WYBRANE METODY SZTUCZNEJ INTELIGENCJI ZAIMPLEMENTOWANE W JĘZYKU PROGRAMOWANIA C/C++

Aleksandra Mreła

Uniwersytet Kazimierza Wielkiego
Instytut Informatyki
ul.Mikołaja Kopernika 1, 85-074 Bydgoszcz
e-mail: aleksandra.mrela@ukw.edu.pl

Streszczenie: *Obecnie coraz częściej wykorzystuje się metody sztucznej inteligencji (AI) do budowania systemów ekspertowych, czy opartych na wiedzy. Jednakże, bardzo często podczas zajęć w szkole podstawowej i średniej podczas zajęć z programowania, lub ogólniej kształcenia myślenia komputacyjnego, uczniowie uczą się algorytmów rozwiązujących problemy w warunkach pewności. Wobec tego większość młodych ludzi nie ma okazji rozważania rozwiązań problemów w zakresie logiki rozmytej. Aby nauczyciele informatyki rozważali z uczniami metody sztucznej inteligencji, należy przygotować proste przykłady algorytmów AI. W artykule przedstawiono kilka prostych przykładów zbiorów i relacji rozmytych oraz prostego systemu wnioskującego zakodowanych w C++.*

Słowa kluczowe: *Edukacja, logika rozmyta, systemy wnioskujące, C++*

Selected methods of artificial intelligence implemented in C/C++ programming language

Abstract: *Currently, artificial intelligence (AI) methods are increasingly used to build expert or knowledge-based systems. However, very often during elementary and high school classes while programming classes, or more general, computational thinking training, students learn algorithms that solve problems in conditions of certainty. Therefore, most young people have no opportunity to consider solutions to fuzzy logic problems. For IT teachers to discuss artificial intelligence methods with students, simple AI algorithms should be prepared. The article presents some simple examples of fuzzy sets and relations as well as a simple inference system coded in C++.*

Keywords: *Education, fuzzy logic, inference systems, C++*

1. Wprowadzenie

Nauka programowania w szkole koncentruje się nauce myślenia komputacyjnego służącego do rozwiązywania różnych problemów, których rozwiązanie polega na znalezieniu algorytmu do ich rozwiązania [1]. Uczniowie, po opracowaniu algorytmu lub jego fragmentu, kodują je korzystając z wybranej technologii. Zgodnie z podstawą programową, uczeń w szkole średniej powinien stosować algorytmy dotyczące m.in. badania czy dana liczba jest pierwsza, wyznaczania NWD i NWW, szyfrowania tekstu metodą Cezara, porządkowania ciągu liczb, wydawania reszty czy obliczania wartości ciągu metodą iteracyjną czy rekurencyjną [2].

Obecnie, w dobie gwałtownego rozwoju sztucznej inteligencji, uczniowie powinni uczyć się także rozwiązywać zadania w warunkach niepewności. Jak twierdzi M. M. Sysło, „przybliżenie sztucznej inteligencji kręgom edukacyjnym z przekonaniem, że ta dziedzina czeka na uwzględnienie jej w realizacji powszechnego kształcenia wszystkich uczniów" [3]. W artykule omówiono przykłady, zakodowane w C++, zbiorów i relacji rozmytych oraz prostego systemu wnioskującego.

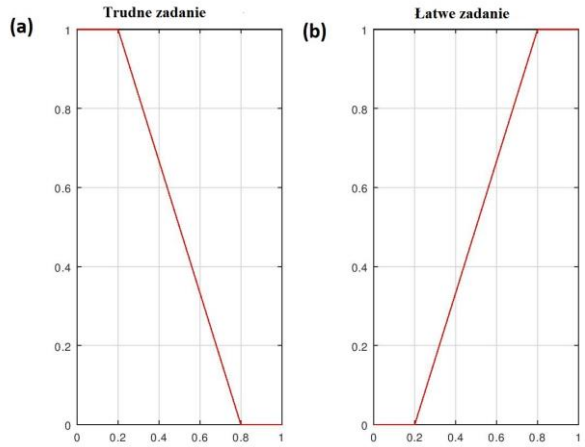
2. Zbiory rozmyte

Idea zbiorów rozmytych została wprowadzona i opracowana przez L. A. Zadeha w 1965 r. [5]. Zbiorem rozmytym A w przestrzeni X jest zbiór par uporządkowanych $\{(x, \mu_A(x), x \in X\}$, gdzie $\mu_A(x): X \rightarrow [0,1]$ jest funkcją przynależności. Wartość $\mu_A(x)$ określa stopień przynależności elementu x do zbioru A. Powyższa definicja jest z pewnością trudna do zrozumienia dla uczniów, ale poprzez różne przykłady można ją im przybliżyć.
Rozważmy przykład zbioru rozmytego z dziedziny bliskiej uczniom, a mianowicie zbioru „trudne

zadanie". Niech $X = [0,1]$ i niech funkcja przynależności będzie określona w następujący sposób:

$$\mu_{trudne\ zadanie}(x) = \begin{cases} 1 & x < 0{,}2 \\ -\left(\frac{5}{3}\right)x + \frac{4}{3} & 0{,}2 \le x \le 0{,}8 \\ 0 & x > 0{,}8. \end{cases}$$

Wykres funkcji przynależności zbioru rozmytego „trudne zadanie" jest przedstawiony na rys. 1 (a).



Rysunek. 1 Wykresy funkcji przynależności zbiorów rozmytych: (a) trudne zadanie, (b) łatwe zadanie.

Po przeanalizowaniu wzoru funkcji i ewentualnie wykresu, można zakodować jej formułę w C++ następująco:

```
double x;
double trudne_zadanie;
  if (x < 0.2)
     trudne_zadanie=1;
  else if (x < 0.8)
     trudne_zadanie = -(5/3) * x + 4/3;
  else
     trudne_zadanie = 0;
```

Zatem uczniowie mogą, podając różne wartości x, wyznaczać wartości funkcji przynależności. Podobnie, można zdefiniować funkcję przynależności zbioru rozmytego „łatwe zadanie zadanie" wzorem

$$\mu_{łatwe\ zadanie}(x) = \begin{cases} 0 & x < 0{,}2 \\ \left(\frac{5}{3}\right)x - \frac{1}{3} & 0{,}2 \le x \le 0{,}8 \\ 1 & x > 0{,}8. \end{cases}$$

Wykres funkcji jest przedstawiony na rys. 1 (b), a kod w C++ jest zapisany w następujący sposób:

```
double x;
  double latwe_zadanie;
    if (x < 0.2)
       latwe_zadanie =0;
    else if (x < 0.8)
       latwe_zadanie = (5/3) * x - (1/3);
    else
       latwe_zadanie = 1;
```

Wobec tego dla różnych wartości x uczniowie mogą określać stopień przynależności do zbiorów rozmytych „zadanie łatwe" i „zadanie trudne". Uczniowie oczywiście będą pytać jak wyznaczyć wartość x. Rozważmy przykład. W klasie jest 30 uczniów, którzy rozwiązywali zadanie z matematyki. Przyjmijmy, że x jest równe stosunkowi uczniów, którzy rozwiązali poprawnie wybrane zadanie, do wszystkich uczniów. Zatem, jeśli 10 studentów rozwiązało to zadanie, to

$$x = \frac{10}{30} = 0{,}3333,$$

więc $\mu_{trudne\ zadanie}(0{,}3333) = 0{,}7778$ oraz

$$\mu_{łatwe\ zadanie}(0{,}3333) = 0{,}2222.$$

Wobec tego powyższe zadanie jest trudne w 78% i łatwe w 22%. Oczywiście, pierwsze spotkanie ze zbiorami rozmytymi może być trudne dla uczniów, to po rozważeniu kilku przykładów, w tym zakodowaniu kilku algorytmów pojęcie zbioru rozmytego stanie się łatwe.

3. Relacje rozmyte

Jeśli rozważaną przestrzenią jest iloczyn kartezjański $X \times Y$, to zbiór rozmyty nazywamy relacją rozmytą. Rozważmy, bliską uczniom, relację rozmytą R_1 =„Wiedza $\times$ Zadanie", gdzie X ="Wiedza" = {X_1 – Miejsca zerowe funkcji liniowych, X_2 – Wartości funkcji liniowych, X_3 – Wykres funkcji liniowych} i Y = Zadanie = {Z_1, Z_2, Z_3, Z_4}. Niech relacja R_1 będzie zdefiniowana w następujący sposób

$$R_1 = \begin{bmatrix} 0{,}8 & 0{,}3 & 0{,}2 & 0{,}4 \\ 0{,}5 & 0{,}6 & 0{,}7 & 0{,}8 \\ 0{,}9 & 0{,}5 & 0{,}6 & 0{,}7 \end{bmatrix}$$

Niech zadanie 1 (Z_1) będzie określone następująco: Niech $y = 2x + 1.$ Wyznaczyć miejsca zerowe funkcji y, obliczyć $y(0)$ oraz narysować wykres tej funkcji. Na tej podstawie przyjęto, że

$$R_1(Miejsca\ zerowe, Z_1) = 0{,}8;$$
$$R_1(Wartości\ funkcji, Z_1) = 0{,}5;$$
$$R_1(Wykres, Z_1) = 0{,}9.$$

Podczas dyskusji z uczniami, można ustalić inne wartości rozważanej relacji. Relację R_1, wraz z operacją wyświetlenia wartości tej relacji na ekranie monitora, można zakodować w C++ następująco:

```
double R1[3][4];
R1[0][0]=0.8;  R1[0][1]=0.3;
R1[0][2]=0.2;  R1[0][3]=0.4;
R1[1][0]=0.5;  R1[1][1]=0.6;
R1[1][2]=0.7;  R1[1][3]=0.8;
R1[2][0]=0.9;  R1[2][1]=0.5;
R1[2][2]=0.6;  R1[2][3]=0.7;
for (int i=0; i<3; i++)
{
  for (int j=0; j<4; j++)
  {
   cout << "R1["<< i << "," << j << "]=" <<
relacja_R1[i][j] << endl;
  }
}
```

Zbudujemy także relację rozmytą R_2 między zadaniami Y a uczniami $Z = \{U_1, U_2, U_3, U_4, U_5\}$, która przyjmuje dwie wartości: 1 oznacza, że uczeń rozwiązał zadanie poprawnie, a 0, że uczeń nie rozwiązał zadania. Niech relacja R_2 będzie określona następująco

$$R_2 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix},$$

którą można zakodować w C++ w następujący sposób

```
double R2[4][5];
R2[0][0]=1;  R2[0][1]=0; R2[0][2]=1;
R2[0][3]=1;  R2[0][4]=0;
R2[1][0]=1;  R2[1][1]=1; R2[1][2]=0;
R2[1][3]=1;  R2[1][4]=1;
R2[2][0]=1;  R2[2][1]=1;  R2[2][2]=1;
R2[2][3]=0;  R2[2][4]=1;
R2[3][0]=0;  R2[3][1]=0;  R2[3][2]=1;
R2[3][3]=1;  R2[3][4]=1;
for (int i=0; i<3; i++)
{
for (int j=0; j<4; j++)
{
cout << "R2["<< i << "," << j <<
"]=" << relacja_R2[i][j] << endl
 }
}
```

Uczniowie chcą ocenić poziom wiedzy z zakresu funkcji liniowych. Do wyznaczenia relacji $R_3 = X \times Z$ wykorzystamy $S - T$-złożenie relacji R_1 i R_2, wykorzystując jako S-konormę funkcję max, a jako T-normę funkcję min. Wobec tego, zgodnie z [4], złożenie relacji $max - min$ jest zapisane następująco:

$$R_3(x,z) = max_{y \in Y}(\min(R_1(x,y), R_2(y,z)).$$

Złożenie relacji $max - min$ jest dosyć trudne do zakodowania, ponieważ wymaga zastosowania trzech pętli, ale oczywiście jest możliwe do opracowania dla uczniów:

```
double R3[3][5];
for (int i=0; i<3; i++)
{
 for (int j=0; j<5; j++)
   {
    R3[i][j]=0;
    for (int k=0;k<4;k++)
     {
      R3[i][j]=max(R3[i][j],
min(R1[i][k],R2[k][j]));
     }
   }
}
```

Wobec tego, relacja R_3 jest równa:

$$R_3 = \begin{bmatrix} 0{,}8 & 0{,}3 & 0{,}8 & 0{,}8 & 0{,}4 \\ 0{,}7 & 0{,}7 & 0{,}8 & 0{,}8 & 0{,}8 \\ 0{,}9 & 0{,}6 & 0{,}9 & 0{,}9 & 0{,}7 \end{bmatrix}.$$

Rozważmy wartość $R_3(X_1, U_1) = 0{,}8$, która oznacza, że uczeń U_1 posiadł wiedzę na temat miejs zerowych funkcji liniowych na poziomie 80%. Podobnie, jak w przypadku zbiorów rozmytych, można stworzyć ze studentami kilka przykładów relacji, zakodować je w C++ i opracować interpretacje uzyskanych wyników algorytmu w języku C++.

4. Systemy wnioskujące

Wraz z uczniami można zastosować i zakodować prosty schemat wnioskowania rozmytego, typu Tagano-Sugeno [4]. Załóżmy, że określamy poziom trudności zadania x jako stosunek liczby osób, które rozwiązały dane zadanie, to liczby osób w grupie czy klasie. Poziom rozwiązania zadania oznacza liczbę punktów y jaką otrzymał rozważany uczeń transformowaną do przedziału $[0,1]$. Ponadto, można określić próg zdawalności zadania, czy wymagany poziom znajomości tematu z.
Jako przesłanki zdefiniujemy cztery zbiory rozmyte „łatwe_zadanie”, „trudne_zadanie”, „slaba_odpowiedz” i „dobra_odpowiedz”, co może być zakodowane w następujący sposób:

```
        double latwe_zadanie,
      trudne_zadanie, slaba_odpowiedz,
      dobra_odpowiedz;
           if (x < 0.2)
      trudne_zadanie=1;
    else if (x < 0.8)
       trudne_zadanie = -(5/3) * x + 4/3;
    else
        trudne_zadanie = 0;
   if (x < 0.2)
       latwe_zadanie=0;
    else if (x < 0.8)
       latwe_zadanie = (5/3) * x - (1/3);
    else
        latwe_zadanie = 1;
    if (y < 0.3)
       slaba_odpowiedz=1;
    else if (y < 0.7)
       slaba_odpowiedz = -2.5 * y + 1.75;
    else
        slaba_odpowiedz = 0;
    if (y < 0.3)
       dobra_odpowiedz=0;
    else if (y < 0.7)
       dobra_odpowiedz = 2.5 * y - 0.75;
    else
        dobra_odpowiedz = 1;
```

Następnie w celu zdefiniowania następnika tworzymy macierz $Y[3]$ złożoną z 4 wartości określających 4 różne poziomy wiedzy określone regułami:

Jeśli zadanie jest latwe i odpowiedz jest slaba, to ocena jest niedostateczna, czyli Y[0] = 2.
Jesli zadanie jest latwe i cdpowiedz jest dobra, to ocena jest dostateczna, czyli Y[1] = 3.
Jesli zadanie jest trudne i odpowiedz jest słaba, to ocena jest dobra, czyli Y[2] = 4.
Jesli zadanie jest trudne i odpowiedz jest dobra, to ocena jest bardzo dobra, czyli Y[3] = 5.

W ostatniej części naszego programu, określamy wagi w przy pomocy funkcji min, wyznaczamy ostateczną ocenę jako średnią ważoną i porównujemy ją z ustalonym wcześniej poziomem znajomości materiału:

```
double w[3], Y[3];
 w[1]=min(latwe_zadanie, slaba_odpowiedz);
 Y[0] =2;
 w[2]=min(latwe_zadanie, dobra_odpowiedz);
 Y[1] = 3;
 w[3]=min(trudne_zadanie, slaba_odpowiedz);
 Y[2] = 4;
 W[4]=min(trudne_zadanie, dobra_odpowiedz);
 Y[3] = 5
 double ostateczna_ocena;
    ostateczna_ocena=(w[0]*Y[0] + w[1]*Y[1]+
w[2]*Y[2] + w[3]*Y[3])/(w[0]+w[1]+w[2]+w[3]);
    cout << "ostateczna ocena wynosi " <<
ostateczna_ocena << endl;

    if (ostateczna_ocena >= z)
    cout << "Umiesz ten rozdzial wystarczajaco
dobrze. Przejdz do nastepnego rozdzialu";
    else
    cout << "Musisz powtorzyc material jeszcze
raz";
```

Po zakodowaniu systemu wnioskującego, można pokazać uczniom, że wnioskowanie rozmyte nie musi być trudne i mogą zaimplementować prosty system wnioskujący, a następnie można tworzyć bardziej skomplikowane systemy wnioskujące.

5. Podsumowanie

Uczniowie szkół średnich, a nawet podstawowych, powinni kształcić proste metody sztucznej inteligencji. Obecnie, wykorzystuje się coraz częściej systemy wnioskujące, więc uczniowie powinni rozumieć podstawy zbiorów i relacji rozmytych. W szkole rozpatruje się tylko jeden z rodzajów niepewności, a mianowicie rachunek prawdopodobieństwa, natomiast w życiu codziennym znajdujemy się w sytuacjach innego rodzaju niepewności, który może być opisany metodami logiki rozmytej.

Język programowania C++ doskonale nadaje się do kodowania zbiorów, czy relacji rozmytych oraz wyznaczania ich sum, różnic, czy złożeń, a także do zbudowania prostego systemu wnioskującego z uczniami podczas zajęć z informatyki. Należy jednakże pamiętać o opracowaniu pojęć logiki rozmytej oraz stosowanych metod na bardzo niskim poziomie abstrakcji.

Literatura

1. Drab T., „Informatyka oparta na rachunkach”, Materiały XVI konferencji Informatyka w Edukacji, Toruń 2019, *https://iwe.mat.umk.pl/tom-iwe2019/19.pdf*, dostęp 11.10. 2019 r.
2. Podstawa programowa, Informatyka, liceum/technikum, Podstawa programowa.pl, *https://podstawaprogramowa.pl/Liceum-technikum/Informatyka*, dostęp 11.10.2019 r.
3. Sysło M.M., „Inteligencja +”, Materiały XVI konferencji Informatyka w Edukacji, Toruń

2019, *https://iwe.mat.umk.pl/tom-iwe2019/03.pdf*, dostęp 11.10.2019 r.

4. Rutkowski L., Metody i techniki sztucznej inteligencji, PWN, Warszawa 2012.
5. Zadeh L. A., Fuzzy sets, Information and Control, 8 (3), 1965, str. 338-353.