Úvod do práce s Neural Network Toolboxem, MATLAB

Podporou pro řešení úloh pomocí umělých neuronových sítí je NN-Toolbox Matlabu. Současné verze umožňují práci s jednoduchými modely, které tvoří základ práce s UNS, ale také s mnoha novými metodami, které byly vyvinuty v posledních letech. Pomocí demonstračních úloh je možné se seznámit s jednotlivými fázemi návrhu sítě, jejího trénování a zpracování výsledků. Toolbox také obsahuje aplikační úlohy. Jeho úspěšné používání předpokládá základní znalosti z programování a z teorie neuronových sítí. V práci uživateli pomáhá Help, který obsahuje všechny potřebné informace. Přesto na tomto místě uvádím alespoň základní pojmy. Podrobnější seznámení s NN-Toolboxem zůstává na uživateli. Pouze experimentování s neuronovými sítěmi umožní porozumět teorii a přispěje k úspěšným aplikacím.

Uvedu zde přehled funkcí, které se v této variantě NN-Toolboxu vyskytují, abych umožnila snazší orientaci při využívání helpu. Na tomto místě uvedu jen názvy hlavních kategorií, pouze u několika z nich přidám více podrobností.

Postup:

- MATLAB
- APPS
  - **Neural Net Clustering**
    Funkce**:** **selforgmap**
    trainbu
    learnsomb
  - **Neural Net Fitting**
    Funkce**:** **fitnet**
    trainlm
    trainscg
  - **Neural Net Pattern Recognition**
    Funkce**:** **patternnet**
    trainscg
  - **Neural Net Time Series**

- Help ---- Search Documentation
  - Neural Network Toolbox
  - Functions
    - Function Approximation and Nonlinear Regression
    - Pattern Recognition and Classification
    - Clustering
    - Time Series and Dynamic Systems
    - Neural Network Control Systems
    - Define Neural Network Architectures

Pro rozdělení funkcí podle typu UNS voláme „Help“ nebo „?**“,** Neural Network Toolbox, Neural Network Toolbox Examples, Functions

Functions in Neural Network Toolbox

Function Approximation and Nonlinear Regression

nnstart	Neural network getting started GUI
nftool	Neural network fitting tool
view	View neural network
fitnet	Function fitting neural network
feedforwardnet	Feedforward neural network
cascadeforwardnet	Cascade-forward neural network
train	Train neural network
trainlm	Levenberg-Marquardt backpropagation
trainbr	Bayesian regularization backpropagation
trainscg	Scaled conjugate gradient backpropagation
trainrp	Resilient backpropagation
mse	Mean squared normalized error performance function
regression	Linear regression
ploterrhist	Plot error histogram
plotfit	Plot function fit
plotperform	Plot network performance
plotregression	Plot linear regression
plottrainstate	Plot training state values
genFunction	Generate MATLAB function for simulating neural network

Pattern Recognition and Classification

nnstart	Neural network getting started GUI
nprtool	Neural network pattern recognition tool
view	View neural network
patternnet	Pattern recognition network
lvqnet	Learning vector quantization neural network
train	Train neural network
trainlm	Levenberg-Marquardt backpropagation
trainbr	Bayesian regularization backpropagation
trainscg	Scaled conjugate gradient backpropagation
trainrp	Resilient backpropagation
mse	Mean squared normalized error performance function
regression	Linear regression
roc	Receiver operating characteristic
plotconfusion	Plot classification confusion matrix
ploterrhist	Plot error histogram
plotperform	Plot network performance
plotregression	Plot linear regression
plotroc	Plot receiver operating characteristic
plottrainstate	Plot training state values
crossentropy	Neural network performance
genFunction	Generate MATLAB function for simulating neural network

Clustering

Self-Organizing Maps

nnstart	Neural network getting started GUI
nctool	Neural network classification or clustering tool
view	View neural network
selforgmap	Self-organizing map
train	Train neural network
plotsomhits	Plot self-organizing map sample hits
plotsomnc	Plot self-organizing map neighbor connections
plotsomnd	Plot self-organizing map neighbor distances
plotsomplanes	Plot self-organizing map weight planes
plotsompos	Plot self-organizing map weight positions
plotsomtop	Plot self-organizing map topology
genFunction	Generate MATLAB function for simulating neural network

Competitive Layers

competlayer	Competitive layer
view	View neural network
train	Train neural network
trainru	Unsupervised random order weight/bias training
learnk	Kohonen weight learning function
learncon	Conscience bias learning function
genFunction	Generate MATLAB function for simulating neural network

Time Series and Dynamic Systems

Modeling and Prediction with NARX and Time-Delay Networks

nnstart	Neural network getting started GUI
ntstool	Neural network time series tool
view	View neural network
timedelaynet	Time delay neural network
narxnet	Nonlinear autoregressive neural network with external input
narnet	Nonlinear autoregressive neural network
layrecnet	Layer recurrent neural network
distdelaynet	Distributed delay network
train	Train neural network
gensim	Generate Simulink block for neural network simulation
adddelay	Add delay to neural network response
removedelay	Remove delay to neural network's response
closeloop	Convert neural network open-loop feedback to closed loop
openloop	Convert neural network closed-loop feedback to open loop
ploterrhist	Plot error histogram
plotinerrcorr	Plot input to error time-series cross-correlation
plotregression	Plot linear regression
plotresponse	Plot dynamic network time series response
ploterrcorr	Plot autocorrelation of error time series
genFunction	Generate MATLAB function for simulating neural network

Creating Simulink Models

gensim	Generate Simulink block for neural network simulation
setsiminit	Set neural network Simulink block initial conditions
getsiminit	Get Simulink neural network block initial input and layer delays states
sim2nndata	Convert Simulink time series to neural network data
nndata2sim	Convert neural network data to Simulink time series

Define Neural Network Architectures

network	Create custom neural network

Neural Network Toolbox Examples

Function Fitting and Approximation
Pattern Recognition and Classification
Clustering
Dynamic Modeling and Prediction
Control Systems and Simulink
Self-organizing Networks
Adaptive Linear Filters
Radial Basis Networks
LVQ Networks
Simple Applications
Hopfield Networks
Perceptrons
Other Examples

Pro snadnější orientaci v NN-Toolboxu je v tabulce D2 uveden rozdíl mezi značením funkcí a proměnných mezi teoretickými úvahami ve skriptu a v monografii (korespondují s velkou většinou literárních pramenů týkajících se umělých neuronových sítí a jejich aplikací ve světě) a značením v toolboxu.

Název - UNS	Značení - UNS	Značení - MATLAB
vstupy do neuronu	x_i, i = 1,…,n	p_i, i = 1,…,R
obvodové funkce	u	n
výstup z neuronu	y	a
aktivační funkce	f	f

Tabulka D2: Značení funkcí a proměnných ve skriptu a NN-Toolboxu v MATLABu.

V následujících řádcích si ukážeme, jak pracovat s toolboxem. Pro jednoduchý neuron platí:

- vstup (*input*) je označen ‘p‘ (obecně je to vektor o R-elementech)
- váha (*weight*) je označena ‘w‘ (obecně matice vah **W**); je-li použita matice vah, představují řádky rozložení neuronů ve vrstvě, sloupce určují připojené vstupy; např. $w_{1,2}$ určuje, že signál je šířen ze vstupu 2 do neuronu 1)

- aktivační (přenosová) funkce (*transfer function*) je označena ‘f’
- výstupní hodnota (*output*) je označen ‘a’
- práh (*bias*) je označen ‘b’.

Pro vrstevnatou UNS platí, že vstupem je buď vektor **p** s R-vstupy, resp. matice vstupů s Q-vektory; pak je dimenze vstupní matice (R,Q). Matice vah $\mathbf{W^1}$ má hodnost (S^1,R), kde S^1 je počet neuronů v první skryté vrstvě a R je počet vstupů.

$$\boldsymbol{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,R} \\ w_{2,1} & w_{2,2} & \dots & w_{2,R} \\ \dots & \dots & \dots & \dots \\ w_{S,1} & w_{S,2} & \dots & w_{S,R} \end{bmatrix}$$

Matice vah $\boldsymbol{W^2}$ má hodnost (S^2, S^1), kde S^1 je počet neuronů v první skryté vrstvě a S^2 je počet neuronů ve druhé skryté vrstvě, resp. ve výstupní vrstvě. Z každé vrstvy vystupují vektory ***a*** a pro jednotlivé neurony vrstvy jsou definovány prahy b**,** které tvoří vektor ***b***, oba mají S^i-elementů. V následujících řádcích ukážeme způsob zápisu vektorů a matic.

```
W =  [1 2]          net.IW{1,1} = [1,2];
b  = [0]            net.b{1} = 0;
```

Vstupní vektory .

$p_1 = [1\ 2]^T$ $p_2 = [2\ 1]^T$ $p_3 = [2\ 3]^T$ $p_4 = [3\ 1]^T$

P = $[p_1, p_2, p_3, p_4]$ = [1 2 2 3; 2 1 3 1];

Výstup z neuronové sítě bude zapsán jako

```
a = sim (net, P)
a =
    5  4  8  5
```

Vstupní váhy se v této verzi NN-Toolboxu označují jako *input weights* – např. $\mathbf{IW^{1,2}}$ (ze vstupu 2 do neuronu1). V této verzi MATLABu platí

$\mathbf{IW^{1,2}}$ ⟶ net.IW{1,2}

Pro váhy ve vrstvě je zavedeno označení *layer weights* – **LW.**

Pro potenciál neuronu lze např. psát:

```
n{1} = net.IW{1,2}*p + net.b{1}
```

Příklad zápisu pro 3-vrstvé (podle značení MATLABu) UNS:

- pro výstupy z jednotlivých vrstev

$a^1 = f^1 (IW^{1,1} p + b^1)$
$a^2 = f^2 (IW^{2,1} p + b^2)$
$a^3 = f^3 (IW^{3,1} p + b^3)$

- pro výstup z celé sítě

$$a^3 = f^3 (IW^{3,2} f^2 (LW^{2,1} f^1 (IW^{1,1} p + b^1) + b^2) + b^3)$$

Počet vstupů je R^1, počet neuronů v první (skryté) vrstvě je S^1, počet neuronů v druhé (skryté) vrstvě je S^2 atd. Tedy např. 3-vrstvá UNS má 1 výstup (layer 3) a 2 skryté vrstvy (layer 1 a layer 2). Vstupy se nepovažují za vrstvu.[1]

Rozlišují se dva základní typy vstupních vektorů, a to vektory vstupující do sítě v jednom čase – *concurrent inputs* (nezáleží na pořadí vektorů) a vektory vstupující do sítě sekvenčně – *sequential inputs* (pořadí vektorů je rozhodující).

Vykreslení aktivačních funkcí

```
n = -5: 0.1:5;          n = -5: 0.1:5;          n = -5: 0.1:5;          n = -5: 0.1:5;
a = purelin(n);         a = satlin(n);          a = satlins(n);         a = tansig(n);
plot (n, a)             plot (n, a)             plot (n, a)             plot (n, a)

n = -5: 0.1:5;          n = -5: 0.1:5;          n = -5: 0.1:5;
a = hardlim(n);         a = radbas(n);          a = logsig(n);
plot (n, a)             plot (n, a)             plot (n, a)
```

[1] Rozdílné značení vrstev v MATLABu od skript.