Systemy rozmyte

W poprzedniej części omówiliśmy teorię i techniki budowy inteligentnych systemów zdolnych do przetwarzania danych w języku naturalnym. Pewne jest, że będzie rosło zapotrzebowanie na maszyny, które mogą wchodzić w interakcje z ludźmi za pomocą języka naturalnego. Aby systemy mogły interpretować język naturalny i reagować w najbardziej rozsądny i niezawodny sposób, systemy wymagają dużego stopnia rozmycia. Mózg biologiczny może bardzo łatwo poradzić sobie z przybliżeniami danych wejściowych w porównaniu z tradycyjną logiką, którą zbudowaliśmy z komputerami. Na przykład, kiedy widzimy osobę, możemy wywnioskować iloraz starości bez wyraźnego poznania wieku osoby. Na przykład, jeśli widzimy dwuletnie dziecko, na iloraz starości interpretujemy to dziecko jako nie stare, a więc młody. Możemy łatwo poradzić sobie z dwuznacznością danych wejściowych. W tym przypadku nie musimy znać dokładnego wieku dziecka, aby uzyskać podstawową i bardzo podstawową interpretację danych wejściowych. Ten poziom rozmycia jest niezbędny, jeśli chcemy budować inteligentne maszyny. W rzeczywistych scenariuszach nie możemy polegać na dokładnym matematycznym i ilościowym danych wejściowych dla naszych systemów do pracy, chociaż nasze modele (na przykład głębokie sieci neuronowe) wymagają rzeczywistych danych wejściowych. Niepewności są częstsze, a charakter scenariuszy w świecie rzeczywistym jest wzmacniany przez niekompletność informacji kontekstowej, charakterystyczną przypadkowość i ignorancję danych. Poziomy ludzkiego rozumowania są w stanie poradzić sobie z tymi atrybutami w prawdziwym świecie. Podobny poziom rozmycia jest niezbędny do budowy inteligentnych maszyn, które mogą uzupełniać ludzkie możliwości, w pełnym tego słowa znaczeniu.Tu zajmiemy się podstawami teorii logiki rozmytej i jej implementacją do budowy następujących elementów:

* Adaptacyjne sieciowe systemy wnioskowania rozmytego

* Klasyfikatory z rozmytymi c-średnimi

* Neuro-rozmyte klasyfikatory

Omówimy następujące tematy:

* Podstawy logiki rozmytej

* Sieć ANFIS

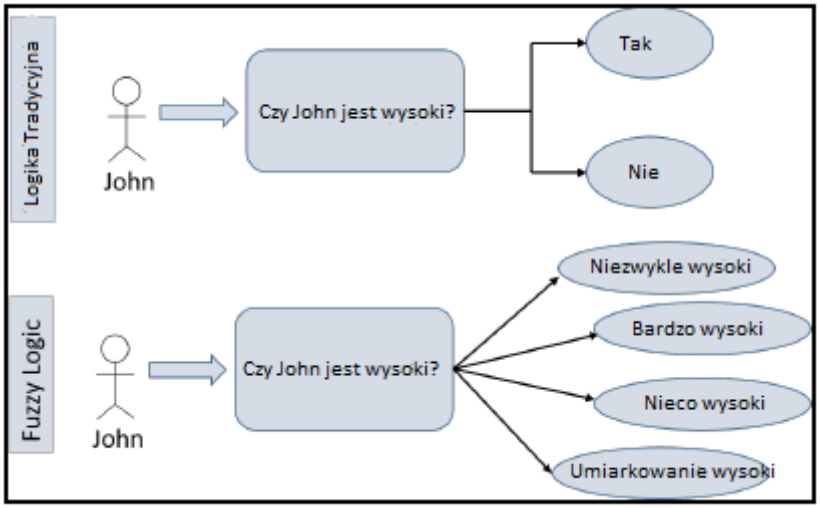
* Rozmyte C-oznacza oznacza grupowanie

* NEFCLASS

Podstawy logiki rozmytej

Szybko zrozumiemy, w jaki sposób interakcje międzyludzkie są płynne, nawet przy pewnym stopniu niejasności w naszych wypowiedziach. Oświadczenie takie jak John jest wysoki, nie ma żadnego wskazania dokładnej wysokości Johna w calach lub centymetrach. Jednak w kontekście rozmowy dwie osoby komunikujące się ze sobą mogą je zrozumieć i wywnioskować. Rozważmy teraz, że ta rozmowa odbywa się między dwoma nauczycielami w szkole na temat ucznia drugiej klasy, Johna. W tym kontekście stwierdzenie, że John jest wysoki, oznacza pewną wysokość, a my jesteśmy naprawdę dobrzy w zrozumieniu i wnioskowaniu o znaczeniu kontekstowym na podstawie tej niejasnej informacji. Podstawowa koncepcja logiki rozmytej wywodzi się z faktu, że wraz ze wzrostem złożoności kontekstu środowiskowego zmniejsza się nasza zdolność do precyzyjnego i dokładnego stwierdzania o stanie, a mimo to ludzki mózg jest w stanie wyciągać dokładne wnioski . Logika rozmyta reprezentuje

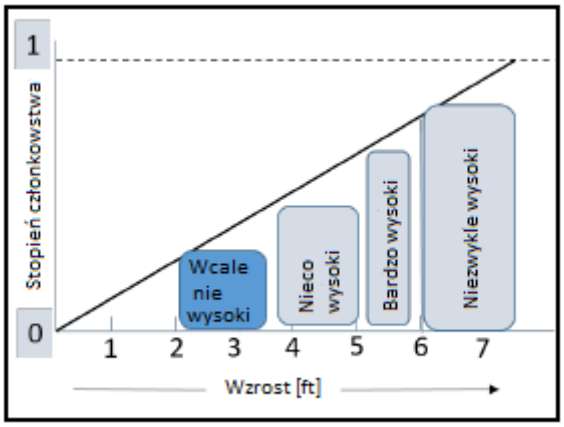
stopień prawdy zamiast prawdy absolutnej (czasami matematycznej). Przedstawmy różnicę między logiką tradycyjną a logiką rozmytą za pomocą prostego schematu:



Podczas gdy tradycyjne ramy obliczeniowe są lepiej dostosowane do tradycyjnej logiki, inteligentne systemy, które zamierzamy zbudować, muszą dostosować się do rozmytych danych wejściowych opartych na kontekście. Ramy obliczeniowe muszą przejść od prawdy absolutnej, tak / nie, do prawdy częściowej, wyjątkowo wysokiej, bardzo wysokiej i tak dalej. Jest to bardzo podobne do paradygmatu rozumowania człowieka, w którym prawda jest stronnicza, a fałsz jest malejącym stopniem prawdy.

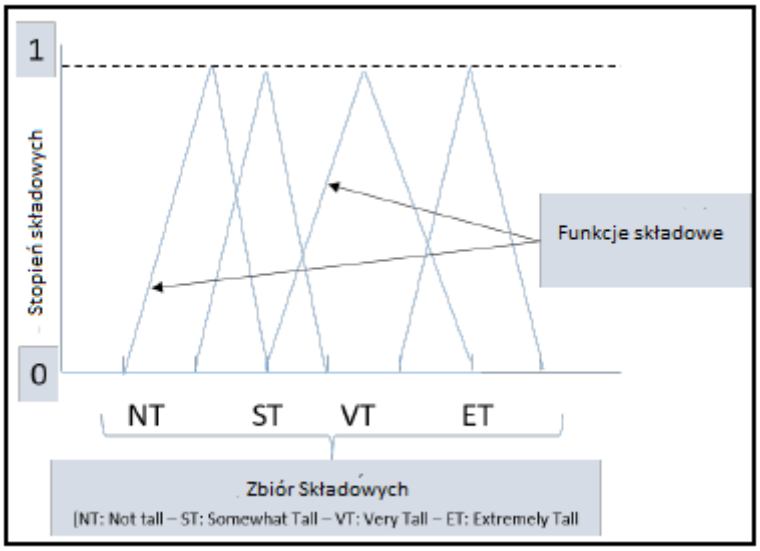
Zestawy rozmyte i funkcje składowe

W naszym przykładzie wszystkie możliwe odpowiedzi na pytanie o wysokość osoby stanowią zbiór. Ponieważ w każdej z wartości występuje wystarczająca niepewność, określa się ją mianem zbioru rozmytego. W tym przypadku rozmytym zestawem jest = k, {„Bardzo wysoki”, „Nieco wysoki”, „Średnio wysoki”}. Każda składowa zbioru ma wartość matematyczną reprezentującą poziom lub stopień członkostwa. W naszym przykładzie zestaw może być reprezentowany wraz ze stopniem członkostwa jako {„Niezwykle wysoki”: 1,0, „Bardzo wysoki”: 0,8, „Nieco wysoki”: 0,6, „Średnio wysoki”: 0,2}. Dane wejściowe można wykreślić na krzywej reprezentującej wartości w rozmytym zbiorze wraz ze stopniami członkostwa:



Zdefiniujmy standardową terminologię dotyczącą zbiorów rozmytych. Zbiór rozmytych jest zazwyczaj oznaczony znakiem „A”, który reprezentuje parametr przestrzeni danych X (w tym przypadku miara

wysokości). Zbiór rozmytych A jest definiowany za pomocą funkcji członkostwa μA (X), która wiąże każdą wartość w obrębie A z liczbą rzeczywistą od 0 do 1, oznaczającą stopień członkostwa w A. Przestrzeń członkostwa jest również nazywana wszechświatem dyskurs, który po prostu odnosi się do wszystkich możliwych wartości w zbiorze A. W przestrzeni wartości funkcja członkostwa musi spełniać tylko jeden warunek: stopień członkostwa dla wszystkich rozmytych elementów zbioru powinien wynosić od 0 do 1. W ramach tego ograniczenia , funkcje członkostwa mogą przybierać dowolną formę (trójkątną, sigmoidalną, krokową, gaussowską itd.) w zależności od zestawu danych i trudny kontekst. Oto reprezentacja funkcji członkowskich dla naszego zestawu danych, która oznacza wysokość dla osoby:



Zmienne językowe (NT / ST / VT / ET) mogą być powiązane ze zmiennymi numerycznymi (rzeczywisty wzrost osoby w calach) z poziomem przybliżenia lub rozmycia.

Atrybuty i notacje wyraźnych zbiorów

Wyraźny zestaw to zbiór bytów, które można wyraźnie oddzielić jako członków od osób niebędących członkami, na przykład zbiór obiektów żywych w porównaniu z obiektami nieożywionymi. W takim przypadku pojemnik całkowicie zawiera lub całkowicie wyklucza elementy. Istnieje kilka sposobów definiowania wyraźnych zestawów:

* Zestaw liczb parzystych większych niż 0 i mniejszych niż 10

* A = {2,4,6,8}

* Zestaw elementów należących do innego zestawu, P i Q

* A = {x | x jest elementem należącym do P i Q}

* $\mu_A(X) = 1$ jeśli $(x \in A)$, 0 jeśli $(x \notin A)$

* Φ: reprezentuje zestaw zerowy lub pusty

* Zestaw zasilania P (A) = {X | $x \subseteq A$}: jest to zestaw zawierający wszystkie możliwe podzbiory A

* W przypadku wyraźnych zestawów A i B zawierających super zestaw elementów w obrębie X:

* $x \subset A$ ==> x należy do A.

* x ∉ A ==> x nie należy do A.

* x ⊂ X ==> x należy do całego wszechświata X

* Rozważ wyraźne zestawy A i B na przestrzeni X.

* A ⊂ B ==> A jest całkowicie częścią B (jeśli x ∈ A to x ∈ B) - domyślnie

rozumowanie

* A ⊆ B ==> A jest zawarte lub równoważne z B.

* A = B ==> A ⊂ B lub B ⊂ A

Operacje na wyraźnych zestawach

Podobnie do liczb matematycznych, możemy wykonywać pewne operacje na wyraźnych zestawach:

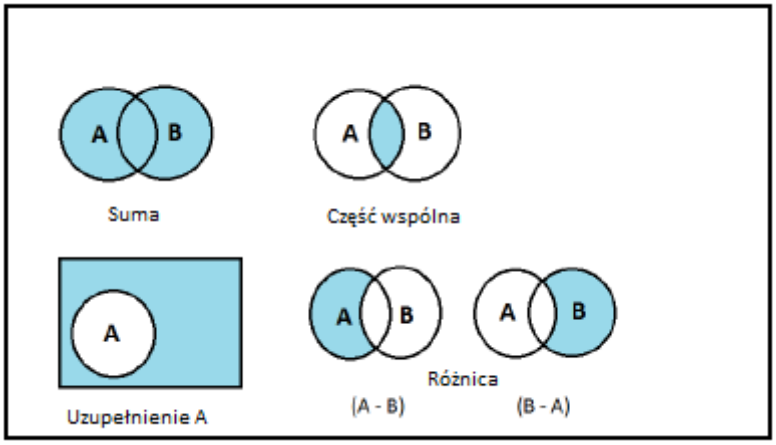
Unia: A ∪ B = {x | x ∈ A OR x ∈ B}

Przecięcie: A ∩ B = {x | x ∈ A AND x ∈ B}

Uzupełnienie: Ā = {x | x ∉ A, x ∈ X}

Różnica: A - B = A | B = {x | x ∉ A i x ∉ B} ==> A - (A ∩ B)

Oto jak reprezentujemy następujące operacje:



Właściwości wyraźnych zbiorów

Zestawy Crisp wykazują następujące właściwości:

Przemienność:

A ∪ B = B ∪ A

A ∩ B = B ∩ A

Stowarzyszenie:

A ∪ (B ∪ C) = (A ∪ B) ∪ C

A ∩ (B ∩ C) = (A ∩ B) ∩ C

Dystrybucja:

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

$A \cap (A \cup C) = (A \cap B) \cup (A \cap C)$

Idempotencja:

$A \cup A = A$

$A \cap A = A$

Przechodniość:

Jeżeli $A \subseteq B \subseteq C$, to $A \subseteq C$

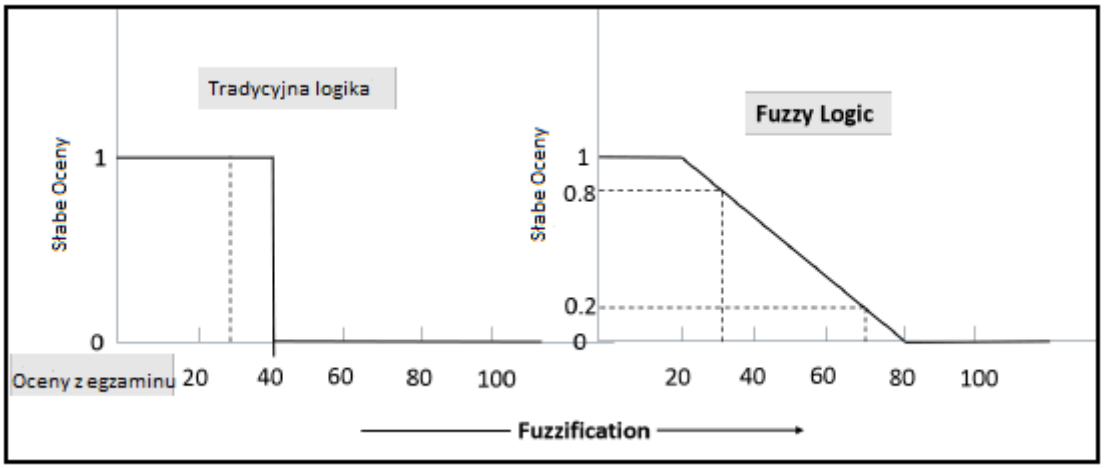
Fuzzification

Komputery cyfrowe są zaprojektowane i zaprogramowane do pracy przede wszystkim z wyraźnymi zestawami. Oznacza to, że są w stanie stosować operacje logiczne i wnioskowanie obliczeniowe w oparciu o zbiory klasyczne. Aby tworzyć inteligentne maszyny, potrzebujemy procesu zwanego fuzzification. W tym procesie wejścia cyfrowe są przekształcane w zestawy rozmyte. Przynależność do zbiorów rozmytych odpowiada pewnemu poziomowi pewności dla zbiorów rozmytych. Fuzzification to proces, w którym stopniowo przechodzimy od precyzyjnych symboli do niejasności dla reprezentacji elementów, co przekłada zmierzone wartości liczbowe na rozmyte wartości językowe. Rozważ zestaw liczb zbliżonych do liczby całkowitej 5:

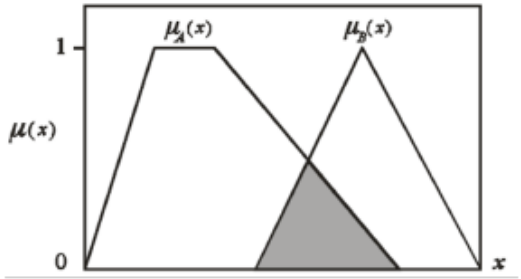
$A_{classic} = \{3,4,5,6,7\}$

$A_{fuzzy} = \{0,6 / 2, 0,8 / 3, 1,0 / 4, 1,0 / 5, 1,0 / 6, 0,8 / 7, 0,6 / 8\}$

Fuzzification to proces określania stopnia członkostwa zestawu członków. W przypadku zestawu klasycznego stopień przynależności wynosi 1 lub 0. Natomiast w zestawie rozmytym stopień przynależności waha się od 0 do 1. Poniższy schemat ilustruje zestaw danych reprezentujący słabość stopni. Załóżmy, że student otrzymuje na egzaminie oceny od 0 do 100. 0 jest minimalną, a zatem najgorszą oceną, a 100 jest najwyższą, a zatem wcale nie jest złą oceną:

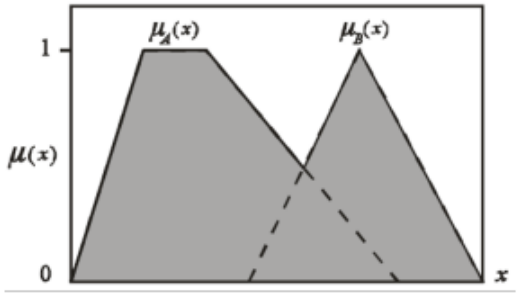


Jeśli uczeń uzyska 30 punktów na egzaminie, z tradycyjną logiką, otrzymał słabe oceny, ponieważ słabość ocen jest funkcją krokową, która traktuje wszystkie oceny poniżej 40 jako słabe, a powyżej 40 jako niezadowalające. W przypadku logiki rozmytej, jeśli uczeń otrzyma 30, ma 0,8 stopnia słabej oceny, a jeśli uczeń uzyska 70, ma 0,2 stopnia słabej oceny. Zestawy rozmyte nie muszą być odrębne i mogą się łączyć, przecinać, uzupełniać i różnicować:



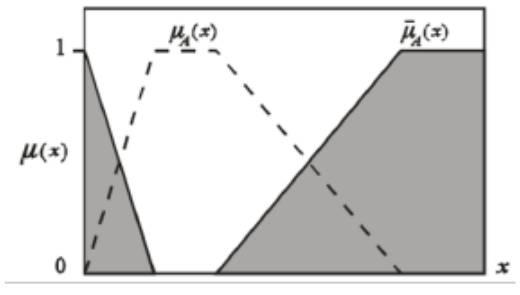
Część Wspólna

$$\mu_{A\cap B}(x) = min\ (\mu_A(x), \mu_B(x))\ \forall_x \in X$$



Suma

$$\mu_{A\cup B}(x) = max\ (\mu_A(x), \mu_B(x))\ \forall_x \in X$$



Uzupełnienie

$$\overline{\mu}_A(\chi) = 1 - \mu_A(\chi) \forall \in X$$

Funkcja rozmyta może przybierać dowolną złożoną formę na podstawie wnioskowania kontekstowego opartego na danych. Członkostwo dla elementów w zestawie rozmytym, które następuje po funkcji rozmytej, można zapewnić na wiele sposobów, w zależności od kontekstu:

* Członkostwo jako podobieństwo

* Członkostwo jako prawdopodobieństwo

* Członkostwo jako intensywność

* Członkostwo jako przybliżenie

* Członkostwo jako kompatybilność

* Członkostwo jako możliwość

* Funkcje członkostwa można generować na dwa sposoby:

* Temat: Intuicja / wiedza specjalistyczna / wiedza

* Automatyczne: grupowanie / sieci neuronowe / algorytmy genetyczne

Defuzzification

Defuzzification to proces, w którym możliwe do uzyskania wyniki są generowane jako wartości kwantyfikowalne. Ponieważ komputery mogą zrozumieć tylko wyraźne zestawy, można to również postrzegać jako proces konwersji rozmytych wartości ustawionych na podstawie kontekstu na wyraźne wyjście. Defuzzification interpretuje wartość członkostwa w oparciu o kształt funkcji członkostwa w rzeczywistą wartość. Odblokowana wartość reprezentuje działanie, jakie ma podjąć inteligentna maszyna na podstawie danych kontekstowych. Dostępnych jest wiele technik defuzzyfikacji; ten, który jest używany dla danego problemu, zależy od kontekstu.

Metody defuzyfikacji

Mamy następujące metody usuwania defuzji:

* Metoda środka sum

* Metoda środka ciężkości (COG) / centriod of area (COA)

* Metoda środka / dwusieczna obszaru (BOA)

* Metoda średniej ważonej

* Metody Maxima:

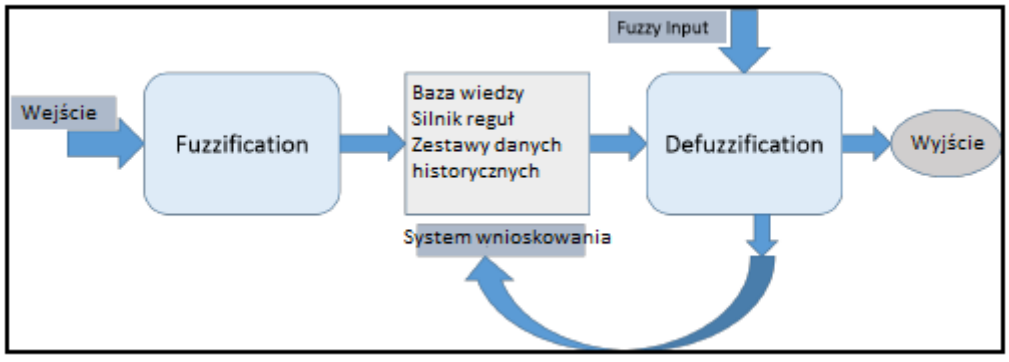
* Pierwsza metoda maksima (FOM)

* Metoda ostatniej z maksimów (LOM)

* Metoda średniej maksimów (MOM)

Wnioskowanie rozmyte

Wnioskowanie rozmyte jest faktycznym procesem, który łączy wszystko razem w celu sformułowania działań dla inteligentnych maszyn. Proces można przedstawić następująco:



W tradycyjnych systemach wejścia są odbierane jako wyraźne zbiory. Wyraziste dane wejściowe są zamazane jako funkcje członkostwa, a zestawy danych rozmytych są agregowane za pomocą technik unii / uzupełnienia / różnicowania. Po uzyskaniu zagregowanej funkcji członkostwa, stosujemy bazę wiedzy, reguły i wykorzystujemy historyczne zbiory danych, zanim odblokujemy zestaw danych wejściowych w możliwej do zastosowania wartości wyjściowej. Nowoczesne inteligentne systemy muszą bezpośrednio pracować z rozmytym wejściem; proces fuzyfikacji jest częścią kontekstu środowiskowego. Maszyny muszą interpretować język naturalny, aby zapewnić użytkownikom

końcowym bezproblemową obsługę. Jednostka fuzyfikująca musi obsługiwać różne metody fuzzyfikacji w celu przekształcenia wyraźnych danych wejściowych w zestawy rozmyte.

Sieć ANFIS

We wcześniejszych częściach widzieliśmy teorię i praktyczne zastosowania ANN. Łącząc ogólną teorię ANN z logiką rozmytą, jesteśmy w stanie uzyskać neuro-rozzmyty system, który jest bardzo wydajnym i potężnym mechanizmem do modelowania danych wejściowych ze świata rzeczywistego w inteligentnych maszynach i generowania wyników opartych na ocenie adaptacyjnej maszyny. To przybliża ramy obliczeniowe do sposobu, w jaki ludzki mózg interpretuje informacje i jest w stanie podjąć działania w ciągu ułamków sekund. Sama logika rozmyta ma zdolność wzajemnego przenikania się interpretacji danych, informacji i wiedzy przez człowieka i maszynę. Nie ma jednak nieodłącznej zdolności do tłumaczenia i modelowania procesu przekształcania ludzkich procesów myślowych w oparte na regułach, samouczące się, rozmyte systemy wnioskowania (FIS). ANN można wykorzystać do automatycznego dostosowania funkcji członkostwa w oparciu o kontekst środowiskowy i interaktywne szkolenie sieci w celu zmniejszenia poziomu błędu. To stanowi podstawę Sztucznych Neuro-Rozmytych Systemów Wnioskowania (ANFIS). ANFIS można uznać za klasę lub typ sieci adaptacyjnych równoważnych wnioskowaniu rozmytemu systemu wykorzystujące hybrydowy algorytm uczenia się.

Sieć adaptacyjna

Jest to rodzaj sprzężonej sieci neuronowej z wieloma warstwami, która często wykorzystuje nadzorowany algorytm uczenia. Ten typ sieci zawiera wiele adaptacyjnych węzłów, które są ze sobą połączone, bez żadnej wartości masy między nimi. Każdy węzeł w tej sieci ma inne funkcje i zadania. Zastosowana reguła uczenia się wpływa na parametry w węźle i zmniejsza poziomy błędów w warstwie wyjściowej. Ta sieć neuronowa jest zwykle trenowana z propagacją wsteczną lub spadkiem gradientu. Ze względu na powolność konwergencji można również zastosować podejście hybrydowe, które przyspiesza konwergencję i potencjalnie pozwala uniknąć lokalnych minimów.

Architektura ANFIS i algorytm uczenia hybrydowego

Podstawą architektury ANFIS jest sieć adaptacyjna, która wykorzystuje algorytm uczenia nadzorowanego. Rozumiemy to na prostym przykładzie. Weź pod uwagę, że istnieją dwa wejścia x i y oraz wyjście z. Możemy rozważyć użycie dwóch prostych reguł w metodzie if-then w następujący sposób:

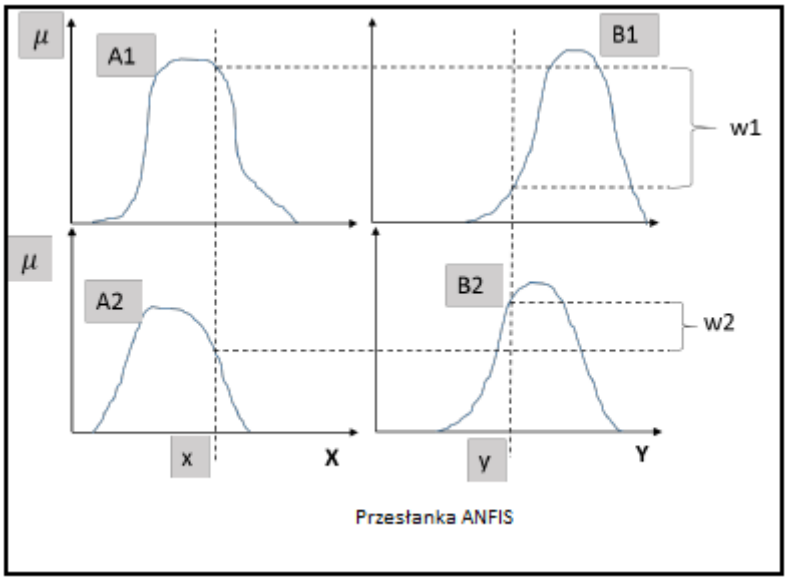
Zasada 1: Jeśli x oznacza A_1, a y oznacza B_1, to $z_1 = p_1x + q_1x + r_1$

Zasada 2: Jeśli x oznacza A_2, a y oznacza B_2, to $z_2 = p_2y + q_2y + r_2$

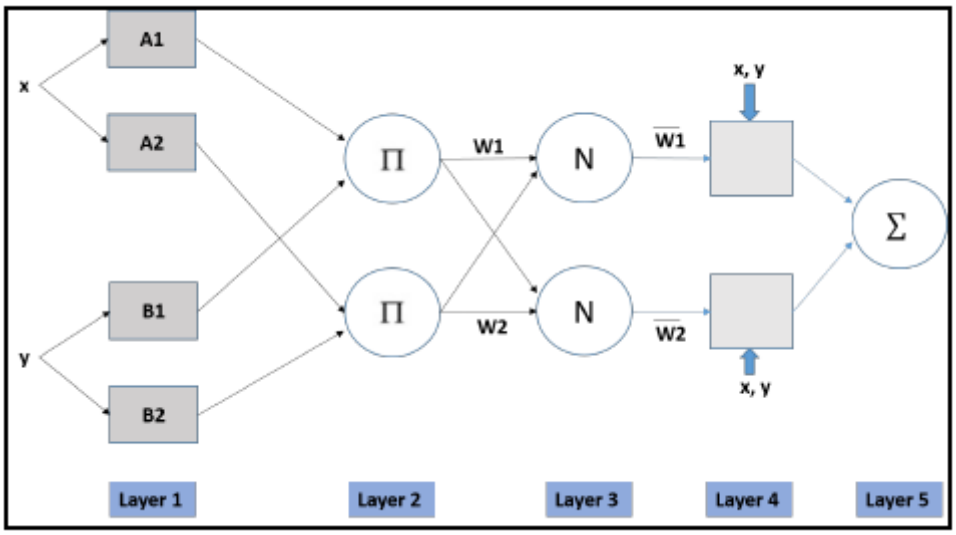
A1, A2 i B1, B2 są funkcjami członkostwa każdego wejścia x i y. p1, q1,

r1 i p2, q2, r2 są parametrami liniowymi modelu wnioskowania rozmytego.

Zilustrujmy to diagramem



Architekturę ANFIS w tym przypadku można uznać za pięciowarstwową sieć neuronową. Pierwsza i czwarta warstwa zawierają węzeł adaptacyjny, a pozostałe warstwy zawierają stałe węzły, jak widzieliśmy już w poprzednich rozdziałach o ANN. Sieć została zilustrowana na poniższym schemacie:



* Warstwa 1: Ta warstwa składa się z dwóch adaptacyjnych węzłów, które dostosowują się do parametru funkcji na podstawie wartości wejściowych (xiy). Dane wyjściowe z każdego z tych węzłów oznaczają stopień członkostwa odpowiadający wartości wejściowej (patrz przesłanka ANFIS na poprzednim schemacie). Funkcja członkostwa, jak widzieliśmy w poprzednich sekcjach, może przybierać dowolną formę (gaussowska, funkcja dzwonka itd.). Parametry w tej warstwie są nazywane parametrami wstępnymi:

z1 = p1x + q1y + r1

z2 = p2x + q2y + r2

* Warstwa 2: Węzły w tej warstwie są stałymi węzłami, które z natury nie są adaptacyjne i przypominają ukryty węzeł warstwy w sieci neuronowej. Sygnał wyjściowy z tych węzłów jest uzyskiwany przez pomnożenie sygnału pochodzącego z węzłów adaptacyjnych i dostarczany do

węzłów następnej warstwy. Węzły w tej warstwie reprezentują siłę strzału każdej z reguł, które są dziedziczone przez węzły adaptacyjne w poprzedniej warstwie.

* Warstwa 3: Węzły na tej warstwie są również stałymi węzłami. Każdy węzeł jest obliczoną wartością stosunku siły strzału n-tej reguły do sumy siły strzału wszystkich reguł. Ogólny wynik reprezentuje znormalizowaną siłę strzału.

* Warstwa 4: Węzły na tej warstwie są węzłami adaptacyjnymi. W tej warstwie znormalizowana siła strzału z poprzednich węzłów warstwy jest mnożona przez wynik funkcji reguł (p1x + q1x + r1 i p2y + q2y + r2). Parametry wyjściowe z tej warstwy nazywane są parametrami wynikowymi.

* Warstwa 5: jest to warstwa wyjściowa i ma jeden stały węzeł wyjściowy podobny do ANN. Ten węzeł dokonuje sumowania sygnałów z poprzedniej warstwy. Jest to ogólna wydajność sieci ANFIS. Przedstawia to ilościowy możliwy do działania wynik z systemu rozmytego. Dane wyjściowe można wykorzystać w pętli sterowania i propagować wstecz w celu szkolenia i optymalizacji, ostatecznie minimalizując błąd.

Po wdrożeniu tej topologii sieci możemy zastosować hybrydowy algorytm uczenia się w celu zoptymalizowania wyników i zmniejszenia błędu. Algorytm hybrydowy zapewnia również, że jesteśmy w stanie szybciej się konwergować i unikać lokalnych minimów. Algorytm hybrydowy to dwuetapowy proces, który zasadniczo poprawia parametry pierwszej i czwartej warstwy adaptacyjnej w oparciu o zestaw reguł. Podczas przejścia do przodu parametry pierwszej warstwy (parametry założenia) są utrzymywane na stałym poziomie, a parametry czwartej warstwy (wynikające z tego parametry) są dostosowywane w oparciu o metodę rekurencyjnego estymatora najmniejszych kwadratów (RLSE). Zauważ, że wynikające z tego parametry warstwy są liniowe i możemy przyspieszyć współczynnik konwergencji w procesie uczenia się. Po uzyskaniu kolejnych wartości parametrów dane są przekazywane przez przestrzeń wejściową i zagregowane funkcje członkostwa, a dane wyjściowe są generowane. Moc wyjściowa jest następnie porównywana z rzeczywistą mocą wyjściową. Po wykonaniu przejścia wstecznego, kolejne parametry uzyskane z pierwszego kroku są utrzymywane na stałym poziomie, a parametry wstępne są modyfikowane za pomocą metody uczenia się opadania gradientu lub propagacji wstecznej. Dane wyjściowe są ponownie generowane ze zmienionymi wartościami parametrów lokalu i porównywane z rzeczywistymi danymi wyjściowymi w celu dalszego strojenia i optymalizacji. Zastosowanie tego hybrydowego algorytmu, który łączy RLSE i opadanie gradientu, zapewnia szybszą konwergencję.

Rozmyte C-means oznacza grupowanie

Wcześniej widzieliśmy algorytm klastrowania k-średnich, który jest iteracyjnym algorytmem bez nadzoru, który tworzy klastry dla zestawu danych na podstawie odległości od losowej centroidu w pierwszym kroku iteracji. Centriody są obliczane w każdej iteracji, aby uwzględnić nowe punkty danych. Proces ten powtarza się, dopóki centriody nie zmienią się znacząco po punkcie. W wyniku algorytmu klastrowania k-średnich otrzymujemy dyskretne klastry z punktami danych. Każdy punkt danych albo należy do klastra, albo nie. Istnieją tylko dwa stany dla punktu danych pod względem członkostwa w klastrze. Jednak w rzeczywistych sytuacjach mamy punkty danych, które mogą należeć do wielu klastrów o różnym stopniu członkostwa. Algorytmy tworzące rozmyte członkostwo zamiast wyraźnego członkostwa w punktach danych w klastrze nazywa się algorytmami miękkiego klastrowania. Klastrowanie C-średnich jest jednym z najpopularniejszych algorytmów, który ma charakter iteracyjny i bardzo podobny do algorytmu grupowania k-średnich. Rozważmy zestaw danych S, który zawiera N punktów danych. Celem jest zgrupowanie tych N punktów danych w klastry C:

$S = \{x_1, x_2, x_3, ..., x_N\}$

Będziemy mieć funkcje członkostwa w klastrze C (oznaczone μ):

$\mu_1 = [\mu_1(x_1), \mu_1(x_2), \mu_1(x_3), ... \mu_1(x_n)]$

$\mu_2 = [\mu_2(x_1), \mu_2(x_2), \mu_2(x_3), ... \mu_2(x_n)]$

.

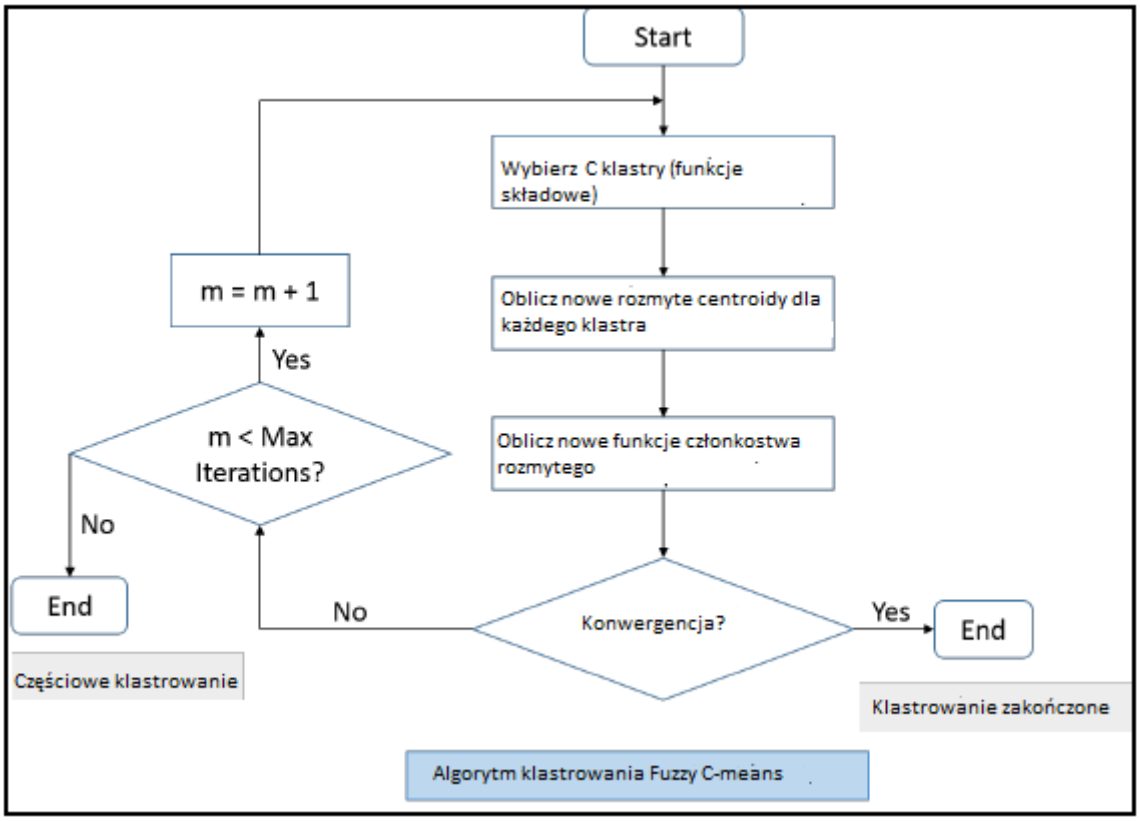
.

$\mu_c = [\mu_c(x_1), \mu_c(x_2), \mu_c(x_3), ... \mu_c(x_n)]$

Dla każdego klastra reprezentowanego przez funkcje członkostwa będziemy mieli centroidowy punkt danych, oznaczony przez Vi, odpowiadający rozmytemu klasterowi Cl_i (i = 1,2,3, ... C). Na podstawie tych informacji ogólnych i tych notacji cel optymalizacji dla algorytmu klastrowania C-means jest zdefiniowany jako:

$$J_m = \sum_{i=1}^{c} \sum_{k=1}^{N_s} [u_i(k)]^m d^2(x_k, V_i)$$

N_s jest całkowitą liczbą wektorów wejściowych; m oznacza wskaźnik niewyraźności dla i-tego skupiska (im wyższa wartość m, tym wyższa jest niewyraźność). Algorytm rozmytych C-średnich minimalizuje J_m, wybierając V_i i μ_i, gdzie i = 1,2,3, ... C w procesie iteracyjnym. Z tymi notacjami i celem algorytmu, oto schemat blokowy reprezentujący rozmyty algorytm C-mean:



Schemat blokowy wyjaśniono w następujący sposób:

*Inicjalizacja (Wybierz takie funkcje członkostwa, aby):

$$0 < \mu_i(x_k) < 1 \qquad \text{dla } i = 1, 2, 3, \ldots\ldots\ldots C$$

$$\sum_{i=1}^{C} \mu_i(x_k) = 1 \qquad \text{dla } k = 1,2,3, \ldots\ldots\ldots N_s$$

$$0 < \sum_{k=1}^{N_s} \mu_i(x_k) < N_s \qquad \text{dla } i = 1,2,3, \ldots\ldots\ldots C$$

* Oblicz rozmyte centroidy dla i = 1,2,3, ... C i k = 1,2,3, ... N_s:

$$V_i = \frac{\sum_{k=1}^{N_s} [\mu_i(x_k)]^m x_k}{\sum_{k=1}^{N_s} [\mu_i(x_k)]^m}$$

* Oblicz nowe rozmyte funkcje śladowe:

$$\mu_i(x_k) = \frac{\left(\frac{1}{d^2(x_k, V_i)}\right)^{\frac{1}{(m-1)}}}{\sum_{j=1}^{C} \left(\frac{1}{d^2(x_k, V_j)}\right)^{\frac{1}{(m-1)}}}$$

* Sprawdź zbieżność:

* Jeśli funkcje członkostwa nie zmieniają się podczas iteracji, iteracje mogą zostać zatrzymane, a algorytm zbiegnie się

* Gdy algorytm zbiegnie się, µi reprezentuje rozmyte klastry

* Jeśli algorytm nie jest zbieżny, a liczba iteracji jest równa maksymalnej liczbie iteracji ustawionych jako parametr, wychodzimy z pętli bez znalezienia optymalnych rozmytych klastrów

Wartości członkostwa dla punktów danych uzyskanych przez algorytm nie są unikalne, ponieważ istnieje zależność od początkowych warunków losowych. Istnieje możliwość konwergencji tego algorytmu do lokalnego minimum. Jeśli ustawimy próg dla wartości członkostwa, możliwe jest tworzenie twardych klastrów (tak samo jak algorytm klastrowania k-średnich). Na przykład możemy ustawić wartość progową na 0,8. Jeśli wartość członkostwa w klastrze jest większa niż 0,8, możemy uznać ją za wyraźną wartość członkostwa wynoszącą 1, a mniejszą niż 0,8 jako 0.

Zaimplementujmy ten algorytm w Spark:

```
import org.apache.spark.mllib.linalg.Vectors
```

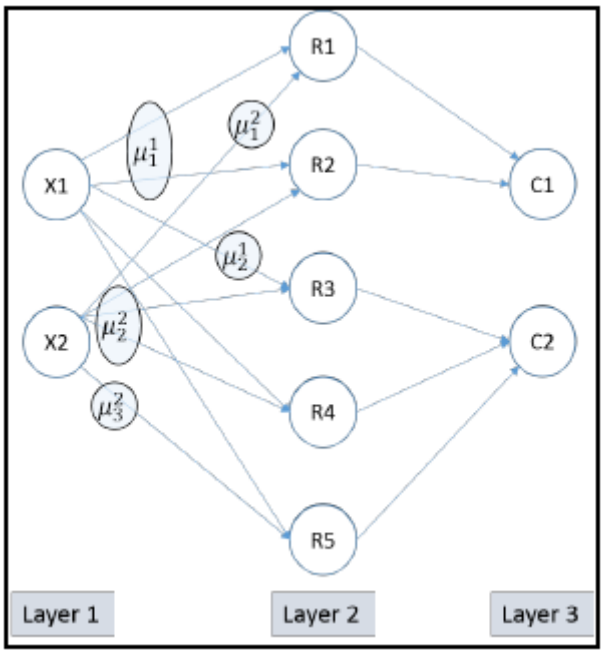
```
import scala.util.Random
import org.apache.spark.mllib.clustering._
import org.apache.spark.ml.clustering._
import org.apache.spark.mllib.clustering.KMeans
import org.apache.spark.mllib.clustering.FuzzyCMeans
import org.apache.spark.mllib.clustering.FuzzyCMeans._
import org.apache.spark.mllib.clustering.FuzzyCMeansModel
val points = Seq(
Vectors.dense(0.0, 0.0),
Vectors.dense(0.0, 0.1),
Vectors.dense(0.1, 0.0),
Vectors.dense(9.0, 0.0),
Vectors.dense(9.0, 0.2),
Vectors.dense(9.2, 0.0)
)
val rdd = sc.parallelize(points, 3).cache()
for (initMode <- Seq(KMeans.RANDOM, KMeans.K_MEANS_PARALLEL)) {
(1 to 10).map(_ * 2) foreach { fuzzifier =>
val model = FuzzyCMeans.train(rdd, k = 2, maxIterations = 10, runs
= 10, initMode, seed = 26031979L, m = fuzzifier)
val fuzzyPredicts = model.fuzzyPredict(rdd).collect()
rdd.collect() zip fuzzyPredicts foreach { fuzzyPredict =>
println(s" Point ${fuzzyPredict._1}")
fuzzyPredict._2 foreach{clusterAndProbability =&gt;
println(s"Probability to belong to cluster
${clusterAndProbability._1} " +
s"is ${"%.6f".format(clusterAndProbability._2)}")
}
}
}
}
```

Program wyświetli to rozmyte grupowanie:

Iteration - 1	Iteration - 10
Point [200.0] Probability to belong to cluster 0 is 0.000219 Probability to belong to cluster 1 is 0.999781 Point [204.0] Probability to belong to cluster 0 is 0.006037 Probability to belong to cluster 1 is 0.993963 Point [5.0] Probability to belong to cluster 0 is 0.998991 Probability to belong to cluster 1 is 0.001009 Point [198.0] Probability to belong to cluster 0 is 0.001727 Probability to belong to cluster 1 is 0.998273 Point [198.0] Probability to belong to cluster 0 is 0.001727 Probability to belong to cluster 1 is 0.998273 Point [4.0] Probability to belong to cluster 0 is 0.999958 Probability to belong to cluster 1 is 0.000042 Point [4.0] Probability to belong to cluster 0 is 0.999958 Probability to belong to cluster 1 is 0.000042 Point [203.0] Probability to belong to cluster 0 is 0.004299 Probability to belong to cluster 1 is 0.995701 Point [2.0] Probability to belong to cluster 0 is 0.997964 Probability to belong to cluster 1 is 0.002036 Point [195.0] Probability to belong to cluster 0 is 0.006995 Probability to belong to cluster 1 is 0.993005 Point [3.0] Probability to belong to cluster 0 is 0.999244 Probability to belong to cluster 1 is 0.000756 Point [201.0] Probability to belong to cluster 0 is 0.001303 Probability to belong to cluster 1 is 0.998697	Point [200.0] Probability to belong to cluster 0 is 0.497199 Probability to belong to cluster 1 is 0.502801 Point [204.0] Probability to belong to cluster 0 is 0.497317 Probability to belong to cluster 1 is 0.502683 Point [5.0] Probability to belong to cluster 0 is 0.502447 Probability to belong to cluster 1 is 0.497553 Point [198.0] Probability to belong to cluster 0 is 0.497136 Probability to belong to cluster 1 is 0.502864 Point [198.0] Probability to belong to cluster 0 is 0.497136 Probability to belong to cluster 1 is 0.502864 Point [4.0] Probability to belong to cluster 0 is 0.502424 Probability to belong to cluster 1 is 0.497576 Point [4.0] Probability to belong to cluster 0 is 0.502424 Probability to belong to cluster 1 is 0.497576 Point [203.0] Probability to belong to cluster 0 is 0.497288 Probability to belong to cluster 1 is 0.502712 Point [2.0] Probability to belong to cluster 0 is 0.502379 Probability to belong to cluster 1 is 0.497621 Point [195.0] Probability to belong to cluster 0 is 0.497036 Probability to belong to cluster 1 is 0.502964 Point [3.0] Probability to belong to cluster 0 is 0.502401 Probability to belong to cluster 1 is 0.497599 Point [201.0] Probability to belong to cluster 0 is 0.497229 Probability to belong to cluster 1 is 0.502771

NEFCLASS

W poprzednich rozdziałach poznaliśmy ogólną teorię sieci neuronowych, które przypominają ludzki mózg pod względem sieci połączonych ze sobą jednostek obliczeniowych. Sieci neuronowe są trenowane przez dostosowanie wag synaps (łączników). Jak widzieliście, sieć neuronowa może być przeszkolona do rozwiązywania problemów z klasyfikacją, takich jak rozpoznawanie obrazów. Sieci neuronowe akceptują wyraźne dane wejściowe i dostosowują wagi, aby uzyskać wartości wyjściowe (klasyfikacja do klasy). Jednak, jak widzieliśmy w tym rozdziale, dane wejściowe w świecie rzeczywistym mają pewien stopień rozmycia, a także pewien stopień niejasności w danych wyjściowych. Przynależność zmiennych wejściowych i wyjściowych do określonego klastra lub typu jest reprezentowana stopniem zamiast wyraźnego zestawu. Możemy połączyć dwa podejścia, aby sformułować neuro-rozmytą klasyfikację (NEFCLASS), która jest oparta na rozmytym wejściu i wykorzystuje elegancję wielowarstwowej sieci neuronowej w celu rozwiązania problemu klasyfikacji. W tej sekcji zrozumiemy algorytm i intuicję. Na wysokim poziomie NEFCLASS składa się z warstw wejściowych, reguł i wyjściowych. Neurony w tych warstwach są więc nazywane neuronami wejściowymi, neuronami rządzącymi i neuronami wyjściowymi. Oto ogólne przedstawienie strukturalne sieci NEFCLASS:



Layer 1 przetwarza dane wejściowe. Funkcja aktywacji w tej warstwie jest zazwyczaj funkcją tożsamości. Neurony w ukrytej warstwie 2 reprezentują rozmyte reguły, które zawierają rozmyte zbiory po stronie przesłanek i wniosków (wejście i wyjście, dla uproszczenia). W matematyce funkcja tożsamości, zwana także relacją tożsamości, mapą tożsamości lub transformacją tożsamości, jest funkcją, która zawsze zwraca tę samą wartość, która została użyta jako argument. W równaniach funkcję tę podaje f (x) = x. Zazwyczaj używane są zestawy rozmyte z trójkątnymi funkcjami członkostwa, a część rozmyta z funkcją członkostwa singleton jest używana w części podsumowującej. Przesłanki reguł rozmytych stają się wagami dla neuronów reguł w Warstwie 2. Wreszcie, konkluzja reguły jest połączeniem neuronu reguły z warstwą wyjściową. Kiedy obliczamy aktywację z neuronów reguł w Warstwie 2, używamy T-normy jako funkcji minimalizacji:

$$a_R{}^p = \min_{x \in U_1} \{W(x, R)(a_x{}^p)\}$$

W (x, R) reprezentuje wagę połączenia między neuronem wejściowym, x, a neuronem reguły, R.

Wagi dla neuronów reguł, podane we wcześniejszej formule, są wspólne dla każdej rozmytej wartości wejściowej i używany jest jeden zestaw rozmytych. Od reguły neuronowej warstwy (Warstwa 2) do warstwy klasyfikacyjnej (Warstwa 3) dołączone jest tylko jedno połączenie. To reprezentuje związek między regułą a klasą. Ostatnią warstwą jest warstwa wyjściowa, która oblicza wartość aktywacji dla danej klasy na podstawie aktywacji reguł wskazujących daną klasę jako wynik. W takim przypadku korzystamy z funkcji maksymalnej wskazanej poniżej:

$$a_c{}^p = \max_{R \in U_2} \{a_R{}^p\}$$

Po obliczeniu aktywacji w neuronach wyjściowych, neuron o najwyższej aktywacji jest wybierany w wyniku klasyfikacji.

Często Zadawane Pytania

P: Dlaczego potrzebujemy rozmytych systemów?

O: W naszym dążeniu do zbudowania inteligentnych maszyn nie możemy nadal modelować świata za pomocą wyraźnych, ilościowych i określonych danych wejściowych. Musimy modelować systemy takie jak ludzki mózg, które mogą łatwo zrozumieć i przetworzyć dane wejściowe, nawet jeśli nie są matematyczne i zawierają pewien stopień niejasności. Potrzebujemy rozmytych systemów, aby interpretować rzeczywiste dane wejściowe i wykonywać określone działania w oparciu o kontekst. Systemy rozmyte mogą zakłócać i odblokowywać dane wejściowe oraz ułatwiać nierozdzielność zdarzeń naturalnych i komputerów.

P: Co to są zestawy wyraźne i rozmyte? Czym się różnią od siebie?

O: Zestawy Crisp mają dwie możliwości dla członków. Konkretnym elementem / punktem danych / zdarzeniem jest element lub element niebędący elementem zestawu danych. Na przykład dni w tygodniu od poniedziałku do niedzieli są członkami wyraźnych dni tygodnia. Cokolwiek innego oprócz siedmiu dni nie jest członkiem zestawu. Z drugiej strony członkowie zbiorów rozmytych należą do zbioru rozmytego z pewnym stopniem członkostwa. W ten sposób odbywają się nasze rozmowy w języku naturalnym. Kiedy mówimy, że dana osoba jest wysoka, nie wspominamy o jej dokładnej wysokości. W ten punkt, jeśli wysokość jest uważana za funkcję przynależności, osoba o pewnej wysokości należy do rozmytego zbioru ze stopniem.

P: Czy zestawy rozmyte obsługują wszystkie operacje obsługiwane przez zestawy wyraźne?

O: Tak, zestawy rozmyte obsługują wszystkie operacje obsługiwane przez zestawy wyraźne, takie jak połączenie, przecięcie, uzupełnienie i różnicowanie.

Podsumowanie

Zrozumieliśmy podstawową teorię logiki rozmytej. Konieczne jest, aby budując inteligentne maszyny z ciągle rosnącymi ilościami danych, które są dostępne z dyskretnych źródeł w postaci ustrukturyzowanej, nieustrukturyzowanej i częściowo ustrukturyzowanej, maszyny potrzebują zdolności do komunikowania się ze światem rzeczywistym w taki sam sposób jak istoty ludzkie robić. Nie potrzebujemy wyraźnych danych matematycznych, aby podejmować nasze decyzje. W ten sam sposób, jeśli będziemy w stanie zinterpretować język naturalny i zastosować do obliczeń techniki rozmyte, będziemy w stanie stworzyć inteligentne maszyny, które naprawdę będą uzupełniać ludzi. Matematyczna teoria systemów rozmytych ma dziesięciolecia. Jednak wraz z nadejściem masowych ram przechowywania i przetwarzania danych możliwe są praktyczne implementacje, szczególnie w przypadku zbieżności logiki rozmytej i głębokich sieci neuronowych, a naprawdę inteligentny, samouczący się system wkrótce stanie się rzeczywistością. Ten rozdział stworzył podstawy do modelowania i przybliżenia naszych systemów do ludzkiego mózgu.