Neuronowe i ewolucyjne metody identyfikacji i dostrajania modeli rozmytych - implementacja i porównanie.

Autor: Stanisław Swianiewicz
Opiekun naukowy: dr inż. Piotr Marusak

9. maja 2010 r.

Modele rozmyte Takagi-Sugeno-Kanga

- Obszar zmienności wejść modelu dzielony na zbiory rozmyte
- Funkcje przynależności
- Reguły wnioskowania
- Liniowe modele lokalne

Modele rozmyte Takagi-Sugeno-Kanga

Zbiór reguł — baza wiedzy

$$
\begin{aligned}
\mathrm{R}^1 &: \text{ JEŚLI } x_1 \text{ jest } X_{11} \text{ i } x_2 \text{ jest } X_{21} \text{ to } y^{11} = f_{11}(x) \\
\mathrm{R}^2 &: \text{ JEŚLI } x_1 \text{ jest } X_{11} \text{ i } x_2 \text{ jest } X_{22} \text{ to } y^{12} = f_{12}(x) \\
\mathrm{R}^3 &: \text{ JEŚLI } x_1 \text{ jest } X_{12} \text{ i } x_2 \text{ jest } X_{21} \text{ to } y^{21} = f_{21}(x) \\
\mathrm{R}^4 &: \text{ JEŚLI } x_1 \text{ jest } X_{12} \text{ i } x_2 \text{ jest } X_{22} \text{ to } y^{22} = f_{22}(x)
\end{aligned}
$$

Oblicznie konkluzji finalnej

- Suma ważona wyjść modeli lokalnych

$$y = \sum_{i=1}^{R} w^i y^i$$

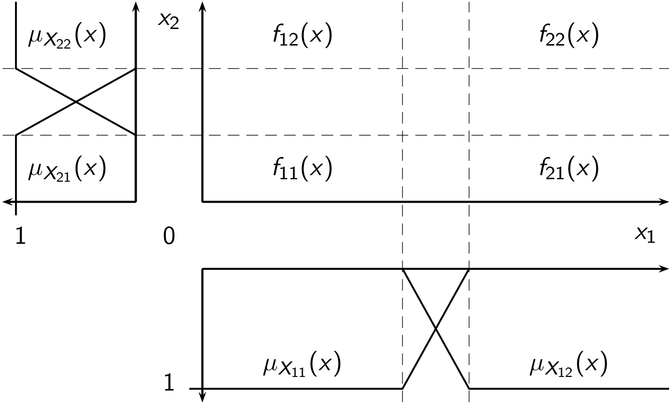
- Średnia wyjść modeli lokalnych

$$y = \frac{\sum_{i=1}^{R} w^i y^i}{\sum_{i=1}^{r} w^i}$$

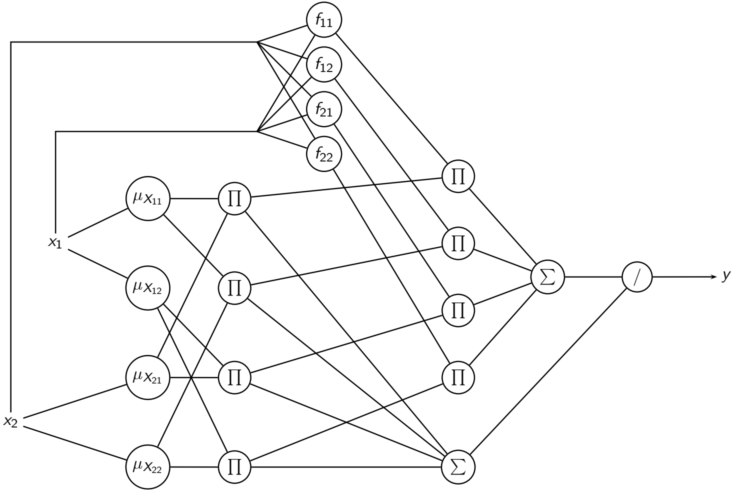
- $w^i = \prod_{j=1}^{N} \mu_{X_j^i}$ – poziomy aktywacji reguł

Modele rozmyte Takagi-Sugeno-Kanga

Funkcje przynależności, wnioskowanie rozmyte



Struktura rozmytej sieci neuronowej



Algorytm uczenia rozmytej sieci neuronowej

- Algorytm hybrydowy
- Dostrajanie parametrów liniowych następników - metoda najmniejszych kwadratów
- Dostrajanie parametrów poprzedników - uczenie SN
  - Optymalizacja gradientowa - algorytm propagacji wstecznej
  - Optymalizacja bezgradientowa

Metoda najmniejszych kwadratów

$$f_i(\mathbf{x}) = p_0^i + \sum_{j=1}^{M} p_m^i x_m$$

$$y = \sum_{i=1}^{R} \tilde{w}_i [p_0^i + \sum_{j=1}^{M} p_j^i x_j]$$

Dla ustalonych wartości wejścia wyjście zależy liniowo od parametrów p_j^i.

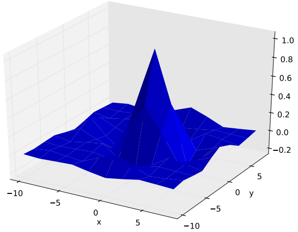
Metoda propagacji wstecznej

$$\frac{\partial E}{\partial c_{j,k,l}} = (y(\mathbf{x}) - d)\sum_{i=1}^{R}[y^i \frac{\partial \tilde{w}_i}{\partial c_{j,k,l}}]$$

Uśrednianie poziomów aktywacji prowadzi do zwiększenia ilości obliczeń związanych z obliczaniem pochodnej ilorazu.

$$\tilde{w}^i = \frac{\prod_{j=1}^{N} \mu_{X_j^i}}{\sum_{k=1}^{R}[\prod_{j=1}^{N} \mu_{X_j^k}]}$$

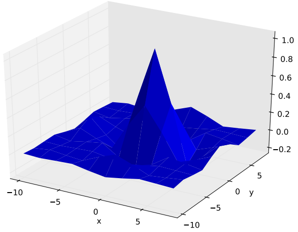
Przykład I



dane testowe

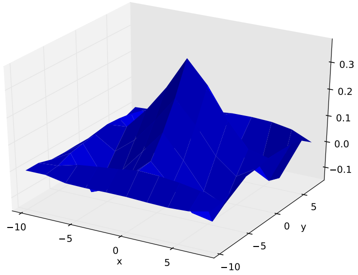
$$y = \frac{\sin(x_1)\sin(x_2)}{x_1 x_2}$$

Przykład I



dane testowe

$$y = \frac{\sin(x_1)\sin(x_2)}{x_1 x_2}$$



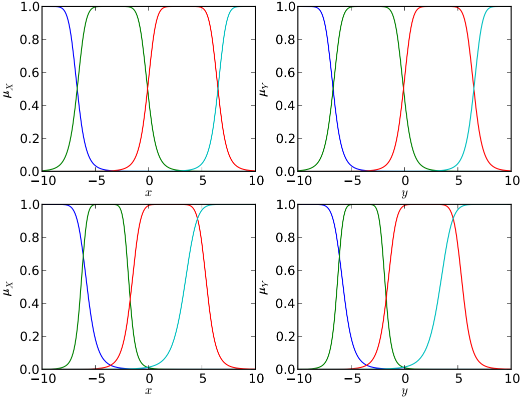
następniki modelu dostrojone mnk

Przykład I

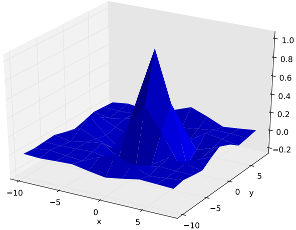
funkcje przynależności

sieć nie uczona

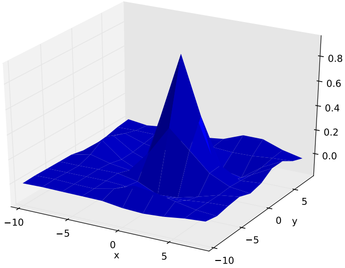
30 epok uczenia



Przykład I



dane testowe



30 epok uczenia

Przykład I

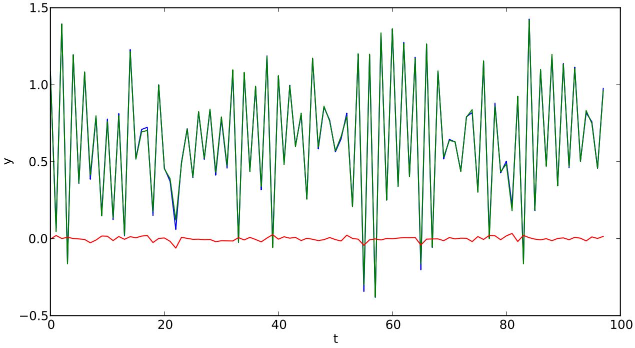
Training error: 0.3717143522138994

Stop training

Przykład II

$$y(i) = \sqrt{x(i)} - 0.5\sin(y(i-1)) + 0.4y(i-2)$$

model rozmyty: 3 wejścia po dwie dzwonowe funkcje przynależności, 8 reguł



Algorytmy ewolucyjne

- Ewolucja sztucznych osobników należących do populacji
- Mutacje
- Rozmnażanie poprzez krzyżowanie
- Dobór naturalny

Algorytmy ewolucyjne w strojeniu modeli rozmytych

- System rozmyty reprezentowany poprzez wektor genotypu
- Genotyp c – wektor parametrów poprzedników systemu
- Następniki zawsze dostrajane metodą najmniejszych kwadratów

Mutacja – mutacja nierównomierna

$$c^{l+1} = \begin{cases} c^l + \Delta(l, \delta_{c_{\max}}), & b = 0 \\ c^l - \Delta(l, \delta_{c_{\max}}), & b = 1 \end{cases}$$

$$\Delta(l, y) = y(1 - r^{(1-\frac{l}{l_{\max}})^b})$$

- $\delta_{c_{\max}}$ - wektor maksymalnych zmian genotypu
- b - liczba losowa ze zbioru $\{0, 1\}$
- r - liczba losowa z przedziału $[0, 1]$
- l, $l_{\max}$ - numer pokolenia, liczba pokoleń

Wielkość mutacji maleje w kolejnych pokoleniach

Krzyżowanie

$$
\begin{array}{ll}
C_1^{l+1} = aC_r^l + (1-a)C_s^l, & C_2^{l+1} = (1-a)C_r^l + aC_s^l, \\
C_3^{l+1} = \min(C_r^l, C_s^l), & C_4^{l+1} = \max(C_r^l, C_s^l)
\end{array}
$$

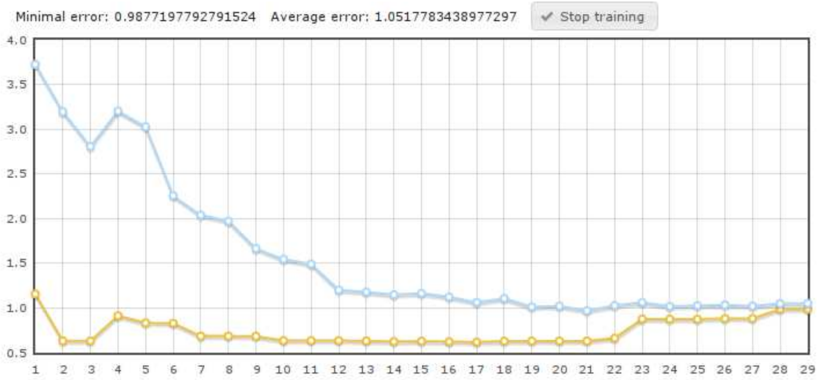
- max, min - ekstremum po współrzędnych
- a - liczba losowa z przedziału $[0, 1]$

Dobór naturalny

- Ruletka – prawdopodobieństwo wylosowania odwrotnie proporcjonalne do wartości błędu
- Możliwe strategie ewolucyjne:
  - μ, λ - przeżywają tylko osobniki z populacji potomnej
  - $\mu + \lambda$ - przeżywają osobniki z populacji potomnej i rodzicielskiej

Przykład III

Dane z przykładu I



Implementacja

- Wygodne i uniwersalne API do tworzenia i strojenia modeli rozmytych
- Elastyczność - swoboda wyboru struktury i parametrów dostrajalnych modelu
- Interfejs graficzny oparty o przeglądarkę
- Możliwość importu i eksportu danych wykorzystywanych przez MATLAB Fuzzy Toolbox
- Wykorzystywane technologie: Python, NumPy, SciPy, Flask, RabbitMQ

Interfejs programisty

3 moduły Pythona:

- `pyfis.struct` – Obiektowa struktura modeli rozmytych TSK
- `pyfis.anfis` – Algorytmy neuronowe strojenia modeli rozmytych
- `pyfis.evofis` – Algorytmy ewolucyjne strojenia modeli rozmytych

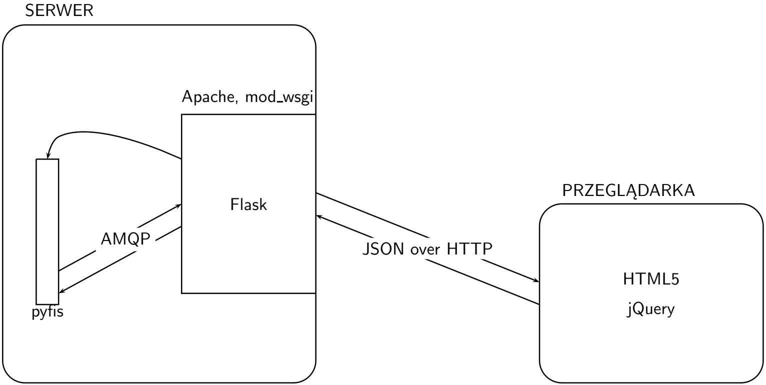
Interfejs użytkownika

```
fis = Fis(defuzzmethod="sum")

for j in range(2):
    inp = Input()
    for i in range(4):
        inp.mem_func.append(BellMemFunc([3.3, 4, -10+i*6.6]))
    fis.inputs.append(inp)

for i in range(4):
    for j in range(4):
        rule = Rule([0, 0, 0])
        rule.inputs.append((0, i))
        rule.inputs.append((1, j))
        fis.rules.append(rule)
```

Architektura

- Interfejs oparty o przeglądarkę
- Długotrwałe zadania obliczeniowe uruchamiane w zewnętrznych procesach
- Komunikacja procesów poprzez protokół AMQP
- Dane składowane w relacyjnej bazie danych

Architektura



Demo

Zadania

- Eksport modeli do MATLABa
- Automatyczne generowanie struktury modeli
- Modyfikacje algorytmu ewolucyjnego
- Zwiększenie szybkości działania aplikacji

Dziękuję za uwagę