Systemy neuronowo-rozmyte – wprowadzenie.

1. Istota systemów rozmytych (przypomnienie)

Systemy rozmyte są systemami, których działanie opiera się o logikę wykorzystującą pojęcia opisowe, takie jak „mały" „średni", „duży", „niski" „wysoki" itp. Pozwala to na zastosowanie w tych systemach wiedzy opisowej i sposobu wnioskowania zbliżonego do stosowanego przez ludzi.

System rozmyty składa się z trzech podstawowych bloków:

- blok rozmywania. Określa on do jakiej klasy i w jakim stopniu dana wartość wejściowa należy. Na przykład wzrost może być określony jako niski, średni i wysoki, zaś blok rozmywania może ustalić że wzrost „170 cm" jest w 0.3 niski a w 0.7 wysoki
- blok wnioskowania. Blok ten zawiera zebraną wiedzę ekspertów w formie opisowej. Przykład: jeśli na podwórku jest ***zimno*** ***_i_*** w domu jest **zimno**, grzej ***mocno.***
- blok wyostrzania (defuzyfikacji). Jego zadaniem jest zamiana wyników otrzymanych z bloku wnioskowania na wartość funkcji wyjściowej. Przykład: jeśli blok wnioskowania wnioskuje żeby grzać ***mocno*** w 70% i grzać ***średnio*** w 30% to blok defuzyfikacji ustawi moc ogrzewania na (np.) 700 W.

Wyróżniamy dwa podstawowe typy systemów rozmytych:

- *Mamdani* – działa jak opisany powyżej
- *Takagi-Sugeno*, który zamiast opisanego wyżej sposobu wyostrzania stosuje w bloku wyostrzania funkcje zmiennych wejściowych.

2. Koncepcja systemów neuronowo-rozmytych.

Logika rozmyta operuje na zmiennych lingwistycznych, ale nie potrafi się uczyć. Sieci neuronowe potrafią się uczyć, ale nauczona sieć stanowi „czarną skrzynkę" i zawarta w niej wiedza rozproszona jest w wartościach wag.
Jeśli opisać każdy blok systemu rozmytego jako jedną lub kilka warstw sieci neuronowej, można do takiego systemu zastosować metody uczenia sieci neuronowej, zachowując strukturę systemu rozmytego. Po nauczeniu takiej sieci wartości wag neuronów poszczególnych warstw możemy zinterpretować jako parametry systemu rozmytego, uzyskując dostęp do zgromadzonej w sieci wiedzy.

W systemie neuronowo-rozmytym typu *Mamdani* warstwa pierwsza jest warstwą wejściową: neuron wejściowy o liniowej funkcji aktywacji i wadze równej 1 służy do rozprowadzenia sygnału wejściowego do kolejnej warstwy.

Warstwa druga wykonuje zadanie rozmywania a jej neurony reprezentują zbiory rozmyte. Każdy neuron sprawdza do jakiego stopnia dana wartość wejściowa należy do zbioru rozmytego przypisanego do tego neuronu.

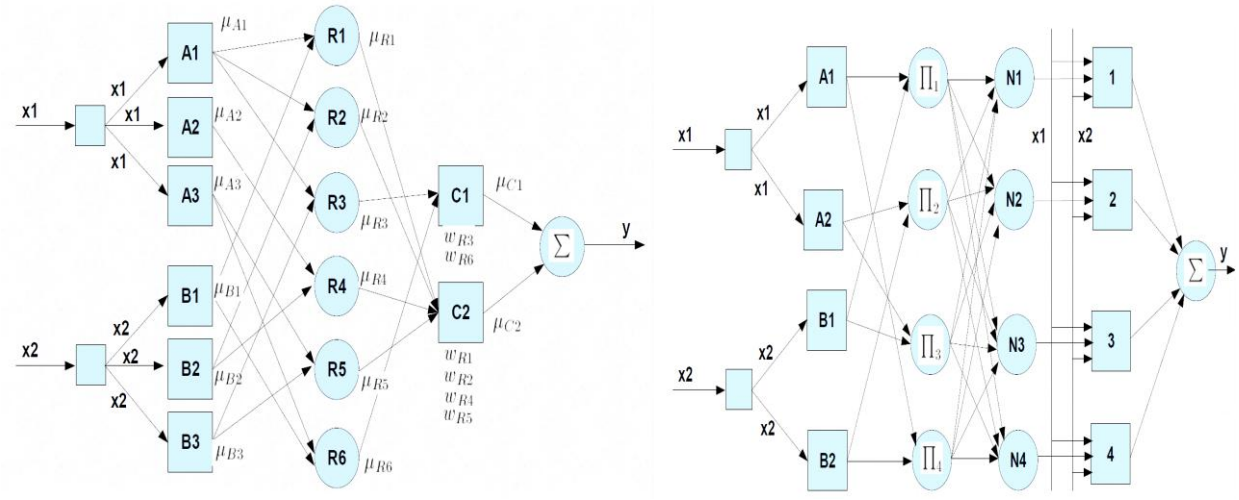
Warstwa trzecia jest warstwą reguł rozmytych. Każdy neuron otrzymuje dane od neuronów warstwy rozmywania i implementuje pojedynczą regułę rozmytą

Neurony warstwy czwartej reprezentują zbiory rozmyte opisujące zmienną wyjściową.

Warstwa piąta (na ogół 1 neuron) wykonuje operację defuzyfikacji.

W systemie typu *Takagi-Sugeno*, który wykorzystuje na wyjściu funkcje zmiennych wejściowych zamiast zbiorów rozmytych, niezbędna jest jeszcze jedna (występująca jako piąta) warstwa neuronów, realizująca wspomniane funkcje. Warstwa czwarta w tym modelu stanowi warstwę normalizacji, niezbędną do poprawnego działania tego modelu.

a) b)



Struktury systemów neuronowo-rozmytych. a) – Mamdani, b) – Takagi-Sugeno.
Źródło: „Systemy Neuronowo-Rozmyte (Neuro-Fuzzy Systems), dr inż. Michał Bereta, Politechnika Krakowska

3. Systemy neuronowo-rozmyte w środowisku Matlab.

Do budowy systemów neuronowo-rozmytych częściej wykorzystuje się model rozmyty *Takagi-Sugeno*. Możliwe jest w ten sposób zbudowanie systemu rozmytego matematycznie równoważnego sieciom LVQ lub RBF. Sieci taki określane są akronimem ANFIS (*adaptive neuro-fuzzy inference system*). Środowisko Matlab udostępnia następujące funkcje związane z tymi systemami:

- *genfis1* – tworzy FIS typu *Takagi-Sugeno* na podstawie tzw *grid partition*
- *genfis2* – tworzy FIS typu *Takagi-Sugeno* używając techniki *subtractive clustering*
- *genfis3* – tworzy FIS typu *Takagi-Sugeno* na podstawie grupowania danych w oparciu o algorytm rozmytych c-średnich (FCM).
- *anfis* – trenuje system neuronowo-rozmyty uzyskany za pomocą jednej z funkcji *genfis*

Składnia funkcji *genfis1* jest następująca:

fismat = genfis1(data,numMFs,inmftype,outmftype)

gdzie:

- fismat – wynikowa struktura FIS
- data – zbiór danych na podstawie którego struktura będzie tworzona. Zbiór powinien składać się z kolumn, które zawierają dane poszczególnych wejść i (ostatnia kolumna) **jednego** wyjścia.
- numMFs – liczba funkcji przynależności generowana dla każdego z osobna lub wszystkich wejść
- inmftype – typ funkcji przynależności
- outmftype – typ wyjściowej funkcji – *linear* lub *constant*

Składnia funkcji *genfis2* jest następująca:

fismat = genfis2(Xin,Xout,radii,xBounds,options,user_centers)

gdzie:

- fismat – wynikowa struktura FIS
- Xin – macierz danych wejściowych
- Xout – macierz danych wyjściowych

- radii – wektor określający zakres wpływu centrum klastra w każdym z wymiarów danych
- xBounds – opcjonalna macierz 2 na N, która określa sposób odwzorowania danych w Xin i Xout w jednostkowym hipersześcianie,
- options – opcjonalny wektor służący do określania parametrów algorytmu
- user_centers – opcjonalna macierz do określania niestandardowych centrów klastrów

Składnia funkcji *genfis3* jest następująca:

fismat = genfis3(Xin,Xout,type,cluster_n)

gdzie:

- fismat– wynikowa struktura FIS
- Xin – zbiór danych wejściowych
- Xout –zmienne wyjściowe
- type – typ struktury FIS, możliwe opcje to Mamdani/Sugeno
- cluster_n – liczba klastrów na które zbiór danych wejściowych zostanie podzielony

Składnia funkcji *anfis*:

[fis,error] = anfis(trnData,initFis,trnOpt,dispOpt,chkData,optMethod)

gdzie:

- fis – nauczony system FIS,
- error – błąd uzyskany podczas uczenia,
- trnData – zbiór danych uczących,
- initFis – zainicjowana struktura FIS (przed uczeniem),
- trnOpt – opcje procesu uczenia – wektor o 5 elementach zawierający w poszczególnych komórkach wartości opisujące parametry. W przypadku potrzeby użycia wartości domyślnych należy wpisać NaN:
    - trnOpt(1) – liczba epok uczenia,
    - trnOpt(2) - maksymalna dopuszczalna wartość błędu,
    - trnOpt(3) – początkowy rozmiar kroku uczenia,
    - trnOpt(4) – rozmiar zmniejszania kroku uczenia,
    - trnOpt(5) – rozmiar zwiększania kroku uczenia.
- dispOpt – opcje wyświetlania wyników cząstkowych na ekranie,
- chkData – zbiór danych testowych zabezpieczających przed przeuczeniem systemu,
- optMethod – sposób optymalizacji funkcji przynależności: 1 – optymalizacja hybrydowa, 0 – optymalizacja w oparciu o algorytm wstecznej propagacji błędu.

Jeśli któregoś z parametrów wywołania funkcji *anfis* nie chcemy przekazywać, należy w jego miejsce wstawić [ ]

Więcej szczegółów dotyczących składni i sposobu używania wyżej wymienionych funkcji można znaleźć w systemie pomocy programu Matlab.