```
import scipy.io as sio
import sklearn.model_selection as sms
import sklearn.naive_bayes as snb
import sklearn.neighbors as sn
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mc
import math
import neurolab
import numpy
import pylab
from sklearn.datasets import load_digits
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn import svm

#1
x = numpy.linspace(0, 6, 20)
size = len(x)
y = numpy.sin(x)
inp = x.reshape(size,1)
net = neurolab.net.newff([[0, 6]],[5, 1])
net.trainf = neurolab.train.train_gd
error = net.train(inp, y.reshape(size, 1), epochs=500, show=100, goal=0.02)

x2 = numpy.linspace(0,6,150)
y2 = net.sim(x2.reshape(x2.size,1)).reshape(x2.size)
y3 = net.sim(inp).reshape(size)

pylab.plot(x2, y2, '-',x , y, '.', x, y3, 'p')
pylab.legend(['wartosc rzeczywista', 'wynik uczenia'])
pylab.show()

x = numpy.linspace(1, 2.5, 20)
size = len(x)
y = numpy.log(x) * 0.5

inp = x.reshape(size,1)
net = neurolab.net.newff([[1, 2.5]],[5, 1])
net.trainf = neurolab.train.train_gd
error = net.train(inp, y.reshape(size, 1), epochs=500, show=100, goal=0.02)

x2 = numpy.linspace(1,2.5,150)
y2 = net.sim(x2.reshape(x2.size,1)).reshape(x2.size)
y3 = net.sim(inp).reshape(size)

pylab.plot(x2, y2, '-',x , y, '.', x, y3, 'p')
pylab.legend(['wartosc rzeczywista', 'wynik uczenia'])
pylab.show()

x = numpy.linspace(1, 6, 20)
size = len(x)
y = numpy.cos(x) * x + numpy.log(x) * 0.3

inp = x.reshape(size,1)
net = neurolab.net.newff([[1, 6]],[5, 1])
net.trainf = neurolab.train.train_gd
error = net.train(inp, y.reshape(size, 1), epochs=500, show=100, goal=0.02)

x2 = numpy.linspace(1,6,150)
y2 = net.sim(x2.reshape(x2.size,1)).reshape(x2.size)
```

```
y3 = net.sim(inp).reshape(size)

pylab.plot(x2, y2, '-',x , y, '.', x, y3, 'p')
pylab.legend(['wartosc rzeczywista', 'wynik uczenia'])
pylab.show()
#1


#2
x = numpy.linspace(1, 2.5, 20)
size = len(x)
y = numpy.log(x) * 0.5

n1 = neurolab.net.newff([[0, 2.5]],[5, 1])
n2 = neurolab.net.newff([[-5, 2.5]],[5, 1])
n3 = neurolab.net.newff([[2, 2.5]],[5, 1])

n1.trainf = neurolab.train.train_gd
n2.trainf = neurolab.train.train_gd
n3.trainf = neurolab.train.train_gd

inp = x.reshape(size, 1)
tar = y.reshape(size, 1)
error1 = n1.train(inp, tar, epochs=500, show=100, goal=0.02)
error2 = n2.train(inp, tar, epochs=500, show=100, goal=0.02)
error3 = n3.train(inp, tar, epochs=500, show=100, goal=0.02)
#2


#3
x_train, x_test, y_train, y_test = train_test_split(load_digits().data,
load_digits().target, train_size=1000, test_size=500)

clf = MLPClassifier()
clf.fit(x_train, y_train)
print(clf.score(x_test, y_test))

clf = MLPClassifier(solver='lbfgs', alpha=0.5)
clf.fit(x_train, y_train).predict(x_test)
print(clf.score(x_test,y_test))

clf = MLPClassifier(solver='lbfgs', alpha=1e-5, random_state=1)
clf.fit(x_train, y_train).predict(x_test)
print(clf.score(x_test,y_test))
#3


#4
data = scipy.io.loadmat('mnist_012.mat')

x_train = data['train_images']
y_train = data['train_labels']
x_test = data['test_images']
y_test = data['test_labels']

ny, nx, nsamples = x_train.shape
x_train = x_train.reshape((nsamples, nx*ny))

ny, nx, nsamples = x_test.shape
x_test = x_test.reshape((nsamples, nx*ny))
```

```
c1 = MLPClassifier(hidden_layer_sizes=(50, 5)).fit(x_train,
y_train.ravel(nsamples))
c2 = MLPClassifier(hidden_layer_sizes=(100, 80)).fit(x_train,
y_train.ravel(nsamples))
c3 = MLPClassifier(hidden_layer_sizes=(5, 2)).fit(x_train,
y_train.ravel(nsamples))

print(c1.score(x_test, y_test))
print(c2.score(x_test, y_test))
print(c3.score(x_test, y_test))
#4


#5
d1 = scipy.io.loadmat('perceptron1.mat')
d2 = scipy.io.loadmat('perceptron1.mat')

net = neurolab.net.newp([[-1, 5],[0, 2]], 1)
error = net.train(d1['data'], d1['labels'], epochs=8, show=1, lr=0.1)
error2 = net.train(d2['data'], d2['labels'], epochs=8, show=1, lr=0.1)
#5


#6
data, target = sklearn.datasets.load_diabetes(True)

x_train, x_test, y_train, y_test = train_test_split(data,     target,
test_size=0.3, random_state=42)

tar = y_train.reshape(len(x_train), 1)

errorP = neurolab.net.newp([[-2, 2], [-2, 2], [-2, 2], [-2, 2], [-2, 2],
[-2, 2], [-2, 2], [-2, 2], [-2, 2], [-2, 2]], 1).train(x_train, tar)

errorMLP = neurolab.net.newff([[-2, 2],[-2, 2], [-2, 2],[-2, 2], [-2, 2],
[-2, 2], [-2, 2], [-2, 2], [-2, 2], [-2, 2]], [3, 1]).train(x_train,
tar)

print(svm.SVC().fit(x_train, tar.ravel()).predict(x_test))
#6


#7
data = scipy.io.loadmat('banana.mat')

train_data = data['train_data']
train_labels = data['train_labels']

perceptron = neurolab.net.newp([[-2, 2],[-2, 2]], 1)

errorP = perceptron.train(train_data, train_labels)
out = perceptron.sim(train_data)
#7


#8
cahracters = ['A', 'T', 'V']
target = numpy.asfarray([[0,0,1,0,0,     0,1,0,1,0,     0,1,1,1,0,
0,1,0,1,0,     0,1,0,1,0],       [1,1,1,1,1,     0,0,1,0,0,     0,0,1,0,0,
0,0,1,0,0,     0,0,1,0,0],       [1,0,0,0,1,     0,1,0,1,0,     0,1,0,1,0,
0,1,0,1,0,     0,0,1,0,0]])
```

```
target[target == 0] = -1

net = neurolab.net.newhop(target)
for i in range(len(target)):
    print(cahracters[i], (net.sim(target)[i] == target[i]).all())
test = numpy.asfarray([0,0,1,0,0,     0,1,0,1,0,     0,1,1,1,0,
0,1,0,1,0,     0,1,0,1,0])
test[test==0] = -1

print((net.sim([test])[0] == target[0]).all(), 'ilosc krokow',
len(net.layers[0].outs))

#Wystepuja utrudnienia przy rozpoznawaniu zaszumionych negatywow wzorcow
uczacych.
#8


#10
with open('kohonen1.mat') as f:
    data = f.read()

kohonen = data.split('\n')
d = []

for entry in kohonen:
    tmp = entry.split(' ')
    try:
        d.append([float(tmp[0]), float(tmp[1])])
    except ValueError, e:
        break

net = neurolab.net.newc([[0.0, 1.0],[0.0, 1.0]], 4)
error = net.train(d, epochs=200, show=20)
#10
```