

Vorlesung Computational Intelligence:

Zusatzmaterial zum Selbststudium:
Künstliche Neuronale Netze in MATLAB

Ralf Mikut, Wilfried Jakob, Markus Reischl

Karlsruher Institut für Technologie, Institut für Angewandte Informatik

E-Mail: ralf.mikut@kit.edu, wilfried.jakob@kit.edu

jeden Donnerstag 14:00-15:30 Uhr, Nusselt-Hörsaal

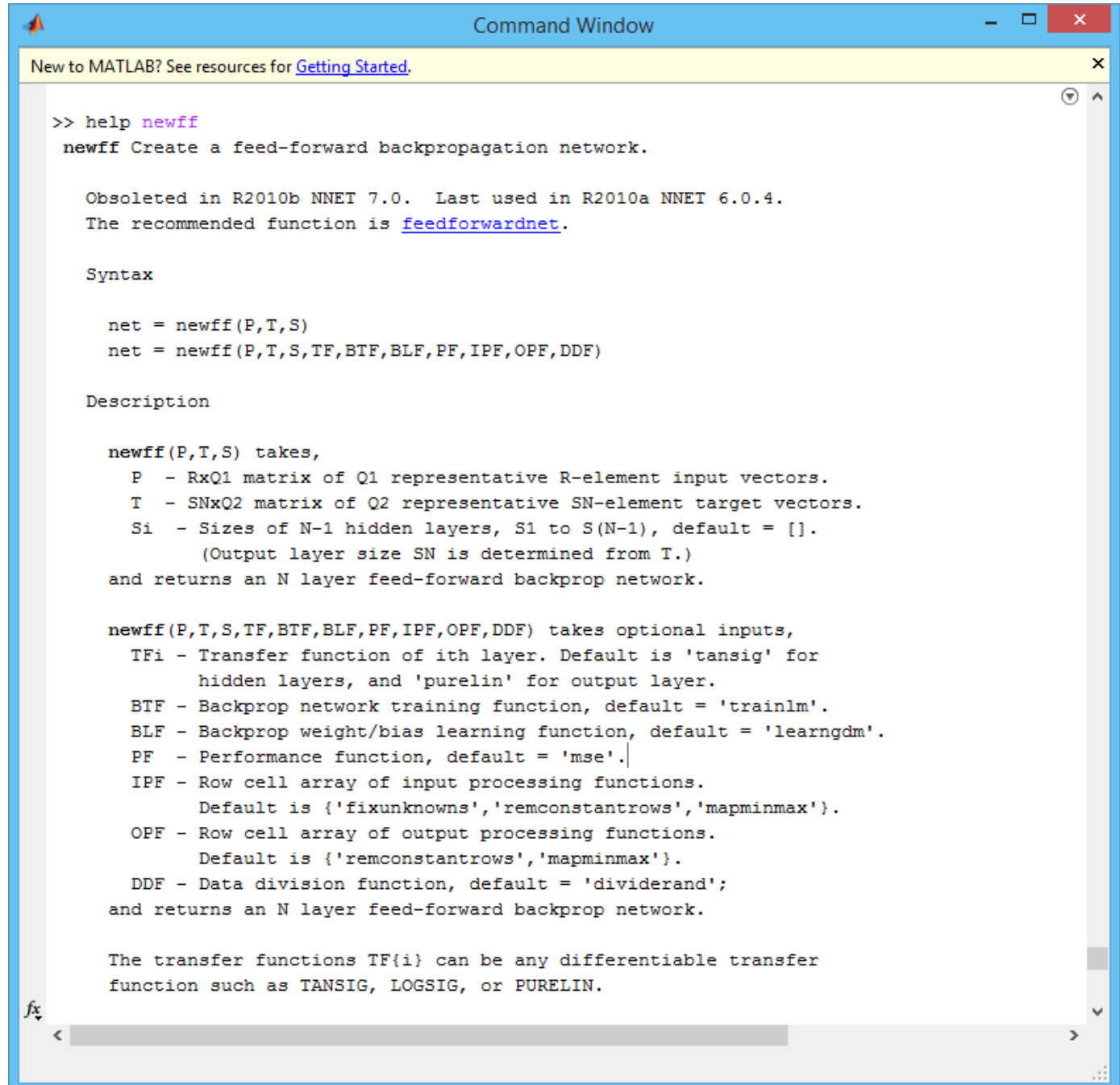
Beispiel MLP-Netze: MATLAB

MATLAB-Befehl für MLPs:

- `newff(.)`
- `feedforwardnet(.)`

Beschreibung:

- `help nnet`
- `help newff`



```
Command Window
New to MATLAB? See resources for Getting Started.

>> help newff
newff Create a feed-forward backpropagation network.

Obsolated in R2010b NNET 7.0. Last used in R2010a NNET 6.0.4.
The recommended function is feedforwardnet.

Syntax

net = newff(P,T,S)
net = newff(P,T,S,TF,BTF,BLF,PF,IPF,OPF,DDF)

Description

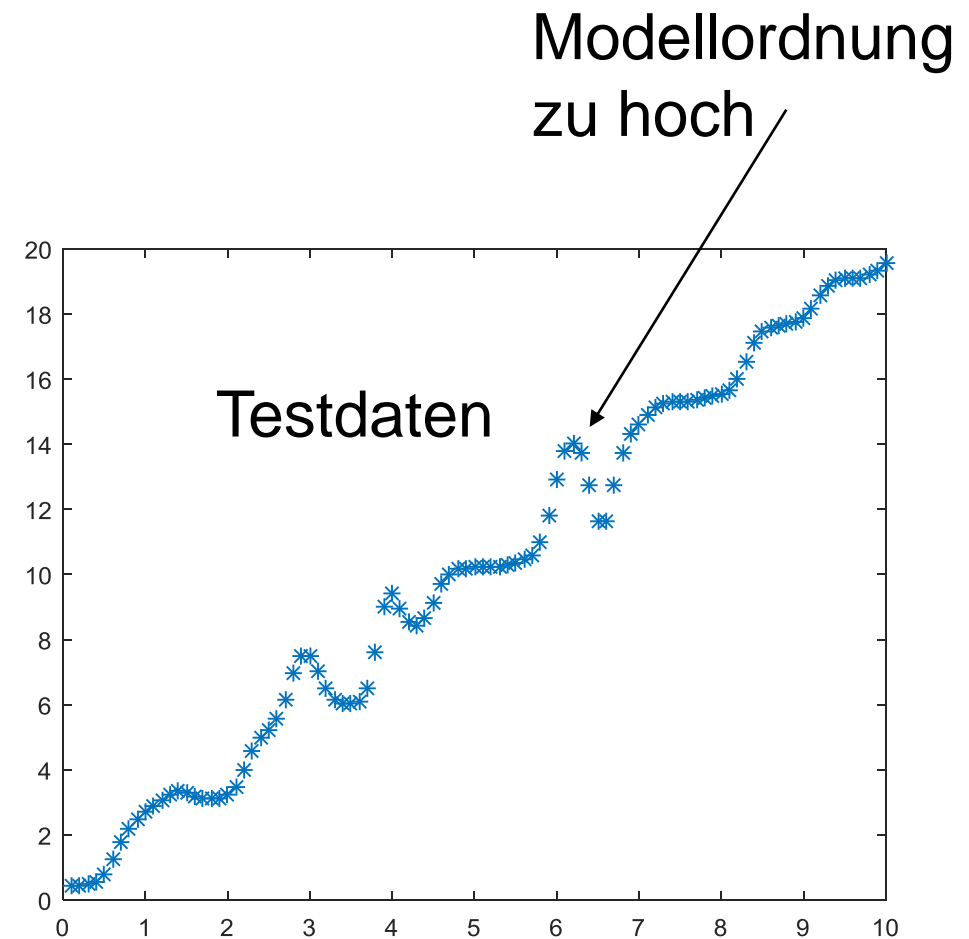
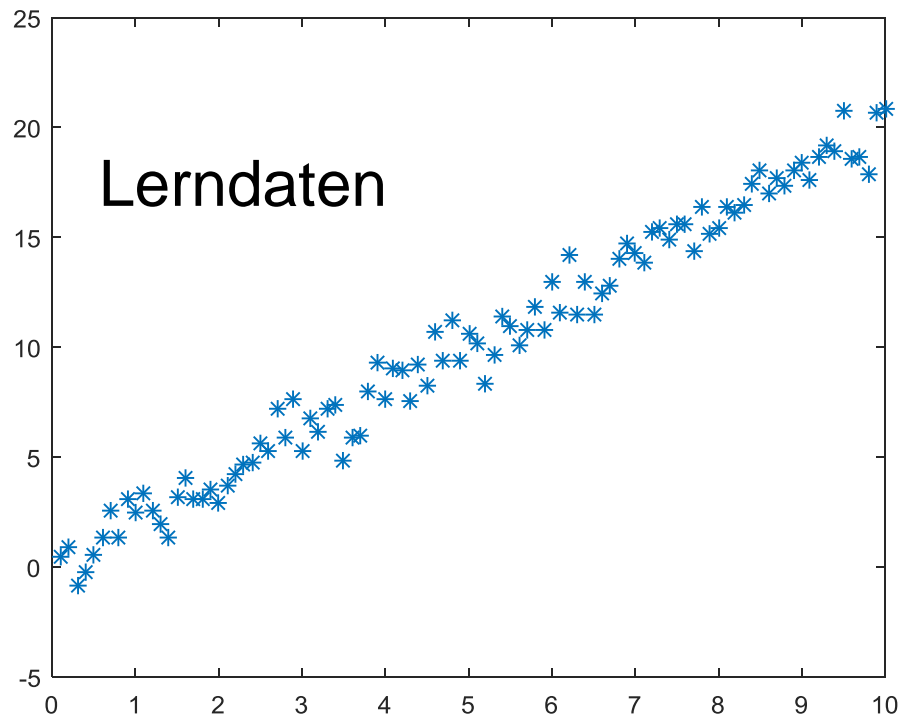
newff(P,T,S) takes,
    P - RxQ1 matrix of Q1 representative R-element input vectors.
    T - SNxQ2 matrix of Q2 representative SN-element target vectors.
    Si - Sizes of N-1 hidden layers, S1 to S(N-1), default = [].
        (Output layer size SN is determined from T.)
and returns an N layer feed-forward backprop network.

newff(P,T,S,TF,BTF,BLF,PF,IPF,OPF,DDF) takes optional inputs,
    TFi - Transfer function of ith layer. Default is 'tansig' for
        hidden layers, and 'purelin' for output layer.
    BTF - Backprop network training function, default = 'trainlm'.
    BLF - Backprop weight/bias learning function, default = 'learngdm'.
    PF - Performance function, default = 'mse'.
    IPF - Row cell array of input processing functions.
        Default is {'fixunknowns','remconstantrows','mapminmax'}.
    OPF - Row cell array of output processing functions.
        Default is {'remconstantrows','mapminmax'}.
    DDF - Data division function, default = 'dividerand';
and returns an N layer feed-forward backprop network.

The transfer functions TF{i} can be any differentiable transfer
function such as TANSIG, LOGSIG, or PURELIN.
```

Beispiel MLP-Netze zur Regression

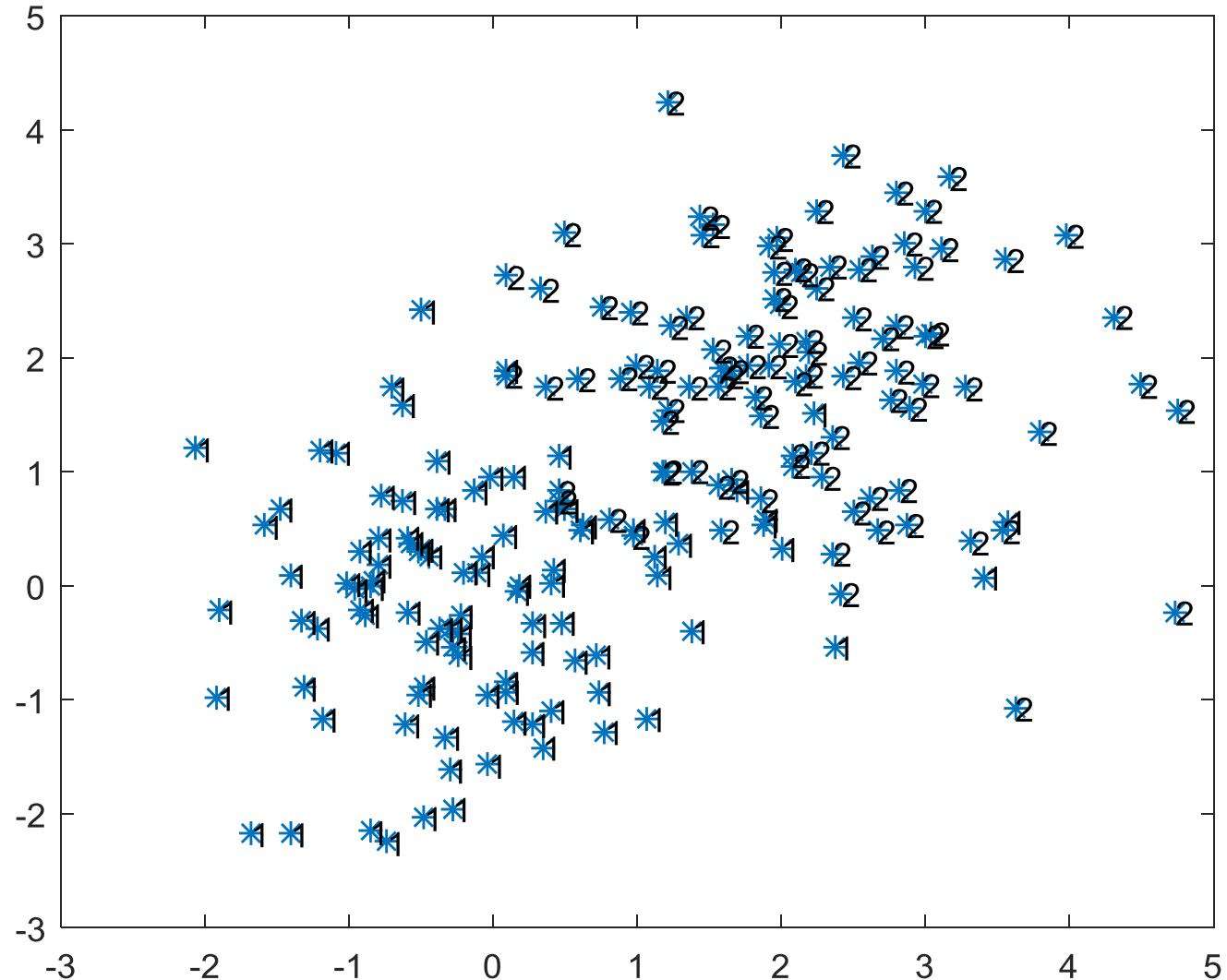
- `anz = 100; x = [0.1:0.1:10]; y = 2*x + randn(1, length(x));`
- `figure; plot(x, y, '*');`
- `net = newff(x, y, 3);`
- `net = train(net, x, y);`
- `y_sim = net(x);`
- `figure; plot(x, y_sim, '*')`



Beispiel MLP-Netze zur Klassifikation

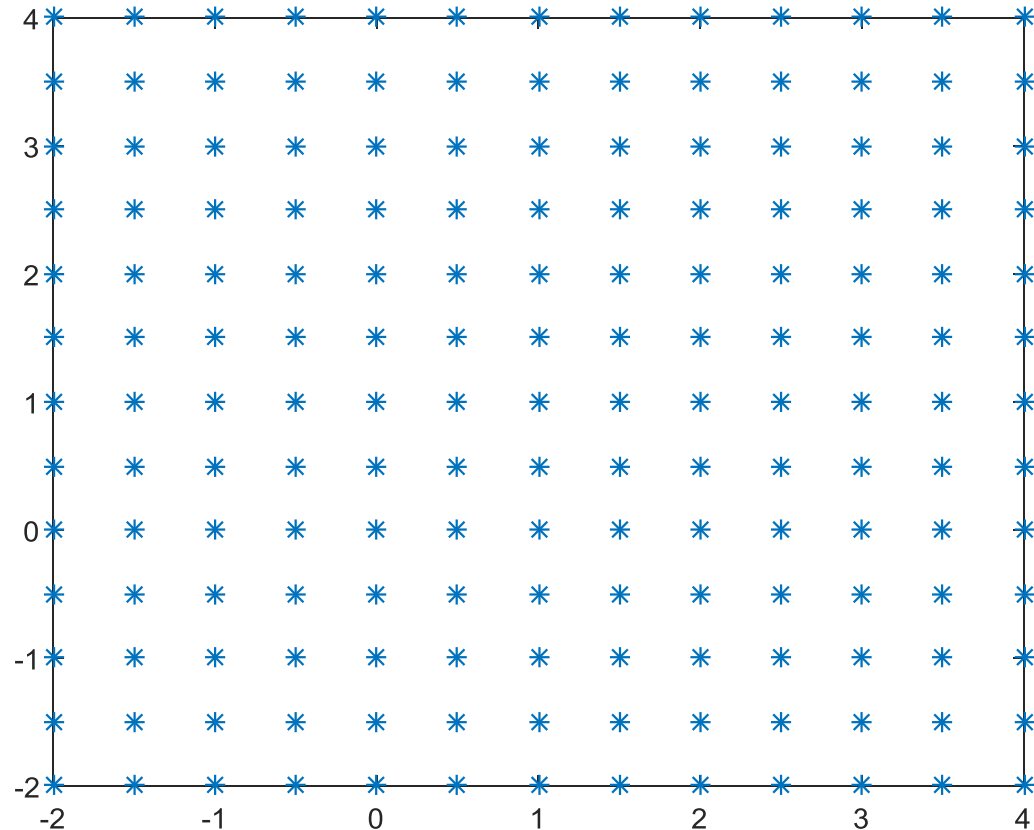
Lerndaten

- `anz = 100;`
`dat = [randn(anz,2);`
`randn(anz,2) + 2];`
- `code = [ones(anz,1);`
`2*ones(anz,1)];`
- `figure; plot(dat(:,1),`
`dat(:,2), '*');`
- `for i=1:size(dat,1),`
`text(dat(i,1), dat(i,2),`
`sprintf('%i', code(i)));`
`end;`



Testdaten

- `dat_test = []; for x=-2:0.5:4, for y = -2:0.5:4; dat_test = [dat_test; [x y]]; end; end;`
- `figure; plot(dat_test(:,1), dat_test(:,2), '*')`



- `net = newff(dat',code',20);`
- `net = train(net, dat',code');`
- `outputs = net(dat_test');`
- `figure; set(gca,'xlim',[-2 4], 'ylim', [-2 4]);`
- `for i=1:size(dat_test,1), text(dat_test(i,1), dat_test(i,2),
sprintf('%1.1f', outputs(i))); end;`

The grid data is as follows:

	-2	-1	0	1	2	3	4
4	2	2	2	2	2	2	2
3	2	2	2	2	2	2	2
2	1	1	1	2	2	2	2
1	1	1	1	1	2	2	2
0	1	1	1	1	1	2	2
-1	1	1	1	1	1	1	2
-2	1	1	1	1	1	1	1

Trainieren und testen (2 Ausgangsneuronen):

- `anz = 100; dat = [randn(anz,2); randn(anz,2) + 2];`
- `code = [ones(anz,1); 2*ones(anz,1)];`
- `code_n = [2-code, code-1];`
- `net1 = newff(dat',code_n',20);`
- `net1 = train(net1, dat',code_n');`
- `outputs1 = net1(dat_test');`

- `[tmp, out] = max(outputs1,[],1)`
- `figure; set(gca,'xlim',[-2 4],
'ylim', [-2 4]);`
- `for i=1:size(dat_test,1),
text(dat_test(i,1),
dat_test(i,2), sprintf('%i',
out (i))); end;`

