**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*