Perceptron

The perceptron network consists of a single layer of M perceptron neurons connected to N inputs through a set of weights $w_{i,j}$,. The network indices i and j indicate that $w_{i,j}$ is the strength of the connection from the j-th input to the i-th neuron.

`net = newp(P,N,TF,LM)` – the statement creates the preceptron network (underlined parameters are optional)

P – matrix defining minimum and maximum value for each input
N – number of neurons
TF – transfer function (**`hardlim`** – is default, **`hardlims`**)
LM – network learning method (**`learnp`** – is default)

An example of creating and learning of the simple perceptron neuron which realizes AND logic gate:

```
inp = [0 0 1 1; 0 1 0 1];            inputs of the neuron
out = [0 0 0 1];                     target output of the neuron
net = newp(minmax(inp), 1);;         neuron creation
y = sim(net, inp);                   simulation of the neuron work
figure(1); plot(abs(y-out));         error plot
title('Error before learning');

net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
net = init(net);                     set random weights in the beginning
net.trainParam.epochs = 50;          maximum number of learning epochs
net = train(net, inp, out);          neuron learning
y = sim(net, inp);                   simulation of the neuron work
figure(2); plot(abs(y-out));         error plot
title('Error after learning');
net.iw{1,1}                          weights of the neuron
net.b{1}                             bias of the neuron

figure(3); plotpv(inp, out);         data plot
plotpc(net.iw{1,1}, net.b{1});       separation line
```

An example of creating and learning of the simple perceptron neuron for data in text files:

```
load percep_i.txt
load percep_o.txt
inp = percep_i’;                  inputs of the neuron
out = percep_o’;                  inputs of the neuron

net = newp(minmax(inp), 1);;      neuron creation
y = sim(net, inp);                simulation of the neuron work
figure(1); plot(abs(y-out));      error plot
title('Error before learning');

net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
net = init(net);                  set random weights in the beginning
net.trainParam.epochs = 50;       maximum number of learning epochs
net = train(net, inp, out);       neuron learning
y = sim(net, inp);                simulation of the neuron work
figure(2); plot(abs(y-out));      error plot
title('Error after learning');
net.iw{1,1}                       weights of the neuron
net.b{1}                          bias of the neuron

figure(3); plotpv(inp, out);      data plot
plotpc(net.iw{1,1}, net.b{1});    separation line

sim(net, [4;6])                   test which class point (4,6) belongs to
```

Tasks to do:

1. Prepare data, create the preceptron neuron and learn it to work as the logic gate OR and XOR, plot data and the separation line for the neuron before and after learning.
2. Read data from files (choose 3 of them), learn the neuron, plot data and the separation line for the neuron before and after learning:
    - percep, data_a – 2 inputs, separable case,
    - data_1, data_2, data_3 – 2 inputs, not separable case,
    - data3d_a – 3 inputs, separable case,
    - data3d_1, data3d_2, data3d_3 – 3 inputs, not separable case,
    - data8d_a – 8 inputs, separable case,
    - data8d_1, data8d_2, data8d_3 – 3 inputs, not separable case,
3. Create yourself data, describing the real classification problem (technical, economic or other). Create the preceptron neuron and learn it. Plot data and the separation line for the neuron before and after learning.
4. Create yourself a MATLAB function, reading data from files and learning neuron using the delta rule.

Prepare the report.

Present the results of neuron learning: weights of neurons, plots illustrating the process of learning and its results, programs created during experiments.