Studia Doktoranckie IBS PAN

nt. „Techniki informacyjne – teoria i zastosowania”

WYKŁAD i Seminarium

nt. Modelowanie rozwoju systemów w środowisku MATLABA i Simulinka

Prof. nadzw. dr hab. inż. Jerzy Tchórzewski, jtchorzewski@interia.pl; Jerzy.Tchorzewski@uph.edu.pl

Warszawa, 28 stycznia 2017 r.,

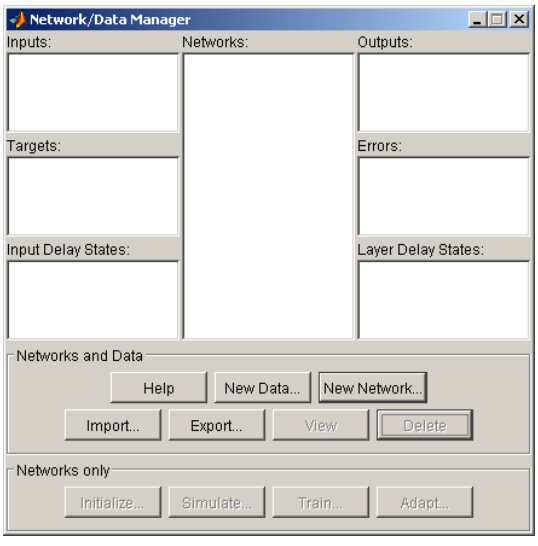
Blok tematyczny II – Sztuczne sieci neuronowe

(środowisko MATLAB i Simulink z wykorzystaniem Neural Network Toolbox),

Uwagi realizacyjne wstępne:

1. Przed zajęciami każdy doktorant powinien zainstalować pakiet MATLAB i Simulink (trial firmy MathWorks[1]) z toolboxami: Symulink, Symbolic Math T., System Identification T., Control System Toolbox, Neural Network Toolbox, itp.
2. Przygotować dane do ćwiczeń laboratoryjnych w środowisku Excel zawierające cztery-pięć wielkości wejściowych i dwie-trzy wielkości wyjściowe w obszarze zainteresowań poszczególnych doktorantów. Wartości poszczególnych zmiennych wygodnie jest zapisać kolumnowo (ok. 100 wartości).
3. Najpierw należy utworzyć w środowisku MATLAB dwie macierze, jedną zawierającą dane wejściowe, a druga zawierającą dane wyjściowe, o identycznej liczbie kolumn (próbki pomiarowe) oraz dowolnej liczbie wierszy (wielkości wejściowe i wielkości wyjściowe), poleceniem: >unntwe=uwe’ oraz ynntwy=ywy’ (w wyniku transformacji macierzy stworzonych dla potrzeb identyfikacji za pomocą SIT) – na wszelki wypadek należy przygotować też znormalizowane dane.
4. **Zapoznać się z toolbox-em MATLAB-a pn. Neural Network Toolbox, a w szczególności zwrócić uwagę m.in. na:**

   4.1 Rozpoczęcie pracy w Neural Network Toolboxie (dalej: NNT) poleceniem >nntool (interfejs graficzny GUI, poznać możliwości biblioteki programów NNT) – rys. 1. Interfejs graficzny GUI upraszcza pracę użytkownika oraz przyspiesza obliczenia. Potrzebne funkcje są dostępne w menu. Pliki wejściowe można importować do GUI NNT (Import), a uzyskiwane sztuczne sieci neuronowe eksportować do przestrzeni roboczej MATLABA (export).



Rys. 1. GUI Neural Network Toolbox-a. Najważniejsze oznaczenia:

Inputs – pole do wprowadzania zmiennych wejściowych uczących (u),

Targets – pole do wprowadzania zmiennych wyjściowych uczących (nauczyciela),

New Network – przycisk uruchamiający tworzenie architektury SSN,

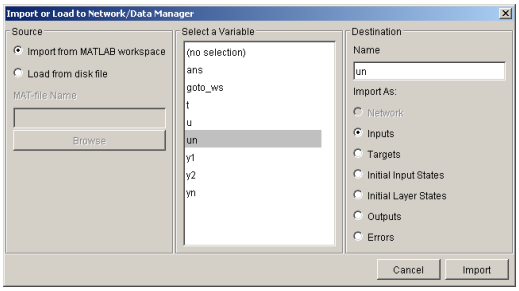
Import – przycisk do importowania danych z Workspace,

Eksport – przycisk do eksportowania wygenerowanego modelu do Workspace,

View – przycisk do oglądania architektury wygenerowanej SSN.

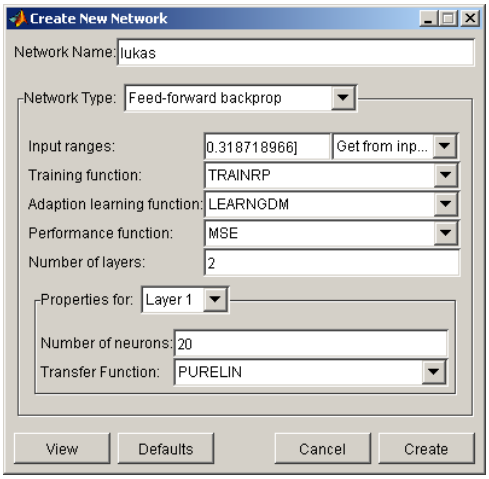
[1] https://uk.mathworks.com

4.2 Zmienne wejściowe pobieramy za pomocą przycisku Import (gdy importujemy u ustawiamy Import as Inputs, a gdy y ustawiamy Targets) – rys. 2.



Rys. 2. Importowanie par trenujących (macierzy **u** oraz **y**)

4.3 NNT umożliwia tworzenie modeli systemów w postaci sztucznych sieci neuronowych np. sztucznych sieci neuronowych wielowarstwowych jednokierunkowych (perceptronowych), itp. przy znajomości sygnałów wejściowych (u) i wyjściowych (y) systemu, przy czym najpierw projektuje się architekturę SSN (ustala się warstwę neuronów wejściowych – odpowiadającą wejściom, warstwy neuronów wyjściowych – odpowiadającą wyjściom oraz 1-2 warstwy ukryte z neuronami ukrytymi – rys. 3.



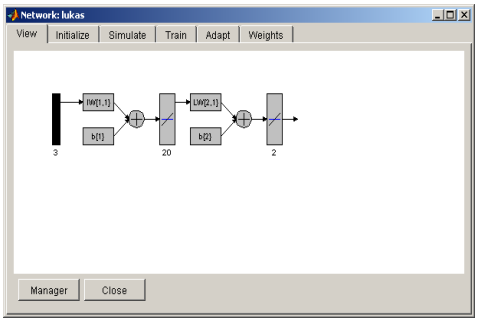
Rys. 3. Projektowanie SSN. Podstawowe oznaczenia:

Get from Input – tutaj wprowadzamy nazwę naszej zmiennej wejściowej – u, Training function – dobieramy funkcję uczenia,

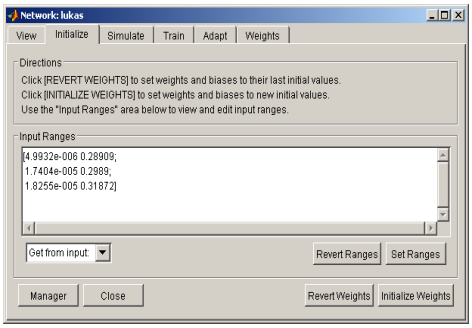
Number of Layers – ustalamy liczbę warstw SSN (oprócz warstwy wyjściowej) i liczbę neuronów w każdej warstwie),

Transfer function – funkcja aktywacji neuronu (purelin – liniowa),

Create – przycisk do utworzenia architektury SSN dla zaprojektowanych parametrów, View – przycisk do obejrzenia SSN (rys. 4)



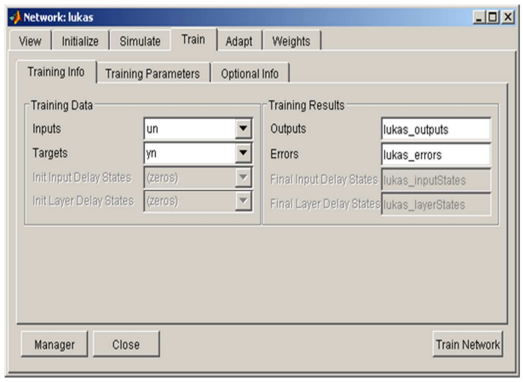
Rys. 4. Utworzona architektura SSN (o liczbie neuronów w warstwie wejściowej – 3, warstwie ukrytej – 20 (w celu zwiększenia pojemności SSN), warstwowe wyjściowej - 2). Oznaczenia: IW{1,1} – macierz wag pomiędzy warstwą wejściową neuronów a warstwą ukrytą, LW{2,1} – macierz wag pomiędzy warstwą ukrytą a warstwą wyjściową neuronów, b{1}, b{2} – biasy, Initialize – przycisk do inicjalizacji macierzy wag (przed uczeniem nadanie wartości macierzom) – rys. 5, Train – uczenie SSN – rys. 6.



Rys. 5. Inicjalizacja SSN. Oznaczenia:

Get from Input – wprowadzamy nasza zmienną wejściową – macierz **u**

Initialize Weitghts – naciskając przycisk nadajemy elementom wartości początkowe poprzedzające proces uczenia.

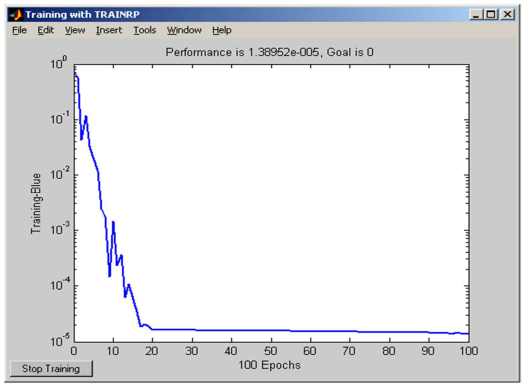


Rys. 6. Uczenie SSN modelu systemu. Oznaczenia:

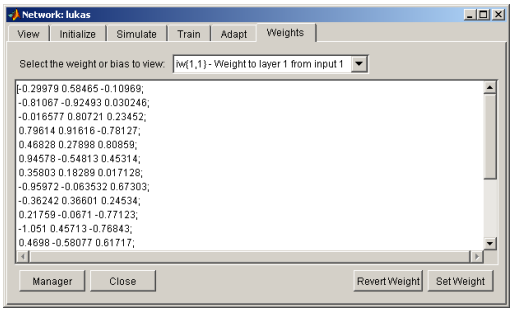
Inputs – wstawiamy naszą macierz wejściową do uczenia sieci, Targets – wstawiamy nasza macierz wyjściową do uczenia sieci,

Taining Parameters – najważniejsze aby ustawić liczbę epok (liczbę par trenujących),Train Networks – uczy SSN modelu systemu,

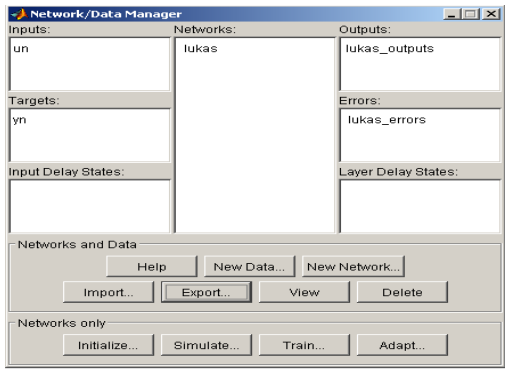
Trainig Info – informacja o przebiegu uczenia SSN – rys. 7



Rys. 7. Przebieg krzywej uczenia. W tym przypadku ustawiono 100 epok – po nauczeniu SSN istnieje możliwość obejrzenia macierzy wag i wektorów biasów – rys. 8 oraz wyeksportowania do Workspace modelu neuralnego systemu – rys. 9.

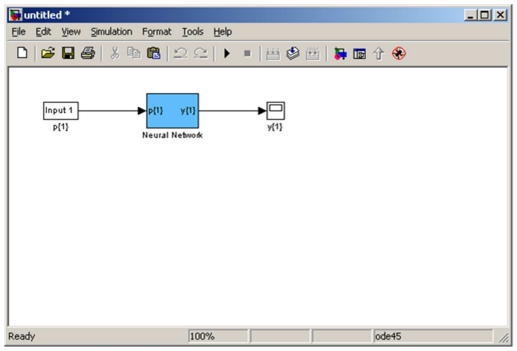


Rys. 8. Wiedza SSN dotycząca modelu systemu zawarta jest w macierzach wag oraz w wektorach biasów.



Rys. 9. Za pomocą przycisku Eksport mogę przenieść model do Workspace MATLAB-a (model ma nazwę w tym przypadku: lukas), a gdy jest model w Workspace to poleceniem:

>gensim(lukas) mogę wygenerować blok ze SSN dla potrzeb wykorzystania jej w Simulinku – rys. 10.



Rys. 10. Model neuronalny systemu w postaci bloczka ze SSN dla potrzeb wykorzystania jej w Simulink-u i dalej budować modele w Simulinku dla celów symulacyjnych i dla celów badania wrażliwości.

5. Neural Network Toolbox zawiera m.in.:

5.1 **Różne architektury sztucznych sieci neuronowych**

- Sieć PERCEPTRON o skokowej funkcji aktywacji,
- Siec typu ADALINE i MADELINE,
- Sieć jednokierunkowa z metodą wstecznej propagacji błędów,
- Sieć typu WTA (Kohenena),
- Sieć rezonansowa ART,
- Sieć ze sprzężeniem zwrotnym Hopfielda.

5.2 **Różne rodzaje funkcje aktywacji**

- hardlim – funkcja skoku jednostkowego przyjmująca wartości 0 lub 1,
- hardlims – funkcja symetryczna przyjmująca wartości -1 lub 1,
- logsig – funkcja sigmoidalna unipolarna,
- tansig – funkcja sigmoidalna bipolarna,
- purelin – funkcja liniowa,
- satlin – funkcja liniowa w zakresie (-1,1),
- compet – funkcja współzawodnictwa neuronowe w warstwie typu WTA.

5.3 **Różne metody uczenia**

- learnbp – algorytm wstecznej propagacji błędu w odniesieniu do jednokierunkowej sieci wielowarstwowej
- learnbpm – algorytm wstecznej propagacji błędu w odniesieniu do jednokierunkowej sieci wielowarstwowej z wykorzystaniem momentum,
- learnh – algorytm Hebba,
- learnhd – algorytm Hebba z zapominaniem,
- learnis – algorytm uczenia INSTAR,
- learnos – algorytm uczenia OUTSTAR,
- learnk – algorytm SOM Kohenena,
- learnp – algorytm uczenia PERCEPTRON-u,
- learnwh – algorytm uczenia Widrowa-Hofa.

Cel wykładu:

Moduł II (28.01.2017 r.) obejmuje bardzo zwięzłe wprowadzenie do modelowania neuralnego systemów technicznych i techniczno-ekonomicznych, w tym krótki przegląd przykładów środowisk programistycznych wykorzystywanych do modelowania neuralnego ze szczególnym zwróceniem uwagi na środowisko MATLAB opracowane i prowadzone przez firmę MathWorks, w tym na Neural Network Toolbox oraz Simulink.

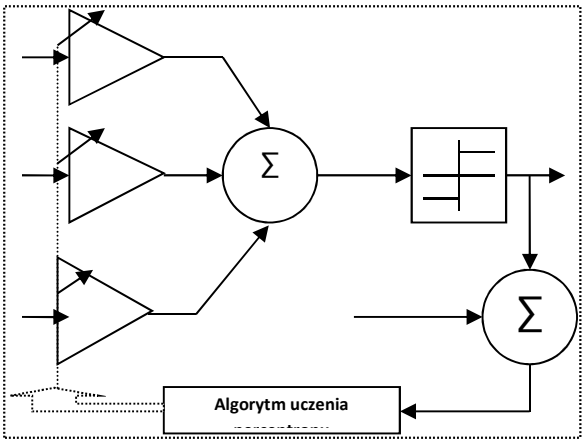
Treści dydaktyczne

Celem wykładu jest wprowadzenie do środowiska MATLAB ze szczególnym zwróceniem uwagi na praktyczne umiejętności projektowania systemów sztucznej inteligencji w środowisku MATLAB z wykorzystaniem umiejętności projektowania i uczenia sztucznych sieci neuronowych modeli systemów, procesów i obiektów przy pomocy Neural Network Toolbox-a i częściowo Simulinka (dalej: SIM). W szczególności zostanie zwrócona uwaga na modele neuronalne typu: SSN PERCEPTRON, SSN HOPFIELDA oraz SSN SOM Kohenena. Do zbudowania SSN jednokierunkowej wielowarstwowej punktem wyjścia był model McCullocha-Pitsa, która nazywa się Perceptronem prostym (rys. 1). Jako funkcję aktywacji przyjęto funkcję bipolarną f(net). Sygnał net na wyjściu części liniowej Perceptronu jest dany wzorem:

$$net = \sum_{i=1}^{N} w_i u_i - v = \sum_{i=0}^{N} w_i u_i \,, \tag{4}$$

gdzie: $w_0 = v$ oraz $u_0 = -1$.

Zadaniem Perceptronu jest klasyfikacja wektora $u=[u_1, \ldots, u_N]^T$ do jednej z dwóch klas oznaczonych literami L_1 oraz L_2. Perceptron klasyfikuje wektor u do klasy L_1, jeżeli sygnał wyjściowy przyjmuje wartość 1 oraz do klasy L_2, jeżeli sygnał wyjściowy przyjmuje wartość -1. Zatem Perceptron dzieli N-wymiarową przestrzeń wektorów wejściowych u na dwie półprzestrzenie rozdzielone N-1 wymiarowa hiperpłaszczyzną o równaniu:



$$\sum_{i=1}^{N} w_i u_i - v = \sum_{i=0}^{N} w_i u_i = 0 \qquad (5)$$

Hiperpłaszczyzna (5) nosi nazwę granicy decyzyjnej (ang. Decision boundary). Jeśli N=2 to granica decyzyjną jest linia prosta o równaniu:

$$w_1u_1+w_2u_2 -v =0 \qquad (6)$$

Rys. 1. Algorytm uczenia SSN.

Pokazany zostanie sposób tworzenia schematów blokowych w Simulinku i prowadzenia badań symulacyjnych z wykorzystaniem SSN. W drugiej części wykładu zostanie pokazany przykład **procesu budowy i uczenia sztucznej sieci neuronowej,** który następnie na ćwiczeniach laboratoryjnych stanowił będzie dla doktorantów przykład do samodzielnego przeprowadzenia badań neuronalnych.

Literatura podstawowa:

[1] MATLAB, Simulink, (+ poszczególne toolboxy). User's Guide. The MathWorks.1988-2016
[2] Osowski S.: Sieci neuronowe do przetwarzania informacji. WNT. Warszawa 2002
[3] Tadeusiewicz R.: Elementarne wprowadzenie do techniki sieci neuronowych z przekładowymi programami .AOW PLJ. Warszawa 1999
[4] Żurada J., i inni: Sztuczne sieci neuronowe. WNT. Warszawa 1996

Literatura uzupełniająca:

[5] Brzózka J., Dorobczyński L., Programowanie w MATLAB, MIKOM, Warszawa 1998
[6] Cichocki A., Osowski S., Siwek K.: MATLAB w zastosowaniu do obliczeń obwodowych i przetwarzania sygnałów. OW PW. Warszawa 2006
[7] Kamińska A., Pińczyk B.: Ćwiczenia z MATLAB. Przykłady i zadania. MIKOM. Warszawa 2002
[8] Mrozek B., Mrozek Z., MATLAB i Simulink. Poradnik użytkownika. Wyd. III, Helion 2010
[9] Mrozek B., Mrozek Z., MATLAB uniwersalne środowisko do obliczeń naukowo-technicznych, Wydawnictwo PLJ, Warszawa 2011
[10] Mulawka J., Systemy ekspertowe. WNT. Warszawa 1996
[11] Tadeusiewicz R.: Sieci neuronowe. AOW RM. Warszawa 1999
[12] Zalewski A., Cegieła R., MATLAB - obliczenia numeryczne ich zastosowanie, NAKOM, Poznań 1996