Systemy neuronowo-rozmyte

Wstęp

Systemy neuronowo rozmyte pozwalają na skorzystanie z zysków, jakie niosą ze sobą systemy rozmyte oraz sieci neuronowe. Innymi słowy klasyczne systemy regałowe bazowały na różnego rodzaju algorytmach przeszukiwania, które cechuje b. duża złożoność obliczeniowa, podczas gdy systemy neuronowo rozmyte pozwalają na wykorzystanie algorytmów gradientowych oraz znanego z sieci neuronowych algorytmu wstecznej propagacji błędu podczas procesu optymalizacji zbioru reguł. Dzięki temu uzyskanie dobrze nauczonego systemu jest stosunkowo prostą i szybką procedurą. Do budowy systemów neuronowo-rozmytych zwykle wykorzystuje się model rozmyty Takagi-Sugeny, czyli taki model, w którym konkluzje reguł stanowią funkcję. Jak pokazała to w swojej pracy z 1996r. L Kuncheva[1] poprzez odpowiedni dobór operatorów rozmytych oraz kształtu funkcji zawartej w konkluzji możliwe jest zbudowanie systemu rozmytego matematycznie równoważnego sieciom LVQ oraz RBF.

Algorytm ANFIS jest jednym z przykładów algorytmu/implementacji systemów nauronowo-rozmytych. Wykorzystuje on zarówno proces optymalizacji z wykorzystaniem algorytmów gradientowych jak i algorytm propagacji wstecznej.

Opis narzędzi Matlaba

Dla danych IRIS (tylko zmienne 3 i 4 + opis etykiet) zainicjuj system rozmyty korzystając z polecenia *genfis1(),genfis3().*

Funkcja *genfis1*: - tworzy FIS typu Takagi Sugeno, na podstawie tzn “grid partition”

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

gdzie:

- fismat – wynikowa struktura FIS
- data – zbiór danych na podstawie którego struktura będzie tworzona
- numMFs – liczba funkcji przynależności generowana dla każdego z osobna lub wszystkich wejść
- inmftype – typ funkcji przynależności (najlepiej trójkątna/Gaussowska)
- outmftype – typ wyjściowej funkcji – uwaga – w FIS typu Takagi-Sugeno wyjsciem nie jest zbiór rozmyty tylko funkcja, dlatego też możliwe opcje to linear lub constant

Funkcja *genfis3*: - tworzy FIS typu Takagi Sugeno, na podstawie grupowania danych w oparciu o algorytmu rozmytych c-średnich (FCM). Domyślnie tworzony system zbudowany jest w oparciu o gaussowskie funkcje przynależności.

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

[1] L. Kuncheva „On the Equivalence Between Fuzzy and Statistical Classifiers” IJUFKBS, 1996

gdzie:

- fismat– wynikowa struktura FIS
- Xin – zbiór danych wejściowych (zmienne używane do predykcji)
- Xout –zmienne wyjściowe – dla nas wyjściem jest kolumna z etykietami
- type – typ struktury FIS, możliwe opcje to Mamdani/Sugeno
- cluster_n – liczba klastrów na które zbiór danych wejściowych zostanie podzielony

Stworzona przez obydwa inicjatory struktur FIS może następnie zostać poddana optymalizacji w oparciu a algorytm ANFIS, dzięki czemu położenie, oraz parametry funkcji przynależności oraz odpowiednie parametry funkcji wyjściowej zostaną automatycznie zoptymalizowane na podstawie danych.

Postać 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 – zainisjowana struktura FIS przez algorytm genfis1 I genfis3
- trnOpt – opcje uprocesu czenia – 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, im więcej tym dłuzej system będzie uczony, ale będzie miał mniejszy błąd. Typowa wartość to 100, domyślna wart. 10
  - trnOpt(2) - maksymalna dopuszczalna wartość błędu, domyślna wart. 0
  - trnOpt(3) – początkowy rozmiar kroku uczenia – domyslna wart. 0.01
  - trnOpt(4) – rozmiar zmniejszania kroku uczenia – domyslna wart. 0.9
  - trnOpt(5) – rozmiar zwiększania kroku uczenia – domyslna wart. 1.1
- dispOpt – opcje wyświetlania wyników cząstkowych na ekranie (na potrzeby lab. nieistotne)
- chkData – zbiór danych testowych umozliwiający ocenę przed przeuczeniem system (na potrzeby lab. nieistotne)
- optMethod – sposób optymalizacji funkcji przynależności, możliwe opcje to 0 lub 1, gdzie 1 – optymalizacja hybrydowa, 0 – optymalizacja w oparciu o alg. Wstecznej propagacji błędu. UWAGA jeśli któregoś z parametrów wywołania struktury FIS nie chcemy przekazywać, należy w jego miejsce wstawić [] np.

```
[fis,error] = anfis(trnData,initFis,[100 0 NaN NaN NaN],[],[],0);
```

Zadania

1. Wczytaj dane odpowiednio obliczenia powtórz dla danych Iris,Iono,WBC,pima

2. Podziel dane na trening / test
3. Zainicjuj strukturę FIS odpowiednio przez genfis1 oraz osobno genfis3. Inicjacji dokonaj na zbiorze treningowym
4. Uruchom system ANFIS jako narzędzie do optymalizacji wcześniej zainicjowanej w pkt. 3 struktury FIS
5. Uruchom zbudowany i zoptymalizowany system FIS za pośrednictwem funkcji evalfis na danych testowych
6. Policz dokładność jak na zajęciach z CI.

Zbadaj wpływ dla genfis1

- Wpływ kształtu funkcji przynależności na dokładność uzyskanych wyników
- wpływ liczby podziałów (numMFs) na dokładność uzyskanych wyników.
- wpływ typu funkcji wyjściowej na dokładność uzyskiwanych wyników

Zbadaj wpływ dla genfis3

- liczby klastrów na dokładność funkcji uzyskanych wyników

Zbadaj wpływ dla anfis

- liczby iteracji na dokładność uzyskanych wyników
- typu optymalizacji funkcji przynależności na dokładność klasyfikacji
- kozystając z funkcji plot(error) – wykreśl błąd procesu uczenia struktury FIS poprzez algorytm an fis.

Wyniki przedstaw w postaci wykresów, w celu uniknięcia problemów losowego podziału na część treningową i testową obliczenia powtórz kilkukrotnie, a na wykresach umieść średnie.

Uwagi

- Zbiór Jonosfera składa się z 33 zmiennych, co powoduje że dla złożoności dla algorytmu genfis1 (liczby generowanych reguł) rzędu n^m i m=33 mogą się pojawić problemy z procesem uczenia wynikające ze zbyt dużej liczby reguł! – wówczas zaznacz odpowiedź – *problem nierozwiązywalny*
- Wynikiem działania systemu FIS podczas predykcji jest liczba ciągła, w związku z tym do obliczenia dokładności klasyfikacji niezbędne jest zaokrąglenie uzyskanych wyników np.

```
(…)
fis = an fis(dataTrain,fis);
wyn = evalfis(dataTest);
wyn = round(wyn);
```

- Uważaj na zmienne wejściowe binarne, gdyż Matlabowa implementacja ANFIS'a dla tego typu zmiennych zachowuje się niestabilnie