



数学建模培训课程论文

第十一次作业

班级：电子科学与技术21-2

姓名：田博松

学号：202111040246

序号：143

组别：A

2022年 7 月

Newp---单层感知器相关函数详解

>> p=[-1,1;-1,1]

p =

-1 1

-1 1

>> t=1;

>> net=newp(p,t)

net =

Neural Network

name: 'Custom Neural Network'

userdata: (your custom info)

dimensions:

numInputs: 1

numLayers: 1

numOutputs: 1

numInputDelays: 0

numLayerDelays: 0

numFeedbackDelays: 0

numWeightElements: 3

sampleTime: 1

connections:

biasConnect: true

inputConnect: true

layerConnect: false

outputConnect: true

subobjects:

input: Equivalent to inputs{1}

output: Equivalent to outputs{1}

inputs: {1x1 cell array of 1 input}

layers: {1x1 cell array of 1 layer}

outputs: {1x1 cell array of 1 output}

biases: {1x1 cell array of 1 bias}

inputWeights: {1x1 cell array of 1 weight}

layerWeights: {1x1 cell array of 0 weights}

functions:

adaptFcn: 'adaptwb'

adaptParam: (none)

derivFcn: 'defaultderiv'

divideFcn: (none)

divideParam: (none)

divideMode: 'sample'

initFcn: 'initlay'

performFcn: 'mae'

performParam: .regularization, .normalization

plotFcns: {'plotperform', 'plottrainstate'}

plotParams: {1x2 cell array of 2 params}

trainFcn: 'trainc'

trainParam: .showWindow, .showCommandLine, .show, .epochs,

.time, .goal, .max\_fail

weight and bias values:

IW: {1x1 cell} containing 1 input weight matrix

LW: {1x1 cell} containing 0 layer weight matrices

b: {1x1 cell} containing 1 bias vector

methods:

adapt: Learn while in continuous use

configure: Configure inputs & outputs

gensim: Generate Simulink model

init: Initialize weights & biases

perform: Calculate performance

sim: Evaluate network outputs given inputs

train: Train network with examples

view: View diagram

unconfigure: Unconfigure inputs & outputs

evaluate: outputs = net(inputs)

>> P=[0,0,1,1;0,1,0,1]

P =

0 0 1 1

0 1 0 1

>> T=[0,1,1,1]

T =

0 1 1 1

>> net=train(net,P,T)

net =

Neural Network

name: 'Custom Neural Network'

userdata: (your custom info)

dimensions:

numInputs: 1

numLayers: 1

numOutputs: 1

numInputDelays: 0

numLayerDelays: 0

numFeedbackDelays: 0

numWeightElements: 3

sampleTime: 1

connections:

biasConnect: true

inputConnect: true

layerConnect: false

outputConnect: true

subobjects:

input: Equivalent to inputs{1}

output: Equivalent to outputs{1}

inputs: {1x1 cell array of 1 input}

layers: {1x1 cell array of 1 layer}

outputs: {1x1 cell array of 1 output}

biases: {1x1 cell array of 1 bias}

inputWeights: {1x1 cell array of 1 weight}

layerWeights: {1x1 cell array of 0 weights}

functions:

adaptFcn: 'adaptwb'

adaptParam: (none)

derivFcn: 'defaultderiv'

divideFcn: (none)

divideParam: (none)

divideMode: 'sample'

initFcn: 'initlay'

performFcn: 'mae'

performParam: .regularization, .normalization

plotFcns: {'plotperform', 'plottrainstate'}

plotParams: {1x2 cell array of 2 params}

trainFcn: 'trainc'

trainParam: .showWindow, .showCommandLine, .show, .epochs,

.time, .goal, .max\_fail

weight and bias values:

IW: {1x1 cell} containing 1 input weight matrix

LW: {1x1 cell} containing 0 layer weight matrices

b: {1x1 cell} containing 1 bias vector

methods:

adapt: Learn while in continuous use

configure: Configure inputs & outputs

gensim: Generate Simulink model

init: Initialize weights & biases

perform: Calculate performance

sim: Evaluate network outputs given inputs

train: Train network with examples

view: View diagram

unconfigure: Unconfigure inputs & outputs

evaluate: outputs = net(inputs)

>> newP=[0,0.9]';

>> newT=sim(net,newP)

newT =

0

>> newP=[0.9,0.9]';

>> newT=sim(net,newP)

newT =

1

newT=sim(net,P)

newT =

0 1 1 1

Sim---单层感知器相关函数详解

>> net=newp([-2,2;-2,2],1)

net =

Neural Network

name: 'Custom Neural Network'

userdata: (your custom info)

dimensions:

numInputs: 1

numLayers: 1

numOutputs: 1

numInputDelays: 0

numLayerDelays: 0

numFeedbackDelays: 0

numWeightElements: 3

sampleTime: 1

connections:

biasConnect: true

inputConnect: true

layerConnect: false

outputConnect: true

subobjects:

input: Equivalent to inputs{1}

output: Equivalent to outputs{1}

inputs: {1x1 cell array of 1 input}

layers: {1x1 cell array of 1 layer}

outputs: {1x1 cell array of 1 output}

biases: {1x1 cell array of 1 bias}

inputWeights: {1x1 cell array of 1 weight}

layerWeights: {1x1 cell array of 0 weights}

functions:

adaptFcn: 'adaptwb'

adaptParam: (none)

derivFcn: 'defaultderiv'

divideFcn: (none)

divideParam: (none)

divideMode: 'sample'

initFcn: 'initlay'

performFcn: 'mae'

performParam: .regularization, .normalization

plotFcns: {'plotperform', 'plottrainstate'}

plotParams: {1x2 cell array of 2 params}

trainFcn: 'trainc'

trainParam: .showWindow, .showCommandLine, .show, .epochs,

.time, .goal, .max\_fail

weight and bias values:

IW: {1x1 cell} containing 1 input weight matrix

LW: {1x1 cell} containing 0 layer weight matrices

b: {1x1 cell} containing 1 bias vector

methods:

adapt: Learn while in continuous use

configure: Configure inputs & outputs

gensim: Generate Simulink model

init: Initialize weights & biases

perform: Calculate performance

sim: Evaluate network outputs given inputs

train: Train network with examples

view: View diagram

unconfigure: Unconfigure inputs & outputs

evaluate: outputs = net(inputs)

>> P=[0,0,1,1;0,1,0,1];

>> T=[0,0,1,1];

>> net=train(net,P,T);

>> Y=sim(net,P)

Y =

0 0 1 1

>> Y=net(P)

Y =

0 0 1 1

Hardlims---单层感知器相关函数详解

>> figure;

>> subplot(2,1,1);

>> n=-5;0.01:5;

>> plot(n,hardlim(n),'LineWidth',2);

>> subplot(2,1,2);

>> plot(n,hardlims(n),'r','LineWidth',2);

>> title('hardlims');

>> subplot(2,1,1);

>> title('hardlim');

>>