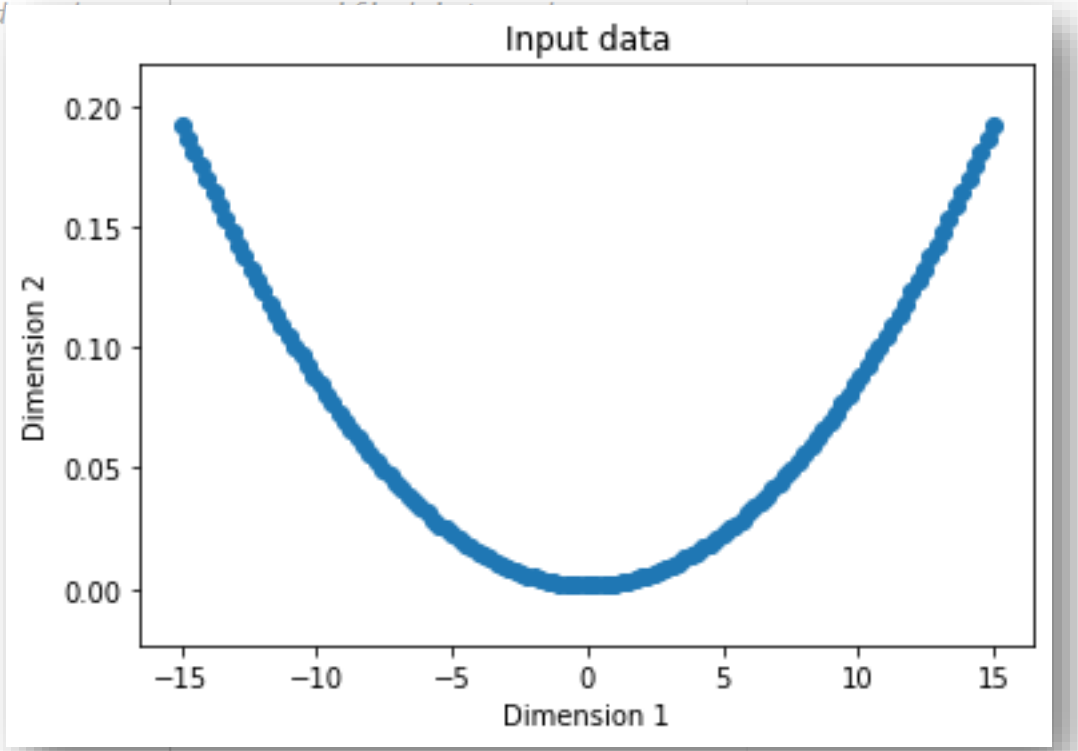
Τα δείγματα σημείων δεδομένων (sample data) βασίζονται στην εξίσωση $y = 3x^2 + 5$ και στη συνέχεια χρησιμοποιούμε τη συνάρτηση norm για να κάνουμε normalization.

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Generate some training data
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points) # Returns evenly spaced numbers over a specified interval.
y = 3 * np.square(x) + 5
y /= np.linalg.norm(y)

# Create data and labels
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)

# Plot input data
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Define a multilayer neural network with 2 hidden layers;
# First hidden layer consists of 10 neurons
# Second hidden layer consists of 6 neurons
# Output layer consists of 1 neuron
nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])

# Set the training algorithm to gradient descent
nn.trainf = nl.train.train_gd

# Train the neural network
error_progress = nn.train(data, labels, epochs=2000, show=100, goal=0.01)

# Run the neural network on training datapoints
output = nn.sim(data)
y_pred = output.reshape(num_points)

```

Αποτυπώστε τα δεδομένα εισόδου (input data).

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Generate some training data
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points) # Returns evenly spaced
y = 3 * np.square(x) + 5
y /= np.linalg.norm(y)

# Create data and labels
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)

# Plot input data
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Define a multilayer neural network with 2 hidden layers;
# First hidden layer consists of 10 neurons
# Second hidden layer consists of 6 neurons
# Output layer consists of 1 neuron
nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])

# Set the training algorithm to gradient descent
nn.trainf = nl.train.train_gd

# Train the neural network
error_progress = nn.train(data, labels, epochs=2000, show=100, goal=0.01)

# Run the neural network on training datapoints
output = nn.sim(data)
y_pred = output.reshape(num_points)
```

Αποτυπώστε τα δεδομένα εισόδου (input data).



```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Generate some training data
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points) # Returns evenly spaced numbers over a specified interval.
y = 3 * np.square(x) + 5
y /= np.linalg.norm(y)

# Create data and labels
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)

# Plot input data
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Define a multilayer neural network with 2 hidden layers;
# First hidden layer consists of 10 neurons
# Second hidden layer consists of 6 neurons
# Output layer consists of 1 neuron
nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])

# Set the training algorithm to gradient descent
nn.trainf = nl.train.train_gd

# Train the neural network
error_progress = nn.train(data, labels, epochs=2000, show=100, goal=0.01)

# Run the neural network on training datapoints
output = nn.sim(data)
y_pred = output.reshape(num_points)

```

Ορίστε την αρχιτεκτονική του νευρωνικού δικτύου.

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Generate some training data
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points) # Returns evenly spaced numbers over a specified interval.
y = 3 * np.square(x) + 5
y /= np.linalg.norm(y)

# Create data and labels
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)

# Plot input data
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Define a multilayer neural network with 2 hidden layers;
# First hidden layer consists of 10 neurons
# Second hidden layer consists of 6 neurons
# Output layer consists of 1 neuron
nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])

# Set the training algorithm to gradient descent
nn.trainf = nl.train.train_gd

# Train the neural network
error_progress = nn.train(data, labels, epochs=2000, show=100, goal=0.01)

# Run the neural network on training datapoints
output = nn.sim(data)
y_pred = output.reshape(num_points)

```

Ρυθμίστε τον αλγόριθμο εκπαίδευσης σε gradient descent και εκπαιδεύστε το μοντέλο.

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Generate some training data
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points) # Returns evenly spaced numbers over a specified interval.
y = 3 * np.square(x) + 5
y /= np.linalg.norm(y)

# Create data and labels
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)

# Plot input data
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Define a multilayer neural network with 2 hidden layers;
# First hidden layer consists of 10 neurons
# Second hidden layer consists of 6 neurons
# Output layer consists of 1 neuron
nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])

# Set the training algorithm to gradient descent
nn.trainf = nl.train.train_gd

# Train the neural network
error_progress = nn.train(data, labels, epochs=2000, show=100, goal=0.01)

# Run the neural network on training datapoints
output = nn.sim(data)
y_pred = output.reshape(num_points)

```

```
Epoch: 100; Error: 0.6958681467110985;
Epoch: 200; Error: 0.014412280638650635;
The goal of learning is reached
```

Ρυθμίστε τον αλγόριθμο εκπαίδευσης σε gradient descent και εκπαιδεύστε το μοντέλο.

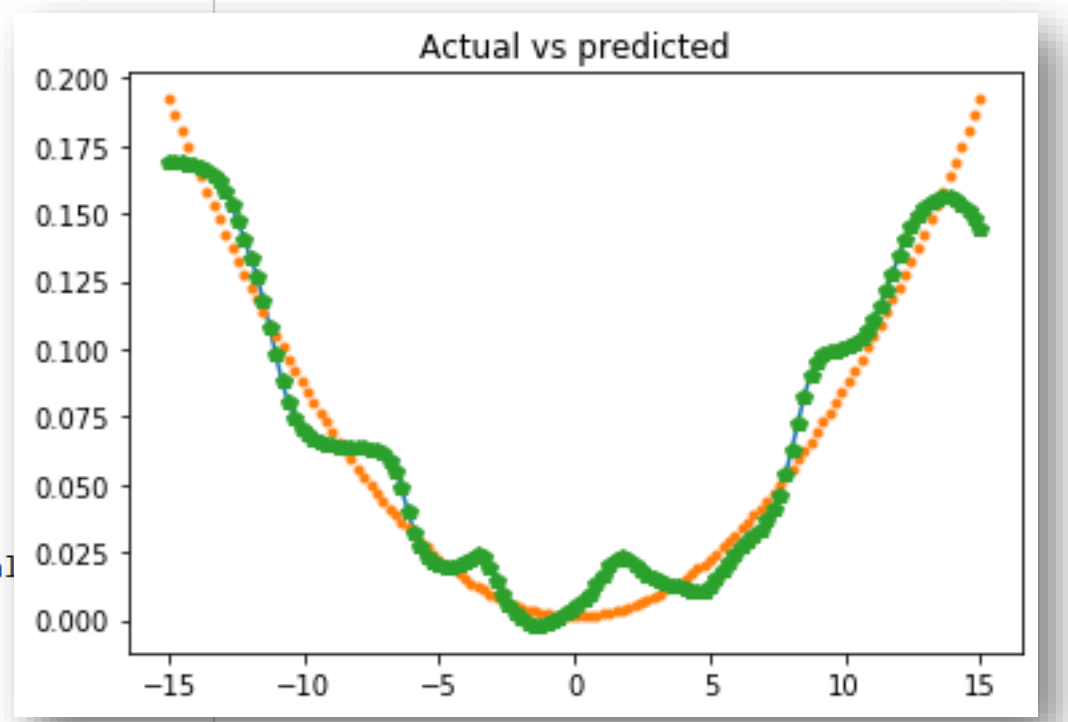
```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Generate some training data
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points) # Returns evenly spaced numbers over a specified interval.
y = 3 * np.square(x) + 5
y /= np.linalg.norm(y)

# Create data and labels
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)

# Plot input data
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Define a multilayer neural network with 2 hidden layers;
# First hidden layer consists of 10 neurons
# Second hidden layer consists of 6 neurons
# Output layer consists of 1 neuron
nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])

# Set the training algorithm to gradient descent
nn.trainf = nl.train.train_gd

# Train the neural network
error_progress = nn.train(data, labels, epochs=2000, show=100, goal=0.01)

# Run the neural network on training datapoints
output = nn.sim(data)
y_pred = output.reshape(num_points)

```

Εκτελέστε το νευρωνικό δίκτυο στα σημεία δεδομένων εκπαίδευσης.

```
# Plot input data
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Define a multilayer neural network with 2 hidden layers;
# First hidden layer consists of 10 neurons
# Second hidden layer consists of 6 neurons
# Output layer consists of 1 neuron
nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])

# Set the training algorithm to gradient descent
nn.trainf = nl.train.train_gd

# Train the neural network
error_progress = nn.train(data, labels, epochs=2000, show=100, goa

# Run the neural network on training datapoints
output = nn.sim(data)
y_pred = output.reshape(num_points)

# Plot training error
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Error')
plt.title('Training error progress')

# Plot the output
x_dense = np.linspace(min_val, max_val, num_points * 2)
y_dense_pred = nn.sim(x_dense.reshape(x_dense.size,1)).reshape(x_dense.size)

plt.figure()
plt.plot(x_dense, y_dense_pred, '-', x, y, '.', x, y_pred, 'p')
plt.title('Actual vs predicted')

plt.show()
```

Αποτυπώστε την πρόοδο της εκπαίδευσης.

```
# Plot input data
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')

# Define a multilayer neural network with 2 hidden layers;
# First hidden layer consists of 10 neurons
# Second hidden layer consists of 6 neurons
# Output layer consists of 1 neuron
nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])

# Set the training algorithm to gradient descent
nn.trainf = nl.train.train_gd

# Train the neural network
error_progress = nn.train(data, labels, epochs=2000, show=100, goal

# Run the neural network on training datapoints
output = nn.sim(data)
y_pred = output.reshape(num_points)

# Plot training error
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Error')
plt.title('Training error progress')

# Plot the output
x_dense = np.linspace(min_val, max_val, num_points * 2)
y_dense_pred = nn.sim(x_dense.reshape(x_dense.size,1)).reshape(x_dense.size)

plt.figure()
plt.plot(x_dense, y_dense_pred, '-', x, y, '.', x, y_pred, 'p')
plt.title('Actual vs predicted')

plt.show()

```



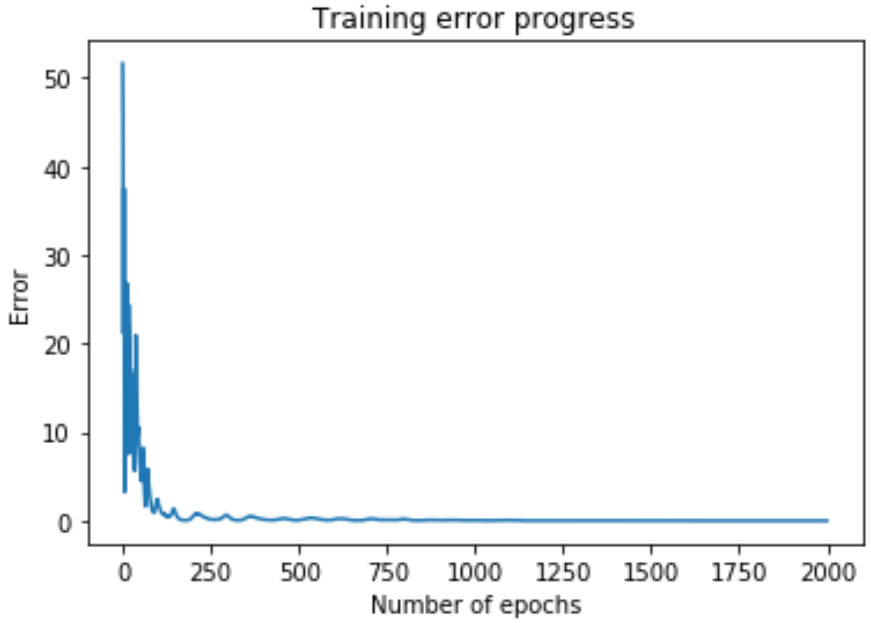
Αποτυπώστε τις προβλέψεις μαζί με τα πραγματικά σημεία

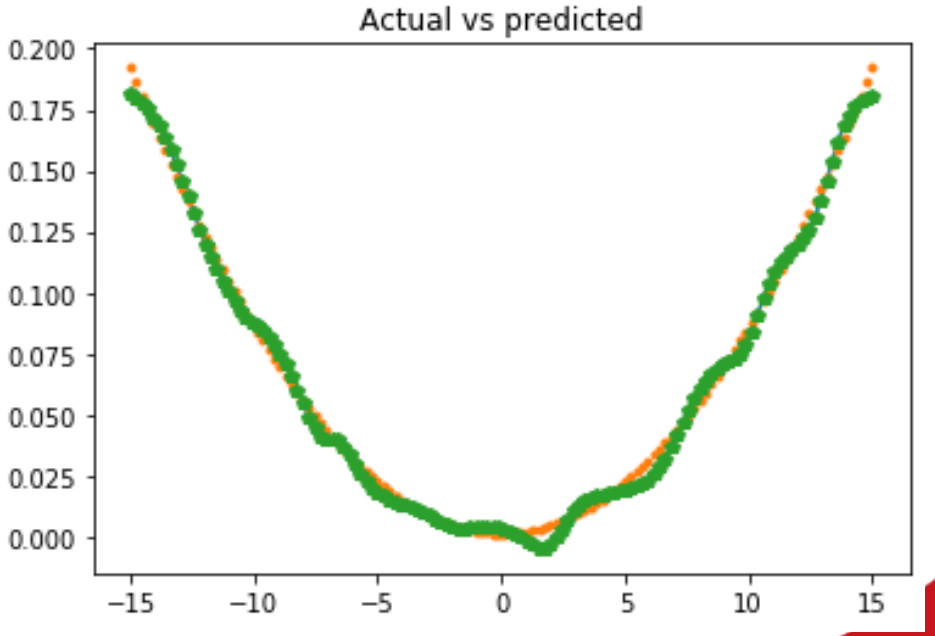
Δημιουργία πολυεπίπεδου νευρωνικού δικτύου
multilayer neural network

- Τι θα συμβεί αν μειώσουμε το όριο σφάλματος ή αυξήσουμε τις εποχές ή και τα δύο;
- Τι πρέπει να παρατηρήσουμε;

Δημιουργία πολυεπίπεδου νευρωνικού δικτύου
multilayer neural network

- Μείωση του στόχου σφάλματος από 0,01 σε 0,001.
- Ο στόχος επιτυγχάνεται, αλλά χρειάζονται και οι 2000 κύκλοι για την εκπαίδευση.





Δημιουργία πολυεπίπεδου νευρωνικού δικτύου
multilayer neural network

- Τι θα συμβεί αν αλλάξουμε την αρχιτεκτονική του νευρωνικού δικτύου;
  - π.χ. αύξηση του αριθμού των νευρώνων σε ένα από τα δύο κρυφά επίπεδα;
- Τι παρατηρούμε;

Optical Character Recognition (OCR)
παράδειγμα

- Κατεβάστε το αρχείο "`ocr.py`" από τη σελίδα του μαθήματος.
- Φορτώστε το στο Spyder IDE.
- Θα χρησιμοποιήσουμε το σύνολο δεδομένων που είναι διαθέσιμο στο http://ai.stanford.edu/~btaskar/ocr.
- Κατεβάστε το αρχείο δεδομένων "`letter.data`" το οποίο είναι επίσης διαθέσιμο στη σελίδα του μαθήματος.
- Τρέξτε το αρχείο "`ocr.py`".
- Αλλά πριν μπορεί να θέλετε να φορτώσετε αυτά τα δεδομένα και να οπτικοποιήσετε τους χαρακτήρες.
  - Ο κώδικας για αυτό βρίσκεται στο αρχείο "`character_visualizer.py`".

Εκπαίδευση νευρωνικού δικτύου

- Το "`character_visualizer.py`" χρησιμοποιεί το πακέτο cv2 από το OpenCV (ελεύθερος κώδικας για computer vision)
- Μεταβείτε στον φάκελο WinPython και εκτελέστε το **WinPython Command Prompt**.
- Στη γραμμή εντολών πληκτρολογήστε:
  - `pip3 install opencv-python`
- Εάν πρέπει να αναβαθμίσετε την έκδοση python, μπορείτε να πληκτρολογήσετε:
  - `python -m pip install --upgrade pip`

Διαδικτυακές πηγές

- 3Blue1Brown
  - Neural Networks series
- A basic introduction to Neural Networks
- The BackPropagation Network: Learning by Example
- Neural network tutorial: The back-propagation algorithm
  - Part I, Part 2
- Neural Networks Demystified
- The Care and Training of Your Backpropagation Neural Net
- Deep Neural Networks or Conviolutional Neural Networks (CNN)
  - How they work (brief, in depth, full course)