Wstęp do inteligencji komputerowej – zajęcia nr 4

Jarosław Stańczak
WSISiZ

Systemy rozmyte (zbiory, logika, liczby i wnioskowanie rozmyte):

- podstawy i zastosowania w sztucznej inteligencji
- działania na zbiorach i liczbach rozmytych
- sterowanie rozmyte.

Uczenie maszynowe:

- uczenie z nadzorem i bez nadzoru
- uczenie się ze wzmocnieniem
- generacja zbioru reguł.

Logika wielowartościowa

Klasyczna logika dwuwartościowa nie zawsze może być wykorzystana do wnioskowania w świecie informacji niepewnej, niepełnej i niedookreślonej. Bardzo często pewne pojęcia, opisujące świat, fakty, zdarzenia, relacje również nie są ostre. Oznacza to, że do opisu realnego świata potrzebne są metody wnioskowania oparte na bardziej złożonym obrazie świata, w którym fakty mogą być prawdziwe lub fałszywe do pewnego stopnia lub też nie zachodzi dla nich prawo wyłączonego środka:

$$(p \vee \neg p).$$

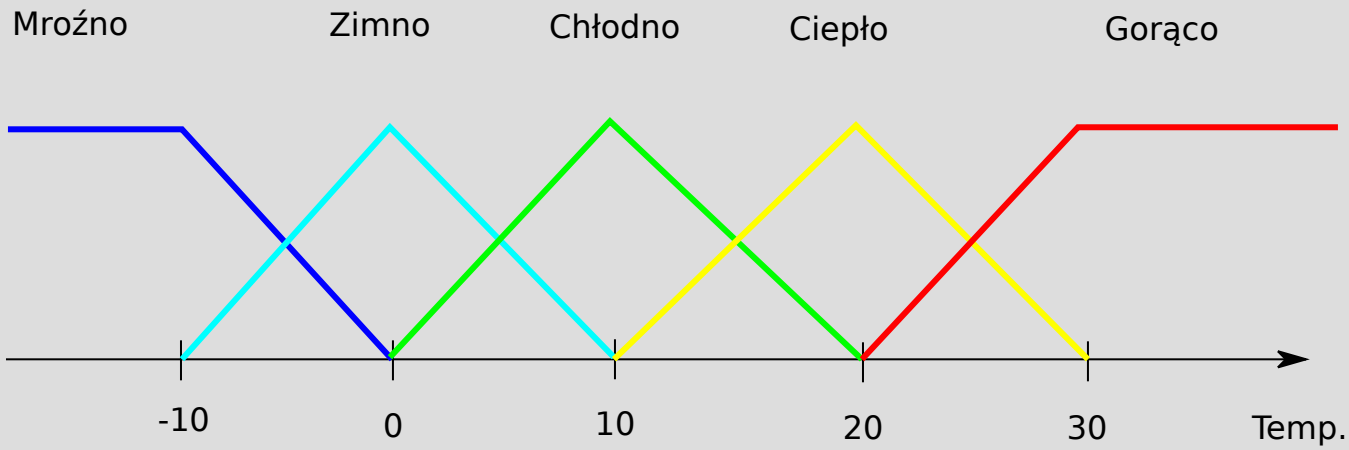
Tego typu właściwości mogą mieć logiki wielowartościowe, a logika rozmyta jest jedną z nich, dodatkowo przejście od prawdy do fałszu jest płynne od 0 do 1 ze wszystkimi możliwymi stanami pośrednimi.

Logika rozmyta a zbiory rozmyte

Logika rozmyta bazuje na pojęciu **zbioru rozmytego,** wprowadzonego przez Lotfiego Zadeha. W klasycznej definicji zbioru **funkcja charakterystyczna** ($\varphi_A(x)$) jego elementów ma wartość 0, gdy element do niego nie należy, i 1, gdy należy. Dla zbioru rozmytego ta funkcja charakterystyczna zastępowana jest tzw. **funkcją przynależności** ($\mu_A(x)$), która jest płynna i elementy mogą do takiego zbioru należeć z wartością funkcji przynależności od 0 do 1. Takie podejście daje możliwość opisu zjawisk o nieprecyzyjnie określonych granicach.
Jest to sytuacja podobna jak w logice rozmytej, w której fakty nie muszą być jednoznacznie określane jako prawdziwe lub fałszywe, lecz może im być przypisywana dowolna wartość z przedziału $\langle 0,1 \rangle$.

Przykłady zbiorów rozmytych

Pojęcie „temperatura” jest tu tzw. zmienną lingwistyczną, która przyjmuje wartości: „mroźno”, „zimno”, „chłodno”, „ciepło”, „gorąco”. Wartości tej zmiennej są zbiorami rozmytymi, opisanymi przedstawionymi na rysunku funkcjami przynależności.
Opis ten jest dość subiektywny i mógłby się zasadniczo różnić, gdyby go dokonał mieszkaniec tropików lub mieszkaniec Arktyki.



Funkcje przynależności do zbioru rozmytego

Nie ma narzuconego przez definicję kształtu funkcji przynależności, a jedynie ogólne właściwości:

- 0 – brak przynależności, 1 – pełna przynależność i możliwość przyjmowania dowolnych wartości między 0 a 1,
- **monotoniczność –** im bardziej dany element "pasuje" do zbioru, tym większą wartość przyporządkowuje mu funkcja przynależności,
- **symetria** – elementy w równym stopniu spełniające kryteria przynależności do zbioru rozmytego muszą mieć takie same wartości funkcji przynależności.

Funkcje przynależności do zbioru rozmytego

Najczęściej stosuje się kształt trójkątny, trapezoidalny, czasem prostokątny lub tzw. singletony rozmyte, czyli „słupki” o wysokości równej wartości funkcji przynależności, z uwagi na prostotę, łatwość zapisu i przeprowadzania obliczeń, ale oczywiście mogą to być również odcinki innych krzywych (np. paraboli, okręgu, funkcji Gaussa).

Zazwyczaj wartości funkcji przynależności dla danego elementu, gdy należy do wielu zbiorów rozmytych, sumują się do 1, ale nie jest to warunek konieczny.

Funkcje przynależności do zbiorów: klasycznego i rozmytego

Wariant dla zbioru klasycznego:

$$\underset{x \in X}{\forall} \mu_A(x) = \begin{cases} 1 \ dla \ x \in A \\ 0 \ dla \ x \notin A \end{cases}$$

Wariant dla zbioru rozmytego:

$$\underset{x \in X}{\forall} \mu_A(x) = \begin{cases} (0,1\rangle \ dla \ x \in A \\ 0 \ dla \ x \notin A \end{cases}$$

Przynależność elementu x do zbioru

Definicja przynależności elementu do zbioru rozmytego wynika bezpośrednio z definicji funkcji przynależności:

$$x \in A \Leftrightarrow \mu_A(x) > 0$$

Operacje na zbiorach rozmytych

Na zbiorach rozmytych można przeprowadzać operacje analogiczne do zbiorów klasycznych: $\cap$ (iloczyn), $\cup$ (suma), ' (dopełnienie zbioru, bywają tu różne oznaczenia: –, $\neg$, $^-$).

Najczęściej operacje te są definiowane przez działania na funkcji przynależności do zbioru $\mu_A(x)$. Dodatkowo wprowadza się tu pojęcia *s*-normy i *t*-normy, które są uogólnieniami odpowiednio sumy i iloczynu zbiorów.

Operacje na zbiorach rozmytych
iloczyn zbiorów (*t*-norma)
różne wersje

Operator	Wzór
minimum	$\mu_{A\cap B}(x)=MIN(\mu_A(x),\mu_B(x))$
iloczyn	$\mu_{A\cap B}(x)=\mu_A(x)*\mu_B(x)$
iloczyn Hamachera	$\mu_{A\cap B}(x)=\frac{\mu_A(x)*\mu_B(x)}{\mu_A(x)+\mu_B(x)-\mu_A(x)*\mu_B(x)}$
iloczyn Einsteina	$\mu_{A\cap B}(x)=\frac{\mu_A(x)*\mu_B(x)}{2-(\mu_A(x)+\mu_B(x)-\mu_A(x)*\mu_B(x))}$
iloczyn drastyczny	$\mu_{A\cap B}(x)=\begin{cases} MIN(\mu_A(x),\mu_B(x)) \ dla \ MAX(\mu_A(x),\mu_B(x))=1 \\ 0 \ dla \ pozostałych \end{cases}$
ograniczona różnica	$\mu_{A\cap B}(x)=MAX(0,\mu_A(x)+\mu_B(x)-1)$

Operacje na zbiorach rozmytych
suma zbiorów (*s*-norma)
różne wersje

Operator	Wzór
maximum	$\mu_{A\cup B}(x)=MAX(\mu_A(x),\mu_B(x))$
suma	$\mu_{A\cup B}(x)=\mu_A(x)+\mu_B(x)-\mu_A(x)*\mu_B(x)$
suma Hamachera	$\mu_{A\cup B}(x)=\frac{\mu_A(x)+\mu_B(x)-2*\mu_A(x)*\mu_B(x)}{1-\mu_A(x)*\mu_B(x)}$
suma Einsteina	$\mu_{A\cup B}(x)=\frac{\mu_A(x)+\mu_B(x)}{1+\mu_A(x)*\mu_B(x)}$
suma drastyczna	$\mu_{A\cup B}(x)=\begin{cases} MAX(\mu_A(x),\mu_B(x)) \ dla \ MIN(\mu_A(x),\mu_B(x))=0 \\ 1 \ dla \ pozostałych \end{cases}$
ograniczona suma	$\mu_{A\cup B}(x)=MIN(1,\mu_A(x)+\mu_B(x))$

Operacje na zbiorach rozmytych
negacja
dwie wersje, równoważne dla zbiorów o ograniczonym nośniku

Negację w zbiorach rozmytych można uzyskać, stosując wzór:

$$\mu_{\neg A}(x)=1-\mu_A(x)$$

lub też jeśli nośnik zbioru jest ograniczony (np. A={1, 2, 3, ... , K}) to:

$$\mu_{\neg A}(x)=\mu_A(K-x)$$

Operacje na zbiorach rozmytych zawieranie i równość zbiorów

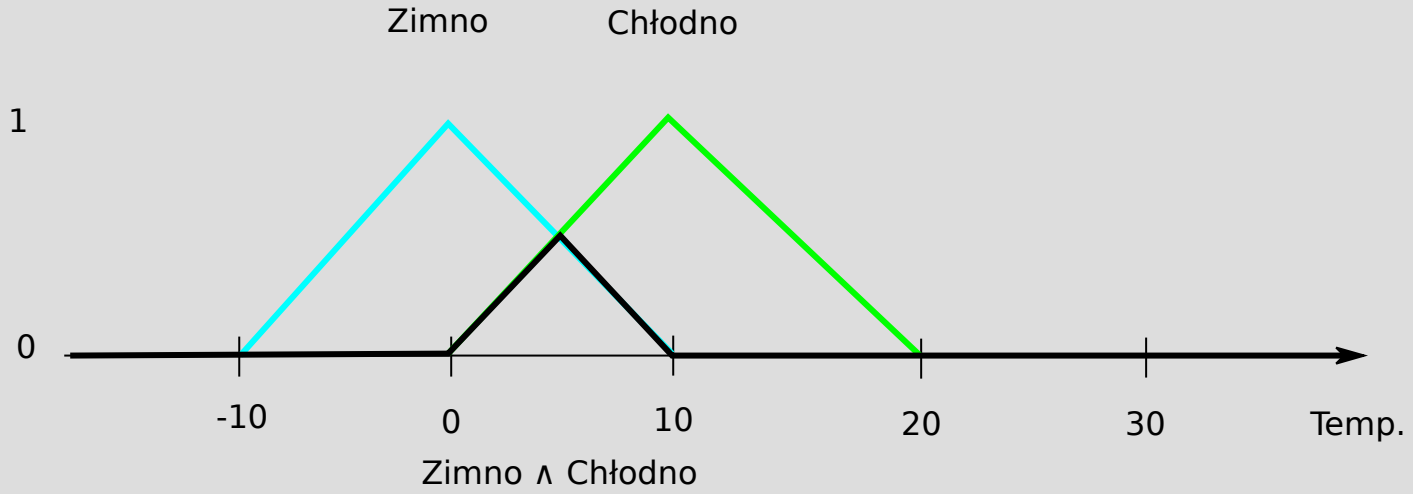
Zawieranie zbiorów rozmytych można zdefiniować następująco:

$$A \subseteq B \Leftrightarrow \underset{x \in X}{\forall} \mu_A(x) \leqslant \mu_B(x)$$

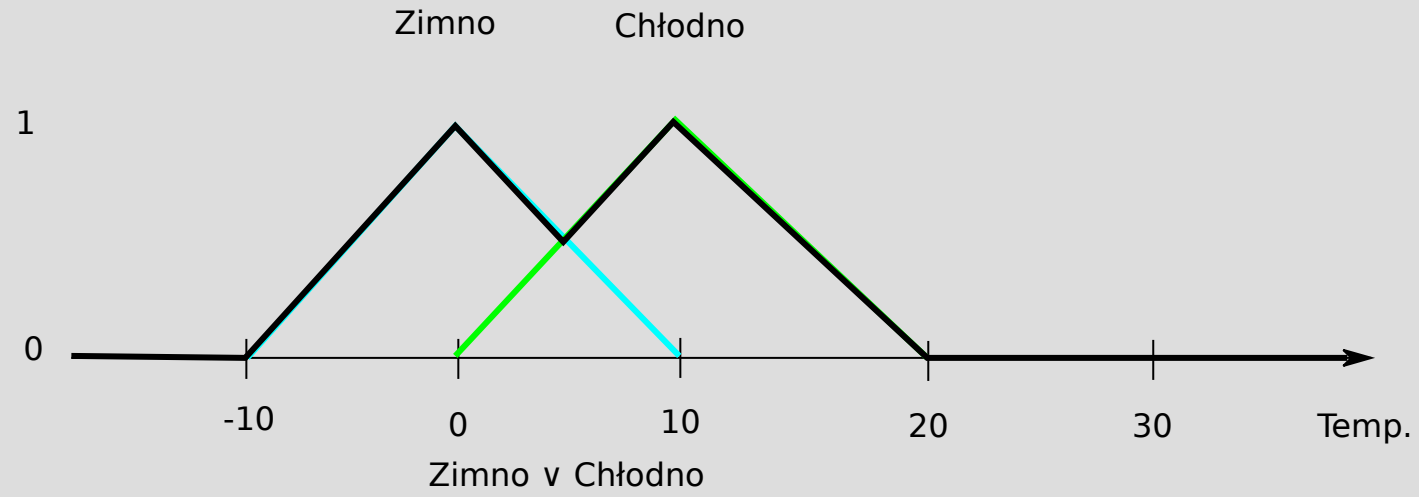
Równość zbiorów rozmytych można zaś zapisać jako:

$$A = B \Leftrightarrow \underset{x \in X}{\forall} \mu_A(x) = \mu_B(x)$$

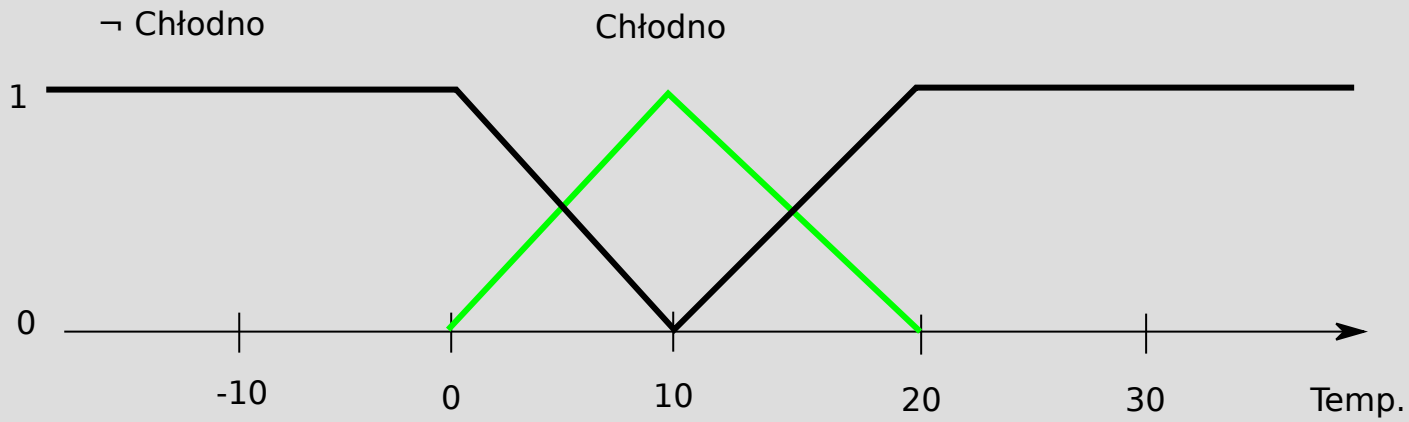
Operacje na zbiorach rozmytych przykład działania *t*-normy (∧) MIN



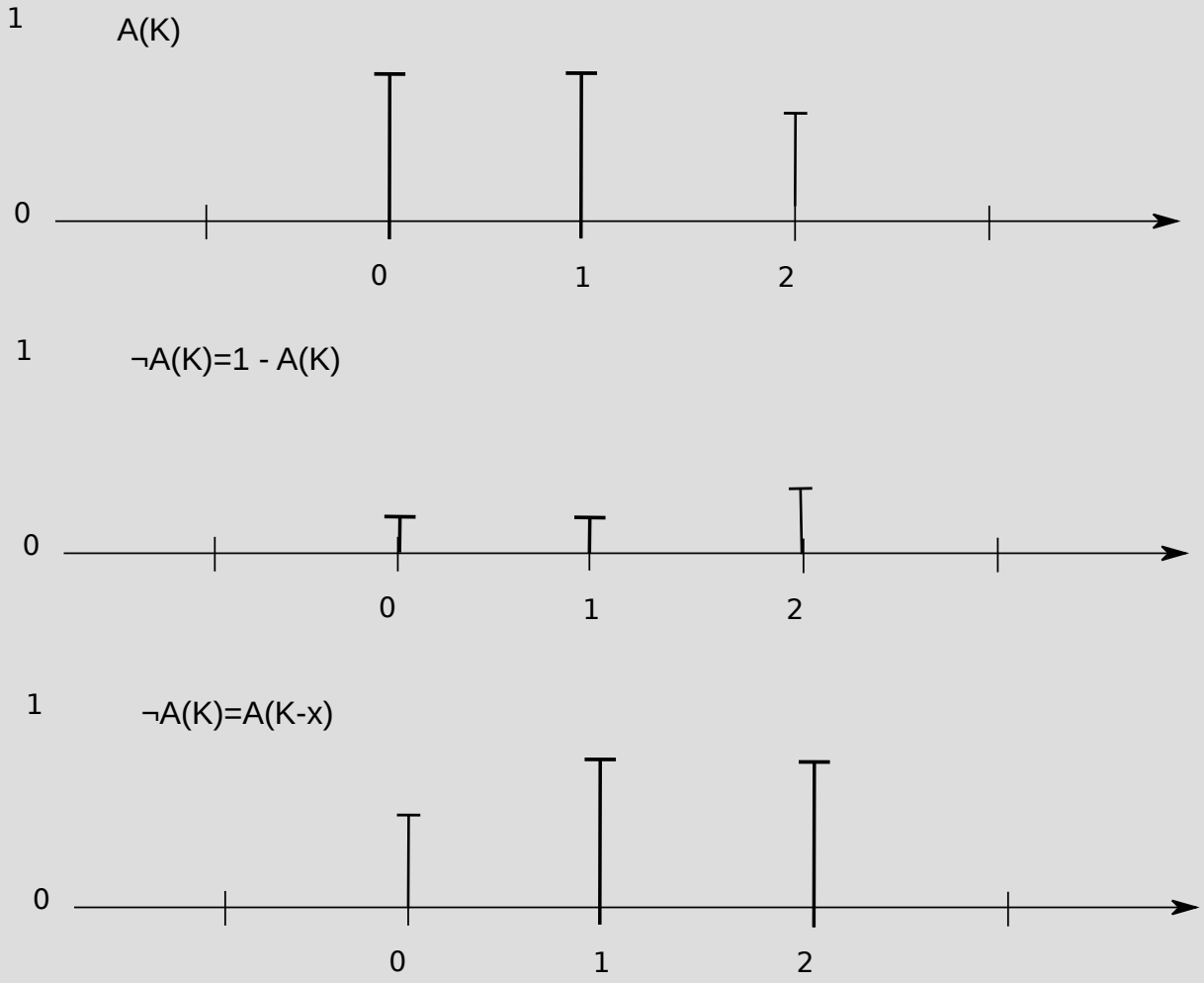
Operacje na zbiorach rozmytych przykład działania *s*-normy ($\vee$) MAX



Operacje na zbiorach rozmytych przykład działania dopełnienia (negacji, ¬) zbioru rozmytego



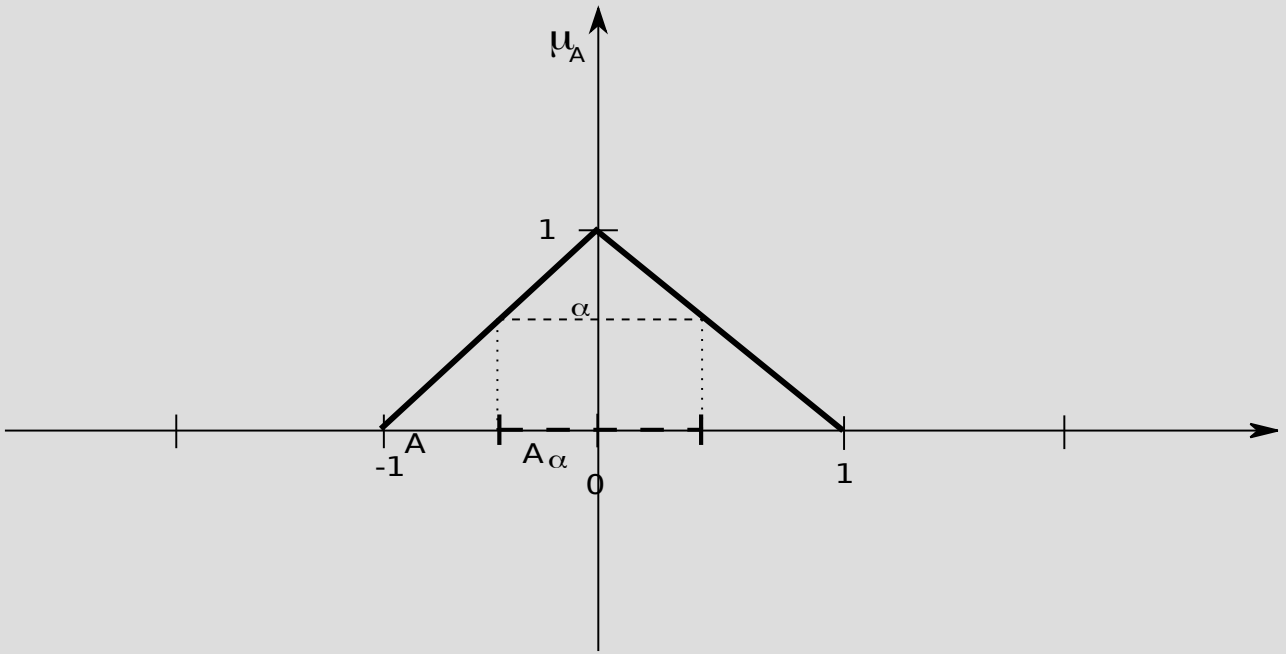
Operacje na zbiorach rozmytych
przykład działania dopełnienia (negacji, ¬) w wersji dla zbioru rozmytego ograniczonego i dyskretnego



Operacje na zbiorach rozmytych
α-cięcie

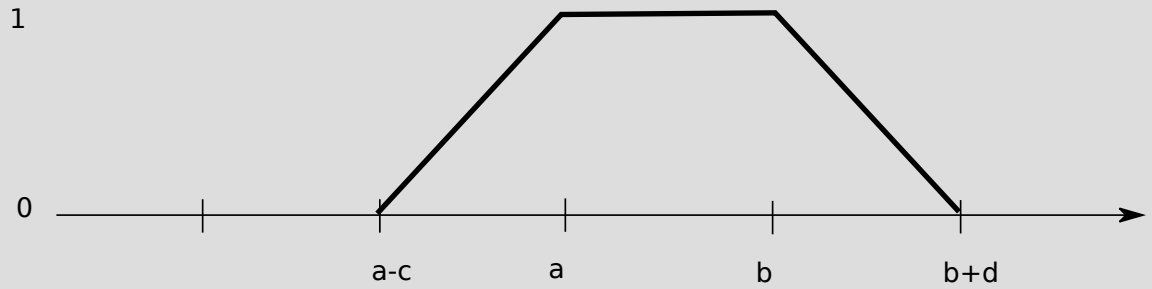
α-cięcie (α-przekrój) zbioru rozmytego A:

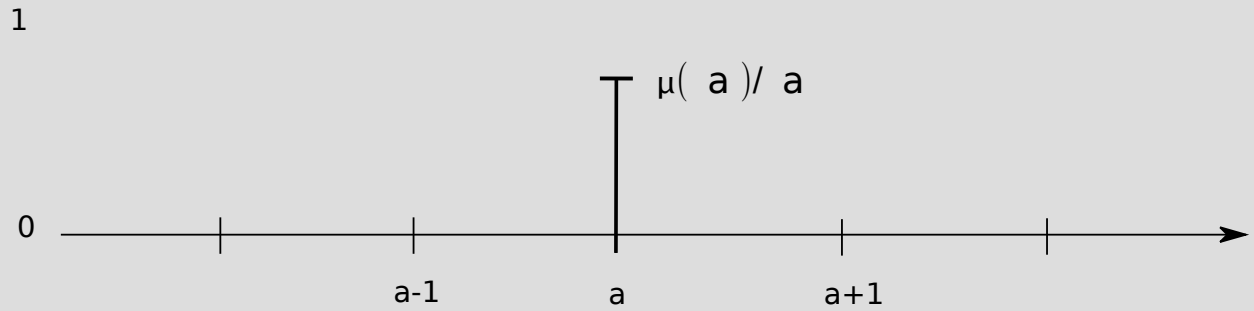
$$A_\alpha = \{x \in \mathbf{X}: \mu_A(x)) > \alpha\}$$



Liczby rozmyte

Rozszerzając pojęcie zbioru rozmytego, można wprowadzić pojęcie liczb rozmytych, czyli reprezentujących je zbiorów rozmytych, zapisanych jako czwórki liczb (a,b,c,d), określających funkcję przynależności do takiej liczby. W opisie takim mieści się też liczba rozmyta będąca singletonem rozmytym (a,a,0,0) lub prościej μ(a)/a.





Liczby rozmyte i działania na nich

- dodawanie

$$\mu_{A+B}(z)=\underset{x+y=z}{MAX}\left(\mu_A(x)\wedge\mu_B(y)\right)$$

- odejmowanie

$$\mu_{A-B}(z)=\underset{x-y=z}{MAX}\left(\mu_A(x)\wedge\mu_B(y)\right)$$

- mnożenie

$$\mu_{A*B}(z)=\underset{x*y=z}{MAX}\left(\mu_A(x)\wedge\mu_B(y)\right)$$

- dzielenie

$$\mu_{A/B}(z)=\underset{x/y=z,\,y\neq 0}{MAX}\left(\mu_A(x)\wedge\mu_B(y)\right)$$

- liczba przeciwna

$$\mu_{-A}(x)=\mu_A(-x)$$

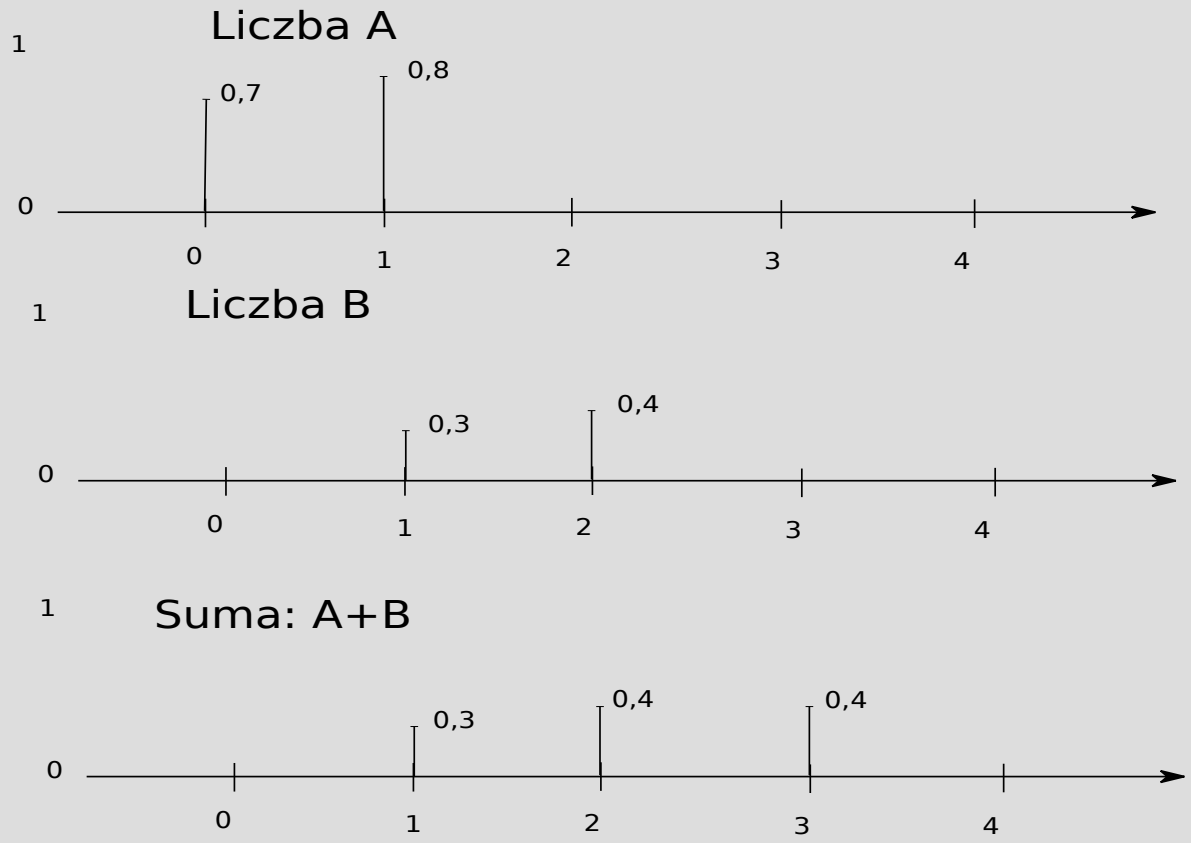
- liczba odwrotna

$$\mu_{A^{-1}}(x)=\mu_A(1/x),\ x\neq 0$$

Liczby rozmyte i działania na nich

przykład dodawania

$$\mu_{A+B}(z) = \underset{x+y=z}{MAX}\left(\mu_A(x) \wedge \mu_B(y)\right)$$



Liczby rozmyte i działania na nich
przykład dodawania

$$\mu_{A+B}(z) = \underset{x+y=z}{MAX}\left(\mu_A(x) \wedge \mu_B(y)\right)$$

A=0,7/0+0,8/1; B=0,3/1+0,4/2;

$\wedge$ - t-norma, w tym przypadku zastosowano operację MIN(...), lecz może być inna z tabeli na slajdzie 9.
MAX(...) - nie jest tu s-normą, lecz operacją wynikająca z tzw. reguły rozszerzania, która umożliwia przetwarzanie działań znanych z algebry liczb nierozmytych na rozmyte i nie może być zamieniona na inną.

1) x+y=1: (0+1) $\mu_{A+B}(1)=MAX(0,7 \wedge 0,3)=0,3$
2) x+y=2: (0+2,1+1) $\mu_{A+B}(2)=MAX(0,7 \wedge 0,4;\ 0,8 \wedge 0,3)=MAX(0,4;\ 0,3)=0,4$
3) x+y=3: (1+2) $\mu_{A+B}(3)=MAX(0,8 \wedge 0,4)=0,4$

A+B=0,3/1+0,4/2+0,4/3

Ćwiczenie (w domu)
Proszę obliczyć analogiczne różnicę, iloczyn i iloraz liczb rozmytych A i B.

Defuzyfikacja (wyostrzenie) zbiorów rozmytych

Używane w wielu zastosowaniach (np. sterownikach rozmytych) zbiory, liczby lub reguły rozmyte muszą w końcu zostać użyte do interakcji z rzeczywistym, nierozmytym obiektem.

Jak to zrobić?

Odpowiedzią jest defuzyfikacja, czyli „wyostrzenie” - zmiana wartości rozmytej w najlepiej odpowiadającą jej wartość nierozmytą. Istnieją różne metody przeprowadzania tej operacji.

Defuzyfikacja zbiorów rozmytych metoda „środka ciężkości”

Metoda „środka ciężkości” jest chyba najpowszechniej stosowaną, choć nie jedyną, metodą defuzyfikacji. Można ją zapisać w postaci następującego wzoru:

$$a = \frac{\sum_{i=1}^{n} x_i * \mu_A(x_i)}{\sum_{i=1}^{n} \mu_A(x_i)}$$

Jej nazwa wywodzi się z podobieństwa do fizycznego wzoru na obliczanie środka ciężkości ciała (w tym wypadku w przestrzeni dwuwymiarowej dyskretnej). W przypadku ciągłym wzór przedstawia się następująco:

$$a = \frac{\int_{x=x_p}^{x_k} x * \mu_A(x)\,dx}{\int_{x=x_p}^{x_k} \mu_A(x)\,dx}$$

Defuzyfikacja zbiorów rozmytych metoda maksimum

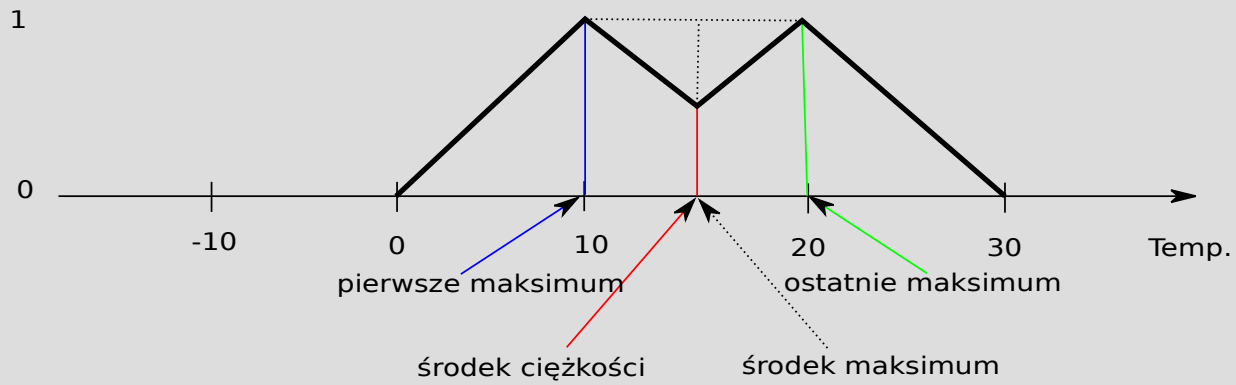
Metoda maksimum (największej wartości funkcji przynależności) jest znacznie prostsza od poprzedniej:

$$a = \sup_{x \in D} \mu(x)$$

Jednakże istnieją różne jej warianty, które mają znaczenie, gdy funkcja przynależności ma więcej niż jedno maksimum:

- metoda pierwszego maksimum
- metoda środka maksimum (środek odległości między maksimami pierwszym i ostatnim)
- metoda ostatniego maksimum.

Defuzyfikacja zbiorów rozmytych metodami maksimum i „środka ciężkości”

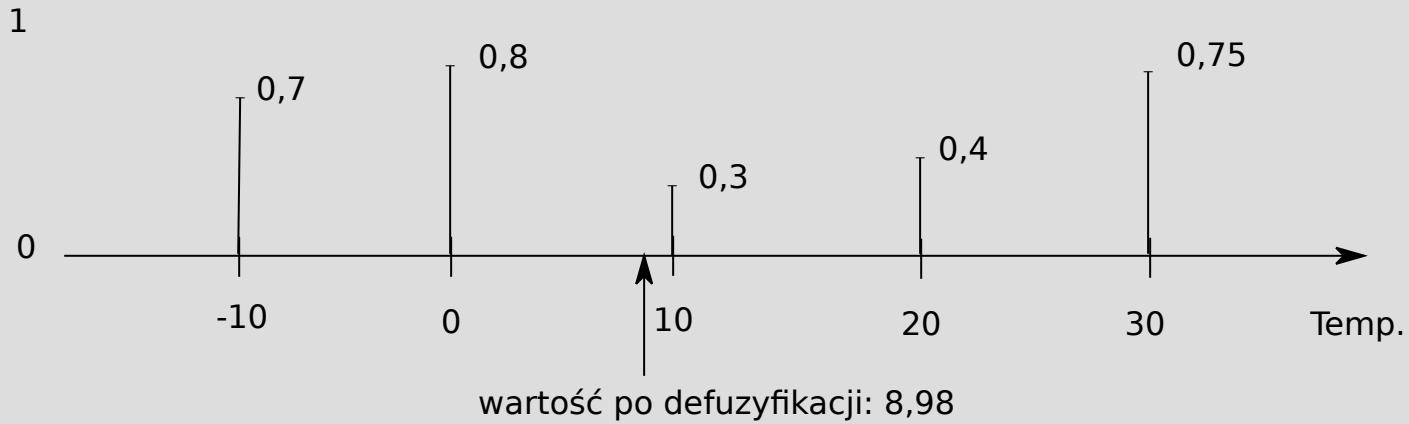


Defuzyfikacja zbiorów rozmytych metoda średnich ważonych

Metoda średnich ważonych przypomina nieco metodę „środka ciężkości” w wariancie dyskretnym (zbiór rozmyty składa się z tzw. singletonów rozmytych), gdyż ma zastosowanie właściwie tylko dla takiego przypadku; wagami są np. stopnie aktywacji reguł (jeśli wynik pochodzi z wnioskowania rozmytego):

$$a = \frac{\sum_{i=1}^{n} c * x_i * \mu_C}{\sum_{i=1}^{n} \mu_C}$$

Defuzyfikacja zbiorów rozmytych metoda środka ciężkości dla zbioru dyskretnego



Wnioskowanie rozmyte

Zbiory i liczby rozmyte stały się podstawą do tworzenia reguł rozmytych, które mogą być wykorzystywane do wnioskowania (np. w systemach ekspertowych), sterowania i podejmowania decyzji. Okazuje się, że taki sposób przetwarzania informacji daje nowe, ciekawe możliwości, których pozbawione były analogiczne systemy bez rozmytości.

Wnioskowanie w logice rozmytej przeprowadzane jest zazwyczaj z wykorzystaniem dobrze znanej z logiki klasycznej (i wspominanej na poprzednim wykładzie) reguły **modus ponens**, która prowadzi do powstania reguł decyzyjnych o postaci "jeśli ... wtedy..." (ang. if... then...). Wartości logiczne takich reguł mogą zawierać się w zakresie $\langle 0, 1 \rangle$, czyli zawsze są prawdziwe lub fałszywe tylko w jakimś stopniu. Przesłanki i konkluzje reguł opisują przynależność zmiennych decyzyjnych do określonych zbiorów rozmytych.

Wnioskowanie rozmyte

Przykładowa reguła może wyglądać następująco:

IF *x* is A AND *y* is B THEN *z* is C

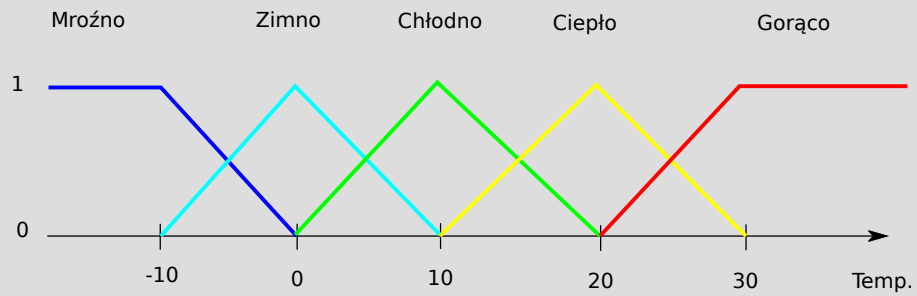
A i B to zbiory rozmyte, *x* i *y* to zmienne lingwistyczne/rozmyte (mogą to być też singletony rozmyte),
C to zbiór rozmyty w przypadku wnioskowania typu Mamdaniego, *z* – jest wtedy zmienną rozmytą, jednakże w prostszej wersji wnioskowania typu Takagi-Sugeno są to zwykłe zbiory/liczby/funkcje i zmienne nierozmyte.
AND jest t-normą (np funkcja MIN),
is – określa stopień przynależności do zbioru.

Wnioskowanie rozmyte w wersji Mamdaniego i Takagi-Sugeno

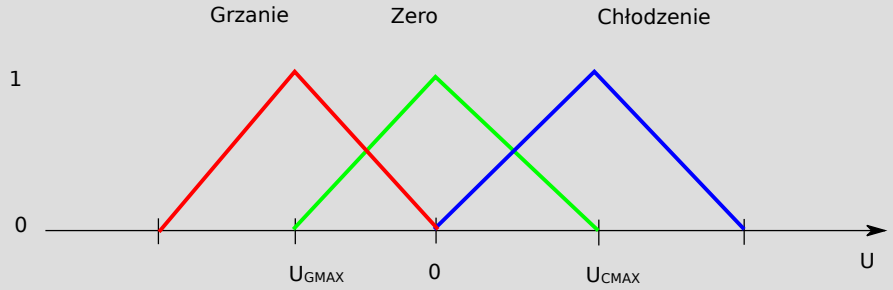
Aby reguła „odpaliła”, x i y muszą mieć niezerowe wartości funkcji przynależności. AND jest t-normą, dzięki której otrzymamy **wagę reguły** c jako wynik działania t-normy na otrzymane wartości funkcji przynależności $\mu_A(x)$ i $\mu_B(y)$. We wnioskowaniu typu Mamdaniego konkluzje są „ważone” wagami ich części warunkowych reguł c (najczęściej wynikowy zbiór rozmyty jest „przycinany” do wartości c, jest to tzw. α-cięcie zbioru).
Jeśli w danym momencie aktywnych jest więcej reguł, to one również są brane pod uwagę, ich konkluzje (zbiory rozmyte wynikowe) są „sumowane” przy użyciu s-normy (np. funkcji MAX).
Aby otrzymać konkretną, nierozmytą wartość wyjścia, otrzymane zbiory poddaje się defuzyfikacji.
W przypadku wnioskowania Takagi-Sugeno konkluzje są „ważone” wagami reguł c, a następnie sumowane - są to wartości nierozmyte, więc nie ma tu potrzeby przeprowadzania defuzyfikacji.

Wnioskowanie rozmyte

przykład: sterowanie ogrzewaniem w wersji Mamdaniego



Przedstawiany wcześniej zestaw zbiorów przesłanek reguł określających temperaturę



Zbiory określające sterowanie w konkluzji (wnioskowanie Mamdaniego)

Wnioskowanie rozmyte

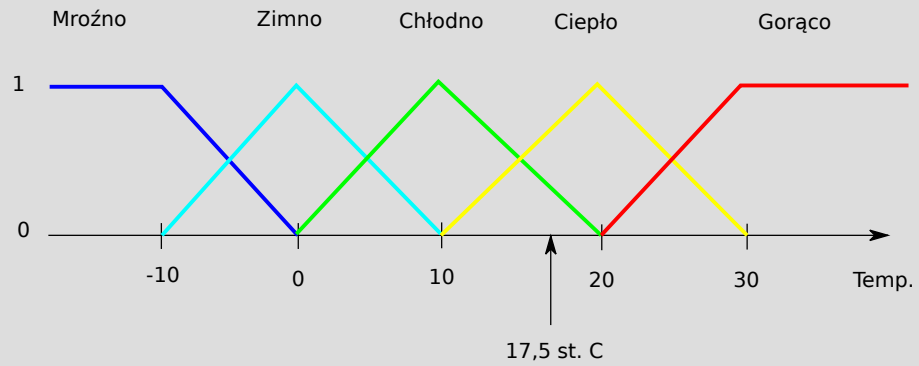
przykład: sterowanie ogrzewaniem w wersji Mamdaniego

R1: IF t is Mroźno THEN u is Grzanie
R2: IF t is Zimno THEN u is Grzanie
R3: IF t is Chłodno THEN u is Grzanie
R4: IF t is Ciepło THEN u is Zero
R5: IF t is Gorąco THEN u is Chłodzenie

Wnioskowanie rozmyte

przykład: sterowanie ogrzewaniem w wersji Mamdaniego

Załóżmy, że t=17,5 st. C



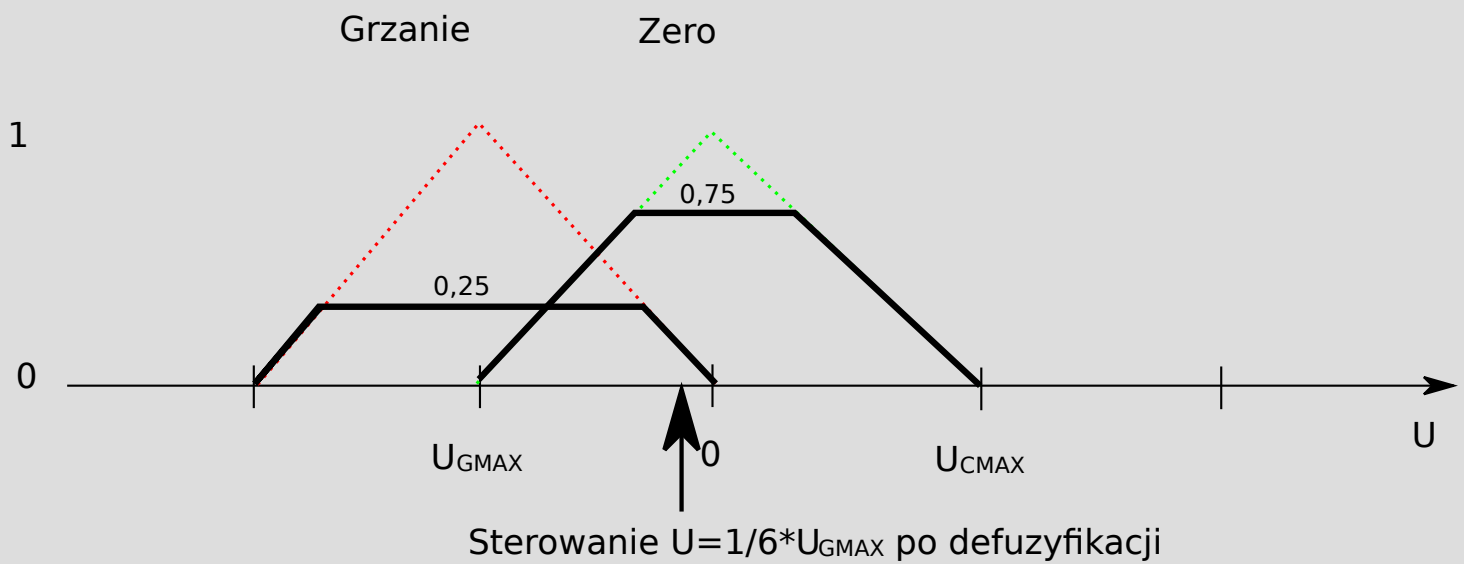
R1: IF t is Mroźno THEN u is Grzanie
R2: IF t is Zimno THEN u is Grzanie
R3: IF t is Chłodno THEN u is Grzanie
R4: IF t is Ciepło THEN u is Zero
R5: IF t is Gorąco THEN u is Chłodzenie

Aktywne są reguły R3 i R4 z odpowiednio
$\mu_{Chłodno}(t)=0,25$ i $\mu_{Ciepło}(t)=0,75$

Wnioskowanie rozmyte

przykład: sterowanie ogrzewaniem w wersji Mamdaniego

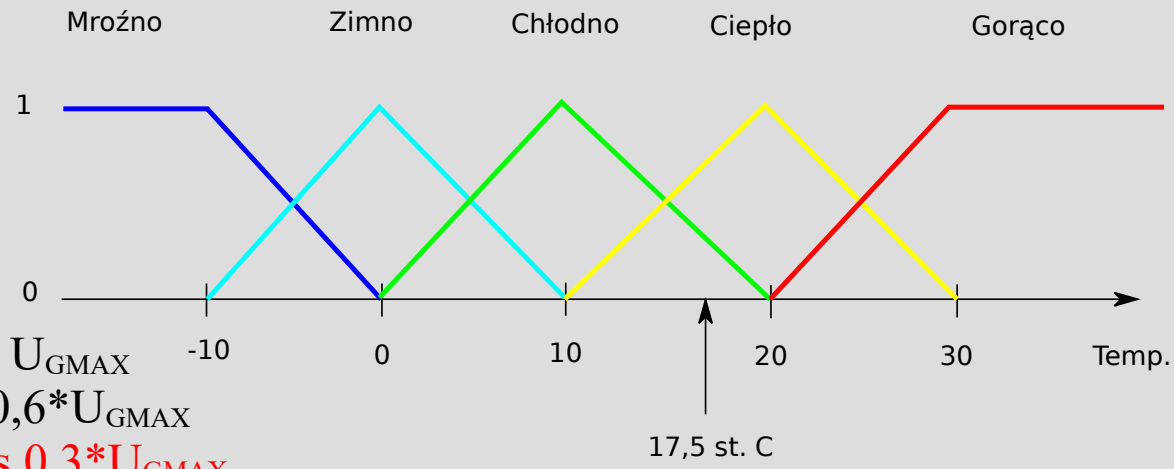
Otrzymujemy następujący zestaw konkluzji i po defuzyfikacji (metodą „środka ciężkości”) wynikające z niego sterowanie:



Sterowanie rozmyte

przykład: sterowanie ogrzewaniem z wnioskowaniem w wersji Takagi-Sugeno

Załóżmy, że t=17,5 st. C



R1: IF t is Mroźno THEN u is U_{GMAX}
R2: IF t is Zimno THEN u is 0,6*U_{GMAX}
R3: IF t is Chłodno THEN u is 0,3*U_{GMAX}
R4: IF t is Ciepło THEN u is 0
R5: IF t is Gorąco THEN u is 0,8*U_{CMAX}
Aktywne są reguły R3 i R4 z odpowiednio
$\mu_{Chłodno}(t)=0,25$ i $\mu_{Ciepło}(t)=0,75$
więc: $U=(0,25*0,3*U_{GMAX}+ 0,75*0)/(0,25+0,75) = 0,075*U_{GMAX}$

W przypadku sterownika w wersji Takagi-Sugeno nie ma potrzeby stosowania defuzyfikacji, choć trzeba policzyć sterowanie wynikowe!

Systemy rozmyte
zastosowania

- sterowanie obiektami dla których trudno jest zastosować konwencjonalne metody sterowania z uwagi na trudną definicję celu bądź trudny opis obiektu sterowania: urządzenia AGD (pralki, zmywarki, TV), autofokus w aparatach fotograficznych,
- sterowanie metrem w mieście Sugeno w Japonii,
- sterowanie autonomicznymi pojazdami
- wnioskowanie rozmyte w systemach ekspertowych,
- opis zjawisk i obiektów trudnych do opisania klasycznymi równaniami.

Uczenie maszynowe

Uczenie maszynowe (UM, *ang. machine learning*, ML) jest dość szerokim pojęciem, gdyż właściwie można pod nie podciągnąć wszystkie dziedziny sztucznej inteligencji, a już na pewno te, w których następuje pozyskiwanie nowych faktów, danych, wiedzy, adaptacja do zmiennych warunków i wszelkie przejawy modyfikacji i poprawy sposobu działania na skutek interakcji z otoczeniem.
Zacznijmy więc od definicji uczenia się (na podst. P. Cichosz „Systemy uczące się”):

„Uczeniem się systemu jest każda autonomiczna zmiana w systemie zachodząca na podstawie doświadczeń, która prowadzi do poprawy jakości jego działania.”

Nie jest to jedyna definicja uczenia, powstało ich dużo więcej.

Uczenie z nadzorem i bez

W systemach uczących mamy pewien element, który zmienia sposób swojego działania w miarę zachodzenia interakcji z otoczeniem. Procesem uczenia nie będzie każda sytuacja zmiany w sposobie działania, ale tylko te, które prowadzą do poprawy jego działania (zepsuty komputer zmienia swój sposób działania w stosunku do sprawnego, ale czy to jest efekt uczenia się?). Z tego wynika, że system taki musi mieć jakąś funkcję oceny.

W przypadku **uczenia z nadzorem** posiadamy pewne przykłady wraz z ich interpretacją, a interpretacja układu uczącego się podlega ocenie, przy czym ocena ta jest tym wyższa (lepsza), im odpowiedź układu jest bliższa interpretacji narzuconej w przykładzie trenującym. Sytuacja jest tu podobna do układu nauczyciel-uczeń.

W przypadku **uczenia bez nadzoru** system uczący otrzymuje tylko przykłady, bez ich interpretacji i sam musi je odpowiednio zinterpretować (najczęściej pogrupować). Funkcja oceny jest tu więc włączona w algorytm przetwarzania danych. Czasem stosuje połączenie obu paradygmatów uczenia i powstaje wtedy **uczenie półnadzorowane**.

Czasem wyróżnia się także **uczenie ze wzmocnieniem** jako osobny wariant, choć jest to specyficzny rodzaj uczenia z nadzorem.

Metody uczenia maszynowego

Metody uczenia maszynowego najłatwiej jest sklasyfikować na podstawie wykorzystywanych metod uczenia:

- uczenie ze wzmocnieniem,
- uczenie Bayesowskie (metody probabilistyczne),
- uczenie na podstawie przykładów,
- uczenie się (indukcja) zbioru reguł,
- uczenie się (indukcja) drzew decyzyjnych,
- uczenie się funkcji logicznych na podstawie przykładów,
- i wiele innych, włączając także sieci neuronowe, metody ewolucyjne i heurystyki obliczeniowe.

Cele uczenia maszynowego

Możemy zdefiniować następujące cele uczenia maszynowego:

- tworzenie lub odkrywanie nowych pojęć,
- wykrywanie regularności, prawidłowości lub zależności w danych,
- formułowanie/generowanie reguł decyzyjnych (np. na podstawie zbioru faktów),
- przyswajanie nowych pojęć i struktur przy pomocy uogólnienia i analogii,
- modyfikowanie, uogólniania i precyzowania danych,
- zdobywania wiedzy poprzez interakcję z otoczeniem,
- formułowania wiedzy w postaci zrozumiałej dla człowieka.

Programy uczące się

Uczenie maszynowe w każdym rozpatrywanym przypadku i metodzie sprowadzi się w końcu do pewnego ogólnego algorytmu, który na podstawie danych doświadczalnych będzie starał się zmienić tak swoje parametry, aby posiadać lepszą wartość jakiejś funkcji oceny. Najczęściej polega to na strojeniu pewnych parametrów algorytmu (często są to pewne parametry liczbowe) – jest to przykład pozyskiwania tzw. **wiedzy deklaratywnej**, opisującej pewne obiekty, sytuacje i związki (pojęcie to pojawiło się też na wykładzie o SE). Można także wyobrazić sobie algorytmy (są one trudniejsze, ale realizowalne), które modyfikują swój sposób działania, strategię działania. Mamy wtedy do czynienia z tzw. **wiedzą proceduralną**, określającą czynności. To drugie podejście jest zdecydowanie rzadziej spotykane z uwagi na trudności teoretyczne i praktyczne modyfikacji działającego programu (tę możliwość wykorzystują niektóre wirusy komputerowe).

Uczenie ze wzmocnieniem

Uczenie ze wzmocnieniem to powszechnie przyjęta nazwa metody uczenia, w której agent (program, maszyna...) uczy się na podstawie interakcji z otoczeniem. Interakcje polegają na tym, że w dyskretnych chwilach czasowych wykonuje on pewne akcje (działania) z pewnego skończonego zbioru możliwych akcji, a następnie bada rekcję otoczenia na nie. Reakcja otoczenia nie musi być natychmiastowa, może być odsunięta w czasie. Ta reakcja otoczenia jest właśnie owym wzmocnieniem (czasem zwanym wypłatą), czyli nagrodą lub karą za podjęte w przeszłości akcje. Jest to więc forma oceny działania agenta za podjęte działania. Akcje agenta wywołują zmiany zarówno w otoczeniu, jak i w nim samym - modyfikują jego strategię działania.

Uczenie ze wzmocnieniem

Generalnie idea metody polega na tym, że wzmocnieniu podlegają akcje lub strategie wykonywania akcji, które przynoszą większe wypłaty. Jest to heurystyka dość powszechnie stosowana przez organizmy żywe i ludzi także, np. jeśli twórca wyda płytę, która się dobrze sprzeda, to potem wydaje kolejną, dość podobną – oczywiście jeśli przesadzi, to może osiągnąć wynik odwrotny od zamierzonego...
Podobna sytuacja jest np. z filmami kinowymi: gdy osiągną sukces, to powstają sequele, prequele, wersje 1,5 i z wieloma innymi dziedzinami życia.

Uczenie ze wzmocnieniem

Celem uczącego się agenta jest wypracowanie strategii decyzyjnej, która zmaksymalizuje otrzymywane przez niego wypłaty (r_t) w długim horyzoncie czasowym. Najczęściej stosowane jest tu kryterium o postaci:

$$Q = \max_{a_0, a_1, \ldots} E\left[\sum_{t=0}^{\infty} \gamma^t * r_t\right]$$

czyli maksymalizujemy po wykonywanych akcjach (a_0, a_1,...) wartość oczekiwaną (agent i/lub środowisko mogą być stochastyczne) ze zdyskontowanej (γ - współczynnik dyskontowania, wybierany z przedziału (0,1)) sumy otrzymanych wypłat – wzmocnień. Współczynnik dyskonta jest odpowiedzialny za różnicowanie wpływu wypłat oddalonych w czasie, im będzie ona bliższa, tym większy będzie miała wpływ, gdyż γ wykładniczo maleje wraz z upływem czasu.

Uczenie ze wzmocnieniem

algorytm czasowego przypisania zasługi (Q-learning)

Wzór podany na poprzednim slajdzie nie jest wygodny do bezpośredniego stosowania, jest on zbyt ogólny. Jedną z grup metod dostosowujących ten ogólny schemat do łatwego użycia są metody czasowego przypisania zasługi, a między innymi metoda Q-learning, w której wyboru akcji dokonuje się na podstawie jak najwyższych wartości funkcji oceny Q:

$$Q_{t+1}(x,a)=\begin{cases}(1-\eta)*Q_t(x_t,a_t)+\eta*(r_t+\gamma*\max\limits_{a'}Q_t(x_{t+1},a')) & dla\ x=x_t, a=a_t \\ Q_t(x,a) & dla\ x\neq x_t, a\neq a_t\end{cases}$$

η - współczynnik szybkości uczenia (0,1), γ - współczynnik dyskontowania (0,1), r_t – wzmocnienie (wypłata lub kara), t – chwila czasowa, x – stan agenta, a – akcja agenta, $\max\limits_{a'}Q_t(x_{t+1},a')$ - oszacowanie maksymalnej wartości funkcji oceny.

Uczenie ze wzmocnieniem

przykład prostego zastosowania heurystyki

Załóżmy, że chcemy rozwiązać problem komiwojażera (TSP): objechać po najkrótszej drodze *n* miast, oczywiście pamiętając o odwiedzeniu wszystkich miast i nieodwiedzaniu żadnego wielokrotnie. Zadanie to jest trudne do rozwiązania, gdyż przestrzeń poszukiwań ma wielkość rzędu *n*!, więc dla liczby miast większej od 100 właściwie niemożliwe jest pełne jej przejrzenie i ocena rozwiązań. Dlatego też do rozwiązywania tego typu zadań stosuje się najczęściej heurystyki (więcej o nich na wykładzie 4).

Uczenie ze wzmocnieniem

przykład prostego zastosowania

Jedną z najprostszych heurystyk jest tzw. algorytm wzrostu, w którym wylosowane rozwiązanie początkowe (rozwiązanie to np. lista miast w kolejności odwiedzania) poddaje się losowej modyfikacji, a otrzymane rozwiązania (może być ich więcej niż 1) ocenia się i jeśli wśród nich jest lepsze od posiadanego bazowego, akceptuje się jako nowe bazowe, podlegające kolejnej modyfikacji. Tak powtarza się sekwencje modyfikacji i ewentualnych akceptacji, aż osiągniemy kryterium stopu, czyli np. określoną liczbę iteracji.

Uczenie ze wzmocnieniem

przykład prostego zastosowania

standardowy algorytm metody (największego) wzrostu

```
begin
    t:=0
    wylosuj rozwiązanie początkowe x_s
    oceń x_s
    x_0:=x_s
    repeat
        wygeneruj n nowych rozwiązań x_1...x_n przez modyfikację x_0 operatorem perturbacji
        oceń rozwiązania x_1...x_n
        wybierz najlepsze rozwiązanie x* z x_1...x_n
        jeśli x* jest lepsze od x_0 to x_0:=x*
        t:=t+1
  until t=MAX
end
```

Uczenie ze wzmocnieniem

przykład prostego zastosowania

W najprostszej wersji stosuje się jeden wariant losowej modyfikacji (perturbacji) rozwiązania. Jednakże można sobie wyobrazić wiele sposobów takich perturbacji (wiele operatorów perturbacji) - stosowanie różnych operatorów perturbacji daje zazwyczaj lepsze efekty niż zastosowanie tylko jednego. Który z nich wybrać oraz jak i kiedy stosować?

Uczenie ze wzmocnieniem

przykład prostego zastosowania

Przykładowe sposoby modyfikacji (perturbacji) rozwiązania:

1. Losowe przestawienie 2 miast w posiadanej liście.
2. Wylosowanie 2 miast i odwrócenie kolejności miast między nimi.
3. Wylosowanie 2 par sąsiednich miast i sprawdzenie, czy nie poprawimy wyniku, prowadząc drogę „na krzyż” między nimi.
4. Wylosowanie pewnego miasta na trasie i przestawienie obok niego miasta najbliższego.
5. Wylosowanie pewnego miasta na trasie i przestawienie obok niego miasta wylosowanego z pewnej grupy najbliższych.
6. Kilkukrotne powtórzenie (liczba powtórzeń może być losowana) jednego z wymienionych wyżej operatorów – traktowane jako osobny operator.
7. … - z pewnością można wymyślić jeszcze sporo innych sposobów modyfikacji rozwiązań w tym problemie.

Uczenie ze wzmocnieniem

przykład prostego zastosowania

Skoro nie wiemy, które z nich będą lepsze, a które gorsze, to może zastosujemy je wszystkie, tylko z odpowiednim mechanizmem wyboru, wzorowanym na uczeniu maszynowym, zgodnie z heurystyką „im większą poprawę rozwiązania daje operator, tym częściej go stosujmy”. Zastosujemy je wobec tego z pewnymi prawdopodobieństwami, które będą proporcjonalne do ich osiągnięć. W każdej iteracji będzie losowany 1 operator, a osiągnięta dzięki jego zadziałaniu „nagroda” podniesie prawdopodobieństwo jego wyboru w kolejnych krokach.

Uczenie ze wzmocnieniem

przykład prostego zastosowania

Prawdopodobieństwa te będą się zmieniać w trakcie poszukiwania rozwiązania problemu, gdyż jest bardzo prawdopodobne, że wymagania metody mogą się zmieniać w trakcie działania, np. operator 3 bardzo dobrze rozplątuje zapętlenia trasy - zapętlenia są zawsze nieoptymalne i przez zmianę dróg między 2 parami miast można je łatwo wyeliminować. Takich zapętleń jest na trasie pewna liczba, a gdy operator je wyeliminuje, nie będzie już potrzebny (przynajmniej dopóki nie powstaną nowe na skutek działania innych operatorów). Widać tu więc, że częstości wywoływania operatorów powinny się zmieniać: jedne maleć, gdy nie przynoszą (na jakimś etapie obliczeń) poprawy, inne rosnąć, gdy przynoszą.

Uczenie ze wzmocnieniem

przykład prostego zastosowania heurystyki z uczeniem

Dlatego też zaprojektujemy i zastosujemy tu mechanizm zarządzania operatorami. Naszym agentem będzie opisana wcześniej metoda obliczeniowa – algorytm wzrostu, która wykonuje akcje – wybiera i uruchamia któryś z operatorów modyfikujących rozwiązanie. Każdy wybrany i uruchomiony operator podlega ocenie. Jeśli któreś z nowych rozwiązań jest lepsze od posiadanego bazowego, otrzymuje nagrodę, jeśli są gorsze lub bez poprawy – karę.
Te nagrody i kary powinny układać się w jakąś prawidłowość, dzięki której powinniśmy przyspieszyć obliczenia w stosunku do metody bez oceny operatorów - powstanie wyuczona i ciągle dostrajana **strategia wybierania operatorów**, a unowocześniona metoda będzie miała zdolność uczenia się i adaptowania do aktualnego stanu środowiska obliczeniowego.

Uczenie ze wzmocnieniem

przykład prostego zastosowania
ulepszony algorytm metody (największego) wzrostu z UM

begin
  t:=0
  wylosuj rozwiązanie początkowe x_s
  ustaw (jednakowe?) oceny startowe $o_1(0)$... $o_k(0)$ operatorów v_1... v_k
  oceń x_s
  x_0:=x_s
  repeat
    wylosuj (wybierz) operator v_i z v_1... v_k na podstawie ocen operatorów $o_1(t)$... $o_k(t)$
    wygeneruj n nowych rozwiązań x_1...x_n przez modyfikację x_0 **wybranym** operatorem o_i
    oceń rozwiązania x_1...x_n
    oceń operatory v_1... v_k odpowiednio modyfikując $o_1(t)$... $o_k(t)$
    wybierz najlepsze rozwiązanie x^* z x_1...x_n
    jeśli x^* jest lepsze od x_0 to x_0:=x^*
    t:=t+1
  until t=MAX
end

Uczenie ze wzmocnieniem

przykład prostego zastosowania heurystyki z uczeniem

Zastosujemy tu wzór na ocenę operatorów, analogiczny do pokazanego wcześniej i przystosowany do konkretnego zadania:

$$V_n(t+1)=\begin{cases}(1-\eta)*V_k(t)+\eta*(r_t+\gamma*V^*(t+1)) \text{ dla } n=k\\ V_n(t) \text{ dla pozostałych } n\end{cases}$$

$V_n(t)$ – ocena operatora, r_t – nagroda lub kara, γ - współczynnik dyskonta, η – współczynnik nauki, $V^*(t+1)$ – oszacowanie maksymalnej wartości oceny, k – indeks operatora wybranego do modyfikacji rozwiązania,
oraz normalizację wypracowanych ocen, aby przekształcić je w prawdopodobieństwa:

$$p_k(t)=\frac{V_k(t)}{\sum_{i=1}^{n} V_i(t)}$$

Uczenie ze wzmocnieniem

przykład prostego zastosowania heurystyki z uczeniem

W taki sposób zastosowaliśmy metodę uczenia ze wzmocnieniem do automatycznego uczenia się sposobu doboru operatorów modyfikujących rozwiązanie w prostej heurystyce obliczeniowej – metodzie wzrostu. Tym samym przekształciliśmy ją w całkiem wyrafinowane narzędzie obliczeniowe, które może rozwiązywać także inne problemy, o ile zmienimy reprezentację rozwiązania, funkcję celu i zapewne także operatory.

Generacja zbioru reguł

Reguła decyzyjna to jedna z metod reprezentacji wiedzy, ich tworzenie na podstawie przykładów to pozyskiwanie wiedzy, czyli uczenie się.

- Reguły mogą przyjmować różne formy, najczęściej występują w postaci:

IF warunek THEN decyzja

warunek to najczęściej koniunkcja selektorów typu $x_1=v_1 \wedge x_2=v_2 \wedge \ldots \wedge x_n=v_n$

- Selektory mogą mieć postać równościową, nierównościową, przedziałową lub podzbiorową, decyzja to pewna akcja wykonywana po stwierdzeniu prawdziwości warunku.
- Zbiór reguł powinien być skonstruowany tak, żeby pokrywał wszystkie dane trenujące, czyli każdy przykład powinien być pokryty przez przynajmniej jedną regułę.
- Zbiór reguł może się czasem mylić w przypadku danych „użytecznych” – dane trenujące mogą nie pokrywać wszystkich możliwych przypadków lub zawierać błędy (również dane „użyteczne” mogą mieć błędy).
- Do generacji reguł używa się różnych metod, m. in. algorytmu AQ.

Generacja zbioru reguł

Wejście: cały zbiór danych.
Wyjście: zbiór reguł.

1. Znajdź niepokrytą daną *x**.
2. Utwórz nową ogólną regułę *r*=(*,*,...,*) (* - warunek spełniany przez wszystkie dane, decyzja – taka jak etykieta danej *x**).
3. Znajdź obiekt *y* z innej klasy decyzyjnej, niż *x**, pasujący do *r*.
4. Jeśli *y* nie istnieje – idź do 7.
5. Znajdź taką najlepszą (ze względu na wsparcie - pokrywanie przykładów) specjalizację *r*, aby *y* nie był pokrywany przez *r*. (Specjalizacja polega na zamianie pewnej „*” na wartość pochodzącą z obiektu *x**.)
6. Jeśli reguła *r* ma nadal kontrprzykłady, wróć do 3.
7. Zapamiętaj *r*, oznacz pokryte obiekty, wróć do 1.

Generacja zbioru reguł

Działanie algorytmu AQ

- wybieramy daną niepasującą do żadnej dotychczas znalezionej reguły (niepokrytą przez żadną regułę),
- znajdujemy najlepszą regułę, do której pasuje (początkowo ogólną),
- w razie istnienia pasujących elementów z innych kategorii uszczegóławiamy jej selektory, wstawiając tam wartości z wybranej niepokrytej danej
- powtarzamy działanie metody aż pokryjemy wszystkie dane.

Dziękuję za uwagę.