Instytut Sterowania i Systemów Informatycznych
Uniwersytet Zielonogórski

Zaawansowane systemy decyzyjne

Laboratorium

prowadzący: Andrzej Czajkowski[1]

Hybrydowe systemy decyzyjne

1 Cel ćwiczenia.

Celem ćwiczenia jest zapoznanie się z możliwościami tworzenia hybrydowych systemów ekspertowych z wykorzystaniem Corvid Exsys oraz aplikacji zewnętrznych stworzonych w środowisku Matlab.

2 Instrukcje do wykonania.

2.1 Kompilacja funkcji w środowisku Matlab.

- Uruchomić środowisko Matlab.
- Wybrać lub utworzyć folder przechowujący pliki projektu (Current Folder // workspace).
- Utworzyć funkcję nnmodel której zadaniem będzie kalkulacja ciepła generowanego przez silnik poprzez wykorzystanie sieci neuronowej w celu estymacji nieliniowej funkcji opisującej silnik.
  - Otworzyć plik do edycji poprzez :
    ```
    >>edit nnmodel
    ```
  - Funkcję należy zdefiniować jako:
    `function cieplo=nnmodel(predkosc_zad,chlodzenie)`
  - Sieć neuronowa opisująca teoretyczny silnik będzie posiadać 2 wejścia (zadana prędkość [zakres 0-5000], chłodzenie[wartość dyskretna 0 1]) oraz jedno wyjście (ciepło [zakres 0-1000]). Na warstwie ukrytej będzie znajdować się 10 nueronów. Zarówno w warstwie ukrytej jak warstwie wyjściowej funkcja aktywacji będzie w postaci tangensa hiperbolicznego.
  - W związku z powyższym należy utworzyć wagi o wymiarach 10x2 i 1x10 i wypełnić je wartościami otrzymanymi w skutek uczenia sieci neuronowej (w celu poznania szczegułów zapoznać się z toolboxem Neural Network Toolbox uruchamianym za pomocą polecenia nntool). Dodatkowo podczas uczenia tworzone są również wagi typu bias. W przypadku rozważanego systemu została utworzona jedna taka macierz wag dla warstwy ukrytej dlatego też jej wymiar wynosi 10x1. Wagi należy zdefiniować jako:

```
W1= [ 4.921578419879503 -0.666871744110809
0.231437255126928 1.418927094934441
2.987145789082478 5.242658109840926
-3.126175022857886 -2.992343163917616
0.643990646119950 -4.382466103064878
1.700398010675582 -14.093296478083557
-3.102227320053625 -3.225232766499700
2.948026641082510 -1.323182177773012
-2.402032234744608 -1.818731345374680
-4.238778178039517 4.171391645761571];

W2=[ 1.323366979890422 -0.186922916782851 0.020174434955355 -0.066151788290780 -0.245675817266941
0.699450301484318 ... 0.101237305396983 0.670410193897250 -0.308488882810207 -0.217493727712114];

b1=[ -3.612632222558453
3.445868534328676
-2.478274010082033
1.634318680048372
```

[1]Andrzej Czajkowski, Institute of Control and Computation Engineering, University of Zielona Góra, ul. Podgórna 50, 65-246 Zielona Góra, Poland. Email: a.czajkowski@issi.uz.zgora.pl

```
-0.496634013278703
0.483648367022102
-1.241989394537247
2.517137676464716
-3.369225341897877
-4.569929485699849];
```

- Ze względu na wykorzystanie funkcji aktywacji w postaci tangensa hiperbolicznego wymagana jest normalizacja wejść i wyjść z sieci (normalizacja ma za zadanie przeskalowanie wartości do zakresu -1:1).
- Normalizacja wejść:
  `input=[predkosc_zad/2500-1;chlodzenie];`
- Wyliczenie wartości wyjścia z sieci dla zadanych wejść:
  `cieplo=tansig(W2*(tansig(W1*input+b1)));`
- Normalizacja wyjść:
  `cieplo = round(500*(cieplo+1));`

- Funkcję należy zapisać i zamknąć po czym można ją przetestować w środowisku Matlab dla przykładowych wartości:
  - Test z wyłączonym chłodzeniem:
    `>> y=[];for i=1:5000, y=[y;nnmodel(i,0)]; end; plot(y);`
  - Test z włączonym chłodzeniem:
    `>> y=[];for i=1:5000, y=[y;nnmodel(i,1)]; end; plot(y);`
- Następnie należy stworzyć pakiet javowy w celu wykorzystania go w applecie uruchamianym w środowisku Corvid Exsys. Wybrać z menu startowego środowiska Matlab następujące podmenu: start->MATLAB->MATLAB Builder JA->Deployment Tool (deploytool) lub poprzez wpisanie w lini komend deploytool
- W oknie dialogowym należy wpisać nazwę projektu (np. zsd) oraz wybrać rodzaj aplikacji jako Java Package.
- Następnie należy w otwartej zakładce dodać nową klasę i nazwać ją NNmodelClass.
- Następnie do klasy należy dołączyć plik z funkcją poprze przycisk "add files" i wybór pliku nnmodel.m.
- Ostatnim krokiem jest stworzenie pliku jar poprzez naciśnięcie przycisku Build. W obecnym folderze powinna zostać utworzona scieżka ZSD/distrib a w niej plik ZSD.jar
- Środowisko Matlab może zostać zamknięte a należy uruchomić środowisko developerskie javy np Eclipse.

!!! W przypadku wyświetlenia błędu kompilacji z informacją o braku javac.exe należy zadbać o dodanie folderu z jdk do zmiennej systemowej PATH. !!!

2.2 Generowanie appletu javy w środowisku Eclipse.

- Uruchomić środowisko Eclipse.
- Utworzyć nowy Java Project o nazwie ZSD_applet.
- W utworzonym projekcie utworzyć pakiet zsd_package.
- W utworzonym pakiecie utworzyć klasę ZsdAppletClass dziedziczącą po klasie Applet.
  `public class ZsdAppletClass extends Applet`
- Należy dodać wygenerowany plik ZSD.jar z funkcją w Matlabie do bibliotek w projekcie (menu Project->properties i właściwość java build path, następnie Add External JARs... i wyszukać ZSD.jar).
- Analogicznie dodać do projektu bibliotekę javabulder.jar, znajdującą się w folderze z instalacją Matlaba (np. MATLAB/2012a/toolbox/javabuilder/jar/).
- Zaimplementować w klasie ZsdAppletClass wymianę informacji z appletem Exsys'a oraz wyliczenie wyjścia z sieci neuronowej za pomocą funkcji NNmodel().
  - W celu stworzenia poprawnej komunikacji pomiędzy oboma appletami należy utworzyć dwa pola w klasie:
    ```
    boolean running = false;
    String origName = "";
    ```

– Następnie należy utworzyć metodę inicjalizującą applet:

```
public void init() {
  if (running == false) {
     origName = this.getName();
     running = true;
  } else {
     processDataRequest(this.getName());
     this.setName(origName);
  }
}
```

Zadaniem tej metody jest przy pierwszym uruchomiue (start systemu ekspertowego) zapisanie we wcześniej utworzonej zmiennej nadanej w pliku html nazwy dla appletu, za pomocą metody `getName()` oraz zapamiętanie informacji o uruchomieniu appletu (zmienna logiczna `running`). Przy kolejnych wywołaniach metody `init()` (właściwe wywołanie w celu wyliczenia wartości w skutek przeprowadzanego wnioskowania) wykonywana jest część instrukcji warunkowej `else` i wywołanie metody przetwarzającej zapytanie, która utworzona będzie w kolejnym kroku. Parametry przesłane z systemu Exsys są pobierane za pomocą metody `getName()` i przesłane do metody przetwarzającej jako łańcuch znaków (**wymagane jest późniejsze parsowanie**). Na koniec w tym bloku następuje ponowne ustawienie nazwy appletu na odstawie nazwy zapamiętanej przy pierwszym uruchomieniu, w celu identyfikacji przez system Exsys (metoda `setName()`).

– Metoda przetwarzająca zapytanie dla systemu Exsys:

```
void processDataRequest(String data) {
   try {
     NNmodelClass obj=new NNmodelClass();
     Object []result=obj.nnmodel(1,Double.parseDouble(data),0.0);
     returnDataToCorvid(""+result[0]);
   } catch (NumberFormatException e) {
     e.printStackTrace();
   } catch (MWException e) {
     e.printStackTrace();
   }
}
```

Zadaniem tej metody jest przetworzenie danych z wykorzystaniem funkcji zaimportowanej ze środowiska MATLAB. Dane z systemu Exsys wprowadzane są w postaci łańcucha znaków w argumencie metody `data`, Następnie tworzony jest obiekt, który pozwala na wywołanie zaimportowanej funkcji (`nnmodel()`). Funkcja ta zgodnie z tym co było utworzone w środowisku MATLAB przyjmuje dwa argumenty a zwraca jedną wartość (wejściowe: prędkość, chłodzenie; wyjściowe: ciepło). Dlatego też funkcja wyeksportowana ma 3 argumenty. Pierwszy argument to liczba zwracanych wartości (w tym przypadku wartość zwracana jest jedna dlatego pierwszym argumentem jest 1). Dwa kolejne argumenty to argumenty bezpośrednio wysylane do funkcji matlabowej czyli prędkość i chłodzenie. Wartość prędkości uzyskiwana jest poprzez zmienna `data` przesłana ze środowiska Exsys. Wartość ta jest otrzymana jako łańcuch znaków (String) dlatego też musi zostać sparsowana do zmiennej typu double (Należy zawsze parsować do zmiennej zmiennoprzecinkowej typu double nawet w przypadku zmiennych typu całkowitego ze względu na brak ich obsługi ze strony MATLABa). Drugi argument domyślnie ustawiamy na 0 (0.0 - w formacie zmiennoprzecinkowym). Wynik zwracany jest do tablicy typu Object i znajduje się pod indexem zerowym, dlatego też wynik taki po skonwertowaniu do łańcucha znaków (w javie dodanie pustego łańcucha znaków "" domyślnie konwertuje zmienne do typu String) wysyłamy do środowiska Exsys.

– Metoda zwracająca obliczone wartości do systemu Exsys na podstawie przetworzonych wcześniej danych:

```
public void returnDataToCorvid(String resultData) {
   Applet corvidRuntime;
   corvidRuntime = getAppletContext().getApplet("CorvidRuntime");
   corvidRuntime.setName(resultData);
   corvidRuntime.start();
}
```

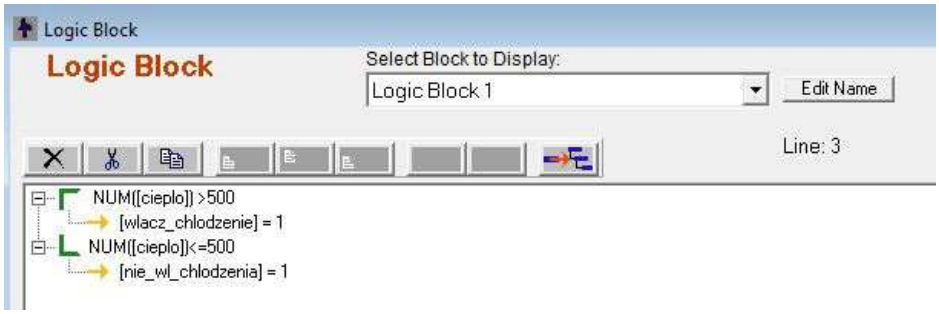
Zadaniem tej metody jest wykorzystanie metody `setName()` do przesłania uzyskanych w skutek działania appletu wyników do appletu Exsysa (systemy ekspertowe w systemie Exsys są kompilowane do appletów i uruchamiane we wbudowanej lub zewnętrznej przeglądarce). W metodzie tej lokalizowany jest po nazwie applet Exsys'a i wysyłane są do niego dane przesłane z metody przetwarzającej za pomocą zmiennej resultData będącej argumentem tej metody.

- Ostatnim krokiem w Eclipsie jest wyeksportowanie utworzonej klasy jako archiwum JAR. Aby to wykonać należy uruchomić eksportera plików poprzez menu File->Export. Następnie wybrać żądany typ czyli JAR

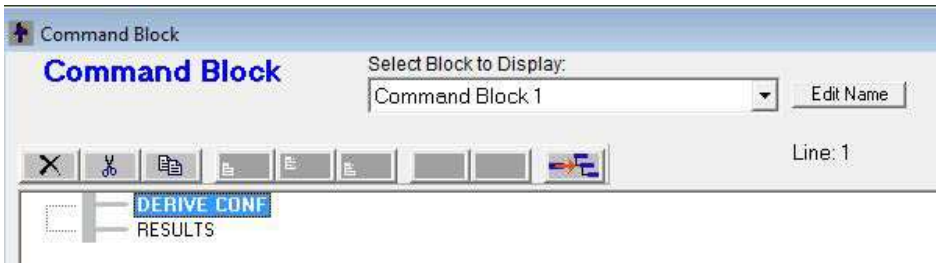
file z folderu Java. W kolejnym kroku należy wyszukać i zaznaczyć klasy, które mają zostać wyeksportowane. Wprowadzić nazwę archiwum (np. ZSD_applet) i kliknąć finish (na potrzeby tej instrukcji nie trzeba zaznaczać nic więcej). Ze względu na implementację interfejsu serializacji przez klasę Applet możliwe jest wyświetlenie o ostrzeżeniach w projekcie (Warning) w trakcie eksportowania. W tym przypadku można je zignorować.

2.3 Utworzenie hybrydowego systemu decyzyjnego w środowisku Exsys.

- Utworzyć nowy projekt w systemie Corvid Exsys.
- Utworzyć zmienne:
  - 2 zmienne typu Confidence (wlacz_chlodzenie i nie_wlaczaj_chlodzenia).
  - Zmienna typu numerycznego predkosc.
  - Zmienna typu numerycznego ciepło - W celu wyliczenia wartości ciepła dla wprowadzonej wartości prędkości za pomocą zewnętrznego appletu należy zmodyfikować pole z zakładki Options dla tej zmiennej. Należy zaznaczyć pole `External Source For Value` (jeśli pole jest nieaktywne na dole okna należy zaznaczyć opcję Show Advanced Options). W polu tekstowym dla tej należy wpisać:
    `APPLET NeuralNetwork [[predkosc]]`
    Pierwszy parametr mówi o wykorzystaniu appletu jako zewnętrznego źródła o nazwie z drugiego parametru. W tym przypadku nadaną nazwą appletu będzie NeuralNetwork. Ostatnimi parametrami są argumenty które mają zostać wysłane do appletu.
- Następnie należy utworzyć logikę w systemie ekspertowym.
  - Logika do utworzenia ma zasugerować włączenie chlodzenia silnika w przypadku przewidywanej wartości ciepla powyżej 500 i zalecić nie włączanie jeśli ciepło jest poniżej 500 stopni poprzez ustawienie odpowiednich zmiennych ufności.



- Kolejnym krokiem jest utworzenie bloku sterowania:
  - Należy wyznaczyć w nim wszystkie zmienne ufności(DERIV CONF) oraz wyświetlić wyniki (RESULTS).



- Ostatnim krokiem jest załadowanie i nazwanie zewnętrznego appletu z potrzebnymi funkcjami. W tym celu należy uruchomić menu File->Properties. I wybrać zakładkę Other HTML. Należy tam wpisać polecenie, które zostanie dodane do strony HTML z systemem.
```
<APPLET CODEBASE = "./" CODE = "zsd_package.ZsdAppletClass" NAME = "NeuralNetwork"
ARCHIVE = "ZSD_applet.jar, javabuilder.jar, ZSD.jar" WIDTH = 0 HEIGHT = 0 HSPACE = 0
VSPACE = 0>
</APPLET>
```
Dzięki wprowadzeniu zer dla wymiarów appletu będzie on niewidoczny dla użytkownika końcowego. Należy zwrócić uwagę na poprawność podanych nazw. Literówki lub zmiany nazw mogą prowadzić do błędów w działaniu.
- W celu uproszczenia należy wszystkie trzy archiwa jar (ZSD_applet.jar, javabuilder.jar, ZSD.jar) umieścić w tym samym folderze co wygenerowany przez Corvida plik html z systemem ekspertowym.

- Po wykonaniu powyższych czynności można uruchomić stworzony hybrydowy system ekspertowy. W przypadku wystąpienia błędów w działaniu należy przejść do kolejnego podpunktu.

2.4 Finalizacja systemu poprzez odpowiednią konfigurację maszyny wirtualnej Javy i systemu Windows

Poniższe kroki są opcjonalne i zależą od konfiguracji posiadanej konfiguracji systemu Windows oraz środowiska uruchomieniowego Javy (JRE). Możliwe problemy przy uruchamianiu stworzonego systemu:

- Komunikat o braku pliku `mclmcrrt7_17.dll` (`java.lang.UnsatisfiedLinkError: Failed to find the library mclmcrrt7_17.dll, required by MATLAB Builder JA, on java.library.path.`).
  System nie widzi bibliotek pochodzących z MATLAB Compiler Runtime (MCR). Należy dodać do systemowej zmiennej %PATH% ścieżkę do bibliotek. Biblioteki znajdują się w folderze z instalacją MATLABA (np. /MATLAB/R2012a/runtime/win32). W przypadku braku instalacji pakietu MATLAB na komputerze, na którym ma być uruchamiany system ekspertowy, można bez opłat ściągnąć MCR ze strony producenta `http://www.mathworks.com/products/compiler/mcr/index.html` i wskazać je po zainstalowaniu (domyślnie ścieżka: C:/Program Files/MATLAB/MATLAB Compiler Runtime/v717/runtime). **UWAGA !!! ze względu, że system Exsys jest programem 32 bitowym korzysta on z 32 bitowego środowiska uruchomieniowego javy (JRE), dlatego też należy udostępnić 32 bitowe biblioteki MCR. W przypadku korzystania z bibliotek pochodzących z instalacji środowiska MATLAB dla systemów 64 bitowych domyślnie instalacja ta jest 64 bitowa a co za tym idzie takie też są wbudowane biblioteki MCR i nie będą one mogły zostać wykorzystane w systemie Exsys. W takim przypadku potrzebna będzie instalacja 32 bitowego pakietu MATLAB lub 32 bitowego pakietu MCR ze strony podanej powyżej.**

- W przypadku wyjątków sygnalizowanych przez konsolę Javy dotyczących braku uprawnień do zasobów (np. `jproxy::exception accesscontrolexception: access denied` lub `java.security.AccessControlException: access denied ("java.util.PropertyPermission" "sun.arch.data.model" "read")`.
  Ze względu na zabezpieczenia javy przed niebezpiecznymi appletami pochodzącymi z niezweryfikowanych źródeł domyślnie nie są dla nich udostępnione lokalne zasoby. W celu ustawienia uprawnień dla stworzonych wsystemie appletów należy zmodyfikować plik java.policy z folderu z instalacją JRE domyślnie C:/Program Files (x86)/Java/jre7/lib/security poprzez dodanie do niego:
  ```
  grant codeBase "file:C:/-" {
  permission java.security.AllPermission;
  };
  ```
  Daje to wszelkie uprawnienia wszystkim appletom pochodzącym z dysku lokalnego C:. Jest to dość niebezpieczne rozwiązanie dlatego prawidłowo należało by wskazać dokładną ścieżkę do wykonywanego appletu lub folderu z appletem np. C:/CorvidPRojekt/*.
  Najbezpieczniejszym rozwiązaniem jednak jest podpisanie certyfikatem archiwum JAR. Aczkolwiek w przypadku nieznanego certyfikatu JRE będzie wymagać zgody użytkownika na działanie tam zawartego appletu. W przypadku posiadania certyfikatu od zaufanego dostawcy np. VeeriSign czy Thawte, proces uruchamiania systemu jest automatyczny.
  W celu podpisania archiwum JAR należy mieć zainstalowane JDK (Java Development Kit) oraz mieć dodaną ścieżkę do niego do zmiennej systemowej PATH. Aby utworzyć certyfikat i podpisać nim archiwum JAR należy w linii komend wykonać następujące instrukcje:
  ```
  keytool -genkey -validity 3650 -keystore pKeyStore -alias keyName
  keytool -selfcert -keystore pKeyStore -alias keyName-validity 3650
  jarsigner -keystore pKeyStore applet.jar keyName
  ```
  W celu podpisania kolejnych archiwum JAR wystarczy wykonać polecenie: `jarsigner -keystore pKeyStore applet_next.jar keyName`

3 Zadanie do samodzielnego wykonania

Zmodyfikować utworzony system tak aby na podsumowaniu działania pokazane zostało ile produkowanego ciepła zostanie zniwelowane po uruchomieniu chłodzenia.